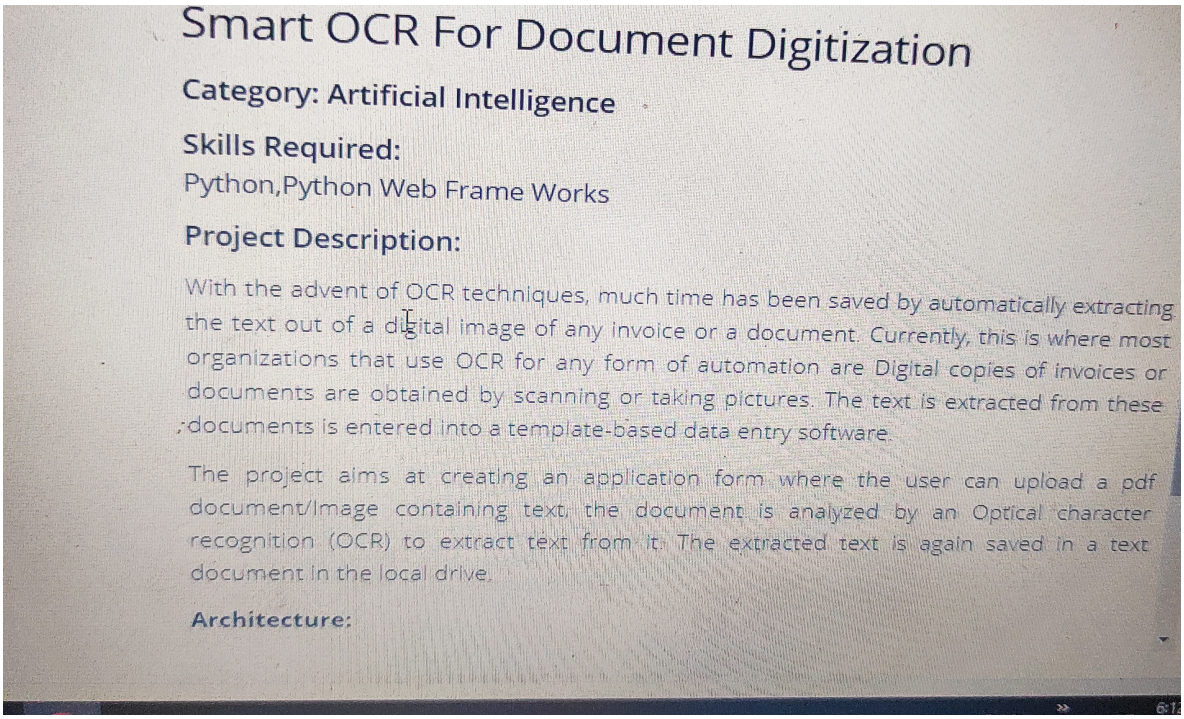
SMART OCR FOR DOCUMENT DIGITALIZATION

Optical character recognition or **optical character reader** (**OCR**) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example: from a television broadcast).[1]

Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs.[2] Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.



SMART OCR FOR DOCUMENT DIGITALIZATION

Learning Objectives & Project Flow

After Completing this project you should be able to

- Use Python, tesseract modules for text extraction
- Flask Web Framework

Project Flow:

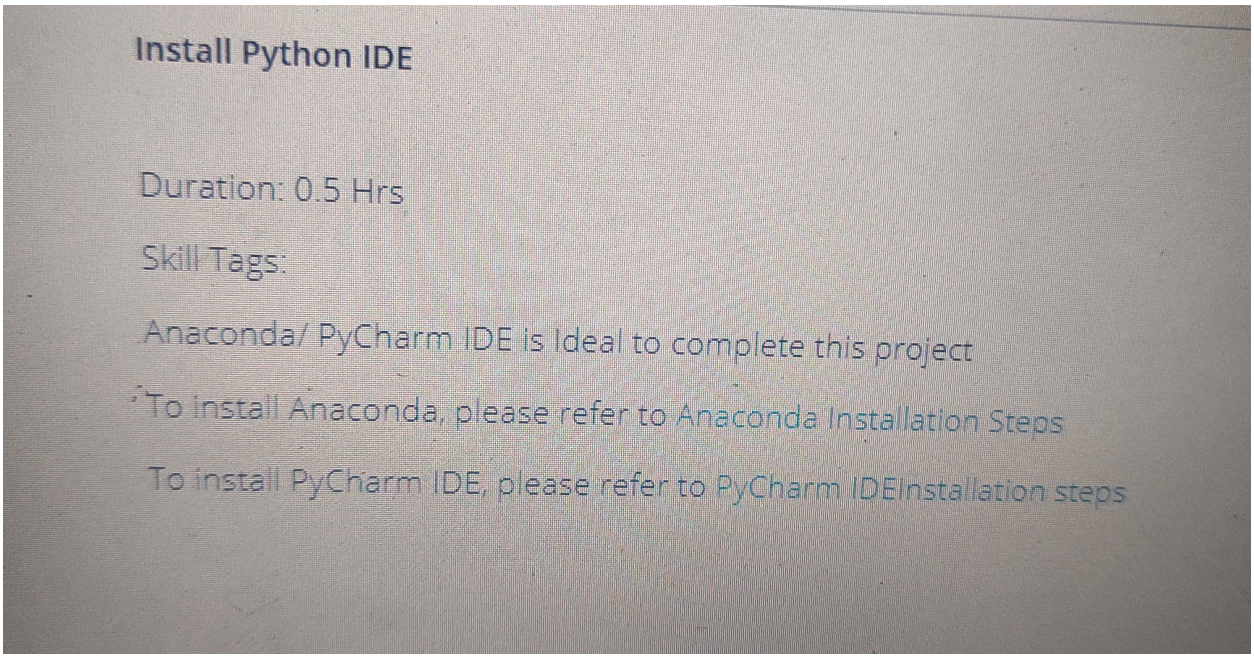
1. Upload a pdf document
2. Convert PDF document to image
3. Extract the text from the image
4. Store the extracted text in the text document

Necessary Installations

To complete this project you should have the following packages and Softwares

- **Python IDE**- For programming
- **pytesseract** -OCR package in python
- **pdf2image**- Converting PDF to Image
- **tesseract-ocr execution file** -Backend used for pytesseract
- **poppler**-Supporting file for pdf2image package
- **Flask**-Build web application

SMART OCR FOR DOCUMENT DIGITALIZATION



Contents

Early optical character recognition may be traced to technologies involving telegraphy and creating reading devices for the blind.[3] In 1914, Emanuel Goldberg developed a machine that read characters and converted them into standard telegraph code.[4] Concurrently, Edmund Fournier d'Albe developed the Optophone, a handheld scanner that when moved across a printed page, produced tones that corresponded to specific letters or characters.[5]

In the late 1920s and into the 1930s Emanuel Goldberg developed what he called a "Statistical Machine" for searching microfilm archives using an optical code recognition system. In 1931 he was granted USA Patent number 1,838,389 for the invention. The patent was acquired by IBM.

Blind and visually impaired users[edit]

In 1974, Ray Kurzweil started the company Kurzweil Computer Products, Inc. and continued development of omni-font OCR, which could recognize text printed in virtually any font (Kurzweil is often credited with inventing omni-font OCR, but it was in use by companies, including CompuScan, in the late 1960s and 1970s[3][6]). Kurzweil decided that the best application of this technology would be to create a reading machine for the blind, which would allow blind people to have a computer read text to them out loud. This device required the invention of two enabling technologies – the CCD flatbed scanner and the text-to-speech synthesizer. On January 13, 1976, the successful finished product was unveiled during a widely reported news conference headed by Kurzweil and the leaders of the National Federation of the Blind.[*citation needed*] In 1978, Kurzweil Computer Products began selling a commercial version of the optical character recognition computer program. LexisNexis was one of the first customers, and bought the program to upload legal paper and news documents onto its nascent online databases. Two years later, Kurzweil sold his company to Xerox, which had an interest in further commercializing paper-to-computer text conversion. Xerox eventually spun it off as Scansoft, which merged with Nuance Communications.

In the 2000s, OCR was made available online as a service (WebOCR), in a cloud computing environment, and in mobile applications like real-time translation of foreign-language signs on a smartphone. With the advent of smart-phones and smartglasses, OCR can be used in internet connected mobile device applications that extract text captured using the device's camera. These devices that do not have OCR functionality built into the operating system will typically use an OCR API to extract the text from the image file captured and provided by the device.[7][8] The OCR API returns the extracted text, along with information about the location of the detected text in the original image back to the device app for further processing (such as text-to-speech) or display.

Various commercial and open source OCR systems are available for most common writing systems, including Latin, Cyrillic, Arabic, Hebrew, Indic, Bengali (Bangla), Devanagari, Tamil, Chinese, Japanese, and Korean characters.

Applications[edit]

OCR engines have been developed into many kinds of domain-specific OCR applications, such as receipt OCR, invoice OCR, check OCR, legal billing document OCR.

They can be used for:

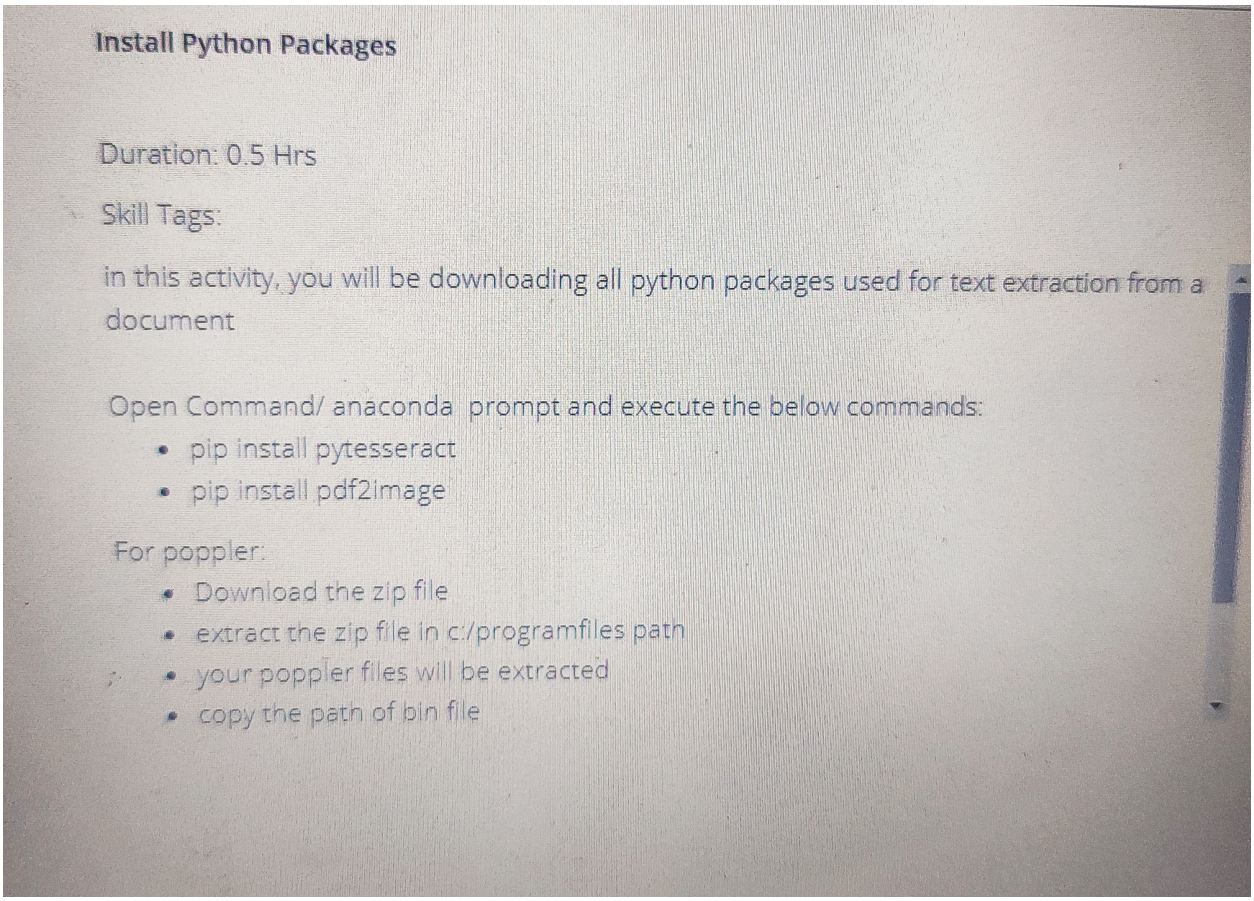
- Data entry for business documents, e.g. Cheque, passport, invoice, bank statement and receipt
- Automatic number plate recognition
- In airports, for passport recognition and information extraction
- Automatic insurance documents key information extraction[*citation needed*]
- Traffic sign recognition[9]
- Extracting business card information into a contact list[10]
- More quickly make textual versions of printed documents, e.g. book scanning for Project Gutenberg
- Make electronic images of printed documents searchable, e.g. Google Books
- Converting handwriting in real-time to control a computer (pen computing)
- Defeating CAPTCHA anti-bot systems, though these are specifically designed to prevent OCR.[11][12][13] The purpose can also be to test the robustness of CAPTCHA anti-bot systems.
- Assistive technology for blind and visually impaired users
- Writing the instructions for vehicles by identifying CAD images in a database that are appropriate to the vehicle design as it changes in real time.
- Making scanned documents searchable by converting them to searchable PDFs

Types[edit]

- Optical character recognition (OCR) – targets typewritten text, one glyph or character at a time.
- Optical word recognition – targets typewritten text, one word at a time (for languages that use a space as a word divider). (Usually just called "OCR".

OCR is generally an "offline" process, which analyses a static document. There are cloud based services which provide an online OCR API service. Handwriting movement analysis can be used as input to handwriting recognition.[14] Instead of merely using the shapes of glyphs and words, this technique is able to capture motions, such as the order in which segments are drawn, the direction, and the pattern of putting the pen down and lifting it. This additional information can make the end-to-end process more accurate. This technology is also known as "on-line character recognition", "dynamic character recognition", "real-time character recognition", and "intelligent character recognition"

SMART OCR FOR DOCUMENT DIGITALIZATION




```
import cv2
import pytesseract
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract'


img = cv2.imread('text.jpg')
width, height = 640,400
ImgResize=cv2.resize(img,(width,height))

img = cv2.cvtColor(ImgResize,cv2.COLOR_BGR2RGB)

##Detecting words
hImg,wImg,_ = img.shape
boxes = pytesseract.image_to_data(img)
for x,b in enumerate(boxes.splitlines()):
  if x!=0:
    #b = b.split(' ') #if we gave space values will print in tab space you can remove it
```

```
        b = b.split()
        print(b)
        if len(b)==12:
            x, y, w, h = int(b[6]), int(b[7]), int(b[8]), int(b[9]) #rectangle box coordinates
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),1)
            #cv2.rectangle(img,(x,hImg-y),(x+w,hImg-h),(0,0,255),1)
            cv2.putText(img,b[11],(x,y),cv2.FONT_HERSHEY_SIMPLEX,1,(50,50,255),2)

cv2.imshow('Result',img)
cv2.waitKey(0)
```

This code is for APP that is Smart OCR running

```
from __future__ import division, print_function
from flask import Flask,request, render_template
#from werkzeug import secure_filename
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer
import numpy as np
import cv2
from PIL import Image
import pytesseract
import sys
from pdf2image import convert_from_path
import os
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
import sys
import os.path
import glob
import random

app = Flask(__name__, static_url_path='')



@app.route('/', methods=['GET'])
def index():
    return render_template('base.html')


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
```

```
f = request.files['image']

basepath = os.path.dirname(__file__)
file_path = os.path.join(
    basepath, 'uploads', secure_filename(f.filename))
f.save(file_path)
PDF_file = file_path
# Store all the pages of the PDF in a variable
pages = convert_from_path(PDF_file, 500,poppler_path=r'C:\Program Files\poppler-0.68.0\bin')

# Counter to store images of each page of PDF to image
image_counter = 1

# Iterate through all the pages stored above
for page in pages:

    # Declaring filename for each page of PDF as JPG
    # For each page, filename will be:
    # PDF page 1 -> page_1.jpg
    # PDF page 2 -> page_2.jpg
    # PDF page 3 -> page_3.jpg
    # ....
    # PDF page n -> page_n.jpg
    filename = "page_"+str(image_counter)+".jpg"

    # Save the image of the page in system
    page.save(filename, 'JPEG')

    # Increment the counter to update filename
    image_counter = image_counter + 1

'''
Part #2 - Recognizing text from the images using OCR
'''
# Variable to get count of total number of pages
filelimit = image_counter-1

# Creating a text file to write the output
basepath = os.path.dirname(__file__)
file_path2 = os.path.join(
    basepath, 'outputs', "output"+str(random.randint(1, 100000))+".txt")



# Open the file in append mode so that
```

```
    # All contents of all images are added to the same file
    f = open(file_path2, "a")

    # Iterate from 1 to total number of pages
    for i in range(1, filelimit + 1):

      # Set filename to recognize text from
      # Again, these files will be:
      # page_1.jpg
      # page_2.jpg
      # ....
      # page_n.jpg
      filename = "page_"+str(i)+".jpg"

      # Recognize the text as string in image using pytesserct
      text = str(((pytesseract.image_to_string(Image.open(filename)))))

      # The recognized text is stored in variable text
      # Any string processing may be applied on text
      # Here, basic formatting has been done:
      # In many PDFs, at line ending, if a word can't
      # be written fully, a 'hyphen' is added.
      # The rest of the word is written in the next line
      # Eg: This is a sample text this word here GeeksF-
      # orGeeks is half on first line, remaining on next.
      # To remove this, we replace every '-\n' to ".
      text = text.replace('-\n', '')

      # Finally, write the processed text to the file.
      f.write(text)

    # Close the file after writing all the text.
    f.close()
  return file_path2


if __name__ == '__main__':
  #port = int(os.getenv('PORT', 8000))
  #app.run(host='0.0.0.0', port=port, debug=True)
  #http_server = WSGIServer(('0.0.0.0', port), app)
  #http_server.serve_forever()
  app.run(debug=False)
```

SMART OCR FOR DOCUMENT DIGITALIZATION

It is for video capture

```
import cv2
import numpy as np
import pytesseract
from PIL import ImageGrab
import time
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract'

cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)
def captureScreen(bbox=(300,300,1500,1000)):
    capScr = np.array(ImageGrab.grab(bbox))
    capScr = cv2.cvtColor(capScr, cv2.COLOR_RGB2BGR)
    return capScr
while True:
    timer = cv2.getTickCount()
    _,img = cap.read()
    img = captureScreen()
#DETECTING CHARACTERES
hImg, wImg,_ = img.shape
boxes = pytesseract.image_to_boxes(img)
for b in boxes.splitlines():
    print(b)
    b = b.split(' ')
    print(b)
    x, y, w, h = int(b[1]), int(b[2]), int(b[3]), int(b[4])
    cv2.rectangle(img, (x,hImg- y), (w,hImg- h), (50, 50, 255), 2)
    cv2.putText(img,b[0],(x,hImg- y+25),cv2.FONT_HERSHEY_SIMPLEX,1,(50,50,255),2)
    fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer);
    cv2.putText(img, str(int(fps)), (75, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (20,230,20), 2);

    cv2.imshow("Result",img)
    cv2.waitKey(1)
```