Similar Document Template Matching Algorithm

Bounding Box Detection and Visualization

In this code, we demonstrate how to detect and visualize bounding boxes around text regions in an image using Python. We utilize the OpenCV library for image manipulation and pytesseract for text recognition.

Code Explanation

1. **Import Required Libraries**: We import the necessary libraries, including `cv2` for OpenCV, `pytesseract` for text extraction, and `Output` from `pytesseract`.
2. **Load the Image**: We load the image `invoice_0_charspace_1.jpg` into the `img` variable using OpenCV's `cv2.imread()` function.
3. **Text Detection with pytesseract**: We use pytesseract's `image_to_data` function to detect text regions in the image. The result is stored in the dictionary `d`, containing information about each detected text region, such as its coordinates, size, and confidence score.
4. **Print Available Information**: We print the keys of the `d` dictionary to see the available information about each detected text region.
5. **Bounding Box Visualization**: We iterate through the detected text regions and check if their confidence score is greater than 60. If the confidence is above the threshold, we draw a green bounding box around the text region using `cv2.rectangle()`.
6. **Display the Image**: We create an OpenCV window named "output," resize the image for better visualization, and display the image with the bounding boxes using `cv2.imshow()`. The `cv2.waitKey(0)` function waits for a key press before closing the window.
7. **Cleanup**: Finally, we destroy the OpenCV window to release resources.

Please make sure to adjust the image path if necessary and ensure that you have OpenCV and pytesseract properly installed in your environment before running this code.

In [ ]:
```
import cv2
import pytesseract
from pytesseract import Output
```

In [2]:
```
img = cv2.imread("/home/neel/invoice_0_charspace_1.jpg")
```

In [3]:
```
d = pytesseract.image_to_data(img, output_type=Output.DICT)
# Print the keys of the resulting dictionary to see the available information
print(d.keys())
```

```
dict_keys(['level', 'page_num', 'block_num', 'par_num', 'line_num', 'word_num', 'lef
t', 'top', 'width', 'height', 'conf', 'text'])
```

In [4]:
```
n_boxes = len(d['text'])
for i in range(n_boxes):
    if float(d['conf'][i]) > 60:  # Check if confidence score is greater than 60
        (x, y, w, h) = (d['left'][i], d['top'][i], d['width'][i], d['height'][i])
        img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)



import cv2
cv2.namedWindow("output", cv2.WINDOW_NORMAL)    # Create window with freedom of dimens
# im = cv2.imread("img.jpg")                    # Read image
imS = cv2.resize(img, (540, 540))                # Resize image
cv2.imshow("output", imS)                       # Show image
cv2.waitKey(0)  z
# cv2.imshow('img', img)
# cv2.waitKey(0)
```

Out[4]: 225

In [ ]:
```
cv2.destroyWindow('img')
```

Getting th Coordinates of Logo

1. **Importing Libraries:** The code starts by importing the necessary libraries.
   - `spacy` is imported for natural language processing tasks, although it's not used in this code.
   - `cv2` (OpenCV) is imported for image processing.
   - `numpy` is imported for numerical operations.
2. **Loading Pretrained YOLO Model:** The code loads a pretrained YOLO (You Only Look Once) model for object detection. YOLO is a deep learning model used for object detection tasks.
3. **Loading Class Names:** It reads a file named "coco.names" that contains the class names used in the YOLO model. These class names include common objects and items that can be detected by the model.
4. **Loading and Preprocessing the Image:** The code loads an image named "Original_01_page-0001.jpg" and preprocesses it for YOLO input. It resizes the image to 416x416 pixels, scales pixel values to a range between 0 and 1, and performs color channel swapping.
5. **Setting Input for the YOLO Model:** The preprocessed image is set as the input to the YOLO model.
6. **Getting Detections:** The code forward-propagates the image through the YOLO model to obtain object detections. The `net.getUnconnectedOutLayersNames()` method retrieves the output layer names of the model.
7. **Looping Over Detections:** The code then loops through the detected objects in the image.
   - It extracts scores, class IDs, and confidence values for each detected object.
   - If the confidence of a detected object is greater than 0.5, it assumes the object is a logo.
8. **Extracting Logo Coordinates:** For each detected logo, the code extracts its coordinates and dimensions.
   - `center_x` and `center_y` are the coordinates of the center of the logo.
   - `width` and `height` represent the dimensions of the bounding box around the logo.
   - `x`, `y`, `x2`, and `y2` calculate the coordinates of the top-left and bottom-right corners of the bounding box.
9. **Cropping the Logo Region:** It crops the region of the image containing the detected logo using the calculated coordinates.
10. **Checking Logo Region:** It checks if the cropped logo region contains non-black and non-white pixels, assuming an RGB image. This step helps verify if the region genuinely contains a logo.
11. **Printing Information:** If the region is considered a logo, the code prints the confidence score and the coordinates of the top-left corner of the logo bounding box.

Overall, this code snippet uses a pretrained YOLO model to detect logos in an image, extracts their coordinates, and checks if the region truly contains a logo based on pixel color. Detected logos meeting the confidence threshold are printed with their coordinates.

In [3]:
```
import spacy
# import en_core_web_sm
# Load the English pre-trained model with NER
nlp = spacy.load('en_core_web_sm')
```

```
/home/neel/anaconda3/lib/python3.8/site-packages/torch/cuda/__init__.py:546: UserWarni
ng: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
```

In [1]:
```
!python3 -m spacy download en_core_web_sm
```

```
/home/neel/anaconda3/lib/python3.8/site-packages/torch/cuda/__init__.py:546: UserWarni
ng: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
Collecting en-core-web-sm==3.6.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_
sm-3.6.0/en_core_web_sm-3.6.0-py3-none-any.whl (12.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 4.6 MB/s eta 0:00:00m eta
0:00:01[36m0:00:01m
Requirement already satisfied: spacy<3.7.0,>=3.6.0 in ./anaconda3/lib/python3.8/site-p
ackages (from en-core-web-sm==3.6.0) (3.6.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in ./anaconda3/lib/python3.
8/site-packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in ./anaconda3/lib/python3.
8/site-packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in ./anaconda3/lib/python3.8/
site-packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (1.0.9)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in ./anaconda3/lib/python3.8/site-p
ackages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (2.0.7)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in ./anaconda3/lib/python3.8/site
-packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (3.0.8)
Requirement already satisfied: thinc<8.2.0,>=8.1.8 in ./anaconda3/lib/python3.8/site-p
ackages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (8.1.12)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in ./anaconda3/lib/python3.8/site-
packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in ./anaconda3/lib/python3.8/site-p
ackages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (2.4.7)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in ./anaconda3/lib/python3.8/si
te-packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (2.0.9)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in ./anaconda3/lib/python3.8/site-
packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (0.9.0)
Requirement already satisfied: pathy>=0.10.0 in ./anaconda3/lib/python3.8/site-package
s (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (0.10.2)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in ./anaconda3/lib/python3.8/s
ite-packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in ./anaconda3/lib/python3.8/site-p
ackages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (4.59.0)
Requirement already satisfied: numpy>=1.15.0 in ./anaconda3/lib/python3.8/site-package
s (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (1.20.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in ./anaconda3/lib/python3.8/si
te-packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (2.25.1)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in ./anaconda3/li
b/python3.8/site-packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (2.3.0)
Requirement already satisfied: jinja2 in ./anaconda3/lib/python3.8/site-packages (from
 spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (2.11.3)
Requirement already satisfied: setuptools in ./anaconda3/lib/python3.8/site-packages
 (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (68.2.1)
Requirement already satisfied: packaging>=20.0 in ./anaconda3/lib/python3.8/site-packa
ges (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (23.1)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in ./anaconda3/lib/python3.8/si
te-packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (3.3.0)
Requirement already satisfied: annotated-types>=0.4.0 in ./anaconda3/lib/python3.8/sit
e-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.7.0,>=3.6.0->en-core-we
b-sm==3.6.0) (0.5.0)
Requirement already satisfied: pydantic-core==2.6.3 in ./anaconda3/lib/python3.8/site-
packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.7.0,>=3.6.0->en-core-web-
sm==3.6.0) (2.6.3)
Requirement already satisfied: typing-extensions>=4.6.1 in ./anaconda3/lib/python3.8/s
ite-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.7.0,>=3.6.0->en-core-
web-sm==3.6.0) (4.7.1)
Requirement already satisfied: chardet<5,>=3.0.2 in ./anaconda3/lib/python3.8/site-pac
kages (from requests<3.0.0,>=2.13.0->spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (4.0.
0)
Requirement already satisfied: idna<3,>=2.5 in ./anaconda3/lib/python3.8/site-packages
 (from requests<3.0.0,>=2.13.0->spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./anaconda3/lib/python3.8/site
-packages (from requests<3.0.0,>=2.13.0->spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0)
 (1.26.4)
Requirement already satisfied: certifi>=2017.4.17 in ./anaconda3/lib/python3.8/site-pa
ckages (from requests<3.0.0,>=2.13.0->spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (202
0.12.5)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in ./anaconda3/lib/python3.8/site-pa
ckages (from thinc<8.2.0,>=8.1.8->spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (0.7.10)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in ./anaconda3/lib/python3.8/s
ite-packages (from thinc<8.2.0,>=8.1.8->spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0)
 (0.1.3)
Requirement already satisfied: click<9.0.0,>=7.1.1 in ./anaconda3/lib/python3.8/site-p
ackages (from typer<0.10.0,>=0.3.0->spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (7.1.
2)
Requirement already satisfied: MarkupSafe>=0.23 in ./anaconda3/lib/python3.8/site-pack
ages (from jinja2->spacy<3.7.0,>=3.6.0->en-core-web-sm==3.6.0) (1.1.1)
DEPRECATION: pyodbc 4.0.0-unsupported has a non-standard version number. pip 23.3 will
 enforce this behaviour change. A possible replacement is to upgrade to a newer versio
n of pyodbc or contact the author to suggest that they release a version with a confor
ming version number. Discussion can be found at https://github.com/pypa/pip/issues/120
63
Installing collected packages: en-core-web-sm
  Attempting uninstall: en-core-web-sm
    Found existing installation: en-core-web-sm 2.2.0
    Uninstalling en-core-web-sm-2.2.0:
      Successfully uninstalled en-core-web-sm-2.2.0
Successfully installed en-core-web-sm-3.6.0
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

In [1]:
```
import spacy
```

```
/home/neel/anaconda3/lib/python3.8/site-packages/torch/cuda/__init__.py:546: UserWarni
ng: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
```

In [2]:
```
nlp = spacy.load('en_core_web_sm')
```

In [3]:
```
with open('Invoice2.pdf', 'r', encoding='utf-8',
                 errors='ignore') as f:
    text = f.read()

# Apply the NER model to the invoice text
doc = nlp(text)
```

In [4]:
```
print(doc)
```


```
%PDF-1.4
1 0 obj
<<
/Title (Document)
/Creator (wkhtmltopdf 0.12.6.1)
/Producer (Qt 4.8.7)
/CreationDate (D:20230911190846Z)
>>
endobj
3 0 obj
<<
/Type /ExtGState
/SA true
/SM 0.02
/ca 1.0
/CA 1.0
/AIS false
/SMask /None>>
endobj
4 0 obj
[/Pattern /DeviceRGB]
endobj
8 0 obj
<<
/Type /Catalog
/Pages 2 0 R
>>
endobj
5 0 obj
<<
/Type /Page
/Parent 2 0 R
/Contents 9 0 R
/Resources 11 0 R
/Annots 12 0 R
/MediaBox [0 0 612 792]
>>
endobj
11 0 obj
<<
/ColorSpace <<
/PCSp 4 0 R
/CSp /DeviceRGB
/CSpg /DeviceGray
>>
/ExtGState <<
/GSa 3 0 R
>>
/Pattern <<
>>
/Font <<
/F6 6 0 R
/F7 7 0 R
>>
/XObject <<
>>
>>
endobj
12 0 obj
[ ]
endobj
9 0 obj
<<
/Length 10 0 R
/Filter /FlateDecode
>>
stream
x[Ko6W
,
N\FP6-b e5Mveo$$5"9qtr;Wvv\U6/5 3d=VjSm}>VX_>i5oGiTYN>VWI
%`LRX{H5|-Sj9$}!jsZvU`-T.~<~f{u~}<zOt oPWxb~&!%8/wiC(qw=c~? ſy:Ij0l{ T
l+4[en7l?pKIQ6\Q+I"kmR"h<GG@w6it$a`F hkbDRdH\.KIx9)|HoD.]Z4;E9]HX@5
\ghR~Qml%      D>Qkpn%7$-B!0fi
x>wS]M@dN2\}.(vo[Zyhn X&?4cz k:5>J##fBB!j1@rY&qTk(M7`}  ;nHMH
@:P#GWnt SLmXO@$Pt]\d        9/Pr'xer\UD]}] TV>U0tsye:pE)u8
i%ft'l@iMy-HXG20%Y`z`=v l)\'d -!B!Og\_*>CeQuBT|NHq}H'Bcf Mo~|5 92\gY}V\5
H]9QnX*;]*zq##Bv
M\iol>*$Z;NrKä:0
^Hp\
yPGe#3u#CI7)r&Yh~4A1H%_A9DWPAQ?@cGr2tHG?QO7!m!5\l5uQK_8ijO!|h!
7*_<'?).beX
endstream
endobj
10 0 obj
1235
endobj
13 0 obj
<< /Type /FontDescriptor
/FontName /QNAAAA+NimbusSans-Regular
/Flags 4
/FontBBox [-210 -299 1032 1075 ]
/ItalicAngle 0
/Ascent 729
/Descent -271
/CapHeight 729
/StemV 50
/FontFile2 14 0 R
>>
endobj
14 0 obj
<<
/Length1 5720
/Length 17 0 R
/Filter /FlateDecode
>>
stream
x8y\SW;[      `1BLb+Y.MbQ6H)"`}SKm-nZ[e>S\~N_;m}
99w!cl !M
=3[WMbu]J!8gh]8 2=?U<YBq\P]XKfE85ES!p$H       $,8bF#-7-AbO?7&O
g       \DjB?;-hf<sFk<`Q
[TY0
~~{bF_NKeR4'"YBc
      HY77GZmVl6d2:n S0 Cz|G:@FU[RS;G7.23>T,:y0I%ar4
mL.
~Z`AWH7GW}
 ~&`f_c+-=ZtoD, /{'N  %9 onzP@QWxC8X4
o5q
~4$=rk8d4<  $2ii9,u-Z<LR/?p>;Lqo/KDQ    tBB5lL[0J2L6r?3s
s@Rl\wYPznZ8<1#9E$sc sv])Ce9c<.da~eG4Y4~
M$Lx({P-6u^>u.,!v-[Z`N\%_}2CjGLCyBZf`:A?D?a%^>Bu;[oIS"XR0~^5|}J0 9r
%u2g  f^MrG/S*:-QcC_S\}@H0ns cg>jkD/z]-9{w;GxN'Nw_j|Dâ.wП0 1-IRAPzJYcJ}P
k p?8:h l3:4!1?%4&vO$PS~S+g8~jE>ic+p5y5SPb]={'
}Fw08%}
U@:W0',jXmp\.H aY Z32g ryUi'
%yjng?MGhb=X b@b#n>!sg ijnl|{#.Z:^`wu4g6jr1UY$xlZ<Xr%O'+'?a~|s_=j{S.e#T
V16S_^K!{OclyxS!i)<d
cM"/(d\
-(C5@
UQhwLeeVfsKN6Y4X`94vIPk\c?nxqQR_OGrb
Ss|E{R|{$$EXm;Oap%j?3=<D`f`"d ILV`R]I
`t aAYw>wk/ZX&7Yr6w_Mv b%a31u$z@;b e{6r:YYI]jeOQVWU<wTk@X@5si 9s
V]*$2?7|OF"<<3 NmL(32 69[h'O&vq.jj\bZyi]I::>99j?5jxvx@)=]u@lQTMAa(grz@ZNN>
2pjd/,5,bPt   U0Xt6,9JZC'Qq AzG:ZV\Q|G$%=Fevl,)\]      ey<lcL/<2R
y@@CgNCv[9p^?[w ~>A/S aQI9M2_
2
r meXi2a2'e<N#'1|+0fuCZpC^WL\}, WnhNCd^`bY*iH
7I5Z:k \#{2qc+z2,ms "@sp$aJtZAzT@/a]:qX9a>|N/IrL6dy'@G5x$)Z
[RT|Uz88kb'Z68n"3MPgs([$0y?J9ZuqH1*8o hRJL]UcZUG!gsa$;o)eK~ZWlKJ-w<WW:,
b<RnI,U.{>~lw@bjt.mEELAa%TrdV0 _|kg*+*NwKWZ: .+~#5uys[2`z_=
S.ox0[QM \b57 beS L!z7J*Y<
HRi/p6I?*GuO2OEG6vM45qA@:X|3?$:|/!IC_>$gZLp =ue(fI4I71+YUS`a7[#4'C6SE7
#:kvb]Q\kW"@~2QPkWAS;rnCcezWG8iEQM~+Vm;T<Z?Y6Xbf ~bDtFG`UTmDY
mvyw=z4<8#b{s1PM:5';DAy+||!|h7u      I!!I    B AiOn n@d^WM#FO~}`nE~aW
jKR/\d6j=%5ou4#48S<e"GIe3Yh[RbS7[zazlXx]g/C-X
SRZ4!qiSj;F?z<_q.Jhmzm[ B+tLyMke;MPUEIW:V8>ge ms|0_](=fPvM;dGâp
$3iF8k7q..8$3By i@8;kzYn!let<B-K"y2C~,      [:(SNt+}#\"W x9*tBF8,U6Vv
`)|<W3H/_mqy}@za,sxoH<,VU5^XF|Z??
GUd@xaJ8/+L< l/,p~|0}]_Y^R93bi;*k:V;\KU
5
UUeFl>otn#fkl$%'LEjnJRN*J|}&|$
~
t-$j %T#U&I1j9UK9g#_Fy9Ky8tI|FE94<
endstream
endobj
17 0 obj
4234
endobj
15 0 obj
<< /Type /Font
/Subtype /CIDFontType2
/BaseFont /NimbusSans-Regular
/CIDSystemInfo << /Registry (Adobe) /Ordering (Identity) /Supplement 0 >>
/FontDescriptor 13 0 R
/CIDToGIDMap /Identity
/W [0 [276 276 716 662 772 716 662 552 276 552 662 220 220 606 552 276 662 552 552 552
552 276 552 552 496 552 276 552 716 276 662 496 826 552 330 552 552 772 716 662 552 27
6 496 330 882 330 ]
]
>>
endobj
16 0 obj
<< /Length 679 >>
stream
/CIDInit /ProcSet findresource begin
12 dict begin
begincmap
/CIDSystemInfo << /Registry (Adobe) /Ordering (UCS) /Supplement 0 >> def
/CMapName /Adobe-Identity-UCS def
/CMapType 2 def
1 begincodespacerange
<0000> <FFFF>
endcodespacerange
2 beginbfrange
<0000> <0000> <0000>
<0001> <002D> [<0049> <004E> <0056> <004F> <0043> <0045> <0023> <0020> <0032> <0042> <
0069> <006C> <0054> <006F> <003A> <0053> <0068> <0070> <0065> <0034> <002C> <0030> <00
33> <0073> <0061> <0066> <0031> <0044> <0074> <0050> <0079> <006D> <006E> <0072> <0075
> <0062> <0051> <0052> <0041> <0024> <002E> <0078> <0028> <0025> <0029> ]
endbfrange
endcmap
CMapName currentdict /CMap defineresource pop
end
end

endstream
endobj
6 0 obj
<< /Type /Font
/Subtype /Type0
/BaseFont /NimbusSans-Regular
/Encoding /Identity-H
/DescendantFonts [15 0 R]
/ToUnicode 16 0 R>>
endobj
18 0 obj
<< /Type /FontDescriptor
/FontName /QSAAAA+NimbusSans-Bold
/Flags 4
/FontBBox [-188 -307 1069 1070 ]
/ItalicAngle 0
/Ascent 729
/Descent -271
/CapHeight 729
/StemV 69
/FontFile2 19 0 R
>>
endobj
19 0 obj
<<
/Length1 2668
/Length 22 0 R
/Filter /FlateDecode
>>
stream
xV{P?B o%V`y.e!a$A44cN5<jkhkmGGcOk'*=w=sw@APUg=-j*  Pv FKC2 qz
$-3*o02;'|>:(
5v5~'0(JQ46/(R{qf`x /7:Ld+g ivN9kJ3ZAH_b1-C5c
 §Ba-H+f.U%FQUf$IQDgg

v8:'_JL,pm/1K+g{pFft4\Uq>Z?p#Y` \f!2(dTj@M_}jK%4K.s  w",Tai&K`@^%:tg/yk]OM
4y(-4ixM:5}x>9S:NMQp/|&df E<-wzek&U(
vat9ELg//X2 k#b{*_T'V h+h0$ub.mQk\jo.iUz<A+<K
",`ZV; VJr6Enpo}#wMW<l`akKayY0Nn\Zti7]'h:OeV|52HQPe(%9l]*O       rD#Ev
g+"NKojVqeuE^De\bX[/~MwW|_cjMg=*]in(5t#E ]@q* /$C;ZYL-,(}L 3G%IT)Yh*Q5q)6
9tctnf|lN2Z9HX~7
5urs7;--cR>7%,47 4Q&P2Q-~Ca}3A>8~ZvuV0w(%+XgQ<*^6~X><$Q/j68tC#WyhK
| YL_V_\f6F7[(==7)o\9&:Kx>^.`ELK)nw#?mY*]d&,$'Z]WY_[+G_c[y      )e|4+E
&}T]5f*WbUc4+f_b=|:WfSE1r,"wj'xyEb-|u        ),m7.ty}L?\y
~k=L4YVFFAP=^Ve$jeZiH|#(oc4?L|!)
{QK$~3WBTX;ka=/.lp\z@"@gcK<F  U]K1]*F3`|V@>&hVbiVBh-p^-0b3O{`&= wVvZ"@
w4l#ErQ!S3pamHW`]>B(0MP4:z
qJ3'wgIBDuJ       >6@5HR        B[f}UC#ZsOU8
h]^_Q
endstream
endobj
22 0 obj
1992
endobj
20 0 obj
<< /Type /Font
/Subtype /CIDFontType2
/BaseFont /NimbusSans-Bold
/CIDSystemInfo << /Registry (Adobe) /Ordering (Identity) /Supplement 0 >>
/FontDescriptor 18 0 R
/CIDToGIDMap /Identity
/W [0 [276 330 606 552 552 552 552 276 716 276 606 552 552 276 716 606 330 ]
]
>>
endobj
21 0 obj
<< /Length 476 >>
stream
/CIDInit /ProcSet findresource begin
12 dict begin
begincmap
/CIDSystemInfo << /Registry (Adobe) /Ordering (UCS) /Supplement 0 >> def
/CMapName /Adobe-Identity-UCS def
/CMapType 2 def
1 begincodespacerange
<0000> <FFFF>
endcodespacerange
2 beginbfrange
<0000> <0000> <0000>
<0001> <0010> [<0066> <0064> <0061> <0073> <0024> <0030> <002E> <0042> <006C> <006E> <
0063> <0065> <0020> <0044> <0075> <003A> ]
endbfrange
endcmap
CMapName currentdict /CMap defineresource pop
end
end

endstream
endobj
7 0 obj
<< /Type /Font
/Subtype /Type0
/BaseFont /NimbusSans-Bold
/Encoding /Identity-H
/DescendantFonts [20 0 R]
/ToUnicode 21 0 R>>
endobj
2 0 obj
<<
/Type /Pages
/Kids
[
5 0 R
]
/Count 1
/ProcSet [/PDF /Text /ImageB /ImageC]
>>
endobj
xref
0 23
0000000000 65535 f
0000000009 00000 n
0000011107 00000 n
0000000177 00000 n
0000000272 00000 n
0000000358 00000 n
0000007697 00000 n
0000010966 00000 n
0000000309 00000 n
0000000674 00000 n
0000001984 00000 n
0000000478 00000 n
0000000654 00000 n
0000002005 00000 n
0000002218 00000 n
0000006564 00000 n
0000006966 00000 n
0000006543 00000 n
0000007841 00000 n
0000008051 00000 n
0000010155 00000 n
0000010438 00000 n
0000010134 00000 n
trailer
<<
/Size 23
/Info 1 0 R
/Root 8 0 R
>>
startxref
11205
%%EOF
```


In [5]:
```
import cv2
import numpy as np

# Load YOLO model
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")

# Load class names
with open("coco.names", "r") as f:
    classes = f.read().strip().split("\n")

# Load image
image = cv2.imread("Original_01_page-0001.jpg")
blob = cv2.dnn.blobFromImage(image, 1/255.0, (416, 416), swapRB=True, crop=False)

# Set the input
net.setInput(blob)

# Get the detections
layer_names = net.getUnconnectedOutLayersNames()
detections = net.forward(layer_names)

# Loop over the detections
for detection in detections:
    for obj in detection:
        scores = obj[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
#
        if confidence > 0.5 :  # Assuming "logo" is the class name for the logo
            label = classes[class_id]

            # Get the coordinates of the bounding box
            center_x = int(obj[0] * image.shape[1])
            center_y = int(obj[1] * image.shape[0])
            width = int(obj[2] * image.shape[1])
            height = int(obj[3] * image.shape[0])

            # Calculate the coordinates of the top-left and bottom-right corners
            x = int(center_x - width / 2)
            y = int(center_y - height / 2)
            x2 = x + width
            y2 = y + height

            # Crop the region containing the detected logo
            logo_region = image[y:y2, x:x2]

            # Check if the region contains non-black/non-white pixels (assuming RGB im
            is_logo = np.any((logo_region > [0, 0, 0]) & (logo_region < [255, 255, 255

            if is_logo:
                print(f"Confidence: {confidence}")
                print(f"Coordinates: x={x}, y={y}")
```

```
Confidence: 0.6662297248840332
Coordinates: x=33, y=71
```

Detecting Fraudulent Invoices Based on Text and Image Similarity

The provided code aims to determine whether two invoice documents (in either image or PDF format) are potentially fraudulent or not based on their similarity scores. It combines text and image analysis to make this determination.

Text and PDF Processing

- The code uses the PyMuPDF library (imported as `fitz`) to extract text from PDF files. The `extract_text_from_pdf` function reads a PDF file and concatenates the text content from all its pages.
- The `extract_images_from_pdf` function extracts images from a PDF file using PyMuPDF. It collects image data from all the pages and returns them as a list of image data.

Text Similarity Calculation

- The `compute_similarity` function calculates the TF-IDF (Term Frequency-Inverse Document Frequency) similarity score between two text documents. It first converts the text into a numerical vector representation using TF-IDF and then computes the cosine similarity between these vectors.

Image Similarity Calculation

- The `compare_images` function compares two images using the structural similarity index (SSIM). It first decodes the image data using OpenCV, and if the images have the same shape, it calculates the similarity score using the template matching technique.

Detecting Fraud

- The `detect_fraud` function uses the previously defined functions to analyze the text and image content of two PDF files. It calculates the text similarity and image similarity scores. However, it seems that the image similarity calculation is currently commented out, and the image similarity is set to 0 in the provided code.
- The `check_similarity_and_fraud` function calls `detect_fraud` to obtain the text and image similarity scores for two PDF files and then checks if both scores are above a threshold of 0.7. If both scores are sufficiently high, it concludes that the documents are similar and potentially fraudulent.

Example Usage

- The code provides an example usage where two PDF file paths are specified (`pdf1_path` and `pdf2_path`). It then calls `check_similarity_and_fraud` to determine whether these two documents are similar and potentially fraudulent based on the text and image analysis.
- Finally, the code prints the result, which includes the similarity scores and a fraudulent indicator.

Please note that in the provided code, the image similarity calculation is commented out, and the image similarity is always set to 0. You may need to uncomment and adapt the image similarity calculation to suit your specific needs. Additionally, you should adjust the file paths for the PDF documents you want to analyze.

In [6]:
```
import fitz  # PyMuPDF for PDF text extraction
import cv2   # OpenCV for image processing
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import DBSCAN
import difflib

# Function to extract text from a PDF file
def extract_text_from_pdf(pdf_file):
    doc = fitz.open(pdf_file)
    text = ""
    for page_num in range(len(doc)):
        page = doc[page_num]
        text += page.get_text()
    return text

# Function to extract images from a PDF file
def extract_images_from_pdf(pdf_file):
    doc = fitz.open(pdf_file)
    images = []
    for page_num in range(len(doc)):
        page = doc[page_num]
        xref_list = page.get_images(full=True)
        for xref in xref_list:
            base_image = doc.extract_image(xref[0])
            image_data = base_image["image"]
            images.append(image_data)
    return images

# Function to compute TF-IDF similarity between two texts
def compute_similarity(text1, text2):
    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform([text1, text2])
    similarity_matrix = cosine_similarity(tfidf_matrix, tfidf_matrix)
    return similarity_matrix[0][1]

# Function to compare images using structural similarity index (SSIM)
def compare_images(image1, image2):
    image1 = cv2.imdecode(np.frombuffer(image1, np.uint8), -1)
    image2 = cv2.imdecode(np.frombuffer(image2, np.uint8), -1)
    if image1.shape != image2.shape:
        return 0.0
    return cv2.matchTemplate(image1, image2, cv2.TM_CCOEFF_NORMED)[0][0]

# Function to detect fraud based on text and image similarities
def detect_fraud(pdf_file1, pdf_file2):
    text1 = extract_text_from_pdf(pdf_file1)
    text2 = extract_text_from_pdf(pdf_file2)

    images1 = extract_images_from_pdf(pdf1_path)
    images2 = extract_images_from_pdf(pdf2_path)

    text_similarity = compute_similarity(text1, text2)

    # image_similarity_scores = []
    # for image1 in images1:
    #     for image2 in images2:
    #         similarity_score = compare_images(image1, image2)
    #         image_similarity_scores.append(similarity_score)

    # image_similarity = max(image_similarity_scores)
    image_similarity = 0
    return text_similarity, image_similarity

# Function to check if documents are similar and potentially fraudulent
def check_similarity_and_fraud(pdf_file1, pdf_file2):
    text_similarity, image_similarity = detect_fraud(pdf_file1, pdf_file2)

    if text_similarity > 0.7 and image_similarity > 0.7:
        return "Similarity Score: {:.2f}, Fraudulent: No".format(text_similarity), Tru
    else:
        return "Similarity Score: {:.2f}, Fraudulent: Yes".format(text_similarity), Fa

# Example usage
pdf1_path = "/home/neel/Invoice1.pdf"  # Replace with the path to the first PDF
pdf2_path = "/home/neel/Invoice2.pdf"  # Replace with the path to the second PDF

result, is_fraudulent = check_similarity_and_fraud(pdf1_path, pdf2_path)
print(result)
```

```
Similarity Score: 0.59, Fraudulent: Yes
```

Clustering PDF Documents Based on Image Category

In this Jupyter Notebook code, we are performing clustering on a collection of PDF documents based on the content of the documents, particularly focusing on the images within them. The goal is to group similar PDF documents together using K-Means clustering. Below is a step-by-step explanation of the code:

Importing Libraries

We begin by importing the necessary Python libraries:

```
import fitz
import cv2
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

    fitz: A library for working with PDF documents.
    cv2: OpenCV for image processing.
    numpy: For numerical operations.
    TfidfVectorizer and cosine_similarity: Used for computing TF-IDF
similarity between texts.
    KMeans: A clustering algorithm for grouping documents.
    StandardScaler and make_pipeline: For data preprocessing.
    matplotlib.pyplot: For creating data visualizations.
    PCA: Principal Component Analysis for dimensionality reduction.
```

Functions for Text and Image Extraction

extract_text_from_pdf(pdf_file)

This function extracts text content from a PDF file using the PyMuPDF library (fitz). extract_images_from_pdf(pdf_file)

This function extracts images from a PDF file. It iterates through the pages of the PDF and collects image data. Functions for Similarity Measurement compute_similarity(text1, text2)

This function computes the TF-IDF similarity between two text documents. It uses a TF-IDF vectorizer and cosine similarity. compare_images(image1, image2)

This function compares two images using the structural similarity index (SSIM). It decodes images and calculates a similarity score. Loading PDF Files

We define a base path for the PDF files and generate file paths for a collection of PDF documents. In this example, it's generating file paths for 50 PDFs. Computing Text and Image Similarity

We create two similarity matrices: text_similarity_matrix and image_similarity_matrix. These matrices store similarity scores between PDF documents based on their text content and images, respectively.

```
For each PDF document, we extract its text and images.
We calculate the text similarity between all pairs of documents.
We calculate image similarity scores for all pairs of documents using
the compare_images function. If a document has no images, its image
similarity score is set to 0.
```

Combining Similarity Scores

We combine the text and image similarity scores into a single combined_similarity_matrix. This matrix contains both text and image similarity information. Normalizing the Combined Similarity Matrix

We standardize (normalize) the combined similarity matrix to make it suitable for K-Means clustering. Applying K-Means Clustering

We apply K-Means clustering to the normalized similarity matrix. The number of clusters (num_clusters) is predefined as 5 in this example. Adjust this number according to your specific dataset and requirements. Printing Clusters

We print the PDF files that belong to each cluster, helping us understand how the documents are grouped. Plotting Results

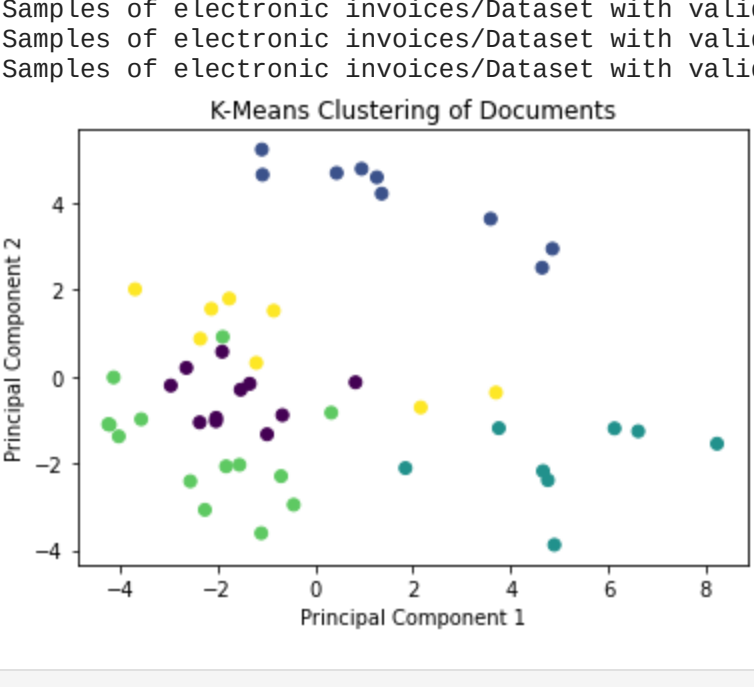
Finally, we use Principal Component Analysis (PCA) to reduce the dimensions of the data for visualization purposes. We then create a scatter plot to visualize the clustering results in a 2D space, coloring the points based on their cluster assignments.

The code provides a way to analyze and group PDF documents based on both text and image content, which can be useful in various document categorization and clustering tasks.

In [1]:

```python
import fitz
import cv2
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
# Function to extract text from a PDF file
def extract_text_from_pdf(pdf_file):
    doc = fitz.open(pdf_file)
    text = ""
    for page_num in range(len(doc)):
        page = doc[page_num]
        text += page.get_text()
    return text

# Function to extract images from a PDF file
def extract_images_from_pdf(pdf_file):
    doc = fitz.open(pdf_file)
    images = []
    for page_num in range(len(doc)):
        page = doc[page_num]
        xref_list = page.get_images(full=True)
        for xref in xref_list:
            base_image = doc.extract_image(xref[0])
            image_data = base_image["image"]
            images.append(image_data)
    return images

# Function to compute TF-IDF similarity between two texts
def compute_similarity(text1, text2):
    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform([text1, text2])
    similarity_matrix = cosine_similarity(tfidf_matrix, tfidf_matrix)
    return similarity_matrix[0][1]

# Function to compare images using structural similarity index (SSIM)
def compare_images(image1, image2):
    image1 = cv2.imdecode(np.frombuffer(image1, np.uint8), -1)
    image2 = cv2.imdecode(np.frombuffer(image2, np.uint8), -1)
    if image1.shape != image2.shape:
        return 0.0
    return cv2.matchTemplate(image1, image2, cv2.TM_CCOEFF_NORMED)[0][0]

# Load your dataset of PDF files here
# The base path for the PDF files
base_path = 'Samples of electronic invoices/Dataset with valid information/invoice_'

# List to store the file paths for 500 images
pdf_files = []

# Loop to generate file paths for 500 images
for i in range(50):
    pdf_file_path = f'{base_path}{i}.pdf'
    pdf_files.append(pdf_file_path)

# Create feature vectors for text and image similarity
text_similarity_matrix = np.zeros((len(pdf_files), len(pdf_files)))
image_similarity_matrix = np.zeros((len(pdf_files), len(pdf_files)))

for i, pdf_file1 in enumerate(pdf_files):
    text1 = extract_text_from_pdf(pdf_file1)
    images1 = extract_images_from_pdf(pdf_file1)

    for j, pdf_file2 in enumerate(pdf_files):
        text2 = extract_text_from_pdf(pdf_file2)
        images2 = extract_images_from_pdf(pdf_file2)

        text_similarity_matrix[i][j] = compute_similarity(text1, text2)
        if(len(images1) == 0 | len(images2) == 0):
            image_similarity_matrix[i][j] = 0;

        else:
            image_similarity_scores = []
            for image1 in images1:
                for image2 in images2:
                    similarity_score = compare_images(image1, image2)
                    image_similarity_scores.append(similarity_score)
            image_similarity_matrix[i][j] = max(image_similarity_scores)


# Combine text and image similarity scores
combined_similarity_matrix = text_similarity_matrix + image_similarity_matrix

# Normalize the combined similarity matrix
scaler = StandardScaler()
normalized_combined_similarity_matrix = scaler.fit_transform(combined_similarity_matri

# Apply K-Means clustering
num_clusters = 5  # You can adjust this based on your dataset and requirements
kmeans = KMeans(n_clusters=num_clusters)
clusters = kmeans.fit_predict(normalized_combined_similarity_matrix)

# Print clusters
for i in range(num_clusters):
    print(f"Cluster {i + 1}:")
    for j, pdf_file in enumerate(pdf_files):
        if clusters[j] == i:
            print(pdf_file)

# Plot the results (for visualization purposes, using the first two dimensions)
pca = PCA(n_components=3)
reduced_features = pca.fit_transform(normalized_combined_similarity_matrix)

plt.scatter(reduced_features[:, 0], reduced_features[:, 1], c=clusters, cmap='viridis
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('K-Means Clustering of Documents')
plt.show()
```

```
Cluster 1:
Samples of electronic invoices/Dataset with valid information/invoice_1.pdf
Samples of electronic invoices/Dataset with valid information/invoice_6.pdf
Samples of electronic invoices/Dataset with valid information/invoice_7.pdf
Samples of electronic invoices/Dataset with valid information/invoice_10.pdf
Samples of electronic invoices/Dataset with valid information/invoice_13.pdf
Samples of electronic invoices/Dataset with valid information/invoice_28.pdf
Samples of electronic invoices/Dataset with valid information/invoice_36.pdf
Samples of electronic invoices/Dataset with valid information/invoice_41.pdf
Samples of electronic invoices/Dataset with valid information/invoice_45.pdf
Samples of electronic invoices/Dataset with valid information/invoice_48.pdf
Samples of electronic invoices/Dataset with valid information/invoice_49.pdf
Cluster 2:
Samples of electronic invoices/Dataset with valid information/invoice_3.pdf
Samples of electronic invoices/Dataset with valid information/invoice_5.pdf
Samples of electronic invoices/Dataset with valid information/invoice_16.pdf
Samples of electronic invoices/Dataset with valid information/invoice_20.pdf
Samples of electronic invoices/Dataset with valid information/invoice_23.pdf
Samples of electronic invoices/Dataset with valid information/invoice_27.pdf
Samples of electronic invoices/Dataset with valid information/invoice_31.pdf
Samples of electronic invoices/Dataset with valid information/invoice_43.pdf
Samples of electronic invoices/Dataset with valid information/invoice_47.pdf
Cluster 3:
Samples of electronic invoices/Dataset with valid information/invoice_0.pdf
Samples of electronic invoices/Dataset with valid information/invoice_8.pdf
Samples of electronic invoices/Dataset with valid information/invoice_11.pdf
Samples of electronic invoices/Dataset with valid information/invoice_14.pdf
Samples of electronic invoices/Dataset with valid information/invoice_15.pdf
Samples of electronic invoices/Dataset with valid information/invoice_19.pdf
Samples of electronic invoices/Dataset with valid information/invoice_25.pdf
Samples of electronic invoices/Dataset with valid information/invoice_38.pdf
Cluster 4:
Samples of electronic invoices/Dataset with valid information/invoice_2.pdf
Samples of electronic invoices/Dataset with valid information/invoice_9.pdf
Samples of electronic invoices/Dataset with valid information/invoice_12.pdf
Samples of electronic invoices/Dataset with valid information/invoice_17.pdf
Samples of electronic invoices/Dataset with valid information/invoice_18.pdf
Samples of electronic invoices/Dataset with valid information/invoice_21.pdf
Samples of electronic invoices/Dataset with valid information/invoice_22.pdf
Samples of electronic invoices/Dataset with valid information/invoice_24.pdf
Samples of electronic invoices/Dataset with valid information/invoice_32.pdf
Samples of electronic invoices/Dataset with valid information/invoice_35.pdf
Samples of electronic invoices/Dataset with valid information/invoice_39.pdf
Samples of electronic invoices/Dataset with valid information/invoice_40.pdf
Samples of electronic invoices/Dataset with valid information/invoice_42.pdf
Samples of electronic invoices/Dataset with valid information/invoice_44.pdf
Cluster 5:
Samples of electronic invoices/Dataset with valid information/invoice_4.pdf
Samples of electronic invoices/Dataset with valid information/invoice_26.pdf
Samples of electronic invoices/Dataset with valid information/invoice_29.pdf
Samples of electronic invoices/Dataset with valid information/invoice_30.pdf
Samples of electronic invoices/Dataset with valid information/invoice_33.pdf
Samples of electronic invoices/Dataset with valid information/invoice_34.pdf
Samples of electronic invoices/Dataset with valid information/invoice_37.pdf
Samples of electronic invoices/Dataset with valid information/invoice_46.pdf
```



In [ ]: