

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



***NHÓM: ..7..***

***ĐỀ TÀI: Table Cell Structure Detection***

**BÁO CÁO BÀI TẬP LỚN MÔN XỬ LÝ ẢNH**

**Ngành: Công nghệ thông tin**

**HÀ NỘI - 2021**

## **I. Giới thiệu**

### *1. Giảng viên*

- Nguyễn Thị Ngọc Diệp

### *2. Thành viên nhóm 7*

- Hoàng Đức Giang - 19020270
- Nguyễn Khánh Quân - 19020400
- Ngô Trung Kiên - 17020840
- Nguyễn Minh Quang - 18021044
- Thái Trần Hồng Quân - 17020989

## **II. Tổng quan bài toán**

### *1. Xử lý ảnh*

- Đọc ảnh png và kết quả xml
- Xử lý ảnh với các hàm trong thư viện
- Lấy thông số ảnh: chiều cao chữ, khoảng cách giữa các dòng, tính khoảng cách các chữ
- Xử lý ảnh theo thông số mới

### *2. Tìm kiếm khoanh vùng chữ trong cell*

- Khoanh cùng các chữ
- Kiểm tra xem có bị chồng chéo và có cùng một cell không sau đó hợp tất cả các khối cho là cùng một cell lại

### *3. Hiển thị kết quả và tính IOU*

- Hiển thị kết quả của nhóm lên hình gốc với màu đỏ
- Hiển thị đáp án với màu xanh
- Tính kết quả IOU cho từng cell và trung bình cho tất cả

### III. Chi tiết thuật toán

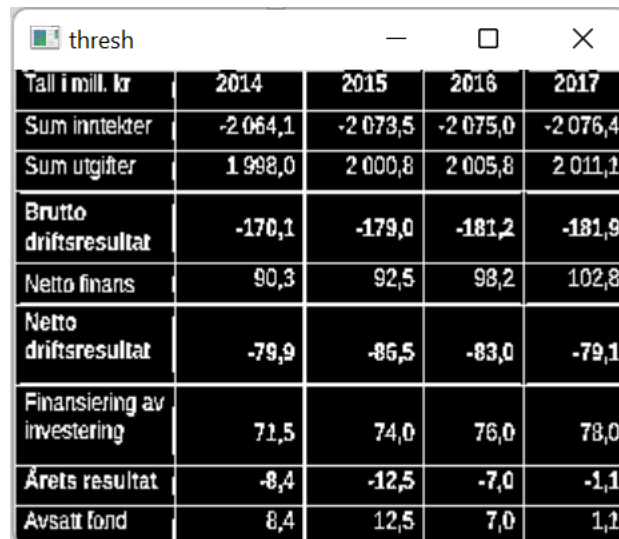
```
if "BTL-DIP" == "BTL-DIP":  
    imgname = "32"  
    img = cv2.imread("data/public/" + imgname + ".png")  
    pathXml = 'data/public/' + imgname + '.xml'
```

- Đọc ảnh png và đọc file xml

```
thresh = processImage(img, (1,1))
```

```
def processImage(img, morph_size):  
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
    thresh = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 11, 10)  
    kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, morph_size)  
    thresh = cv2.dilate(thresh, kernel, iterations=1)  
    return thresh
```

- Xử lý ảnh đầu vào



Tall i mill. kr	2014	2015	2016	2017
Sum inntekter	-2 064,1	-2 073,5	-2 075,0	-2 076,4
Sum utgifter	1 998,0	2 000,8	2 005,8	2 011,1
Brutto driftsresultat	-170,1	-179,0	-181,2	-181,9
Netto finans	90,3	92,5	93,2	102,8
Netto driftsresultat	-79,9	-86,5	-83,0	-79,1
Finansiering av investering	71,5	74,0	76,0	78,0
Årets resultat	-8,4	-12,5	-7,0	-1,1
Avsatt fond	8,4	12,5	7,0	1,1

```
correction = getCorrection(thresh)
```

- Tính thông số ảnh, hàm này sẽ trả về khoảng cách giữa các dòng và chiều cao của chữ

```
def getCorrection(thresh,with_ = 3,height_ = 1):
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (with_, height_))
    thresh = cv2.dilate(thresh, kernel, iterations=1)
    boxes = findContours(thresh)
    line_spacing = lineSpacing(boxes)
    averageHeight = get_htb(boxes)
    return [averageHeight,line_spacing]
```

- Tính toán khoảng cách giữa các dòng: sắp xếp lại mảng theo chiều tăng dần của trục y lúc này sẽ tính khoảng cách liên tiếp giữa các y và trả về giá trị trung bình tổng các y lớn hơn 10

```
def lineSpacing(boxes):
    boxes.sort(key=lambda x: x[1], reverse=False)
    MaxLine = 0
    a = 0
    for i in range(len(boxes)-1):
        print((boxes[i + 1][1] - boxes[i][1]))
        if MaxLine < (boxes[i + 1][1] - boxes[i][1]) and (boxes[i + 1][1] - boxes[i][1]) > 10:
            MaxLine = MaxLine + (boxes[i + 1][1] - boxes[i][1])
            a += 1
    MaxLine /= a
    return MaxLine
```

- tính chiều cao của các khối box tìm được và trả về giá trị trung bình

```
def get_htb(boxes):
    sum_height = 0
    lenboxes = len(boxes)
    for box in boxes:
        sum_height = sum_height + box[3]
    return sum_height/lenboxes
```

- Gán thông số mới tìm được và xử lý lại ảnh với hàm *processImage*

```
correction = getCorrection(thresh)
height_text = correction[0]
lineToLine = correction[1]
letter_spacing = round(height_text*0.75)
line_spacing = lineToLine - round(height_text*1.3)
thresh2 = processImage(img,(round(height_text/2.5),1))
```

- Dọn dẹp các box tìm được với hàm *clearBoxes*: hợp những khối gần nhau một khoản nằm ngang nhau là  $r$  và nằm trên dưới nhau 1 khoảng là  $d$

```
def clearBoxes(boxes,r,d):
    finished = False
    while not finished:
        finished = True

        index = len(boxes)-1
        while index >=0:
            (x, y, w, h) = boxes[index]
            x -= r
            y -= d
            w += 2*r
            h += 2*d
            overlaps = getAllOverlaps(boxes,(x, y, w, h),index)
            con = []
            for box in overlaps:
                (x1,y1,w1,h1) = box
                tl = [x1,y1]
                br = [x1 + w1, y1 + h1]
                con.append(tl)
                con.append(br)
            con = np.array(con)
            (x2,y2,w2,h2) = cv2.boundingRect(con)
            for ind in overlaps:
                if ind in boxes:
                    boxes.remove(ind)
            boxes.append((x2,y2,w2-1,h2-1))
            if len(overlaps) > 1:
                finished = False;
                break
            index -= 1

    return boxes
```

- Hàm clearBoxes sẽ lấy từng khối, nở khối ra theo  $r$  và  $d$ , phát hiện những khối overlaps bằng hàm *getAllOverlaps* sau đó sẽ tạo ra khối bao quanh các khối overlaps và chèn vào boxes rồi xóa những khối overlaps đi cuối cùng sẽ tra về boxes được thay đổi
  - o Lấy những khối overlaps với hàm *getAllOverlaps*: sử dụng hàm *testDNA* để kiểm tra overlap

```
def getAllOverlaps(bboxes, bounds, index):
    overlaps = []
    for i in range(len(bboxes)):
        if testDNA(bounds, bboxes[i]):
            overlaps.append(bboxes[i])
    return overlaps
```

- Hàm *testDNA*: kiểm tra theo chiều x và y nếu cả hai chiều được chồng chéo nên nhau thì sẽ trả về là true nếu không trả về false

```
def testDNA(box1, box2):
    (x, y, w, h) = box1
    (x1, y1, w1, h1) = box2
    if (x+w >= x1) and (x1+w1 >= x) and (y+h >= y1) and (y1+h1 >= y):
        return True
    return False
```

- Tạo ra khối mới với những khối overlaps tìm được là :  
(x2,y2,w2,h2)

```
overlaps = getAllOverlaps(bboxes, (x, y, w, h), index)
con = []
for box in overlaps:
    (x1, y1, w1, h1) = box
    tl = [x1, y1]
    br = [x1 + w1, y1 + h1]
    con.append(tl)
    con.append(br)
con = np.array(con)
(x2, y2, w2, h2) = cv2.boundingRect(con)
```

- Xóa những chồng chéo và thêm khối mới

```
for ind in overlaps:
    if ind in bboxes:
        bboxes.remove(ind)
bboxes.append((x2, y2, w2-1, h2-1))
if len(overlaps) > 1:
    finished = False;
    break
```

- Kiểm tra nếu hơn một khối chồng chéo (tức chính nó) sẽ đổi finished thành false để tiếp tục kiểm tra khối tiếp theo và break.
- Sắp hiển thị kết quả tìm được trên ảnh gốc với màu đỏ

```
for box in results:
    (x, y, w, h) = box
    cv2.rectangle(img, (x, y), (x + w - 1, y + h - 1), (0, 0, 255), 1)

resultsXml = readXml(pathXml)
for x in resultsXml:
    cv2.rectangle(img, (x[0], x[1]), (x[2], x[3]), (36,255,12), 1)
results = [(box[0], box[1], box[2] + box[0], box[3] + box[1]) for box in results]
```

- Đọc file dữ liệu từ file xml và trả về kết quả là mảng vào *resultsXml*,

```
def readXml(pathXml):
    tree = ET.parse(pathXml)
    root = tree.getroot()
    a = []
    for child in root:
        b = []
        if child.tag == 'object':
            for con in child:
                if con.tag == 'bndbox':
                    for con2 in con:
                        b.append(con2.text)
            a.append(b)
    a = np.array(a, dtype=np.int32)
    return a
```

- Hiển thị kết quả từ file xml với màu xanh

```
resultsXml = readXml(pathXml)
for x in resultsXml:
    cv2.rectangle(img, (x[0], x[1]), (x[2], x[3]), (36,255,12), 1)
```

- Sắp xếp lại mảng để phù hợp với vị trí tìm được

```
results.sort(key=lambda x: x[0], reverse=False)
results.sort(key=lambda x: x[1], reverse=False)
rs = []
for r in resultsXml:
    (a,b,c,d) = r
    rs.append((a,b,c,d))
rs.sort(key=lambda x: x[0], reverse=False)
rs.sort(key=lambda x: x[1], reverse=False)
```

- Tính kết quả IOU với hàm *final*: hàm sẽ print ra độ chính xác tại từng cell và trả về độ chính xác trung bình

```
print("IOU: ",final(results,rs))
cv2.imshow("thresh2",thresh2)
cv2.imshow("thresh",thresh)
cv2.imshow("img",img)
cv2.waitKey()
```

- Cuối cùng hiển thị tất cả quá trình và kết quả

Tall i mill. kr	2014	2015	2016	2017
Sum inntekter	2 064,1	2 073,5	2 075,0	2 076,4
Sum utgifter	1 998,0	2 000,8	2 005,8	2 011,1
Brutto driftsresultat	-170,1	-179,0	-181,2	-181,9
Netto finans	90,3	92,5	98,2	102,8
Netto driftsresultat	-79,9	-86,5	-83,0	-79,1
Finansiering av investering	71,5	74,0	76,0	78,0
Årets resultat	-8,4	-12,5	-7,0	-1,1
Avsatt fond	8,4	12,5	7,0	1,1

Tall i mill. kr	2014	2015	2016	2017
Sum inntekter	-2 064,1	-2 073,5	-2 075,0	-2 076,4
Sum utgifter	1 998,0	2 000,8	2 005,8	2 011,1
Brutto driftsresultat	-170,1	-179,0	-181,2	-181,9
Netto finans	90,3	92,5	98,2	102,8
Netto driftsresultat	-79,9	-86,5	-83,0	-79,1
Finansiering av investering	71,5	74,0	76,0	78,0
Årets resultat	-8,4	-12,5	-7,0	-1,1
Avsatt fond	8,4	12,5	7,0	1,1



thresh2				
Tài sản, tr	2014	2015	2016	2017
Sum intider	-2064,1	-2073,5	-2075,0	-2076,4
Sum ntider	1998,0	2000,8	2005,8	2011,1
Đổi đổi	-176,1	-172,7	-169,2	-165,3
Neto finans	90,3	92,5	96,2	102,8
Neto đổi	-79,9	-80,5	-83,8	-79,1
Financing or investing	71,5	74,0	76,0	78,0
Assets resultat	-8,4	-12,5	-7,0	-1,1
Assets fund	8,4	12,5	7,0	1,1

```

Vị trí 0 có độ chính xác là : 0.72727272727273
Vị trí 1 có độ chính xác là : 0.7386363636363636
Vị trí 2 có độ chính xác là : 0.743801652892562
Vị trí 3 có độ chính xác là : 0.7386363636363636
Vị trí 4 có độ chính xác là : 0.8385744234800838
Vị trí 5 có độ chính xác là : 0.7655502392344498
Vị trí 6 có độ chính xác là : 0.8133971291866029
Vị trí 7 có độ chính xác là : 0.8
Vị trí 8 có độ chính xác là : 0.7655502392344498
Vị trí 9 có độ chính xác là : 0.413986013986014
Vị trí 10 có độ chính xác là : 0.7792207792207793
Vị trí 11 có độ chính xác là : 0.7851239669421488
Vị trí 12 có độ chính xác là : 0.7851239669421488
Vị trí 13 có độ chính xác là : 0.7851239669421488
Vị trí 14 có độ chính xác là : 0.9209100758396533
Vị trí 15 có độ chính xác là : 0.8
Vị trí 27 có độ chính xác là : 0.8484848484848485
Vị trí 28 có độ chính xác là : 0.8211143695014663
Vị trí 29 có độ chính xác là : 0.9283854166666666
Vị trí 30 có độ chính xác là : 0.8311688311688312
Vị trí 31 có độ chính xác là : 0.8311688311688312
Vị trí 32 có độ chính xác là : 0.8311688311688312
Vị trí 33 có độ chính xác là : 0.8080808080808081
Vị trí 34 có độ chính xác là : 0.8523391812865497
Vị trí 35 có độ chính xác là : 0.8391608391608392
Vị trí 36 có độ chính xác là : 0.8181818181818182
Vị trí 37 có độ chính xác là : 0.7954545454545454
Vị trí 38 có độ chính xác là : 0.8391608391608392
Vị trí 39 có độ chính xác là : 0.8478632478632478
Vị trí 40 có độ chính xác là : 0.7777777777777778
Vị trí 41 có độ chính xác là : 0.7822222222222223
Vị trí 42 có độ chính xác là : 0.7777777777777778
Vị trí 43 có độ chính xác là : 0.7822222222222223
Vị trí 44 có độ chính xác là : 0.8404040404040404
IOU: 0.8001714570535365

```