

Visão Computacional: reconhecimento de texto com OCR e OpenCV

cursos.alura.com.br/course/visao-computacional-reconhecimento-texto-ocr-opencv/task/112870

Nas aulas anteriores, nós aprendemos a detectar palavras, colocar *bounding boxes* e a adicionar textos que o OCR também estava identificando. Agora, vamos aprender a retirar/capturar textos específicos.

Vamos imaginar algumas situações em que buscamos um texto específico. Primeiro, você está lendo um documento e deseja encontrar os e-mails de uma ou várias pessoas. Segundo, você está lendo vários documentos e precisa encontrar uma expressão que é regular.

Considerando estes casos, nós usaremos a biblioteca **RE** ou **Regular Expressions**. Ela nos ajudará a encontrar as expressões regulares, como e-mails ou datas, enfim, elementos que aparecem com muita frequência. No site da RE, disponível neste link, nós encontramos algumas dessas expressões frequentes. Aliás, elas têm um formato muito semelhante, o que facilita a nossa busca. Vamos importar a biblioteca RE.

```
import re
```

Em seguida, importaremos uma imagem da aula 4, uma tabela de teste, "text-recognize > Aula4-tabela_teste.png". Basta copiar o caminho e substituir na mesma sintaxe de código que usamos em outras aulas.

```
img = cv2.imread('/content/text-recognize/Imagens/Aula4-tabela_teste.png')  
rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  
cv2_imshow(rgb)
```

Portanto, definimos `img` igual a `cv2.imread()`, passamos o caminho da imagem da aula 4, realizamos a conversão de valores de BGR para RGB e, por fim, solicitamos a visualização da imagem em RGB.

O resultado é a imagem de uma tabela de despesas bancárias.

DESPESAS BANCÁRIAS			
Tarifa Manutenção de Conta - Sicredi Norte SC - CESTA DE RELACIONAMENTO	02/2021	10/02/2021	Débito Aut.
Tarifa de Compensação de Boletos - Sicredi Norte SC - TARIFA COM R LIQUIDAÇÃO	02/2021	04/02/2021	Débito Aut.

Coluna 1, linha 2: tipo de despesa - Tarifa de Compensação de Boletos - Sicredi Norte SC - TARIFA COM LIQUIDAÇÃO; mês em que a compra foi feita - 02/2021; Data sem identificação - 04/02/2021; tipo de pagamento - Débito Aut.]

Alguns dados que a tabela apresenta, são: tarifa de manutenção de conta; o mês em que a compra foi feita; algumas datas sem identificação; o tipo de pagamento, débito automático. É muito provável, por exemplo, que essas datas sejam encontradas com frequência. Isso é uma expressão regular.

Já sabemos encontrar os textos nas imagens e podemos aplicar o OCR para que leia o texto e destaque a expressão regular, por exemplo, das datas. Além disso, queremos que ele adicione texto à parte das datas e não em toda a tabela. Começaremos fazendo o OSD com a imagem para que ele gere os resultados.

```
config_tesseract = "--tessdata-dir tessdata"
resultado = pytesseract.image_to_data(rgb, config=config_tesseract, lang="por",
output_type=Output.DICT)
resultado
```

O retorno é um dicionário bastante extenso. Você pode conferi-lo por completo [neste link](#). Temos um trecho do resultado a seguir:

```
1,
  1,
  1],
'text': ['',
'',
'',
'',
'',
'DESPESAS',
'BANCÁRIAS',
'',
'',
'',
'',
'Tarifa',
'Manutenção',
'de',
'Conta',
```

A leitura está bem difícil, mas, ele conseguiu retirar todos os valores: despesas bancárias, tarifa, manutenção de conta. Também aparece a parte das confianças: -1, 91, 92. Enfim, todos os valores estão corretos. Podemos seguir para a próxima etapa.

No site da expressão regular ou regex, existe um padrão de dados específicos, que eu copie, salvei e que nós usaremos no código. Então, escreveremos `padrao_data` igual ao código do regex que retiramos do site.

```
padrao_data = '^(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[012])/(19|20)\d\d$'
```

Esse é o padrão de data que usaremos para fazer a seguinte verificação: se esse código funcionar, isto é, for igual ao padrão de data que já está dentro do nosso texto, podemos prosseguir e adicionar texto. Agora que já colocamos o padrão de data, podemos pegar a nossa função anterior e tornar o padrão de data uma nova função.

```

img_copia = rgb.copy()
for i in range(0, len(resultado['text'])):
    confianca = int(resultado['conf'][i])
    if confianca > min_conf:
        texto = resultado['text'][i]

        if re.match(padrao_data, texto):
            x, y, img = caixa_texto(resultado, img_copia)
            img_copia = escreve_texto(texto, x, y, img_copia, fonte, 12)
        else:
            x, y, img_copia = caixa_texto(resultado, img_copia)

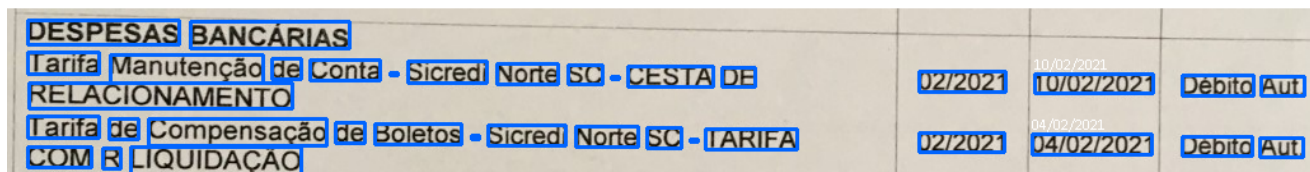
cv2_imshow(img_copia)

```

Começamos fazendo a cópia da nossa imagem, `img_copia`. Depois, fazemos o `for` de zero até o resultado, o número de textos. Em seguida, ele vai conferir a `confianca`. Se ela for igual ao valor mínimo, `confianca > min_conf`, que é 40, ele vai passar para a próxima linha: `texto = resultado['text'][i]`. Essa parte nós já havíamos feito.

A segunda parte é nova: se dentro da biblioteca que importamos, que é a `re`, o padrão de data for igual ao texto, faremos tanto a caixa de texto quanto a escrita do texto; se o padrão não for igual, faremos apenas a caixa de texto e não escreveremos nada. Por fim, mostraremos a imagem. Ele está fazendo um if/else dentro do nosso próprio `if`.

O `texto` ficou antes do `caixa_texto` para que ele pudesse participar do nosso `if` que está mais abaixo. Vamos rodar e conferir o resultado.



02/2021	10/02/2021	Débito Aut
02/2021	04/02/2021	Débito Aut

As *bounding boxes* aparecem em todo o texto. Inclusive, assim como esperamos, ele está considerando os sinais de "igual (=)" como um texto. A data aparece escrita em letras brancas, assim como na imagem. Portanto, aconteceu a combinação entre o padrão de data e o texto. Se fosse um caso com e-mails, poderíamos colocar um padrão "e-mail" e procurar no regex? A resposta é sim.

A junção do regex ao OCR nos permite realizar uma série de aplicações e, por isso, merece a nossa atenção. Podemos usá-la, como no caso desta aula, para destacar informações e deixá-las um pouco mais visíveis.