Visão Computacional: reconhecimento de texto com OCR e OpenCV

cursos.alura.com.br/course/visao-computacional-reconhecimento-texto-ocr-opencv/task/112874

Boas-vindas!! Chegamos à última aula do nosso curso. Já aprendemos vários conceitos sobre Tesseract OCR, Visão computacional e Open CV e agora aplicaremos todos eles em um **Projeto final** que você poderá colocar no seu portfolio.

Agora, nós trabalharemos em um novo Colab, disponível no Projeto final do curso. Nesse projeto, nosso objetivo, é:

- Buscar termos específicos nas imagens;
- Salvar o resultado de cada uma das imagens em um arquivo TXT;
- Mostrar os resultados sobre as imagens dos termos específicos;

Vamos aplicar todos os conhecimentos obtidos a partir das aulas para realizarmos um trabalho espetacular! A primeira parte, que é preparar o ambiente, está pronta. Eu fiz o `!pip install` de todos os nossos ambientes requeridos: Open CV, PyTesseract e Tesseract OCR.

```
!pip install opencv-python==4.6.0
!sudo apt install tesseract-ocr
!pip install pytesseract==0.3.9
```

Também fiz o `git clone`, que é de onde retiraremos os nossos dados outra vez. A pasta "text-recognize" já está disponível com todas as imagens que usaremos.

```
! git clone https://github.com/sthemonica/text-recognize
```

O resultado, é:

```
Cloning into 'text-recognize'...
remote: Enumerating objects: 103, done.
remote: Counting objects: 100% (103/103), done.
remote: Compressing objects: 100% (93/93), done.
remote: Total 103 (delta 12), reused 96 (delta 9), pack-reused 0
Receiving objects: 100% (103/103), 11.98 MiB | 18.92 MiB/s, done.
Resolving deltas: 100% (12/12), done.
```

Além disso, importei as bibliotecas com todos os módulos necessários. Nós já utilizamos essas bibliotecas durante o projeto, exceto a OS.

```
import pytesseract
import numpy as np
import cv2
import re
import os
import matplotlib.pyplot as plt

from PIL import ImageFont, ImageDraw, Image
from pytesseract import Output
from google.colab.patches import cv2_imshow
```

Depois, conferi se as versões do PyTesseract e do OpenCV são as que utilizamos na aula passada e também as que pedimos para instalar.

```
pytesseract.__version__
```

> 0.3.9

```
cv2.__version__
```

> 4.6.0

Construímos a pasta `tessdata` e instalamos o português e o inglês, assim como fizemos na atividade de desafio.

```
!mkdir tessdata
!wget -O ./tessdata/por.traineddata https://github.com/tesseract-
ocr/tessdata/blob/main/por.traineddata?raw=true
!wget -O ./tessdata/eng.traineddata https://github.com/tesseract-
ocr/tessdata/blob/main/eng.traineddata?raw=true
```

Nosso ambiente está pronto e podemos seguir para a etapa de análise dos dados. Primeiro, precisamos entender quais são os nossos dados e de onde estão vindo. Eles estão vindo do mesmo GitHub que estávamos utilizando nas aulas anteriores.

Dentro da pasta "text-recognize > Imagens" encontraremos outra pasta que se chama "Projeto". Ela é composta por quatro imagens de arquivo novas, com bastante texto, inéditas no nosso curso: artigo-desbalanceamento.png; artigo-eng-dados.png; artigo-spark.png; e artigo-termos-ML.png.

Com o que aprendemos nas aulas anteriores, conseguimos trabalhar com textos maiores. Nós já lidamos com imagens de livros, por exemplo, e temos uma ideia de como proceder. Antes, essas imagens estavam soltas, mas, agora construiremos um projeto com todas as imagens concatenadas, isto é, trabalharemos com todas ao mesmo tempo.

Então, quando pesquisarmos uma palavra, desejamos pesquisar em toda a pasta "Projeto". Com isso, otimizaremos nosso tempo. Esse atalho será muito útil, por exemplo, no caso de buscarmos uma palavra, termo, e-mail ou data específica com o regex em uma pasta de 500 imagens.

Faremos a variável `projeto` igual ao caminho da pasta "Projeto", não o caminho das imagens. Na linha abaixo, faremos outra variável, chamada `caminho`, igual a `[os.path.join(projeto, f)`. Depois, faremos um `for` na mesma linha, `for f in os.listdir(projeto)`.

```
projeto = "/content/text-recognize/Imagens/Projeto"
caminho = [os.path.join(projeto, f) for f in os.listdir(projeto)]
print(caminho)
```

> ['/content/text-recognize/Imagens/Projeto/artigo-termos-ML.png', '/content/text-recognize/Imagens/Projeto/artigo-eng-dados.png', '/content/text-recognize/Imagens/Projeto/artigo-desbalanceamento.png', '/content/text-recognize/Imagens/Projeto/artigo-spark.png']

A biblioteca `os` tem as rotinas do nosso computador. Nessa linha, ele está fazendo um `.join`, isto é, está juntando o nosso `projeto` com o `f`. Já o `f`, está percorrendo toda a nossa pasta `projeto` e pegando o diretório em que elas estão.

Ao final, faremos um `print()` do caminho e ele vai gerar uma lista que começa com "Projeto", que já havíamos definido, e termina com o nome de cada uma das imagens. Agora temos o caminho das quatro imagens. O próximo passo é fazer uma função que nos permita mostrar as imagens. Essa função já está no nosso Colab, pronta e comentada, assim ficará mais fácil estudá-la e entendê-la.

```
def mostrar(img):
  fig = plt.gcf() # busca a figura atual
  fig.set_size_inches(20, 10) #define o tamanho
  plt.axis("off") #remove a visualização dos eixos
  plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)) #faz a conversão de cores com o OpenCV
  plt.show() # mostra a imagem
```

Vamos fazer um `def mostrar()`, passando o parâmetro `img`, referente à imagem que estamos recebendo. Após definirmos a função, faremos `fig` igual a `plt.gcf()` e não passaremos nenhum parâmetro. O `gcf()` vai buscar a figura atual, ou seja, pegará o primeiro caminho, `/content/text-recognize/Imagens/Projeto`.

Em seguida, faremos `fig.set_size_inches()` e passaremos, como parâmetro, `20, 10`. Com este parâmetro, definimos o tamanho das imagens. Na próxima linha, faremos `plt.axis()`, passando, como parâmetro, `off`. Com isso, removeremos a visualização dos eixos dentro do Matplotlib.

Em uma das aulas anteriores, nós visualizamos uma imagem no Matplotlib e ela ficou com os eixos, como se estivesse em um gráfico. O `plt.axis()` desativa os eixos, impedindo que isso aconteça.

Na outra linha, faremos `plt.imshow()`, passando o comando do `cv2.cvtColor()`. Dentro dos parênteses, passaremos `img` e `cv2.COLOR_BGR2RGB`, ou seja, estamos construindo a nossa função de dentro para fora: primeiro convertemos as cores de BGR para RGB e depois passamos para o Matplotlib.

Por último, com o `plt.show()`, mostraremos a imagem com o MatplotLib. Agora, a nossa função `mostrar` está pronta! Vamos rodá-la para que ela fique salva dentro do nosso código.

Prosseguindo, vamos pensar em uma forma dinâmica de passar as nossas quatro imagens. Uma possibilidade, é fazer `for imagem in caminho:` e, na próxima linha, `imagem = cv2.imread(imagem)`. Por fim, a função `mostrar(imagem)`, que é o que estamos passando como parâmetro.

```
for imagem in caminho:
  imagem = cv2.imread(imagem)
  mostrar(imagem)
```

Ele agrupou os quatro *plots* em um único resultado do Google Colab, sendo que cada uma das imagens está separada por uma linha branca. Para que na próxima aula seja possível realizar novas formulações de reconhecimento do texto, vamos deixar pronta uma função para o Tesseract. Anteriormente, nesse vídeo, nós fizemos uma importação do `config_tesseract`, o `mkdir`.

Agora, importaremos o `config_tesseract` como `tessdata` e, ao invés de construirmos um código que mostre todos os OSDs e nos permita fazer a marcação ou leitura da imagem, com o OCR, a leitura será realizada com uma função.

```
config_tesseract = "--tessdata-dir tessdata"

def OCR_processa(img, config_tesseract):
  texto = pytesseract.image_to_string(img, lang='por', config=config_tesseract)
  return texto
```

Estamos usando uma nova função chamada `OCR_processa`. Então, `def OCR_processa()`, recebendo os parâmetros `img`, que é a nossa imagem, e `config_tesseract`, que estamos definindo acima.

Na próxima linha, definiremos `texto = pytesseract.image_to_string()`. Neste trecho, estamos passando a imagem para *string*. Dentro dos parênteses, o primeiro parâmetro é a imagem, `img`, o segundo é `lang='por'`, e o terceiro é `config=config_tesseract`. O retorno que nós teremos será `texto`. Fazendo isso, não será necessário definir outras vezes a função `texto`.

Na próxima aula, começaremos a retirar os textos da imagem.