Rocketlab - Recife - Visagio

**Grupo Piseiro:** Emerson Guilherme, Felipe Mororo, Matheus Vieira, Thiago Botelho.

# 1. Objetivo do desafio:

Construir um script para extrair textos de imagens de uma pasta utilizando a técnica de OCR e criar arquivos txt com o mesmo nome das imagens disponibilizadas, contendo o texto lido em cada imagem.

# 2. Metodologia :

O problema foi dividido em passos para facilitar a sua resolução:

* Passo 1: Listar todos os arquivos no diretório que contém as imagens
* Passo 2: Inserir os caminhos dos diretórios onde serão salvos os arquivos .txt gerados na leitura das imagens.
* Passo 3: Lê a imagem
* Passo 4: Analisa o tamanho da imagem
* Passo 5: Realiza o pré-processamento da imagem (nos testes com pré-processamento)
* Passo 6: Usa a biblioteca escolhida para realizar a extração dos textos das imagens
* Passo 7: Salva em um arquivo com o nome do arquivo da imagem
* Passo 8: Salva a confiabilidade para cada extração de textos
* Passo 9: Move o arquivo para pasta de destino

Primeiramente foi testado a biblioteca *Pytesseract* para realizar a extração do texto das imagens, mas percebemos que a leitura não estava sendo precisa em algumas imagens, então começamos a inserir pré-processamentos para facilitar a leitura. Em paralelo, vimos que havia outra biblioteca que também executava a leitura de imagens, a *Easy\_OCR*, que se mostrou mais precisa na extração dos textos, mas para ter a certeza de qual desempenhava melhor, decidimos realizar quatro testes e comparar os resultados. Os testes foram realizados com quatro códigos diferentes onde utilizamos as bibliotecas citadas sem o pré-processamento e com o pré-processamento. No pré-processamento transformamos a imagem em preto e branco e aplicamos a função *threshold* nessa imagem tratada com o método *THRESH\_BINARY* e escolhendo o parâmetro 127 para indicar que queremos transformar em preto tons abaixo de 127 e transformar em branco tons acima de 127.

Ao final, utilizamos um dado que as próprias bibliotecas retornam, que é a probabilidade de acerto considerada ao ler uma pequena parte do texto da imagem, onde pegamos todas as probabilidades de todas as partes da imagem analisada e tiramos a média para ter a confiabilidade média na extração.

## 2.1 Pytesseract sem pré-processamento:

Link do notebook no Colab: [Pytesseract sem pré-processamento](https://colab.research.google.com/drive/1czVXYcD6S8gwuYnI-woEgtMe5uejovds?usp=sharing)

Os textos gerados por este teste podem ser encontrados na pasta **pytesseract\_sem\_preprocessamento** dentro de **resultado\_dos\_testes.**

## 2.2 Pytesseract com pré-processamento:

Link do notebook no Colab: [Pytesseract com pré-processamento](https://colab.research.google.com/drive/12pILQbZy0POA-XQtpOe9yfj-KDq4hh5T?usp=sharing)

Os textos gerados por este teste podem ser encontrados na pasta **pytesseract\_com\_preprocessamento** dentro de **resultado\_dos\_testes.**

## 2.2 Easy\_OCR sem pré-processamento:

Link do notebook no Colab: [Easy OCR sem pré-processamento](https://colab.research.google.com/drive/11AeVx4fVa1EavPvOOFsI8k-kiCVG22i5?usp=share_link)

Os textos gerados por este teste podem ser encontrados na pasta **Resultados\_EasyOCR\_Sem\_Pre-Processamento** dentro de **resultado\_dos\_testes.**

## 2.2 Easy\_OCR com pré-processamento:

Link do notebook no Colab: [Easy OCR com pré-processamento](https://colab.research.google.com/drive/1-RS4UQLZM0i796yI_c-3t4SZyQBawWBA?usp=share_link)

Os textos gerados por este teste podem ser encontrados na pasta **Resultados\_Easy\_ocr\_com\_preprocess** dentro de **resultado\_dos\_testes.**

**3. Resultados:**

Após a execução dos scripts é possível comparar os textos presentes na imagem com a string resultante no arquivo *txt*. Desse modo foi possível classificar entre *sem texto*, *ruim*, *bom* e *ótimo*. Com essa classificação e com os dados de confiabilidade da biblioteca foi feito um Dashboard a fim de facilitar a análise. Segue o link do arquivo:

[Dashboard\_OCR\_Time\_Piseiro](https://docs.google.com/spreadsheets/d/1oV-j2fz8Rqo4j93lRRxDvTVQSbbNNQvcYebGs0FH9qA/edit?usp=sharing)

Podemos observar que o Pytesseract define um valor alto de confiabilidade para situações na qual não é gerado nenhum texto ou até gera o texto mas de acordo com a classificação dos autores a leitura é ruim, e nesses casos temos um falso positivo causando problema na interpretação desse parâmetro.

Também é possível observar que no geral o pré-processamento não obteve melhora significativa na geração dos textos e isto é causado pelas seguintes situações:

* A própria biblioteca já faz um processamento de imagem, então nesse caso pode estar ocorrendo que mais um processamento não ajude;
* O pré-processamento definido pelos autores melhora a performance em algumas imagens e atrapalha em outras, então na média acaba compensando e não impactando o resultado geral;

Durante os testes foram definidos vários processamentos de imagem porém não foi observado melhoras significativas no reconhecimento de letras/palavras, desse modo, o problema pode ser originado na maneira na qual a IA foi treinada.

# 4. Conclusão:

Analisando os resultados, foi definido que a biblioteca **EasyOcr sem o pré-processamento** obteve a melhor performance em extração de textos nas imagens de redes sociais, e apesar do tempo de processamento ser alto, ele é compensado pela maior exatidão nos textos gerados.