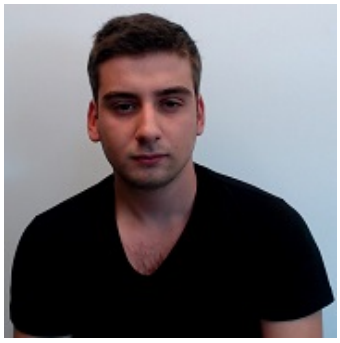
Universidade do Minho

Departamento de Informática

Ciências da Computação - LCC

Projeto - HOCR to Markdown converter



Bruno Silva A91649

Gonçalo Rodrigues A91641

Tiago Guimarães A91689

Coordenador

José João Alemeida

Junho de 2023

Conteúdo

Capítulo 1

Introdução

1.1 Contextualização

O seguinte relatório tem como principal objetivo clarificar e analisar o projeto realizado, o projeto em causa trata-se de um conversor **HOCR to Markdown**. Através da ferramenta **Pytesseract** que tem como função ler um texto captado de uma imagem e converte para um ficheiro **".hocr"**.Ficheiro esse que irá ser trabalhado e mais tarde convertido em **Markdown** como estado final.

Capítulo 2

Análise e Especificação

2.1 Descrição

Existem já várias ferramentas de **OCR** que exportam a informação extraída num formato textual. Neste projeto foi recomendado, pelo coordenador, que utilizássemos o Pytesseract. Esta ferramenta recebe uma imagem e retorna um ficheiro **.hocr**. A partir daqui coube a este grupo que tratasse a informação.

O primeiro passo foi criar um parser que permitisse organizar todos os dados de modo a que o ficheiro **.hocr** se dividisse em classes.

Depois de termos as classes preenchidas com a informação recebida foi necessário perceber como é que a ferramenta seleciona os textos e imagens. Para isso, criamos uma série de funções que desenham caixas (bbox) em torno de certas áreas que definem imagens, parágrafos, conjunto de parágrafos e linhas baseado em coordenadas que o **Pytesseract** fornece no ficheiro **.hocr**.

Seguidamente implementamos uma função capaz de extrair as imagens que a ferramenta deteta como **"Photo"**, mais uma vez baseado em coordenadas de bbox.

Após esta abordagem mais geral, começamos a tratar o texto recebido, foram adicionadas 3 funções que limpam o texto. Muitas vezes o **Pytesseract**, devido à má qualidade da foto entre outros fatores, deteta letras como símbolos, ou até mesmo imagens como texto, duas das funções foram criadas para eliminar esse "lixo". A terceira serve para eliminar translineações para que o texto fique mais legível.

Implementamos mais uma função que agora vai passar a informação contida nas classes para um ficheiro .md, que depois vai ser utilizado com a ferramenta pandoc que nos permite visualizar o **Markdown**.

Por fim, fizemos um pequeno menu que permite ao utilizador selecionar as opções que quer usar na conversão.

2.2 Requisitos

2.2.1 Tipo de dados

Para o tipo Page guardamos o seu id e criamos duas listas, uma que contém as Careas e outra que contém as imagens da página.

Figura 2.1: Tipo de uma página.

Para o tipo Carea guardamos o seu id, coordenadas da bbox e os parágrafos que contém são guardados numa lista.

Figura 2.2: Tipo de uma Carea.

Para o tipo Par guardamos o seu id, coordenadas da bbox, a linguagem (lang) e uma lista que contém as linhas correspondentes a esse parágrafo.

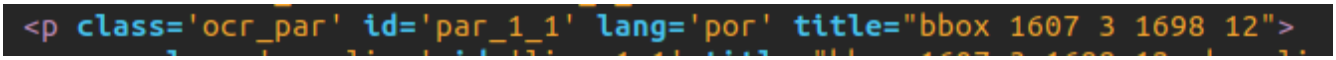


Figura 2.3: Tipo de um parágrafo.

Para o tipo Line guardamos o seu id, coordenadas da bbox, o tamanho de letra (xsize) e uma lista que contém as palavras contidas numa linha.

Figura 2.4: Tipo de uma linha.

Para o tipo Word guardamos o seu id, coordenadas da bbox, a confiança (atributo que o **Pytesseract** atribui para definir a confiança na captação de uma palavra) e o texto propriamente dito.

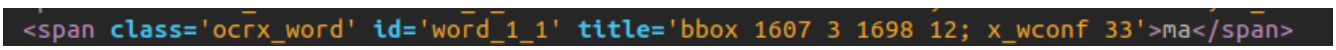


Figura 2.5: Tipo de uma palavra.

Para o tipo **Photo** guardamos o seu id e as coordenadas da bbox.

```
<div class='ocr_photo' id='block_1_3' title="bbox 1011 0 1125 55"></div>
```

Figura 2.6: Tipo de uma imagem.

2.2.2 Parser

A função **parseHocr** recebe o ficheiro .hocr gerado pelo **Pytesseract** e devolve um objeto do tipo Page. O tipo page contém todos os dados que utilizamos para o desenvolvimento até ao resultado **Markdown**.

2.2.3 Visualização do resultado do Pytesseract

Existem quatro funções que nos permitem visualizar a bbox que o **Pytesseract** selecionou e extraiu o texto.

A função **drawCareaBoxes** que, com base nas coordenadas das Careas, desenha um retângulo que representa a Carea que a ferramenta selecionou.

A função **drawParBoxes** que, com base nas coordenadas dos Pars, desenha um retângulo que representa o parágrafo que a ferramenta selecionou.

A função **drawLinesBoxes** que, com base nas coordenadas das Linhas, desenha um retângulo que representa a linha que a ferramenta seleciona.

A função **drawImageBoxes** quem com base nas coordenadas das Photos, desenha um retângulo que representa a imagem que a ferramenta seleciona.

2.2.4 Extração de imagens

Para extrair as imagens usamos a função extractImages que beneficia das utilidades da biblioteca **PIL**. A função a partir das coordenadas da bbox de um tipo Photo é capaz de selecionar a imagem identificada pelo Pytesseract.

2.2.5 Limpeza de texto

De forma a que o texto fique mais limpo adicionamos três funções para esse efeito, são elas **cleanTxt**, **confCheck** e **removeCarateresNS**.

A função **cleanTxt** recebe como argumento uma Page e percorre-a, sempre que encontra uma translineação junta as duas palavras na linha da segunda palavra. A função está preparada para quando a translineação ocorre em parágrafos diferentes.

A função **confCheck** recebe como argumento uma **Page** e um inteiro, mais uma vez, percorre-a, sendo que a cada linha faz uma média da confiança, que o **Pytesseract** tem, de todas as palavras presentes nessa linha, caso encontre uma linha com uma média de confiança inferior ao inteiro (escolhido pelo utilizador) elimina essa mesma linha.

A função **removeCarateresNS** recebe, também, uma Page como argumento,e irá eliminar aqueles caracteres que estão sozinhos e não fazem sentido no contexto do artigo, o que permite uma melhor leitura.

2.2.6 Tradução da informação para Markdown

O foco e o objetivo principal deste projeto é a passagem da informação recebida através do **Pytesseract** e depois de devidamente tratada e organizada pelas funções enunciadas anteriormente está agora na altura de converter a informação trabalhada para **Markdown** e para isso acontecer temos a função **createMarkdown**, que começa por receber por parâmetro **articles** que são os artigos já depois de todo o tratamento, vão ser escritos para um ficheiro **outputFile** convertendo assim para **Markdown** e para uma melhor vizualização deste **Markdown** utilizamos a ferramenta **Pandoc** .

2.2.7 Command line interface

Implementamos também uma "Command line interface"para que o utilizador possa ver os comandos à sua disposição. Nenhum desses comandos é obrigatório e todos os que pedem um argumento têm um valor por default.

```
NAME
    hocr2md - hocr to markdown

SYNOPSIS
    hocr2md sourceImage [OPTION]

DESCRIPTION
    Converts image into hocr file, via tesseract, and hocr into markdown.
        -h, --help              display this help menu

        -o, --output            choose name and location to output markdown file (location must exist)

        -e, --extractImages     extract images into nameOfInputFile_Images/

        -p, --psm               choose psm to be used by tesseract (default=3)

        -l, --lang              choose language to be used by tesseract (default=por)

        --extractImagesFolder   choose folder to extract images

        --conf                  choose value for line confidence (default=40, line is deleted if below confidence)

        --dc                    show image with Careas limits drawn

        --dp                    show image with Pars limits drawn

        --dl                    show image with Lines limits drawn

        --di                    show image with Images limits drawn

        --da                    show image with Articles limits drawn

    Some page segmentation modes, get a full list with tesseract --help-psm:
         1                      Automatic page segmentation with OSD.
         3                      Fully automatic page segmentation, but no OSD. (Default)
         4                      Assume a single column of text of variable sizes.
         5                      Assume a single uniform block of vertically aligned text.
         6                      Assume a single uniform block of text.
        11                      Sparse text. Find as much text as possible in no particular order.
        13                      Raw line. Treat the oppenedImage as a single text line,
                                bypassing hacks that are Tesseract-specific.
```

Figura 2.7: Command line interface.

Capítulo 3

Código e ferramentas utilizadas

Para a execução deste trabalho foi utilizada a linguagem Python e a ferramenta utilizada como outrora menciona foi o Pytesseract que tem como função principal capturar texto de imagens, imagens essas que neste caso prático derivam marioritariamente de jornais antigos.

Capítulo 4

Conclusão

Para a conclusão deste relatório, o projeto a nós entregues tinha como foco converter um ficheiro **Hocr** para **Markdown**, para a sua realização foram necessárias ferramentas como o pytesseract que extraia informação de texto de uma imagem, apesar da sua enorme eficáfia algumas vezes o **pytesseract** devido à qualidade das imagens não ser a melhor possível ou de erros internos da ferramenta, havia erros, erros que tiveram de ser tratados pelos membros do grupo, como translineações, retificação ou remoção de caracteres invulgares e errados que por vezes surgiam entre outras.Além do tratamento de erros, existiram outros focos como replicar o texto da forma como era visto nos jornais.Apesar de tudo isso, não foi possível por nós a aplicação de uma àlgebra de "caixas"que teria como objetivo ligar caixas de texto que pertenciam a um mesmo artigo mas que o **pytesseract** não era capaz de assinalar,mas devido à sua complexidade não foi a nós possível a sua implementação que tornaria o nosso projeto um pouco mais completo. Agradecemos ao coordenador do nosso projeto, o professor José João Alemeida, pelo o apoio e disponiblidade.