OSDOcr Modules

Gonçalo Afonso

November 21, 2023

1 Introduction

In this document, we will formalize the Old Structured Document OCR (OSDOcr) modules.

2 OCR box Module

2.1 OCR Box

A OCR box represents a container element for a region in a document. Each container may include other containers of lower levels, with the lowest being a word container. Based on a n-ary tree structure.

- **level** : text level of the box. $\{1 : \text{page}, 2 : \text{block}, 3 : \text{paragraph}, 4 : \text{line}, 5 : \text{word}\}$
- **page_num** : only meaningful when multiple pages are processed.
- **block_num** : block identifier in which box is inserted
- **line_num** : line identifier in which box is inserted
- **word_num** : word identifier (applicable if level is word)
- **box** : instance of box class (stores coordinates of bounding box)
- **text** : text recognized inside the box
- **conf** : level of confidence in the text
- **id** : box identifier
- **type** : type of box. ['delimiter', 'image', 'text']
- **children** : children boxes (all of lower level and contained within itself)
- **parent** : parent box (box of higher level that contains it)

2.2 Methods

- **is_empty** : $OCR\_Box \rightarrow Bool$

  Checks if a box container is empty. Every box of level 5 (word) within it has to be empty for a positive result.

- **is_delimiter** : $OCR\_Box \rightarrow Bool$

  Checks if a box group is a delimiter. A delimiter is an empty box container that follows the rule:

  $$box.width \geq box.height \times 4 \vee box.height \geq box.width \times 4 \qquad (1)$$

  where *box* is the OCR box's Box instance.

- **get_id** : $(OCR\_Box, id : Str, level : Int) \rightarrow OCR\_Box$

  Finds a box container, within higher level box, or itself. The box container is identified by the *id* and the *level*.
- **calculate_mean_height** : $OCR\_Box \rightarrow Float$

  Calculates the mean height of a box group.
- **is_text_size** :
  $(OCR\_Box, text\_size : Float, mean\_height : Float?, range : Float) \rightarrow Float$

  Checks if a box is of a text size. A bpx is of text size if the mean height of the box group is within the range of the text size. Range is by default 0.3.
- **get_delimiters** :
  $(OCR\_Box, search_area : Box, orientation : Str, conf : Int) \rightarrow [OCR\_Box]$

  Gets the delimiter boxes in a box group. The delimiter blocks are the blocks that are delimiters and are inside the search area and respect the given orientation.

3 Engine Module

tesseract_search_img : $img\_path : Str \rightarrow Dict$

Searches text in an image using tesseract. The result is a dictionary with bounding boxes.

tesseract_convert_to_ocrbox : $Dict \rightarrow OCR\_Box$

Turns a dictionary of tesseract results into a OCR box instance.

4 OCR Analysis Module

analyze_text : $OCR\_Box \rightarrow Dict$

Analyzes a box group. The analysis result returns the value of *normal_text_size*, *normal_text_gap*, *number_lines*, *number_columns* and *columns*.

Algorithm:

```

    # get lines
    lines = OCR_Box.get_box_level(4)

    # save line sizes and margins
    line_sizes = []
    left_margin_n = {}
    right_margin_n = {}
    for line in lines:
        * save line size
        * save left margin
        * save right margin

    # estimate normal text size
    # calculate until good standard deviation is found
    while deviation > normal_text_size*2:
        * remove outlier
        * recalculate normal_text_size and deviation


    # estimate normal text gap
    for line in lines:
        if lines are in sequence and of normal text size:
            * save text gap
    normal_text_gap = sum(text_gaps)/len(text_gaps)

    # estimate number of lines per column
    number_lines = (highest_normal_text_size - lowest_normal_text_size) /
    normal_text_gap

    # estimate number of columns
    number_columns = sort number of left margins, if value is close to number_lines,
                     then it is a column

    # create columns bounding boxes
    columns = []
    for column in number_columns:
        if first:
            box = Box(left_border,first_margin,highest_normal_text_size,
    lowest_normal_text_size)
        else:
            box = Box(previous_margin,margin,highest_normal_text_size,
    lowest_normal_text_size)

    return {'normal_text_size':normal_text_size,
            'normal_text_gap':normal_text_gap,
            'number_lines':number_lines,
            'number_columns':number_columns,
            'columns':columns}
```

Listing 1: analyze_text algorithm

draw_bounding_boxes : $(OCR\_Box, image\_path : Str, draw\_levels : [Int], id : Bool) \rightarrow img : MatLike$

Draws bounding boxes in an image. The image is loaded from *image_path* and the bounding boxes are drawn in the image according with boxes group given and the levels in *draw_levels*. If *id* is true, the id of each box is also drawn in the image.

estimate_journal_header : $(OCR\_Box, image\_info : Dict) \rightarrow Box$

Estimates the journal header using its box group. The header is estimated by finding the blocks that are delimiters and follow the rule:

$$delimiter['bottom'] \geq image\_info['bottom'] \times 0.5 \wedge delimiter['width'] \geq image\_info['width'] \times 0.3 \quad (2)$$

Algorithm:

```
        # get horizontal delimiters
        delimiters = OCR_Box.get_delimiters(upper_half_image,'horizontal')

        delimiters = sort delimiters by width

        widest_delimiter = delimiters[0]
        if widest delimiter is 30% of image width or more:

            * calculate header area
            return header_area
```

Listing 2: estimate_journal_header algorithm

estimate_journal_columns : $(OCR\_Box, image\_info : Dict, header : Box?, footer : Box?) \rightarrow [Box]$

Estimates the journal columns using its box group. The columns are estimated by finding the blocks that are vertical delimiters and are within the area between the header and the footer if they exist (ortherwise within the page).

Algorithm:

```

        # get potential column delimiters
        delimiters = OCR_Box.get_delimiters(body_area,'vertical')

        # clean/join delimiters
        delimiters = join_aligned_delimiters(delimiters)

        # sort delimiters from left to right
        delimiters = sort delimiters by left

        # estimate column boxes
        for delimiter in delimiters:
            if first:
                column = Box(left_border,delimiter['left'],top_body,bottom_body)
            else:
                column = Box(previous_margin,delimiter['left'],top_body,bottom_body)

        return columns


```

Listing 3: estimate_journal_columns algorithm

estimate_journal_template : $(OCR\_Box, image\_info : Dict) \rightarrow Dict$

Estimates the journal template using its box group. Returns a dictionary with the header and the columns.

5 OCR Box Fix Module

improve_bounds : $OCR\_Box \rightarrow OCR\_Box$

Improves the bounds of a box group. Not yet finished.

block_box_fix : $OCR\_Box \rightarrow OCR\_Box$

Fixes the blocks boxes in box group. Eliminates empty, non delimiter boxes and eliminates intersections.

Algorithm:

```

        # get blocks
        blocks = OCR_Box.get_box_level(2)


        for block in blocks:
            * choose current block to analyse if no current block
            ** if block is empty and not delimiter:
                * remove block

            * check current block for intersections
            ** if intersection:
                * treat intersection by separating blocks

            * check for inner blocks
            ** if inner blocks:
                * keep outer block if inner blocks are empty and not delimiters

            * if last block check for current block:
                ** if blocks to check:
                    *** choose next current block

        return blocks


```
Listing 4: block_box_fix algorithm

join_aligned_delimiters : $(delimiters : [OCR\_Box], orientation : Str) \to [OCR\_Box]$

Joins aligned delimiters. The delimiters are aligned if they have the same horizontal or vertical value within a range (*is_aligned* for further reading), depending on the orientation.

6 Information Extraction Module

journal_template_to_text : $(journal\_teamplate, OCR\_Box) \to str$

Converts ocr$_r esultstotextusingjournal_t emplate.$

Algorithm:

```

        # Treat header
        header_ boxes = journal_template['header']
        text = header_boxes.to_text()

        # Treat columns
        for column in journal_template['columns']:
            # separate column in articles
            * get article delimiters
            * get article boxes
            * create article
            ** analyse article text (analyse\_text)
            ** search for potential article atributes
            *** title
            *** author
            *** body


        # Add articles to text
        for article in articles:
            * add article text to text

        # Treat footer
        footer_boxes = journal_template['footer']
        text += footer_boxes.to_text()




        return text

```
Listing 5: journal_template_to_text algorithm

7 Output Converter Module

boxes_to_text : $OCR\_Box \to Str$

Converts a box group into a string. The string is the concatenation of the text of each box in the group.