Automatic Recognition of Historical Newspaper Content

Generated by Doxygen 1.8.12

Contents

Chapter 1

mdp-newspaper-segmentation

This repository will house our code for prototyping and production. In our weekly meetings, we will discuss appropriate documentation techniques, subdivide programming tasks, and assign issues accordingly.

MDP 2016 Proquest News

End-to-End System

The End-to-End system is what we use to evaluate segmentation algorithms. On one end, the system uses a segmentation algorithm to produce a predicted segmentation of a newspaper image. The output of the segmentation algorithm is passed to the other end of the system, which evaluates the predicted segmentation against a ground-truth segmentation.

Required Software

General: python, a LaTeX generator (such as pdflatex) Python packages: numpy, matplotlib, PIL

Usage

1. Set up configuration file

The configuration file serves as the input to the end-to-end evaluation system. The configuration file details the following:

1. A list of segmentation algorithms to test. This is the “Metrics” field. This field should be a list of .py files. These files are the evaluation metrics.
2. A list of evaluation metrics with which to evaluate the segmentation algorithms. This is the “Implementations” field. The “Implementations” field should be a list of .py files. These files are the segmentation algorithms. More than one segmentation algorithm can be used; each will be evaluated individually. Each segmentation algorithm produces a .json file.
3. A path to the newspaper and ground-truth data, with which we will produce and evaluate segmentations. This is the “Data” field. The path should be to a directory (or a .jp2 if only evaluating one image).

4. A path to the location of the predicted segmentations (i.e. output path), and a path to evaluation output. This is the “Outpath” field. This should be a list of two paths: one for the segmentation output, a second for the evaluation output.

An example configuration file is displayed below. Let it be named "test.config". The configuration file must have the following format:

```
{
    “Metrics”: [“EvalOneToMany.py”],
    “Implementations”: [“textblocksBS.py”],
    “Data”: [“/path/to/data_and_groundtruth/”],
    “Outpath”: [“/path/to/segmentation_output”, "/path/to/evaluation_output"]
}
```

2. Running the End-to-End system

Navigate to the code/ directory and in the terminal do:

```
python end2end.py test.config
```

Output will be sent to the output path detailed in test.config.

3. Output

The configuration file's "Outpath" field lists two paths: one path to a location for the segmentation output, a second path to the evaluation output.

1. Segmentation Output: The end-to-end system places the .json files produced by the segmentation algorithms in this path. The segmentation algorithms are those detailed in the configuration file's "Implementations" field.
2. Evaluation Output: The end-to-end system places a .pdf report of the evaluation into this path. This .pdf file is for human digestion, as it nicely summarizes the results of each segmentation algorithm. Additionally, a text file containing the results is placed into this path.

Chapter 2

Todo List

Member Box.Box.from_str (self, string)

: allow easy corrections for height/width / scale tranformations

Member Image.NewsImage.get_total_blackness (self, y0, x0, y1, x1)

: fix (should be four terms)

Class processXML.ProcessXML

: combine getTBData & writeTBData (have latter call former)

Member Segmentation.Segmentation.pair_recall (self, truth)

: make robust to overlaps...

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Box Namespace Reference

Classes

- class Box
  A box is an axis-aligned rectangle, represented by [mincoors, maxcoors] For us, coordinates are length-2 lists of form [y, x], in units of pixels of the original image.

6.2 end2end Namespace Reference

Classes

- class EndToEnd

Functions

- def main ()

6.2.1 Function Documentation

6.2.1.1 def end2end.main ( )

6.3 EvalOneToMany Namespace Reference

Classes

- class EvalOneToMany

Functions

- def main ()

Variables

- int NUM_CLASS = 3
- float ONETOONE_THRESHOLD = 0.85
- float ONETOMANY_THRESHOLD = 0.1
- list LABELS = ['text', 'title', 'other']

6.3.1 Function Documentation

6.3.1.1 def EvalOneToMany.main ( )

6.3.2 Variable Documentation

6.3.2.1 list EvalOneToMany.LABELS = ['text', 'title', 'other']

6.3.2.2 int EvalOneToMany.NUM_CLASS = 3

6.3.2.3 float EvalOneToMany.ONETOMANY_THRESHOLD = 0.1

6.3.2.4 float EvalOneToMany.ONETOONE_THRESHOLD = 0.85

6.4 Image Namespace Reference

Classes

- class NewsImage

  Processes image .jpgs and .xmls for evaluation pipeline.

6.5 imageBS Namespace Reference

Variables

- f_image = sys.argv[1]
- f_xml = sys.argv[2]
- f_out = sys.argv[3]
- f = f_image.split('/')[1].split('.')[0]
- string f_csv = 'images/'
- list boxes = [ ]
- dat = line.strip('\n')
- dictionary seg = {"annotations":[ ]}
- int id = 0
- dictionary tblnfo = {}
- indent

6.5.1 Detailed Description

```
python 2.7
```

6.5.2 Variable Documentation

6.5.2.1 list imageBS.boxes = [ ]

6.5.2.2 imageBS.dat = line.strip('\n')

6.5.2.3 imageBS.f = f_image.split('/')[1].split('.')[0]

6.5.2.4 string imageBS.f_csv = 'images/'

6.5.2.5 imageBS.f_image = sys.argv[1]

6.5.2.6 imageBS.f_out = sys.argv[3]

6.5.2.7 imageBS.f_xml = sys.argv[2]

6.5.2.8 int imageBS.id = 0

6.5.2.9 imageBS.indent

6.5.2.10 imageBS.seg = {"annotations":[ ]}

6.5.2.11 dictionary imageBS.tblInfo = {}

6.6 Latex Namespace Reference

Functions

- def replace_dict (string, replacements)
- def gen_latex (title, fnameout)
- def gen_pdf (fnametex, fnamepdf)

Variables

- latex_template = f.read()

6.6.1 Function Documentation

6.6.1.1 def Latex.gen_latex (*title*, *fnameout*)

6.6.1.2 def Latex.gen_pdf (*fnametex*, *fnamepdf*)

6.6.1.3 def Latex.replace_dict (*string*, *replacements*)

6.6.2 Variable Documentation

6.6.2.1 Latex.latex_template = f.read()

6.7 pipe_to_jason Namespace Reference

Variables

- fnamein
- fnameout
- list polys = [l.split('|') for l in f.read().split('\n') if l]
- list labpolys = [(p[0],eval('[%s]'%(','.join(p[1:])))) for p in polys]
- int sh = 6351
- int sw = 3960
- dictionary json

6.7.1 Variable Documentation

6.7.1.1 pipe_to_jason.fnamein

Author

Anonymous

6.7.1.2 pipe_to_jason.fnameout

6.7.1.3 dictionary pipe_to_jason.json

Initial value:

```
= {'annotations': [
    {'id':pi,'height':(b[1][0]-b[0][0])*sh,'width':(b[1][1]-b[0][1])*sw,'y':(b[0][0])*sh,'x':b[0][1]*sw, '
    type':'rect', 'class':lab}
    for pi, (lab,pol) in enumerate(labpolys) for b in pol]}
```

6.7.1.4 list pipe_to_jason.labpolys = [(p[0],eval('[%s]'%(','.join(p[1:])))) for p in polys]

6.7.1.5 list pipe_to_jason.polys = [l.split('|') for l in f.read().split('\n') if l]

6.7.1.6 int pipe_to_jason.sh = 6351

6.7.1.7 int pipe_to_jason.sw = 3960

6.8 Polygon Namespace Reference

Classes

- class Polygon

  A Polygon is a list of Boxes, presumed pairwise non-overlapping.

6.9 processXML Namespace Reference

Classes

- class ProcessXML

  class for processing .xml files to get ocr information

6.10 readJSON Namespace Reference

Functions

- def seg_from_json (fname, gt_flag)

  This script reads in a json file of this format.

6.10.1 Function Documentation

6.10.1.1 def readJSON.seg_from_json (*fname*, *gt_flag*)

This script reads in a json file of this format.
data["annotations"] returns a list of dicts, where each dict is a segmentation item of this format:

```
{

    "id": 0, (non-negative integer)

    "height": 1448.0,

    "width": 1536.0,

    "y": 1328.0,

    "x": 1288.0,

    "type": "rect",

    "class": "article" ("title", "other")

}
```

python 2.7

Author

: stefan larson

6.11 Segmentation Namespace Reference

Classes

- class Segmentation

  A Segmentation is a list of Polygons, presumed pairwise non-overlapping.

6.12 textblocksBS Namespace Reference

Functions

- def size_of_image (imname)

Variables

- f_out_folder = sys.argv[1]
- f_image = sys.argv[2]

  <image> is a jp2 or jpg
- f_xml = sys.argv[3]

  <xml> is xml file associated with the image
- f_out = sys.argv[4].split('/')[-1]
- pxml = processXML.ProcessXML(f_xml)
- tbList = pxml.getTBData()
- h
- w
- hj
- wj
- hs
- ws
- dictionary seg = {"annotations":[ ]}
- int id = 0
- dictionary tbInfo = {}
- f
- indent

6.12.1 Function Documentation

6.12.1.1 def textblocksBS.size_of_image (*imname*)

6.12.2 Variable Documentation

6.12.2.1 textblocksBS.f

6.12.2.2 textblocksBS.f_image = sys.argv[2]

<image> is a jp2 or jpg

6.12.2.3 textblocksBS.f_out = sys.argv[4].split('/')[-1]

6.12.2.4 textblocksBS.f_out_folder = sys.argv[1]

6.12.2.5 textblocksBS.f_xml = sys.argv[3]

<xml> is xml file associated with the image

6.12.2.6 textblocksBS.h

6.12.2.7 textblocksBS.hj

6.12.2.8 textblocksBS.hs

6.12.2.9 int textblocksBS.id = 0

6.12.2.10 textblocksBS.indent

6.12.2.11 textblocksBS.pxml = processXML.ProcessXML(f_xml)

6.12.2.12 textblocksBS.seg = {"annotations":[ ]}

6.12.2.13 dictionary textblocksBS.tblInfo = {}

6.12.2.14 list textblocksBS.tbList = pxml.getTBData()

6.12.2.15 textblocksBS.w

6.12.2.16 textblocksBS.wj

6.12.2.17 textblocksBS.ws

Chapter 7

Class Documentation

7.1 Box.Box Class Reference

A box is an axis-aligned rectangle, represented by [mincoors, maxcoors] For us, coordinates are length-2 lists of form [y, x], in units of pixels of the original image.

Public Member Functions

- def __init__ (self, coor0=None, coor1=None, string=None)

    The 0th coor is [miny,minx]; 1st coor is [maxy,maxx]; we may also initialize from a string, e.g.
- def from_point (self, coor)

    Initializes box as breadthless, lengthless, and centered at the inputted coordinate.
- def from_corners (self, coorA, coorB)

    Constructs the box from any two nonadjacent corners, in any order.
- def flatten (self)

    Converts internal [[miny,minx],[maxy,maxx]] into [miny,minx,maxy,maxx].
- def __str__ (self)

    String representation of box is simply that of its coordinates, e.g.
- def from_str (self, string)
- def area (self, image=None)

    Returns image area, weighted by image-values if an image is given.
- def center (self)

    returns [centery,centerx]
- def disconnect_distance (self, other)

    returns the largest possible rectangle — infinite in one of the vertical or horizontal directions — that separates the two boxes.
- def overlaps (self, other)

    Returns whether do the boxes share geometric area strictly greater than 0?
- def join (self, other)

    returns the smallest box containing both inputs
- def intersect (self, other)

    returns the largest box contained in both inputs, or else the [[0,0],[0,0]] box if there's no shared area

Public Attributes

- points
  [1,2] is a point.

7.1.1 Detailed Description

A box is an axis-aligned rectangle, represented by [mincoors, maxcoors] For us, coordinates are length-2 lists of form [y, x], in units of pixels of the original image.

1 is a 'coordinate'.

[1,2] is a point.

[[1,2],[3,4]] is a box.

zerobox = Box([0.0,0.0], [0.0,0.0])

Author

Samuel Tenka

7.1.2 Constructor & Destructor Documentation

7.1.2.1 def Box.Box.__init__ (*self*, *coor0* = `None`, *coor1* = `None`, *string* = `None`)

The 0th coor is [miny,minx]; 1st coor is [maxy,maxx]; we may also initialize from a string, e.g.

'[[1.0,0.0],[2.0,3.0]]'. If only coor0 is specified, then a point [coor0,coor0] is created.

Member variables:

0. coors represents coordinates, e.g. [[1.0,0.0],[2.0,3.0]].

7.1.3 Member Function Documentation

7.1.3.1 def Box.Box.__str__ (*self*)

String representation of box is simply that of its coordinates, e.g.

'[[1.0,0.0],[2.0,3.0]]'

7.1.3.2 def Box.Box.area (*self*, *image* = `None`)

Returns image area, weighted by image-values if an image is given.

7.1.3.3 def Box.Box.center (*self*)

returns [centery,centerx]

7.1.3.4 def Box.Box.disconnect_distance (*self, other*)

returns the largest possible rectangle — infinite in one of the vertical or horizontal directions — that separates the two boxes.

if the boxes overlap, this margin will be negative.

7.1.3.5 def Box.Box.flatten (*self*)

Converts internal [[miny,minx],[maxy,maxx]] into [miny,minx,maxy,maxx].

7.1.3.6 def Box.Box.from_corners (*self, coorA, coorB*)

Constructs the box from any two nonadjacent corners, in any order.

Canonicalizes the coordinates into [mincoors,maxcoors] form.

7.1.3.7 def Box.Box.from_point (*self, coor*)

Initializes box as breadthless, lengthless, and centered at the inputted coordinate.

7.1.3.8 def Box.Box.from_str (*self, string*)

Todo : allow easy corrections for height/width / scale tranformations

7.1.3.9 def Box.Box.intersect (*self, other*)

returns the largest box contained in both inputs, or else the [[0,0],[0,0]] box if there's no shared area

7.1.3.10 def Box.Box.join (*self, other*)

returns the smallest box containing both inputs

7.1.3.11 def Box.Box.overlaps (*self, other*)

Returns whether do the boxes share geometric area strictly greater than 0?

We don't consider image-values here.

7.1.4 Member Data Documentation

7.1.4.1 Box.Box.points

[1,2] is a point.

The documentation for this class was generated from the following file:

- Box.py

7.2 end2end.EndToEnd Class Reference

Public Member Functions

- def __init__ (self, config_filename='config.txt')
- def segment (self)
- def evaluate (self)
- def collect_data (self)
- def plot_performance_curve (self)
- def generate_report (self)

Public Attributes

- metrics
- implementations
- files
- seg_out_path
- eval_out_path
- eval_results

7.2.1 Detailed Description

```
assumes segmentation algo.s take command line arguments as follows:
     python dilate_segmenter.py <imagename> <xmlname> <outname>
  e.g.
     python dilate_segmenter.py 0005.jp2 0005.xml 0005_seg.json
```

7.2.2 Constructor & Destructor Documentation

7.2.2.1 def end2end.EndToEnd.__init__ (*self*, *config_filename* = 'config.txt')

7.2.3 Member Function Documentation

7.2.3.1 def end2end.EndToEnd.collect_data (*self*)

```
read in output data from evaluation metrics
    e.g. assume we have an image file named 0005.jpg, and segmentation algorithm
    textblocksBS.py, then this function will read data from 0005.jpg.textblcoksBS.py
    data will be saved in a dictionary in the following format:
    {"textblocksBS.py":
{
    precision:
    recall:
    score:
    history:
    {
        (precision1, recall1, score1),
        (precision2, recall2, score2)
    }
}
```

7.2.3.2 def end2end.EndToEnd.evaluate (*self*)

```
assumes format groundtruth - ....gt.json
```

7.2.3.3 def end2end.EndToEnd.generate_report (*self*)

7.2.3.4 def end2end.EndToEnd.plot_performance_curve (*self*)

7.2.3.5 def end2end.EndToEnd.segment (*self*)

7.2.4 Member Data Documentation

7.2.4.1 end2end.EndToEnd.eval_out_path

7.2.4.2 end2end.EndToEnd.eval_results

7.2.4.3 end2end.EndToEnd.files

7.2.4.4 end2end.EndToEnd.implementations

7.2.4.5 end2end.EndToEnd.metrics

7.2.4.6 end2end.EndToEnd.seg_out_path

The documentation for this class was generated from the following file:

- end2end.py

7.3 EvalOneToMany.EvalOneToMany Class Reference

Public Member Functions

- def __init__ (self, out_folder, seg_folder, gt_path, seg_path, img_path=None, xml_path=None, imp_↩
  name=None)
- def evaluate (self)
- def save_output (self, record)

Public Attributes

- seg_path
- gt_path
- xml_path
- out_folder
- img_path
- history_path
- ground_truth
- seg_to_eval

Static Public Attributes

- list eval_history = [ ]

7.3.1 Constructor & Destructor Documentation

7.3.1.1 def EvalOneToMany.EvalOneToMany.__init__ (*self*, *out_folder*, *seg_folder*, *gt_path*, *seg_path*, *img_path* = None, *xml_path* = None, *imp_name* = None)

7.3.2 Member Function Documentation

7.3.2.1 def EvalOneToMany.EvalOneToMany.evaluate (*self*)

7.3.2.2 def EvalOneToMany.EvalOneToMany.save_output (*self*, *record*)

7.3.3 Member Data Documentation

7.3.3.1 list EvalOneToMany.EvalOneToMany.eval_history = [ ] [static]

7.3.3.2 EvalOneToMany.EvalOneToMany.ground_truth

7.3.3.3 EvalOneToMany.EvalOneToMany.gt_path

7.3.3.4 EvalOneToMany.EvalOneToMany.history_path

7.3.3.5 EvalOneToMany.EvalOneToMany.img_path

7.3.3.6 EvalOneToMany.EvalOneToMany.out_folder

7.3.3.7 EvalOneToMany.EvalOneToMany.seg_path

7.3.3.8 EvalOneToMany.EvalOneToMany.seg_to_eval

7.3.3.9 EvalOneToMany.EvalOneToMany.xml_path

The documentation for this class was generated from the following file:

- EvalOneToMany.py

7.4 Image.NewsImage Class Reference

Processes image .jpgs and .xmls for evaluation pipeline.

Public Member Functions

- def __init__ (self, root, gamma=gamma_default)
  The argument 'root' could be, for example, root='../Data/0005', on which we append '.jpg' and '.xml'.
- def read_blackness (self)
  presumes .jpg to be grayscale; precomputes area blacknesses for more efficient evaluation
- def get_total_blackness (self, y0, x0, y1, x1)

Public Attributes

- filenames
- gamma
- blacknesses

Static Public Attributes

- float gamma_default = lambdax:1.0

7.4.1 Detailed Description

Processes image .jpgs and .xmls for evaluation pipeline.

For example, pre-computes area blacknesses for more efficient evaluation.

Author

Samuel Tenka

7.4.2 Constructor & Destructor Documentation

7.4.2.1 def Image.NewsImage.__init__ (*self*, *root*, *gamma* = gamma_default)

The argument 'root' could be, for example, root='../Data/0005', on which we append '.jpg' and '.xml'.

member functions:

0. 'gamma' transforms a grayscale value into a blackness measure.

7.4.3 Member Function Documentation

7.4.3.1 def Image.NewsImage.get_total_blackness (*self*, *y0*, *x0*, *y1*, *x1*)

Todo : fix (should be four terms)

7.4.3.2 def Image.NewsImage.read_blackness (*self*)

presumes .jpg to be grayscale; precomputes area blacknesses for more efficient evaluation

7.4.4 Member Data Documentation

7.4.4.1 Image.NewsImage.blacknesses

7.4.4.2 Image.NewsImage.filenames

7.4.4.3 Image.NewsImage.gamma

7.4.4.4 float Image.NewsImage.gamma_default = lambdax:1.0 `[static]`

The documentation for this class was generated from the following file:

- Image.py

7.5 Polygon.Polygon Class Reference

A Polygon is a list of Boxes, presumed pairwise non-overlapping.

Public Member Functions

- def __init__ (self, boxes=[ ], label='text', string=None)
- def check_valid (self)

  ensures no overlaps
- def area (self)
- def intersect (self, other)
- def __str__ (self)
- def from_str (self, string)
- def create_jaccard (denom_func, docstring=None)

  returns a method to compute overlaps of form intersect/denominator

Public Attributes

- label
- boxes

Static Public Attributes

- weight_image = None
- jaccard_precision
- jaccard_recall
- jaccard_similarity

7.5.1 Detailed Description

A Polygon is a list of Boxes, presumed pairwise non-overlapping.

The whole Polygon may be labeled with a `label` in ('text', 'title', etc.). Polygon also supports image-weighted areas: if `weight_image`, a member of the class on whole, is set to a NewsImage object, then Polygons' area-calculations will be weighted by that object's blackness values.

Author

Samuel Tenka

7.5.2 Constructor & Destructor Documentation

7.5.2.1 def Polygon.Polygon.__init__ (*self*, *boxes* = `[]`, *label* = `'text'`, *string* = `None`)

7.5.3 Member Function Documentation

7.5.3.1 def Polygon.Polygon.__str__ (*self*)

7.5.3.2 def Polygon.Polygon.area (*self*)

7.5.3.3 def Polygon.Polygon.check_valid (*self*)

ensures no overlaps

7.5.3.4 def Polygon.Polygon.create_jaccard (*denom_func*, *docstring* = `None`)

returns a method to compute overlaps of form intersect/denominator

7.5.3.5 def Polygon.Polygon.from_str (*self*, *string*)

7.5.3.6 def Polygon.Polygon.intersect (*self*, *other*)

7.5.4 Member Data Documentation

7.5.4.1 Polygon.Polygon.boxes

7.5.4.2 Polygon.Polygon.jaccard_precision `[static]`

Initial value:

```
= create_jaccard(lambda s,t,i: s.area(),
        docstring=)
```

7.5.4.3 Polygon.Polygon.jaccard_recall [static]

Initial value:

```
= create_jaccard(lambda s,t,i: t.area(),
        docstring=)
```

7.5.4.4 Polygon.Polygon.jaccard_similarity [static]

Initial value:

```
= create_jaccard(lambda s,t,i: s.area()+t.area()-i,
        docstring=)
```

7.5.4.5 Polygon.Polygon.label

7.5.4.6 Polygon.Polygon.weight_image = None [static]

The documentation for this class was generated from the following file:

- Polygon.py

7.6 processXML.ProcessXML Class Reference

class for processing .xml files to get ocr information

Public Member Functions

- def __init__ (self, fname)
- def getTIFdimensions (self)

  returns (height, width) of the .tif data in the xml file.
- def getTextblocks (self)

  create list of textblocks
- def getTextLines (self, textblock)

  loop thru textlines in a textblock
- def getStrings (self, textline)

  return list of strings from a textline
- def getSpaces (self, textline)

  return a list of spaces from a textline
- def writeStSpData (self, wname)

  write the coordinate data of strings and spaces to wname file
- def writeTBData (self, wname)

  write the coordinate data of textblocks to wname file
- def getTBData (self)

  returns a list of (hpos,vpos,width,height) info for each textblock

Public Attributes

- filename

  .xml file
- textblocks

  list of textblocks

7.6.1 Detailed Description

class for processing .xml files to get ocr information

the hierarchy is like this:

```
    page
        textblock
            textline
                    string
                    space
```

Author

Stefan Larson, Panfeng Cao

Todo : combine getTBData & writeTBData (have latter call former)

7.6.2 Constructor & Destructor Documentation

7.6.2.1 def processXML.ProcessXML.__init__ (*self, fname*)

7.6.3 Member Function Documentation

7.6.3.1 def processXML.ProcessXML.getSpaces (*self, textline*)

return a list of spaces from a textline

7.6.3.2 def processXML.ProcessXML.getStrings (*self, textline*)

return list of strings from a textline

7.6.3.3 def processXML.ProcessXML.getTBData (*self*)

returns a list of (hpos,vpos,width,height) info for each textblock

7.6.3.4 def processXML.ProcessXML.getTextblocks (*self*)

create list of textblocks

7.6.3.5 def processXML.ProcessXML.getTextLines (*self,* *textblock*)

loop thru textlines in a textblock

7.6.3.6 def processXML.ProcessXML.getTIFdimensions (*self*)

returns (height, width) of the .tif data in the xml file.

we will use this data to change scales to the .jp2 image size

7.6.3.7 def processXML.ProcessXML.writeStSpData (*self,* *wname*)

write the coordinate data of strings and spaces to wname file

7.6.3.8 def processXML.ProcessXML.writeTBData (*self,* *wname*)

write the coordinate data of textblocks to wname file

7.6.4 Member Data Documentation

7.6.4.1 processXML.ProcessXML.filename

.xml file

7.6.4.2 processXML.ProcessXML.textblocks

list of textblocks

The documentation for this class was generated from the following file:

- processXML.py

7.7 Segmentation.Segmentation Class Reference

A Segmentation is a list of Polygons, presumed pairwise non-overlapping.

Public Member Functions

- def __init__ (self, segments=[ ], string=None, fname=None)
- def __str__ (self)
- def from_str (self, string)
- def pair_recall (self, truth)

  Returns probability that two random points from truth's segments will be classified the same way by truth and self (as belonging either to the same or to different articles)'''.
- def pair_precision (self, truth)
- def pair_fscore (self, truth)
- def jaccard_precision (self, truth, gamma=1.0)
- def jaccard_recall (self, truth, gamma=1.0)
- def jaccard_fscore (self, truth, gamma=1.0)

  the higher the gamma, the more perfection is prized

Public Attributes

- segs

7.7.1 Detailed Description

A Segmentation is a list of Polygons, presumed pairwise non-overlapping.

Author

Samuel Tenka

7.7.2 Constructor & Destructor Documentation

7.7.2.1 def Segmentation.Segmentation.__init__ (*self*, *segments* = `[]`, *string* = `None`, *fname* = `None`)

7.7.3 Member Function Documentation

7.7.3.1 def Segmentation.Segmentation.__str__ (*self*)

7.7.3.2 def Segmentation.Segmentation.from_str (*self*, *string*)

7.7.3.3 def Segmentation.Segmentation.jaccard_fscore (*self*, *truth*, *gamma* = `1.0`)

the higher the gamma, the more perfection is prized

7.7.3.4 def Segmentation.Segmentation.jaccard_precision (*self*, *truth*, *gamma* = `1.0`)

7.7.3.5 def Segmentation.Segmentation.jaccard_recall (*self*, *truth*, *gamma* = `1.0`)

7.7.3.6 def Segmentation.Segmentation.pair_fscore (*self*, *truth*)

7.7.3.7 def Segmentation.Segmentation.pair_precision (*self*, *truth*)

7.7.3.8 def Segmentation.Segmentation.pair_recall (*self*, *truth*)

Returns probability that two random points from truth's segments will be classified the same way by truth and self (as belonging either to the same or to different articles)'''.

Todo : make robust to overlaps...

7.7.4 Member Data Documentation

7.7.4.1 Segmentation.Segmentation.segs

The documentation for this class was generated from the following file:

- Segmentation.py

Chapter 8

File Documentation

8.1 Box.py File Reference

Classes

- class Box.Box

  A box is an axis-aligned rectangle, represented by [mincoors, maxcoors] For us, coordinates are length-2 lists of form [y, x], in units of pixels of the original image.

Namespaces

- Box

8.2 end2end.py File Reference

Classes

- class end2end.EndToEnd

Namespaces

- end2end

Functions

- def end2end.main ()

8.3 EvalOneToMany.py File Reference

Classes

- class EvalOneToMany.EvalOneToMany

Namespaces

- EvalOneToMany

Functions

- def EvalOneToMany.main ()

Variables

- int EvalOneToMany.NUM_CLASS = 3
- float EvalOneToMany.ONETOONE_THRESHOLD = 0.85
- float EvalOneToMany.ONETOMANY_THRESHOLD = 0.1
- list EvalOneToMany.LABELS = ['text', 'title', 'other']

8.4 Image.py File Reference

Classes

- class Image.NewsImage

  Processes image .jpgs and .xmls for evaluation pipeline.

Namespaces

- Image

8.5 imageBS.py File Reference

Namespaces

- imageBS

Variables

- imageBS.f_image = sys.argv[1]
- imageBS.f_xml = sys.argv[2]
- imageBS.f_out = sys.argv[3]
- imageBS.f = f_image.split('/')[1].split('.')[0]
- string imageBS.f_csv = 'images/'
- list imageBS.boxes = [ ]
- imageBS.dat = line.strip('\n')
- dictionary imageBS.seg = {"annotations":[ ]}
- int imageBS.id = 0
- dictionary imageBS.tblnfo = {}
- imageBS.indent

8.6 Latex.py File Reference

Namespaces

- Latex

Functions

- def Latex.replace_dict (string, replacements)
- def Latex.gen_latex (title, fnameout)
- def Latex.gen_pdf (fnametex, fnamepdf)

Variables

- Latex.latex_template = f.read()

8.7 pipe_to_jason.py File Reference

Namespaces

- pipe_to_jason

Variables

- pipe_to_jason.fnamein
- pipe_to_jason.fnameout
- list pipe_to_jason.polys = [l.split('|') for l in f.read().split('\n') if l]
- list pipe_to_jason.labpolys = [(p[0],eval('[%s]'%(','.join(p[1:])))) for p in polys]
- int pipe_to_jason.sh = 6351
- int pipe_to_jason.sw = 3960
- dictionary pipe_to_jason.json

8.8 Polygon.py File Reference

Classes

- class Polygon.Polygon

  A Polygon is a list of Boxes, presumed pairwise non-overlapping.

Namespaces

- Polygon

8.9 processXML.py File Reference

Classes

- class processXML.ProcessXML

  class for processing .xml files to get ocr information

Namespaces

- processXML

8.10 readJSON.py File Reference

Namespaces

- readJSON

Functions

- def readJSON.seg_from_json (fname, gt_flag)

  This script reads in a json file of this format.

8.11 README.md File Reference

8.12 Segmentation.py File Reference

Classes

- class Segmentation.Segmentation

  A Segmentation is a list of Polygons, presumed pairwise non-overlapping.

Namespaces

- Segmentation

8.13 textblocksBS.py File Reference

Namespaces

- textblocksBS

Functions

- def textblocksBS.size_of_image (imname)

Variables

- textblocksBS.f_out_folder = sys.argv[1]
- textblocksBS.f_image = sys.argv[2]

  <image> is a jp2 or jpg
- textblocksBS.f_xml = sys.argv[3]

  <xml> is xml file associated with the image
- textblocksBS.f_out = sys.argv[4].split('/')[-1]
- textblocksBS.pxml = processXML.ProcessXML(f_xml)
- textblocksBS.tbList = pxml.getTBData()
- textblocksBS.h
- textblocksBS.w
- textblocksBS.hj
- textblocksBS.wj
- textblocksBS.hs
- textblocksBS.ws
- dictionary textblocksBS.seg = {"annotations":[ ]}
- int textblocksBS.id = 0
- dictionary textblocksBS.tbInfo = {}
- textblocksBS.f
- textblocksBS.indent

Index