

Automatic Recognition of Historical Newspaper Content

Generated by Doxygen 1.8.12

Contents

1	mdp-newspaper-segmentation	1
2	Todo List	3
3	Namespace Index	5
3.1	Namespace List	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	Box Namespace Reference	11
6.2	end2end Namespace Reference	11
6.2.1	Function Documentation	11
6.2.1.1	main()	11
6.3	EvalOneToMany Namespace Reference	11
6.3.1	Function Documentation	12
6.3.1.1	main()	12
6.3.2	Variable Documentation	12
6.3.2.1	LABELS	12
6.3.2.2	NUM_CLASS	12
6.3.2.3	ONETOMANY_THRESHOLD	12
6.3.2.4	ONETOONE_THRESHOLD	12

6.4	Image Namespace Reference	12
6.5	imageBS Namespace Reference	12
6.5.1	Detailed Description	12
6.5.2	Variable Documentation	13
6.5.2.1	boxes	13
6.5.2.2	dat	13
6.5.2.3	f	13
6.5.2.4	f_csv	13
6.5.2.5	f_image	13
6.5.2.6	f_out	13
6.5.2.7	f_xml	13
6.5.2.8	id	13
6.5.2.9	indent	13
6.5.2.10	seg	13
6.5.2.11	tblInfo	13
6.6	Latex Namespace Reference	13
6.6.1	Function Documentation	14
6.6.1.1	gen_latex(title, fframeout)	14
6.6.1.2	gen_pdf(fnametex, fframepdf)	14
6.6.1.3	replace_dict(string, replacements)	14
6.6.2	Variable Documentation	14
6.6.2.1	latex_template	14
6.7	pipe_to_json Namespace Reference	14
6.7.1	Variable Documentation	14
6.7.1.1	fframein	14
6.7.1.2	fframeout	14
6.7.1.3	json	14
6.7.1.4	labpolys	15
6.7.1.5	polys	15
6.7.1.6	sh	15

6.7.1.7	sw	15
6.8	Polygon Namespace Reference	15
6.9	processXML Namespace Reference	15
6.10	readJSON Namespace Reference	15
6.10.1	Function Documentation	15
6.10.1.1	seg_from_json(fname, gt_flag)	15
6.11	Segmentation Namespace Reference	16
6.12	textblocksBS Namespace Reference	16
6.12.1	Function Documentation	16
6.12.1.1	size_of_image(imname)	16
6.12.2	Variable Documentation	16
6.12.2.1	f	16
6.12.2.2	f_image	16
6.12.2.3	f_out	17
6.12.2.4	f_out_folder	17
6.12.2.5	f_xml	17
6.12.2.6	h	17
6.12.2.7	hj	17
6.12.2.8	hs	17
6.12.2.9	id	17
6.12.2.10	indent	17
6.12.2.11	pxml	17
6.12.2.12	seg	17
6.12.2.13	tblInfo	17
6.12.2.14	tbList	17
6.12.2.15	w	17
6.12.2.16	wj	17
6.12.2.17	ws	17

7 Class Documentation	19
7.1 Box.Box Class Reference	19
7.1.1 Detailed Description	20
7.1.2 Constructor & Destructor Documentation	20
7.1.2.1 <code>__init__(self, coor0=None, coor1=None, string=None)</code>	20
7.1.3 Member Function Documentation	20
7.1.3.1 <code>__str__(self)</code>	20
7.1.3.2 <code>area(self, image=None)</code>	20
7.1.3.3 <code>center(self)</code>	21
7.1.3.4 <code>disconnect_distance(self, other)</code>	21
7.1.3.5 <code>flatten(self)</code>	21
7.1.3.6 <code>from_corners(self, coorA, coorB)</code>	21
7.1.3.7 <code>from_point(self, coor)</code>	21
7.1.3.8 <code>from_str(self, string)</code>	21
7.1.3.9 <code>intersect(self, other)</code>	21
7.1.3.10 <code>join(self, other)</code>	21
7.1.3.11 <code>overlaps(self, other)</code>	21
7.1.4 Member Data Documentation	22
7.1.4.1 <code>points</code>	22
7.2 end2end.EndToEnd Class Reference	22
7.2.1 Detailed Description	22
7.2.2 Constructor & Destructor Documentation	22
7.2.2.1 <code>__init__(self, config_filename='config.txt')</code>	22
7.2.3 Member Function Documentation	22
7.2.3.1 <code>collect_data(self)</code>	22
7.2.3.2 <code>evaluate(self)</code>	23
7.2.3.3 <code>generate_report(self)</code>	23
7.2.3.4 <code>plot_performance_curve(self)</code>	23
7.2.3.5 <code>segment(self)</code>	23
7.2.4 Member Data Documentation	23

7.2.4.1	eval_out_path	23
7.2.4.2	eval_results	23
7.2.4.3	files	23
7.2.4.4	implementations	23
7.2.4.5	metrics	23
7.2.4.6	seg_out_path	23
7.3	EvalOneToMany.EvalOneToMany Class Reference	23
7.3.1	Constructor & Destructor Documentation	24
7.3.1.1	__init__(self, out_folder, seg_folder, gt_path, seg_path, img_path=None, xml_path=None, imp_name=None)	24
7.3.2	Member Function Documentation	24
7.3.2.1	evaluate(self)	24
7.3.2.2	save_output(self, record)	24
7.3.3	Member Data Documentation	24
7.3.3.1	eval_history	24
7.3.3.2	ground_truth	24
7.3.3.3	gt_path	24
7.3.3.4	history_path	24
7.3.3.5	img_path	24
7.3.3.6	out_folder	24
7.3.3.7	seg_path	24
7.3.3.8	seg_to_eval	24
7.3.3.9	xml_path	24
7.4	Image.NewslImage Class Reference	24
7.4.1	Detailed Description	25
7.4.2	Constructor & Destructor Documentation	25
7.4.2.1	__init__(self, root, gamma=gamma_default)	25
7.4.3	Member Function Documentation	25
7.4.3.1	get_total_blackness(self, y0, x0, y1, x1)	25
7.4.3.2	read_blackness(self)	26
7.4.4	Member Data Documentation	26

7.4.4.1	blacknesses	26
7.4.4.2	filenames	26
7.4.4.3	gamma	26
7.4.4.4	gamma_default	26
7.5	Polygon.Polygon Class Reference	26
7.5.1	Detailed Description	27
7.5.2	Constructor & Destructor Documentation	27
7.5.2.1	__init__(self, boxes=[], label='text', string=None)	27
7.5.3	Member Function Documentation	27
7.5.3.1	__str__(self)	27
7.5.3.2	area(self)	27
7.5.3.3	check_valid(self)	27
7.5.3.4	create_jaccard(denom_func, docstring=None)	27
7.5.3.5	from_str(self, string)	27
7.5.3.6	intersect(self, other)	27
7.5.4	Member Data Documentation	27
7.5.4.1	boxes	27
7.5.4.2	jaccard_precision	27
7.5.4.3	jaccard_recall	28
7.5.4.4	jaccard_similarity	28
7.5.4.5	label	28
7.5.4.6	weight_image	28
7.6	processXML.ProcessXML Class Reference	28
7.6.1	Detailed Description	29
7.6.2	Constructor & Destructor Documentation	29
7.6.2.1	__init__(self, fname)	29
7.6.3	Member Function Documentation	29
7.6.3.1	getSpaces(self, textline)	29
7.6.3.2	getStrings(self, textline)	29
7.6.3.3	getTBData(self)	29

7.6.3.4	getTextblocks(self)	30
7.6.3.5	getTextLines(self, textblock)	30
7.6.3.6	getTIFdimensions(self)	30
7.6.3.7	writeStSpData(self, wname)	30
7.6.3.8	writeTBData(self, wname)	30
7.6.4	Member Data Documentation	30
7.6.4.1	filename	30
7.6.4.2	textblocks	30
7.7	Segmentation.Segmentation Class Reference	30
7.7.1	Detailed Description	31
7.7.2	Constructor & Destructor Documentation	31
7.7.2.1	__init__(self, segments=[], string=None, fname=None)	31
7.7.3	Member Function Documentation	31
7.7.3.1	__str__(self)	31
7.7.3.2	from_str(self, string)	31
7.7.3.3	jaccard_fscore(self, truth, gamma=1.0)	31
7.7.3.4	jaccard_precision(self, truth, gamma=1.0)	32
7.7.3.5	jaccard_recall(self, truth, gamma=1.0)	32
7.7.3.6	pair_fscore(self, truth)	32
7.7.3.7	pair_precision(self, truth)	32
7.7.3.8	pair_recall(self, truth)	32
7.7.4	Member Data Documentation	32
7.7.4.1	segs	32
8	File Documentation	33
8.1	Box.py File Reference	33
8.2	end2end.py File Reference	33
8.3	EvalOneToMany.py File Reference	33
8.4	Image.py File Reference	34
8.5	imageBS.py File Reference	34
8.6	Latex.py File Reference	35
8.7	pipe_to_json.py File Reference	35
8.8	Polygon.py File Reference	35
8.9	processXML.py File Reference	36
8.10	readJSON.py File Reference	36
8.11	README.md File Reference	36
8.12	Segmentation.py File Reference	36
8.13	textblocksBS.py File Reference	36
	Index	39

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:

```
1 {  
2     "Metrics": ["EvalOneToMany.py"],  
3     "Implementations": ["textblocksBS.py"],  
4     "Data": ["/path/to/data_and_groundtruth/"],  
5     "Outpath": ["/path/to/segmentation_output", "/path/to/evaluation_output"]  
6 }
```

2. Running the End-to-End system

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

```
1 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:

Box	11
end2end	11
EvalOneToMany	11
Image	12
imageBS	12
Latex	13
pipe_to_jason	14
Polygon	15
processXML	15
readJSON	15
Segmentation	16
textblocksBS	16

Chapter 4

Class Index

4.1 Class List

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

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	19
end2end.EndToEnd	22
EvalOneToMany.EvalOneToMany	23
Image.NewsImage	Processes image .jpgs and .xmls for evaluation pipeline	24
Polygon.Polygon	A Polygon is a list of Boxes, presumed pairwise non-overlapping	26
processXML.ProcessXML	Class for processing .xml files to get ocr information	28
Segmentation.Segmentation	A Segmentation is a list of Polygons, presumed pairwise non-overlapping	30

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Box.py	33
end2end.py	33
EvalOneToMany.py	33
Image.py	34
imageBS.py	34
Latex.py	35
pipe_to_jason.py	35
Polygon.py	35
processXML.py	36
readJSON.py	36
Segmentation.py	36
textblocksBS.py	36

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 `tblInfo` = {}
- `indent`

6.5.1 Detailed Description

python 2.7

6.5.2 Variable Documentation

6.5.2.1 `list imageBS.bboxes = []`

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, frameout)
- 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_json 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_json.fnamein`

Author

Anonymous

6.7.1.2 `pipe_to_json.fnameout`

6.7.1.3 dictionary `pipe_to_json.json`

Initial value:

```
1 = {'annotations': [
2   {'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}
3   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.tbInfo = {}`

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.NewslImage.__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.NewslImage.get_total_blackness(self, y0, x0, y1, x1)`

Todo : fix (should be four terms)

7.4.3.2 `def Image.NewslImage.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.NewslImage.blacknesses`

7.4.4.2 `Image.NewslImage.filenames`

7.4.4.3 `Image.NewslImage.gamma`

7.4.4.4 `float Image.NewslImage.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.bboxes`

7.5.4.2 `Polygon.Polygon.jaccard_precision` `[static]`

Initial value:

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

7.5.4.3 Polygon.Polygon.jaccard_recall [static]

Initial value:

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

7.5.4.4 Polygon.Polygon.jaccard_similarity [static]

Initial value:

```
1 = create_jaccard(lambda s,t,i: s.area()+t.area()-i,
2     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.bboxes = []`
- `imageBS.dat = line.strip("\n")`
- `dictionary imageBS.seg = {"annotations":[]}`
- `int imageBS.id = 0`
- `dictionary imageBS.tblInfo = {}`
- `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

- `__init__`
 - `Box::Box`, 20
 - `end2end::EndToEnd`, 22
 - `EvalOneToMany::EvalOneToMany`, 24
 - `Image::NewsImage`, 25
 - `Polygon::Polygon`, 27
 - `processXML::ProcessXML`, 29
 - `Segmentation::Segmentation`, 31
- `__str__`
 - `Box::Box`, 20
 - `Polygon::Polygon`, 27
 - `Segmentation::Segmentation`, 31
- `area`
 - `Box::Box`, 20
 - `Polygon::Polygon`, 27
- `blacknesses`
 - `Image::NewsImage`, 26
- `Box`, 11
- `Box.Box`, 19
- `Box.py`, 33
- `Box::Box`
 - `__init__`, 20
 - `__str__`, 20
 - `area`, 20
 - `center`, 20
 - `disconnect_distance`, 21
 - `flatten`, 21
 - `from_corners`, 21
 - `from_point`, 21
 - `from_str`, 21
 - `intersect`, 21
 - `join`, 21
 - `overlaps`, 21
 - `points`, 22
- `boxes`
 - `imageBS`, 13
 - `Polygon::Polygon`, 27
- `center`
 - `Box::Box`, 20
- `check_valid`
 - `Polygon::Polygon`, 27
- `collect_data`
 - `end2end::EndToEnd`, 22
- `create_jaccard`
 - `Polygon::Polygon`, 27
- `dat`
 - `imageBS`, 13
 - `disconnect_distance`
 - `Box::Box`, 21
 - `end2end`, 11
 - `main`, 11
 - `end2end.EndToEnd`, 22
 - `end2end.py`, 33
 - `end2end::EndToEnd`
 - `__init__`, 22
 - `collect_data`, 22
 - `eval_out_path`, 23
 - `eval_results`, 23
 - `evaluate`, 22
 - `files`, 23
 - `generate_report`, 23
 - `implementations`, 23
 - `metrics`, 23
 - `plot_performance_curve`, 23
 - `seg_out_path`, 23
 - `segment`, 23
 - `eval_history`
 - `EvalOneToMany::EvalOneToMany`, 24
 - `eval_out_path`
 - `end2end::EndToEnd`, 23
 - `eval_results`
 - `end2end::EndToEnd`, 23
 - `EvalOneToMany`, 11
 - `LABELS`, 12
 - `main`, 12
 - `NUM_CLASS`, 12
 - `ONETOMANY_THRESHOLD`, 12
 - `ONETOONE_THRESHOLD`, 12
 - `EvalOneToMany.EvalOneToMany`, 23
 - `EvalOneToMany.py`, 33
 - `EvalOneToMany::EvalOneToMany`
 - `__init__`, 24
 - `eval_history`, 24
 - `evaluate`, 24
 - `ground_truth`, 24
 - `gt_path`, 24
 - `history_path`, 24
 - `img_path`, 24
 - `out_folder`, 24
 - `save_output`, 24
 - `seg_path`, 24
 - `seg_to_eval`, 24
 - `xml_path`, 24
 - `evaluate`
 - `end2end::EndToEnd`, 22

- EvalOneToMany::EvalOneToMany, 24
- f
 - imageBS, 13
 - textblocksBS, 16
- f_csv
 - imageBS, 13
- f_image
 - imageBS, 13
 - textblocksBS, 16
- f_out
 - imageBS, 13
 - textblocksBS, 16
- f_out_folder
 - textblocksBS, 17
- f_xml
 - imageBS, 13
 - textblocksBS, 17
- filename
 - processXML::ProcessXML, 30
- filenames
 - Image::NewsImage, 26
- files
 - end2end::EndToEnd, 23
- flatten
 - Box::Box, 21
- fnamein
 - pipe_to_jason, 14
- fnameout
 - pipe_to_jason, 14
- from_corners
 - Box::Box, 21
- from_point
 - Box::Box, 21
- from_str
 - Box::Box, 21
 - Polygon::Polygon, 27
 - Segmentation::Segmentation, 31
- gamma
 - Image::NewsImage, 26
- gamma_default
 - Image::NewsImage, 26
- gen_latex
 - Latex, 14
- gen_pdf
 - Latex, 14
- generate_report
 - end2end::EndToEnd, 23
- get_total_blackness
 - Image::NewsImage, 25
- getSpaces
 - processXML::ProcessXML, 29
- getStrings
 - processXML::ProcessXML, 29
- getTBData
 - processXML::ProcessXML, 29
- getTIFdimensions
 - processXML::ProcessXML, 30
- getTextLines
 - processXML::ProcessXML, 30
- getTextblocks
 - processXML::ProcessXML, 29
- ground_truth
 - EvalOneToMany::EvalOneToMany, 24
- gt_path
 - EvalOneToMany::EvalOneToMany, 24
- h
 - textblocksBS, 17
- history_path
 - EvalOneToMany::EvalOneToMany, 24
- hj
 - textblocksBS, 17
- hs
 - textblocksBS, 17
- id
 - imageBS, 13
 - textblocksBS, 17
- Image, 12
- Image.NewsImage, 24
- Image.py, 34
- Image::NewsImage
 - __init__, 25
 - blacknesses, 26
 - filenames, 26
 - gamma, 26
 - gamma_default, 26
 - get_total_blackness, 25
 - read_blackness, 25
- imageBS.py, 34
- imageBS, 12
 - boxes, 13
 - dat, 13
 - f, 13
 - f_csv, 13
 - f_image, 13
 - f_out, 13
 - f_xml, 13
 - id, 13
 - indent, 13
 - seg, 13
 - tblInfo, 13
- img_path
 - EvalOneToMany::EvalOneToMany, 24
- implementations
 - end2end::EndToEnd, 23
- indent
 - imageBS, 13
 - textblocksBS, 17
- intersect
 - Box::Box, 21
 - Polygon::Polygon, 27
- jaccard_fscore
 - Segmentation::Segmentation, 31
- jaccard_precision

- Polygon::Polygon, [27](#)
 - Segmentation::Segmentation, [31](#)
- jaccard_recall
 - Polygon::Polygon, [27](#)
 - Segmentation::Segmentation, [32](#)
- jaccard_similarity
 - Polygon::Polygon, [28](#)
- join
 - Box::Box, [21](#)
- json
 - pipe_to_jason, [14](#)
- LABELS
 - EvalOneToMany, [12](#)
- label
 - Polygon::Polygon, [28](#)
- labpolys
 - pipe_to_jason, [14](#)
- Latex, [13](#)
 - gen_latex, [14](#)
 - gen_pdf, [14](#)
 - latex_template, [14](#)
 - replace_dict, [14](#)
- Latex.py, [35](#)
- latex_template
 - Latex, [14](#)
- main
 - end2end, [11](#)
 - EvalOneToMany, [12](#)
- metrics
 - end2end::EndToEnd, [23](#)
- NUM_CLASS
 - EvalOneToMany, [12](#)
- ONETOMANY_THRESHOLD
 - EvalOneToMany, [12](#)
- ONETOONE_THRESHOLD
 - EvalOneToMany, [12](#)
- out_folder
 - EvalOneToMany::EvalOneToMany, [24](#)
- overlaps
 - Box::Box, [21](#)
- pair_fscore
 - Segmentation::Segmentation, [32](#)
- pair_precision
 - Segmentation::Segmentation, [32](#)
- pair_recall
 - Segmentation::Segmentation, [32](#)
- pipe_to_jason, [14](#)
 - fnamein, [14](#)
 - fnameout, [14](#)
 - json, [14](#)
 - labpolys, [14](#)
 - polys, [15](#)
 - sh, [15](#)
 - sw, [15](#)
- pipe_to_jason.py, [35](#)
- plot_performance_curve
 - end2end::EndToEnd, [23](#)
- points
 - Box::Box, [22](#)
- Polygon, [15](#)
- Polygon.Polygon, [26](#)
- Polygon.py, [35](#)
- Polygon::Polygon
 - __init__, [27](#)
 - __str__, [27](#)
 - area, [27](#)
 - boxes, [27](#)
 - check_valid, [27](#)
 - create_jaccard, [27](#)
 - from_str, [27](#)
 - intersect, [27](#)
 - jaccard_precision, [27](#)
 - jaccard_recall, [27](#)
 - jaccard_similarity, [28](#)
 - label, [28](#)
 - weight_image, [28](#)
- polys
 - pipe_to_jason, [15](#)
- processXML.ProcessXML, [28](#)
- processXML.py, [36](#)
- processXML::ProcessXML
 - __init__, [29](#)
 - filename, [30](#)
 - getSpaces, [29](#)
 - getStrings, [29](#)
 - getTBData, [29](#)
 - getTIFdimensions, [30](#)
 - getTextLines, [30](#)
 - getTextblocks, [29](#)
 - textblocks, [30](#)
 - writeStSpData, [30](#)
 - writeTBData, [30](#)
- processXML, [15](#)
- pxml
 - textblocksBS, [17](#)
- README.md, [36](#)
- read_blackness
 - Image::NewsImage, [25](#)
- readJSON.py, [36](#)
- readJSON, [15](#)
 - seg_from_json, [15](#)
- replace_dict
 - Latex, [14](#)
- save_output
 - EvalOneToMany::EvalOneToMany, [24](#)
- seg
 - imageBS, [13](#)
 - textblocksBS, [17](#)
- seg_from_json
 - readJSON, [15](#)
- seg_out_path

- end2end::EndToEnd, [23](#)
- seg_path
 - EvalOneToMany::EvalOneToMany, [24](#)
- seg_to_eval
 - EvalOneToMany::EvalOneToMany, [24](#)
- segment
 - end2end::EndToEnd, [23](#)
- Segmentation, [16](#)
- Segmentation.py, [36](#)
- Segmentation.Segmentation, [30](#)
- Segmentation::Segmentation
 - __init__, [31](#)
 - __str__, [31](#)
 - from_str, [31](#)
 - jaccard_fscore, [31](#)
 - jaccard_precision, [31](#)
 - jaccard_recall, [32](#)
 - pair_fscore, [32](#)
 - pair_precision, [32](#)
 - pair_recall, [32](#)
 - segs, [32](#)
- segs
 - Segmentation::Segmentation, [32](#)
- sh
 - pipe_to_json, [15](#)
- size_of_image
 - textblocksBS, [16](#)
- sw
 - pipe_to_json, [15](#)
- tblInfo
 - imageBS, [13](#)
 - textblocksBS, [17](#)
- tbList
 - textblocksBS, [17](#)
- textblocks
 - processXML::ProcessXML, [30](#)
- textblocksBS.py, [36](#)
- textblocksBS, [16](#)
 - f, [16](#)
 - f_image, [16](#)
 - f_out, [16](#)
 - f_out_folder, [17](#)
 - f_xml, [17](#)
 - h, [17](#)
 - hj, [17](#)
 - hs, [17](#)
 - id, [17](#)
 - indent, [17](#)
 - pxml, [17](#)
 - seg, [17](#)
 - size_of_image, [16](#)
 - tblInfo, [17](#)
 - tbList, [17](#)
 - w, [17](#)
 - wj, [17](#)
 - ws, [17](#)
- textblocksBS, [17](#)
 - weight_image
 - Polygon::Polygon, [28](#)
 - wj
 - textblocksBS, [17](#)
 - writeStSpData
 - processXML::ProcessXML, [30](#)
 - writeTBData
 - processXML::ProcessXML, [30](#)
 - ws
 - textblocksBS, [17](#)
- xml_path
 - EvalOneToMany::EvalOneToMany, [24](#)

w