**Natural Language Processing for PDF/TIFF/Image Documents  
SPLITTER Module  
High Precision PDF Page Splitting/OCR/Text Extraction  
Technical Specification, v1.4**

# 1 Document

## 1.1 Document Overview

The document classifier contains the following primary classes, and their relationships:

* Document – This is the base class for the representation of a stored document. The constructor for the class object takes as parameters the stored path to the document, optionally a directory path for storing extracted pages and text, and optionally an event completion handler when processing the document asynchronously.

document = Document(“/somedir/mydocument.pdf”, “/mypages/mydocument”)

The constructors calls the \_exists() and \_collate() private methods for the specified document.

* <Category>Document – This is a derived class for the representation of a category specific stored, such as a medical document. It inherits from the Document base class.
* Page – This is a base class for the representation of an extracted page from the document. The document class contains a list (index) of the extracted pages as Page objects.

**List**

**Page (Base Class)**

**Document (Base Class)**

**Page (Base Class)**

**<Category>Document (Derived Class)**

**Page (Base Class)**

**Fig. 1a High Level view of Document Class Object Relationships**

## 1.2 Document Properties

The document class contains the following properties:

* document – The path to the stored document.
* name – The root name of the document (e.g., /mydir/mydocument.pdf -> mydocument).
* text – The raw extracted text; returned as a list, one entry per page.
* pages – Index of the page objects, one per page.
* dir – Location to store extracted pages and text.
* size – The byte size of the document.
* type – The file format of the document (e.g., pdf).
* scanned – If document is a scanned PDF, TIFF facsimile or image captured.
* time – The elapsed time in seconds for collating the document.
* class – The document classification.

The document class contains the following static variables:

* RESOLUTION – The image resolution when converting PDF to PNG for OCR (default 300).

## 1.3 Document Overridden Operators

The following operators have their implementations (inherited from the base Object) class overridden:

* len() – The \_\_len\_\_() method is overridden to return the number of pages in the document.
* str() – The \_\_str\_\_() method is overridden to return the document classification (label).
* [] – The \_\_getitem\_\_() method is overridden to return the page object at the specified index (i.e., page number – 1). The \_\_setitem\_\_() method is overridden to replace the page object at the specified index (i.e., page number – 1).
* += The \_\_iadd\_\_() method is overridden to add a page object (i.e., append).

## 1.4 Document Public Methods

The document class contains of following public methods:

* There are no public methods

## 1.5 Document Private Methods

The document class contains of following private methods:

* \_exists() – This method checks if the document exists at the specified stored path. If not, a FileNotFound exception is thrown.
* \_collate() – This method performs the collation task, which includes:
  + Determines the number of pages in the document.
  + Splits the document into individual pages, where each page is individually stored in the same format as the document. The pages are named as follows:

<name><pageno>.<suffix>

Each page is stored in the subdirectory specified by the property dir. If dir is None, then the page is stored in the same directory where program is ran; otherwise, if the subdirectory does not exist, it is created.

* + If the page is a scanned PDF page, the scanned image is extracted and saved as a PNG image. The PNG image is then OCR’d to convert to text.

<name><pageno>.png

* + If the page is a TIFF facsimile, the TIFF image is then OCR’d to convert to text.

<name><pageno>.tif

* + If the page is an image capture (e.g. camera capture), the captured image (e.g., JPG) is then OCR’d to convert to text.

<name><pageno>.jpg

* + Extracts the raw text from the page , where each page is individually stored in a raw text format. The pages are named as follows:

<name><pageno>.txt

Each page is stored in the subdirectory specified by the property dir. If dir is None, then the page is stored in the same directory where program is ran.

* + Create a Page object for each page and adds them to the pages index property.
  + If the document format is raw text, then:
    - Treats as a single page.
    - Stores only a single page text file.
  + If the document format is PDF, then page splitting and extraction of the raw text per page is done with the open source version of Ghostscript. If the document is a scanned PDF, the image is extracted and converted to PNG using Ghostscript and then OCR’d using open source Tesseract.
  + If the document format is TIFF, then page splitting is done with the open source Magick and then OCR’d using open source Tesseract.
* \_async() – This method performs asynchronous processing of the \_collate() function, when the optional ehandler parameter to the constructor is not None. When processing is completed, the ehandler parameter value is called as a function to signal completion of the processing, and the document object is passed as a parameter.

# 2 Page

## 2.1 Page Overview

The page classifier contains the following primary classes, and their relationships:

* Page – This is a base class for the representation of an extracted page from a document. The constructor for the class object takes optionally as parameters the stored path to the page, and the extracted raw text.

page = Page( ‘/mypages/page1.pdf’, ‘some text’)

* <Category>Page – This is a derived class for representation of an extracted page from a category specific document, such as a medical document.
* Words – This is a base class for representation of the text as NLP preprocessed list of words.

**Words (Base Class)**

**Page (Base Class)**

**<Category>Page (Derived Class)**

**Fig. 2a High Level view of Page Class Object Relationships**

## 2.2 Page Properties

The page class contains the following properties:

* path – The path to the stored document.
* text – The raw extracted text.
* class – The page classification
* size – The byte size of the raw extracted text.
* pageno – The page number in the parent document
* words – The page text converted to a sequenced tokenized set of words which has been NLP preprocessed.
  + Acronym, Number, Dates, Date of Birth, Telephone, SSN, and Proper Names Detection
  + Punctuation Removal
  + Lowercased
  + Stopword Removal
  + Word Stemming

## 2.3 Page Overwritten Operators

The following operators have their implementations (inherited from the base Object) class overridden:

* len() – The \_\_len\_\_() method is overridden to return the number of tokenized words in the page.
* str() – The \_\_str\_\_() method is overridden to return the page classification (label).
* += The \_\_iadd\_\_() method is overridden to add text to the page (append).

## 2.4 Page Private Methods

The page class contains of following private methods:

* There are no private methods.

## 2.5 Page Public Methods

The page class contains of following public methods:

* store() – Store the NLP tokenized list to a file. The tokenized list is written in JSON format.
* load() – Load the NLP tokenized list from a file into the words list.

# APPENDIX I: Updates

*v1.1*

1. Added time property.
2. Added scanned property.
3. Added support for TIFF and JPG/PNG.

*V1.3*

1. Add direct read of PDF resource element to determine if scanned page.
2. Fix not detecting scanned PDF if text extraction produced noise.

*V1.4*

1. Add pageno property to Page class.
2. Add methods store() and load() to Page class to store/load NLP tokenized words to file.

# APPENDIX II: Anticipated Engineering

The following has been identified as enhancement/issues to be addressed in subsequent update:

1. What does it mean to add text to a document.
2. Break raw text into pages for > 50 lines
3. Refactor page counting for faster performance
4. Add page split endpoint for streaming interface and URL
5. Add more pdf test files

**Proprietary and Confidential Information  
Copyright ©2018, Epipog, All Rights Reserved**