

PDF Extraction Toolkit – Version 0.9

The **PDF Extraction Toolkit** (formerly “PDF Analyser”) is a framework for performing document analysis of PDF files and creating custom conversion methods. **GraphWrap**, a system for graph-based wrapping, or semi-automatic data extraction, from PDF files, is also included within the PDF Extraction Toolkit. The main toolkit (including GraphWrap) is released under the Apache licence, which allows it to be freely incorporated into proprietary software.

A GUI is also included, built upon the “XMillum” library, which enables the results of the document analysis process to be visualized. Also, an interactive graph visualization is provided to view the graph structures created by the system and allow the interactive creation and testing of graph-based wrappers on PDF documents. This GUI is released under the GPL licence.

Getting started

Version 0.9 (2011-02-22) of the PDF Extraction Toolkit consists of three project downloads:

- **PDF Analyser** – the backend, including GraphWrap, published under the Apache licence
- **PDF Analyser GUI** – the GUI, published under the GPL licence
- **TouchGraph-Modified** – a minor modification made to the “TouchGraph” library, also published under the Apache licence.

Both **PDF Analyser** and **PDF Analyser GUI** are standalone projects. The compiled **PDF Analyser JAR** is included within **PDF Analyser GUI** (but not vice versa).

The compiled **TouchGraph-Modified JAR** is included within the project **PDF Analyser**; you only need to download it for reference or if you wish to modify the TouchGraph library further.

Code structure

Package structure:

- **at.ac.tuwien.dbai.pdfwrap**
 - contains the main classes **ProcessFile** (to process a PDF) and **GraphMatcher** (to perform graph-based text extraction)
- **at.ac.tuwien.dbai.pdfwrap.model.document**
 - contains the elements for the document model representation:
 - **GenericSegment** is the base class of a rectangular block
 - **TextSegment** adds text and font fields to **GenericSegment**
 - **CompositeSegment** is the base class of a block containing a **List<? Extends GenericSegment>** of sub blocks
 - a document is a **List<Page>**
 - **Page** is a **CompositeSegment<GenericSegment>**
 - **TextBlock** is the granular block as returned by the segmentation algorithm
 - a **TextBlock** is built up of **TextLines**; single lines of text (enforced by generics)

- a **TextLine** is built up of **LineFragments**; usually just a single one, unless font changes occur within the line (a **LineFragment** may not have any font changes)
- a **LineFragment** consists of **TextFragments** (single COS (sub)instructions, typically 2-3 characters long, sometimes only 1 character)
- a **TextFragment** consists of one or more **CharSegments**
- Graphical items include **Images** (bitmap images), **LineSegments** (straight lines) and **RectSegments** (rectangular segments; filled or unfilled). Currently, there is no hierarchy imposed.
- **at.ac.tuwien.dbai.pdfwrap.model.graph**
 - **AdjacencyGraph** – neighbourhood graph built up from the document model plus **AdjacencyEdges** in the four directions of the compass
 - for segmentation and document analysis, this graph is used
 - **DocumentGraph**: for graph display and graph-based information extraction, a simpler model is used. Currently, only one single hierarchical level is supported.
 - **DocumentGraph(AdjacencyGraph ag)** – constructor creates a **DocumentGraph** out of an **AdjacencyGraph**
 - **DocNode** – essentially a **TextSegment**; includes matching criteria; extends TouchGraph **Node** class
 - **DocEdge** – joins two **DocNodes**; includes relation string (currently only adjacency relations are used) includes matching criteria; extends TouchGraph **Edge** class
- **at.ac.tuwien.dbai.pdfwrap.analysis**
 - classes, which perform document analysis, i.e.: grouping of blocks, segmentation, recognition of larger structures, etc.
- **at.ac.tuwien.dbai.pdfwrap.comparators**
 - comparators
- **at.ac.tuwien.dbai.pdfwrap.gui**
 - **EdgeSegment** – used to display edges in the XMillum view
- **at.ac.tuwien.dbai.pdfwrap.exceptions**
 - **DocumentProcessingException**
- **at.ac.tuwien.dbai.pdfwrap.operator**
 - location of the methods overriding PDFBox's native methods that are carried out when a particular COS operator is encountered
 - **Resources/PDFObjectExtractor.properties** is the mapping from the operator to operator method – please note the **NotImplemented** operators!
- **at.ac.tuwien.dbai.pdfwrap.pdfread**
 - classes for reading in the PDF:
 - **PDFPage** – intermediate representation of a page before it is analysed
 - **PDFObjectExtractor** – methods for extracting low-level objects from the PDF from the COS stream
- **at.ac.tuwien.dbai.pdfwrap.utils**
 - utility methods

Coordinates

Throughout the document model, PDF coordinates are used (72ppi; (0,0) at bottom left). For output to XMillum, these coordinates are converted to screen coordinates ((0,0) at top left).

Main methods:

- `at.ac.tuwien.dbai.pdfwrap.ProcessFile.main`
- `at.ac.tuwien.dbai.pdfwrap.GraphMatcher.main`

Processing a PDF using the command line

See the file **process-file.sh**. Running the command without parameters will print out the usage syntax.

Processing a PDF using the Java method

in class `at.ac.tuwien.dbai.pdfwrap.ProcessFile`:

```
public static List<Page> processPDF(byte[] theFile, int processType, boolean rulingLines,
    boolean ignoreSpaces, int startPage, int endPage, String encoding, String password,
    int maxIterations, List<AdjacencyGraph<GenericSegment>> adjGraphList, boolean GUI)
    throws DocumentProcessingException
```

theFile	the input file
processType	PageProcessor.PP_BLOCK – the blocks after segmentation or PageProcessor.PP_MERGED_LINES for individual lines of text as TextBlocks (note: the TextBlocks include the individual lines anyway)
rulingLines	usually set to true ; if false , then ruling lines will be ignored in the segmentation process
ignoreSpaces	usually set to false ; might be necessary for monospaced text
startPage and endPage	if set to -1 , will use the first and last page respectively
encoding	default is "" (or null)
password	default is "" (or null) – unless a password is required to open the file
maxIterations	default is 0 ; other values for testing purposes only
adjGraphList	default null ; include an empty list if you wish to receive the generated graphs
GUI	default false ; true only if called by the GUI

Extracting text

Look at the **TextBlock** objects that have been returned and their **getText()** methods. Note that a **TextSegment** (incl. **TextBlock**) can only have one font, fontsize and style. To access font changes within the block, you need to go down to the **LineSegment** level. See the **TextBlock.addAsXHTML()** method.

Converting PDF to (X)HTML

```
processFile.sh input-file.pdf output-file.xml -xhtml [-startPage 1 -endPage 1 -password pw]
```

Pages are processed independently of each other and in sequence. Each individual **TextBlock** is represented as a HTML `<p>` element (see **TextBlock.addAsXHTML()** method). Graphic objects are currently ignored.

GraphWrap

Please see the following URL for information about the GraphWrap system:

<http://www.tamirhassan.com/graphwrap.html>

You can find a link to an instruction leaflet (in English and German) on the bottom of this page demonstrating how to interactively create a wrapper. Please note that these instructions are from a previous, demonstration version of the software.

In the **graphwrap-example** subdirectory of the **PDF Analyser** project, you can find this example wrapper. The file **graphwrap-example.sh** shows how GraphWrap can be executed from the command line.

Tamir Hassan
pdfxTk@tamirhassan.com
23 February 2011