Recognition and Enrichment of Archival Documents

D6.13.
Document Understanding Tools P1

Hervé Déjean, Jean-Luc Meunier
Xerox Research Centre Europe

Distribution:

http://read.transkribus.eu/

READ
H2020 Project 674943

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 674943

Project ref no.	H2020 674943
Project acronym	**READ**
Project full title	**Recognition and Enrichment of Archival Documents**
Instrument	H2020-EINFRA-2015-1
Thematic Priority	EINFRA-9-2015 - e-Infrastructures for virtual research environments (VRE)
Start date / duration	01 January 2016 / 42 Months

Distribution	Public
Contractual date of delivery	31/12/2016
Actual date of delivery	
Date of last update	22/12/2016
Deliverable number	6.13
Deliverable title	Document Understanding Tools P1
Type	Demonstrator
Status & version	1.0
Contributing WP(s)	WP5, WP6, WP7, wP8
Responsible beneficiary	XRCE
Other contributors	
Internal reviewers	ASV, UIBK
Author(s)	Hervé Déjean, Jean-Luc Meunier
EC project officer	Martin Majek
Keywords	Document Understanding, Workflow, Conditional Random Fields, Pattern Mining

Contents

Executive Summary

This document presents the work done during the first year for the Document Understanding (DU). Based on the Transkribus REST API, Layout Analysis tools and Handwritten Text Recognition tools are integrated in DU workflows. Two scenarios have been designed: one using supervised Machine Learning algorithms, where annotated documents are provided by a Transkribus user. The second scenario uses Sequential Pattern Mining algorithms to automatically mine and generate document templates.

The toolkit is built upon open-source software and available on the Transkribus GitHub repository. The READ wiki pages are constantly updated with last developments. See references Section 4.

1. Building Document Understanding Workflows

1.1. Overview

The Document Understanding toolkit aims at providing tools in order to perform "*logical and semantic analysis of documents to extract human understandable information and codify it into machine-readable form*." [1] Strongly relying on READ partners' tools (Layout Analysis and HTR), we are currently using the following tools:

- Region/line/word detection (NSCR, CVL)
- Baseline detection (NSCR)
- Graphical line detection (CVL)
- HTR (URO)
- Transkribus User Interface/ REST API (UIBK)

Some partners' tools are available through the Transkribus REST API, other were provided by partners and installed locally.

Regarding data, two collections were targeted: the StAZH collection and the ABP collection. The first one is used for the first scenario (generic label and supervised Machine Learning; see Section 2), and the second is used so as to assess the scenario 2 (sequential pattern mining).

End-to-end DU workflows require a large combination of tools and resources (such as specific language resources). Since the Transkribus platform only provides a subset of the required tools/services, our choice was to design DU workflows by combining services through the REST API and local tools and resources. This has several advantages:

- Specific DU workflows may always require specific resources not available through Transkribus
- The consistent use of the Transkribus REST API would allow for a quick integration of our tools in the Transkribus platform once decided
- This has been a required solution for the first year of the project, where not all tools have been integrated into the Transkribus client.

1.2. Transkribus REST API

We heavily rely on the REST API to access partners' tools, to get document and store them. A python-based library has been developed and allows for handling locally data (downloading/uploading documents) and tools (applying tools or uploading DU outputs). Document Understanding workflows are based on this software. The Transkribus User Interface is used in order to visualize and correct the Document Understanding inputs/outputs.

This software is also a way to provide a tutorial and examples to users willing to use the Transkribus REST services.

This work has been done with the support of UIBK.

This Python-based Transkribus API is available in the Transkribus GitHub repository. It currently covers 25% of the methods exposed by Transkribus server. It also provides a set of command-line utilities. See This READ Wiki page (or Annex 1).

2. Approach 1: Supervised Machine Learning

The split onto two scenarios is motivated by the different usages encountered: On the one hand we see motivated users willing to annotate a couple pages of their documents to obtain accurate and specific semantic representation of their documents. In our previous work, we explored several supervised machine learning approaches for a similar problem. It turned out that structured machine learning was required to tackle the task. The Conditional Random Fields [3] model gave good results for labelling document elements. Following this approach, we are interested in modelling a full document as a graph:

- where nodes reflect text blocks,
- where edges reflect relevant relations between text blocks,
- where feature vectors are associated to both nodes and edges,
- and to collectively predict the class of each node of the document.

In literature, it is common to build an acyclic graph per page, mostly the minimum spanning tree of the page [2]. Typically, the centroid of each block is reflected as a node and an edge links two such centroids. We rather build a graph at document-level. A node reflects a text block. An edge links two nodes, either on same page, or on consecutive pages. We call the former intra-page edge, and the other cross-page edge. An edge reflects a neighbouring relationship between two blocks. Due to its construction, the graph is almost always cyclic.

For training our CRF model, we use the open source pystruct library (pystruct.models.EdgeFeatureGraphCRF and pystruct.learners.OneSlackSSVM). The learning is achieved using a one slack structured SVM algorithm. Inference is done using AD3 dual decomposition. See [3,4,5].

The mid-term goal in 2017 is to integrate the notion of constraints, which should allow us to handle situations such as forcing at most one page number per page.

2.1. Dataset Evaluation

The StAZH Collection has been selected as first use-case. Some documents have been annotated with generic labels: headings, page number, page header, catchwords, and marginalia (see Figure 1).

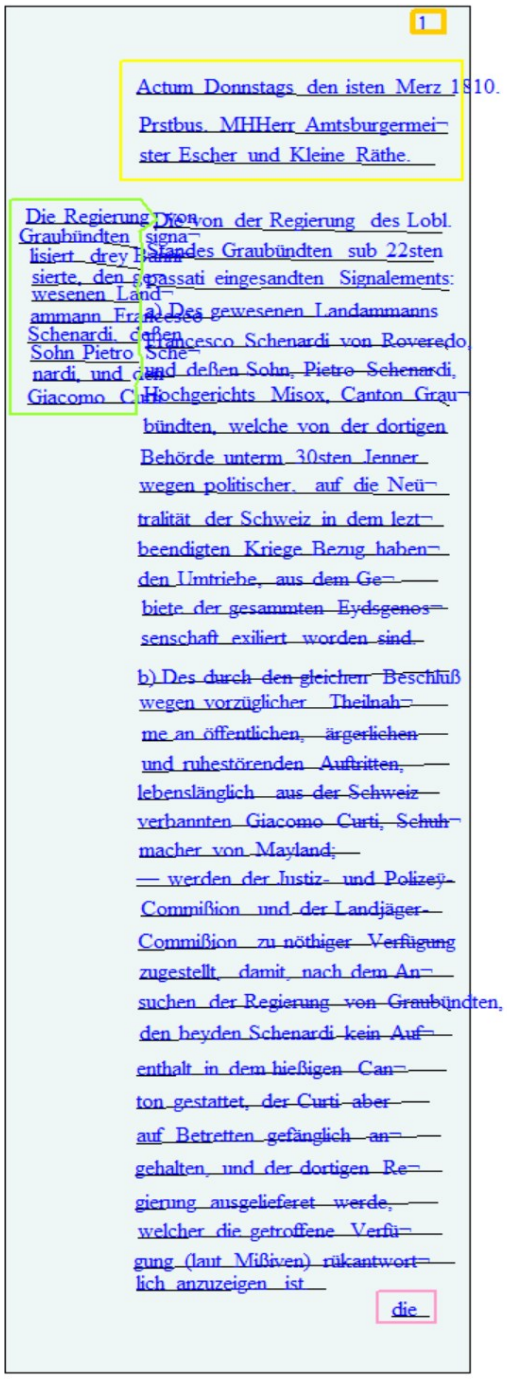

Figure 1: Example of annotated page with page number, heading, marginalia, catch-word. Document from the StAHZ collection.

A first evaluation was done using the ground-truth version, which shows very good results (around 90% F1 score for all labels). This first experiment is described in this wiki page, showing the various evaluations, and explanations on how to reproduce the experiments. A large-scale evaluation of this method will be conducted in 2017, with various large

collections, and in a “end-to-end” setting (considering image as input, while we are still currently using ground-truthed XML).

3. Approach 2: Sequential Pattern Mining

Complementary to the supervised Graph-CRF based approach, we are also working on a unsupervised approach, which does not require annotated documents, and relies on Sequential Pattern Mining algorithm [6]. The usages for such an approach are the following:

- Processing very large collections (since no annotation is required)
- Automatically assessing other layout Analysis tools (Are we able to generate the templates from Layout Analysis outputs?). This can be used as Quality Control tool.
- Providing structural features for supervised Machine Learning tools.

A high-level presentation of this approach can be found at [7]. The method mines a document in order to generate document objects templates. Once found these templates can be used in order to analyse the document (for instance segmenting a set of lines into paragraphs using the template paragraph).

A toy example is available on the READ wiki and shows how this approach can be used in order to generate column template using the line regions as input. This allows us to correct wrong lines (spanning over two columns). [7] shows how the method can be used for content-based document structures (wedding records).

2016 was also dedicated to rewriting some background software (Xerox IP), so that they can be open-source.

3.1. Sequential Pattern Mining for Document Mining

To achieve our Document Understanding task, we rely on two strong assumptions:

1. Documents (their layout and content) are not at all randomly organized and very often follow some templates which help a lot for ‘understanding’ them.
2. These templates can be discovered by mining documents and searching for regularities.

In order to mine the document, we use the PrefixSpan algorithm [6]. The document objects are associated a set of features and are viewed as sequences. Then the sequential pattern mining algorithm is used in order to generate sequential patterns that correspond to layout structures or data structure.

3.2. Dataset and Evaluation

We are currently using the ABP collection in our experiments. This collection is challenging in terms of Layout Analysis. A manual annotation is scheduled for 12/2016-01/2017, as well as the final evaluation.

HTR model training for this collection was delayed, but toy workflows were tested with a generic HTR model and some specific documents (extraction of first names in marginalia). An evaluation will be conducted as soon as an HTR model for this collection will be available.

This approach will also be used in order to provide rich structural information for the CRF-based method, and evaluation will show its usefulness.

4. Resources:

4.1. Software Repositories

TranskribusPyClient: https://github.com/Transkribus/TranskribusPyClient, *A Pythonic API and some command line tools to access the Transkribus server via its REST API*

Transkribus DU toolkit: *https://github.com/Transkribus/TranskribusDU, Document Understanding tools*

- **crf**: (graph-CRF; Approach 1): core ML components for training and applying CRF models
- **spm**: (Sequential Pattern Mining; Approach 2): core components for mining documents
- **use-cases**: examples of end-to-end workflows (current more toy examples)
    - **StaZH**

4.2. Related documentation under WIKI:

The READ wiki page is constantly updated with last developments.

https://read02.uibk.ac.at/wiki/index.php/Document_Understanding_: main page entry for DU activities

https://read02.uibk.ac.at/wiki/index.php/Transkribus_Python_API: page describing the Python REST API (see also annex 1)

http://read02.uibk.ac.at/wiki/index.php/Document_Understanding/First_CRF_Experiment: page describing the first use-case with CRF and the StAHZ collection

4.3. Data under Transkribus

- READDU (collection ID: 3571).StaZH documents annotated with logical labels
    - Ask permission to access this collection (contact us)

5. References

1. Dengel, A., Shafait F., *Analysis of the Logical Layout of Document*s, DOI: 10.1007/978-0-85729-859-1_6
2. *Tao X., Tang Z., Xu C., Document page structure learning for fixed-layout e-books using conditional random fields*, Proc. SPIE 9021, Document Recognition and Retrieval XXI, 90210I (December 27, 2013); doi:10.1117/12.2039492
3. Joachims,T. ,FinleyT, Yu C., *Cutting-plane training of structural SVMs*, JMLR 2009

4. Mueller, A., *Methods for Learning Structured Prediction in Semantic Segmentation of Natural Images*, PhD Thesis. 2014
5. Mueller A., Behnke S., *Learning a Loopy Model For Semantic Segmentation Exactly*, VISAPP 2014
6. Pei, J. et al, PrefixSpan: *Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth*,DOI: 10.1109/ICDE.2001.914830
7. Déjean H. Mining Documents to Understand them, https://read02.uibk.ac.at/wiki/images/5/5d/Xrce-blog-read-v5.pdf

Annex 1: Transkribus Python API

From READ Wiki: Transkribus Python API ; date: 22/12/2016

(We recommend you to click on the link to access an update version.

Contents

1. Reference Documents:

https://transkribus.eu/wiki/index.php/REST_Interface

https://transkribus.eu/TrpServer/Swadl/wadl.html

2. Code

See in git DLA/src/read/TranskribusPyClient . The module client.py offers a subset of the server API.

in sub-package test, there are some example of use.

DLA/src/read/TranskribusCommands contains command line routines.

2.1. Note on the proxy settings

The Proxy can be indicated as usual in environment variables. (HTTPS_PROXY) Or it can be passed as parameters to the code. (See the __init__ constructor)

2.2. on Transkribus Login

Pass your login/password as code parameters. Or consider having a *Transkribus_credential.py* file, where your login password are stored, like below:

```
# -*- coding: utf-8 -*-
login = "jean-luc.meunier@xrce.xerox.com"
password = "my-password-is-here"
```

Contact person: JL Meunier

3. Command Line Utilities

3.1. Persistent login

In order to provide your Transkribus credentials to each command, there 3 possible ways:

1. you create a *Transkribus_credential.py* module as explained in previous session (**set the access right properly to protect your passsword!**)
2. you provide your Transkribus credentials at each command using the *--login* and *--pwd* options.
3. you provide you credentials once and persist them using the *--persist* option . They are stored on disk with appropriate access rights, in a *.trnskrbs* folder.

```
do_login.py --persist --login <login> --pwd <password>
#To use the persisted session, set the --persist option in next commands.
#To clean the persistent session:
do_logout.py
```

3.2. Collections

Add Document(s) to Collection

Command to add one or several documents to a target collection.

These objects are all specified by their unique identifier (a number).

NOTE: the documents are duplicated! It is the same document that will appear in target collection in addition to some other collection(s).

```
 USAGE: TranskribusCommands/do_addDocToCollec.py [--persist] <colId>  [
<docId> | <docIdFrom>-<docIdTo> ]+
 Documents are specified by a space-separated list of numbers, or number
ranges, e.g. 3-36.
```

Copy Document(s) from Collection to Collection

Command to copy one or several documents from a source collection to a target collection.

These objects are all specified by their unique identifier (a number).

NOTE 1: the new document inherits the name from the source collection.

NOTE 2: Access rights in source collection is required.

```
 Usage: ./TranskribusCommands/do_copyDocToCollec.py [--persist]
<from_colId>  <to_colId> [ <docId> | <docIdFrom>-<docIdTo> ]+
 Documents are specified by a space-separated list of numbers, or number
ranges, e.g. 3-36.
```

Create a Collection

Command to create one collection.

```
 Usage: ./TranskribusCommands/do_createCollec.py [--persist]
<collection_name>
```

Delete a Collection

Command to delete one collection.

```
 Usage: ./TranskribusCommands/do_deleteCollec.py [--persist] <colId>
```

List a Collection

Command to list the content of a collection.

```
 Usage: ./TranskribusCommands/do_listCollec.py [--persist] <colId>
```

Transkribus_downloader

Utility to download a full collection from Transkribus and store it in a conventional DS folder structure. PageXml XMLs, and optionally the images, are downloaded.

In addition a "multi-page" PageXml file is generated for each document. (PageXml is a single page standard, so we changed it, see in: *read.xml_formats.multipagecontent.xsd*)

Viewing those xml files (.mpxml and .pxml) is possible using wxvisu and its specific wxvisu_PageXml.ini configuration file. See in read.visu.

NOTE: this downloader is lazy and will download the content of a document only if the document timestamp on server is more recent than the one on disk. On the other hand, when renewing the content of a document on disk, it downloads again the whole contents (xml and images), irrespective on which page or transcript was modified

```
(C:\Anaconda2) python Transkribus_downloader.py --help
Usage: Transkribus_downloader.py <colid> [<directory>]
```

```
Extract a collection from transkribus and create a DS test structure
containing that collection
Options:
--version show program's version number and exit
-h, --help show this help message and exit
-s SERVER, --server=SERVER Transkribus server URL
-f, --force Force rewrite if disk data is obsolete
-l LOGIN, --login=LOGIN Transkribus login (consider storing your
credentials in 'transkribus_credentials.py')
-p PWD, --pwd=PWD Transkribus password
--https_proxy=HTTPS_PROXY proxy, e.g. http://cornillon:8000
> python
C:\Local\meunier\git\DLA\src\read\TranskribusCommands\Transkribus_downloade
r.py 3571
 - Done
- Downloading collection 3571 to folder .
- creating folder: .\trnskrbs_3571
INFO:root:- downloading collection 3571 into folder .\trnskrbs_3571\col
(bForce=False)
 INFO:root:- downloading collection 3571, document 7749 into folder
.\trnskrbs_3571\col\7749 (bForce=False)
 INFO:root:- DONE (downloaded collection 3571, document 7750 into folder
.\trnskrbs_3571\col\7750 (bForce=False))
INFO:root:- DONE (downloaded collection 3571 into folder
.\trnskrbs_3571\col (bForce=False))
- Done
- Generating multi_page PageXml
 - .\trnskrbs_3571\col\7750
 - .\trnskrbs_3571\col\7750.mpxml
 - .\trnskrbs_3571\col\7749
 - .\trnskrbs_3571\col\7749.mpxml
- Done, see in .\trnskrbs_3571
```

TranskribusDU_transcriptUploader

Utility to upload the transcripts from a MultiPageXml XML file to a Transkribus collection. The MultiPageXml is plit into PageXML single-page transcripts, which are then uploaded to become a new version of the transcript for each page of the document(s).

This utility expects to work from the folders created by Transkribus_downloader.py .

```
 Usage: Transkribus_transcriptUploader.py   <directory>   <coldId>
[<docId>]
```

3.3. LA (Layout Analysis)

apply a batch LA

Apply an HTR models onto the pages of a document: do_htr.py

```
 usage : do_LA_batch.py <colId> <docId> [<pages>] <doNotBlockSeq>
<doNotLineSeg>
 by default blocks and lines seg are performed. Indicate (0=False/1=True)
which actions will be performed
 Display the job ID
 do_LAbatch.py  1949 10462 4
 35531
```

- Done

3.4. Recognition

list the HTR Models

List the HTR models:

```
$ do_listHtrModels.py
```

```
modelName|modelId|isUsableInTranskribus|nrOfTokens|nrOfDictTokens|nrOfLines
   ------------------------------|-------|--------------------|----------|--------------|---
------
                     Alcaraz_UPVLC|    276|                   0|         0|             0|
0
                           Bentham|     86|                   1|         0|             0|
0
                         Bentham_2|     87|                   1|         0|             0|
0
                     Bentham_UPVLC|      4|                   1|         0|             0|
0
                       Bozen_HS37a|      1|                   1|         0|             0|
0
                 Forrest_Collection|      6|                   1|         0|             0|
0
               Forrest_Collection_2|      7|                   1|         0|             0|
0
               Forrest_Collection_3|     46|                   1|         0|             0|
0
                   Frisch-Sklaverei|     25|                   1|         0|             0|
0
                 Frisch-Sklaverei_2|    356|                   1|     74392|         10742|
8755
                              Fuchs|     89|                   1|         0|             0|
0
                             GeoIII|    297|                   1|     10435|          2372|
1220
                           GeoIII_2|    357|                   1|     16617|          3152|
1857
                             Girona|    376|                   0|         0|             0|
0
                             Goethe|     90|                   1|         0|             0|
0
                    Goethe_Schiller|     93|                   1|         0|             0|
0
                           IO_Botany|    177|                   1|     27163|          5106|
4822
                         IO_Botany_2|    336|                   1|     96507|         10026|
14801
                         Just_Anton|    216|                   1|      4967|          1702|
595
                  Kerr_Collection_1|     66|                   1|         0|             0|
0
                     Kochbuch_Binder|    156|                   1|         0|             0|
0
               Konzilsprotokolle_1|    113|                   1|         0|             0|
0
   Konzilsprotokolle_1_extended_LM|    154|                   1|         0|             0|
0
               Konzilsprotokolle_2|    176|                   1|     30121|          5256|
3982
               Konzilsprotokolle_3|    316|                   1|     68370|          9609|
9808
                        Marine_Lives|     45|                   1|         0|             0|
0
                          NAF_504_v1|    296|                   1|      8436|          2778|
1498
          Nuernberger_Briefbuecher|    256|                   1|     10912|          2172|
1163
             Reichsgericht_Training|      2|                   1|         0|             0|
0
          Reichsgericht_artificial|     91|                   1|         0|             0|
0
```

```
        Reichsgericht_artificial_2|     92|                    1|         0|             0|
0
                       Resolutions|    155|                    1|         0|             0|
0
                          Schiller|     88|                    1|         0|             0|
0
                 Schweinfurth_Georg|    236|                    1|      4776|          1766|
811
                           Wieland|    134|                    1|         0|             0|
0
                        WrDiarium|    116|                    1|         0|             0|
0
                      WrDiarium_BW|    115|                    1|         0|             0|
0
                   WrDiarium_Farbe|    114|                    1|         0|             0|
0
                           Wydemann|    196|                    1|     16200|          4662|
1256
                        Zwettl_30|      3|                    1|         0|             0|
0
```

- Done

apply an HTR Model

Apply an HTR models onto the pages of a document: do_htr.py

```
usage : do_htr.py <model-name> <colId> <docId> [<pages>]
do_htr.py Wydemann 3829 8620 1
- Done
35313
```

list the HTR RNN Models and Dictionaries

List the names of the RNN HTR models and the names of the dictionaries. **do_listHtrRnn.py**

```
$ do_listHtrRnn.py

--- Models ----------------------------
20160408_htrts_midfinal_11.sprnn
meganet_hist_01_crx.sprnn
meganet_us1900_05_us_an_crx.sprnn
meganet_usaddr_12_pp_crx.sprnn
net_160201_trained_noise.sprnn
net_fraktur_0000_2000.sprnn
South_Carolina_1720.sprnn
GEO_1-3.sprnn
GEO_1-3_v2.sprnn
Reichsgericht_v4.sprnn
IO_Botany_v1.sprnn
Resolutions_v1.sprnn
Bozen.sprnn
escher_v3.sprnn
Frisch-Sklaverei.sprnn
IO_Botany_v2.sprnn
Konzilsprotokolle_v1.sprnn
hervetest.sprnn.sprnn
StAZH_v1.sprnn
Cyrillic_20th_Century.sprnn
Hyde_Reel_1_Session_2.sprnn
Gothic_Letter_1622.sprnn
NB_Norway_Koren.sprnn
Egypt_diary.sprnn
Sutor.sprnn
```

```
--- Dictionaries ----------------------
alvermann_train.dict
deutsch.dict
deutscheNachnamen.dict
eng.dict
fracture.dict
frau.dict
htrts15_all_sorted.dict
mann.dict
Bozen_v1.dict
Resolutions_v1.dict
Reichsgericht_v1.dict
StAZH_v1.dict
Cyrillic_20th_Century.dict
Gothic_Letter_1622.dict
NB_Norway_Koren.dict

- Done
```

apply an HTR RNN Model

Apply an HTR RNN model and dictionary onto the pages of a document: do_htrRnn.py

```
  usage : do_htrRnn.py <model-name> <dictionary-name> <colId> <docId>
[<pages>]
  $ do_htrRnn.py StAZH_v1.sprnn StAZH_v1.dict 3829 8620 3-4
  35442
  - Done
```

4. (Non-Urgent) Questions

4.1. Locking

The API includes lock-related methods (listPageLocks, lockPage(<bool>), isPageLocked). Should we lock a page we work on? Like when a DU tool does automatic annotation?

4.2. Page Status

If some DU tools do automatic annotation, which status should we set? For now on, we don't set it, and it becomes "in progress", which is ok BTW.

4.3. Storing Data

If we generate a ML model for a collection, can we store it somewhere in Transkribus and associate it with the collection?