READ

Recognition and Enrichment of Archival Documents

D6.14.
Document Understanding Tools P2

Hervé Déjean, Jean-Luc Meunier, Stéphane Clinchant
Naver Labs 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	31/12/2017
Actual date of delivery	
Date of last update	21/12/2017
Deliverable number	6.14
Deliverable title	Document Understanding Tools P2
Type	Demonstrator
Status & version	4.3
Contributing WP(s)	WP5, WP6, WP7, WP8
Responsible beneficiary	NLE
Other contributors	
Internal reviewers	ASV, UIBK
Author(s)	Hervé Déjean, Jean-Luc Meunier, Stéphane Clinchant
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 second year for the Document Understanding (DU) work package. TrankribusPyClient, the Python RESTful client has been updated to reflect changes in the RESTful API and extended in order to support collection and workflow design and management. TranskribusDU, the Document Understanding package per se, has been improved, enriched with new components (Information Extraction and Graph Convolutional Networks), and evaluated against several use cases.

Two main use-cases have been addressed: the ABP use case, focusing on Information Extraction from tables, and the BAR use case, focusing on Document format conversion. A first milestone has been reached in processing collections of tens of thousands of pages.

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 6.

1. Introduction

Year 2 was dedicated to improve the existing Document Understanding components and to design workflows for specific use-cases. Figure 1 shows how the TranskribusPyClient and TranskribusDU components interact with the Transkribus platform in order to design a DU workflow.

Section 1 will briefly present the improvements and new developments done for the TranskribusPyClient. Section 2 will present the new Machine Learning components. Then Use cases will be presented, as well as their workflow, and some evaluation.

The reader will find links to the READ Wiki pages which contain more detailed information, as well as in [1].

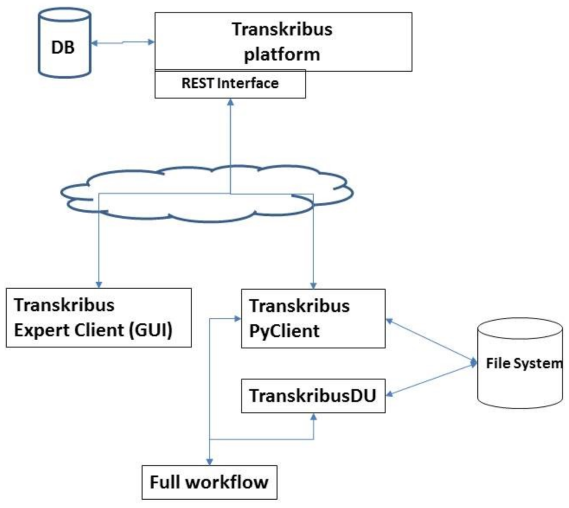


Figure 1: : Interactions between the Transkribus components.

2. TranskribusPyClient

2.1. Overview

TranskribusPyClient is a Python module allowing you to interact with the Transkribus platform through its RESTful interface [2]. Beyond the wrapping of the services offered by the Transkribus RESTfull API, a strong need appeared for some functionalities which would be too tedious through the Transkribus User Interface such as: having an efficient transcripts version management, or automate as much as possible some Machine Learning operations (such as full training configuration with parameter tuning.). With these new commands, full workflow can now be designed for most use cases (combined with TranskribusDU components).

2.2. Year 2 improvements

Here is the list of new or updated functionalities offered by PyClient. Please see the READ Wiki for the full list (or see Annex 1).

1. Managing transcripts of a document

In Transkribus, a document contains a sequence of pages, and each page contains a sequence of transcripts for versioning purpose. Modifying anything on a page, either manually or programmatically, and saving leads to the creation of a new transcript. Managing tens of transcripts for hundreds of pages is a typical order of magnitude, so we propose a new service to manage the mass of transcripts of a document. Essentially, the service selects the transcripts based on some criteria. After selection, some check can be specified, and finally an operation can be performed. For the time being, the criteria are date, time, user and status. The operation is one of list, removal, status update. See the corresponding READ Wiki page for more details.

This service is clearly of use to any tech-savvy practitioner. It can also help informing the design of an equivalent function in the Transkribus GUI

2. Transkribus_uploader

In relation with the previous service for transcript management, we have adapted the download and upload services, so that they can work on a sub-set of transcripts of a document, which were selected by the previous service. This is of use when for instance a subset of the pages was manually annotated to train some DU model, which is then applied on the rest of the document. There is then the need to download and upload subsets of transcripts, based on their annotation state. See this READ Wiki page, in particular the –trp option.

3. New Layout Analysis

This command allows one to perform text lines detection on a set of documents (using URO baseline finder).See this READ Wiki page.

4. Train an HTR model

With this command, the training and test sets can be specified easily and several training jobs can be launched at once, with various values for the main parameters (number of epochs, learning rate, batch size). This allows us to quickly perform a grid-search for selecting the best parameter values for a training set. See this READ Wiki page. See also deliverable D.4.5, Section 3.2

5. Apply an HTR model

This command allows us to select an existing HTR model, a dictionary, and to apply the HTR recognition at document, page or region level. The RESTful API allows us to also perform some recognition as region level with a specific dictionary. This new API extension will be tested with the Table use-case where columns correspond to specific data (names, location, etc.). See this READ Wiki page.

6. Upload specific dictionary

This allows you to upload specific dictionaries, which can be used with the new HTR service at region level (see above). See this READ Wiki page.

3. TranskribusDU

3.1. Overview

TranskribusDU is a Python library allowing you to perform some Document Understanding tasks. It allows you to build your own workflow in Python by easily combining layout analysis tools, TranskribusDU tools and your Python tools. For image processing and Layout Analysis, we rely on the tools available through the Transkribus RESTful API.

In the READ project, two main technologies for Document Understanding are used: a supervised Machine Learning component based on Conditional Random Fields (CRF) or Graphical Convolution Networks (GCN) and a mining (unsupervised) component based on Sequential Pattern Mining (SPM).

In the remaining, we model each page as a graph, where each node reflects one text line (see Figure 2). An edge in the graph reflects a neighbouring relationship between two text lines, possibly long distance ones. More precisely, whenever there is horizontal, respectively vertical, significant and direct overlap between two bounding boxes of two text lines, we create a vertical, respectively horizontal, edge. 'Significant' means that the overlap must be higher than a certain threshold. 'Direct' means that the two bounding boxes must be in line of sight of each other, i.e. without any obstructing block in between.

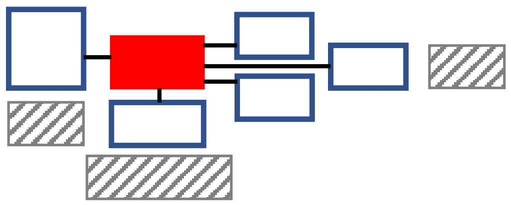


Figure 2: The neighbours of the red-filled block are in blue and are linked to it. Grey strikethrough blocks are not in its neighbourhood.

This particular structure is arbitrary but based on the intuition that neighbouring relations are playing an important role. We have also experimented with acyclic structures based on minimum spanning tree but we observed a degradation of performance.

3.2. Conditional Random Field component

During the first year of the project, we chose to experiment with a structured machine learning approach and implemented it as a Conditional Random Field (CRF) model, where a (mostly always) cyclic graph reflects the objects laid out on a page, or on the pages of a document. During the second year, we pursued this CRF approach further and made progress along three lines:

1. Extending the CRF model itself to a multitype CRF model;
2. Extending the single- and multi-type CRF models to support logical constraints;
3. Making practical improvements to TranskribusDU tool to better support use cases.

We detail below each line of work.

1. Multitype CRF

We have been using CRF to jointly classify all the objects of a page, or of a document, and we observed that joint classification was valuable (compared to separately classifying each object). However, this approach requires object *homogeneity*, i.e. all objects share the same set of classes, or labels. This is a limitation when considering that the "objects" laid out on a page of a document are of several types, e.g. textual, graphical.... Indeed, each type of object has a specific set of labels, e.g. heading, marginalia… for textual objects, and picture, separator

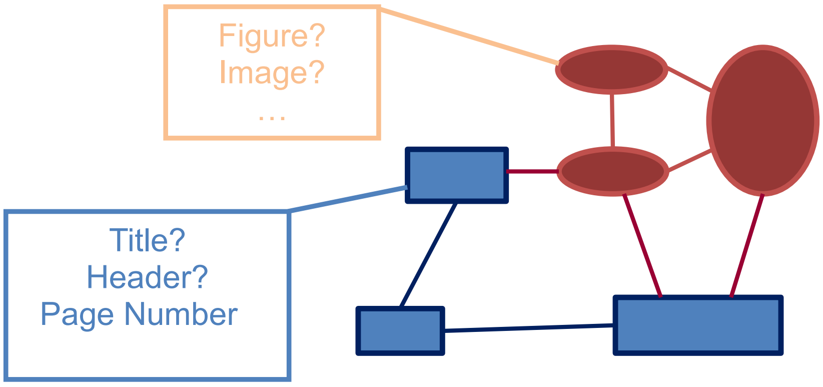


Figure 3: One example of multi-type CRF graph

line…, for graphical objects. They do not share the same set of classes. Finally, when it comes to interpreting the role of the objects on a page, we believe one has to consider all types of objects, in a joint manner because they generally depend on each other, e.g. the position of a caption text relates to the position of its corresponding image. This is why we looked after a method able to **jointly classify heterogeneous objects**. We therefore extended the PyStruct CRF library to perform structured machine learning on typed CRF graphs, i.e. a graph where each node is associated with a type that determines its label space. Generalizing from the conventional CRF model, there is one distinct unary potential function per type, and one distinct pairwise potential function per pair of types. Learning and inference can then be adapted to this new setting as explained in [3].

This CRF extension required extending both the PyStruct CRF library and the AD3 inference library. The extensions are respectively available from https://github.com/jlmeunier/pystruct and https://github.com/jlmeunier/AD3 . In a second publication [1], we gave a more practical view of the new CRF library. The respective owners of the original libraries showed interest in our pull requests. Some work is required from all parties in order to merge those codes (ongoing).

2. Logical Constraints in (single- or multi-type) CRF

While adapting the AD3 Python API to support multi-type CRF, we also extended this API to support logical constraint during inference. Indeed, AD3 natively supports logical constraints on binary graphs and CRF graphs can be binarised as explained in [4**Error! Reference source not found.**]. We then also extended the binarisation procedure to multi-type CRF as explained

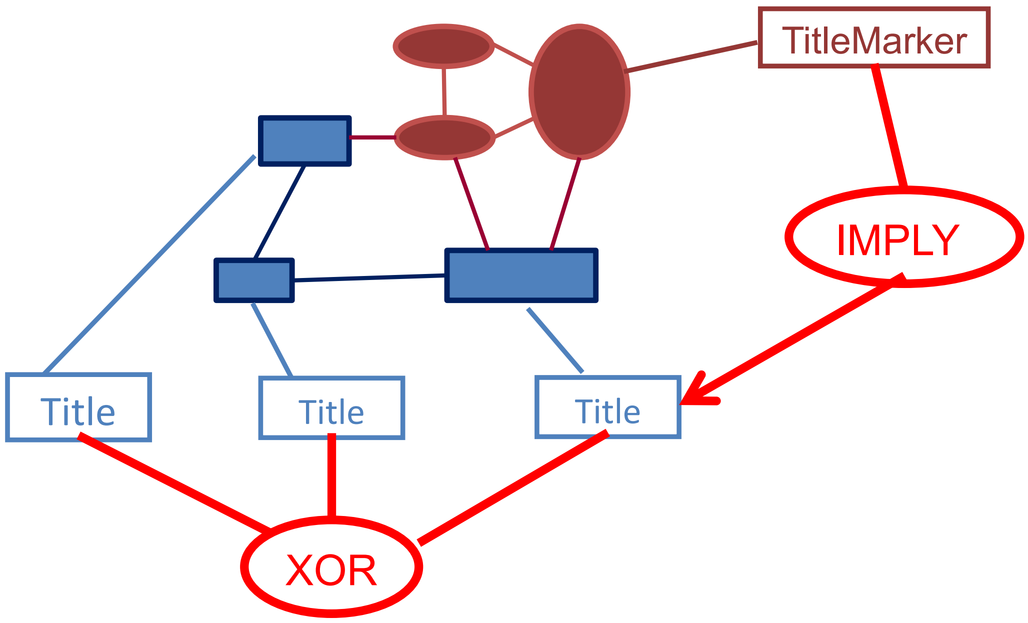


Figure 4: Using logical constraint to express a priori knowledge

in [3].

In turn, it is possible to express certain logical constraint on the labels of the nodes of the graphs. For instance, if one a priori knows that there is at most one page-number per page, this knowledge can be encoded as a AT_MOST_ONE logical constraint over all page number labels of all nodes of the sub-graph of objects corresponding to the page.

3. Extending the TranskribusDU Tool

In the course of our experiments with use cases, we incrementally added relevant features to our DU tool:

- Computing statistics for an annotated collection: number of pages per document, distribution of labelled objects per document, distribution over documents per labelled object, distribution of label per labelled object. Those statistics help the user, or developer, to assess the coverage and balance of the annotation of a collection.
  See in READ Wiki.
- Cross-validation and parameter tuning: the DU tool is able to perform all steps of a cross-validation experiment from partitioning the data to computing performance statistics and comparing to some baseline standard methods.
  See in READ Wiki.
- Visualizing the CRF model convergence and warm-starting a training.
  See in READ Wiki.

3.3. Graph Convolutional Network

Graph Convolutional Networks (GCNs) [5] have been proposed recently for classifying nodes in a graph. The underlying idea is to use the node adjacency matrix as ‘convolution’ (see Section 3.1 which explains how the graph is built). Although GCNs are standard feed-forward networks, there is one noticeable difference to standard networks which process elements independently from each other: GCNs operate on all the nodes of the graph at the same time and therefore introduce implicitly some dependencies among the predictions for unlabelled nodes. We tested this new Machine Learning approach and compared it to our CRF method. While results of both methods are comparable, one main advantage of the neural network approach we foresee over the CRF approach, beyond using mainstream technology, is the capacity to easily scale up regarding the training set size. The architecture used is described in [6], as well as the comparison with CRF for the ABP Table Information Extraction use case.

3.4. Sequential Pattern Mining

For the second year, no major improvement can be reported, but we performed some comprehensive evaluation of the use of Sequential Pattern Mining component for finding the layout and reading order in a multi-page, multi-column document. A collection of 30 different books was selected (provided by ABP) for evaluation. After having applied layout analysis tools at the page level, a sequential pattern mining algorithm is applied at the document level in order to identify the layout templates used in the document. In this experiment the goal was to identify the vertical structures (column and margins vertical boundaries). Then these templates will be used in order to correct errors done by the LA tools used at page level. Figure 5 shows an example of corrected page.

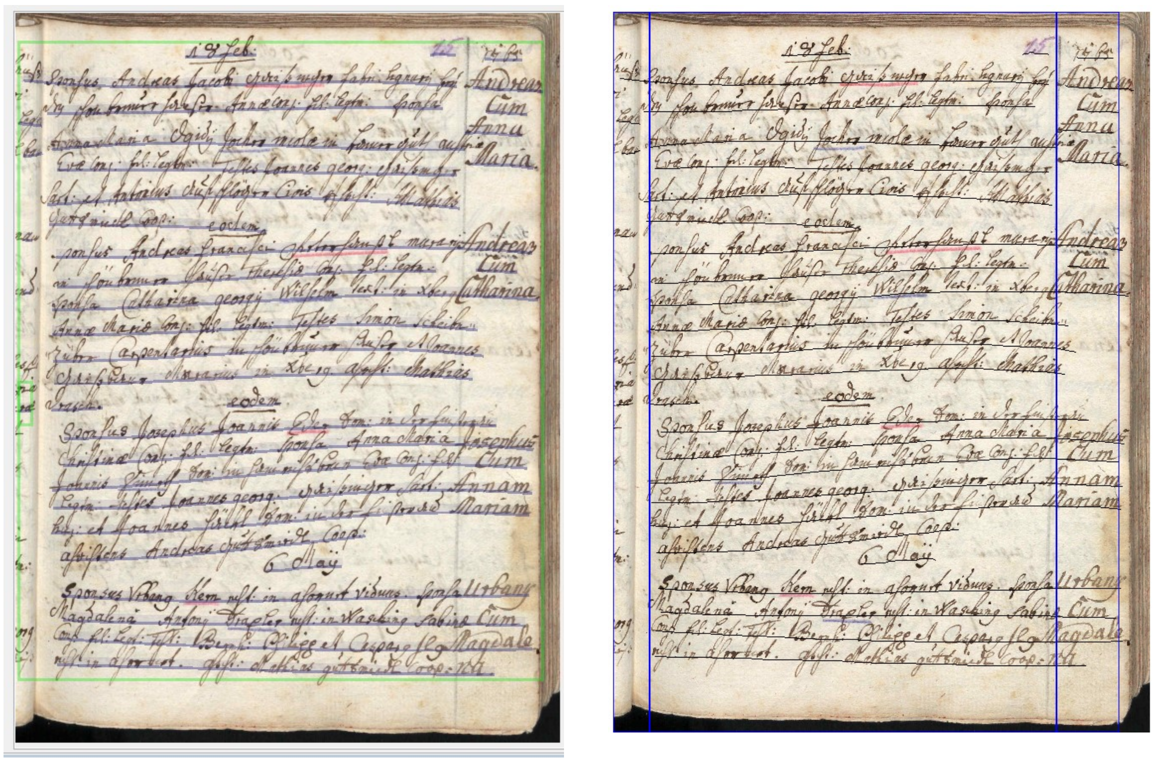


Figure 5: Initial lines and regions (green; left image), and corrected regions (blue, right). Text part of the left page are also excluded.

Table 1 shows the evaluation for the 30 books (first 30 pages considered)

Table 1: Evaluation of the column vertical boundaries with the SPM method.

	Our Method		
Doc#	**Precision**	**Recall**	**F1**
1	97.7	80.6	88.3
2	91.7	91.7	91.7
3	98.6	100	99.3
4	100	100	100
5	100	100	100
6	83	82.1	82.5
7	100	100	100
8	100	93.5	96.7
9	100	93.5	96.7
10	100	100	100
11	96.6	71.3	82.1
12	85	24.8	38.3
13	61.3	100	76
14	100	100	100
15	66.7	78.4	72
16	100	100	100
17	75	70.8	72.8
18	100	66.7	80
19	100	33.3	50
20	94.3	98.8	96.5
21	86	87.5	86.7
22	91.9	87.7	89.8
23	96.7	75	84.5
24	96.7	100	98.3
25	100	100	100
26	81.7	43.8	57
27	100	100	100
28	100	30.5	46.8
29	100	66.7	80
30	100	75	85.7
Micro	**91.4**	**72.5**	**80.8**

This component is used in the BAR use-case in order to build the reading order of the documents (mirrored left-right pages with marginalia) and correct some text lines (no line break between the marginalia and the main body). We also foresee to use this component in order to automatically build table templates for a document, and use the CVL table matching tool to compute the table structure (especially for noisy pages).

3.5. Table Understanding

Tables are an important document object. A set of components has been designed to analyse table structures. We rely on the table template tool designed by CVL (see [7]) for column recognition. For table row detection, our approach has been to use Machine Learning. The way we formulate the row detection problem is as follows: Once the columns and the text lines have been identified, each text line will be tagged with one of the following categories: B, I, E, S, O, which correspond of the following situation:

Table 2: Explanation of the BIESO labels used for table row segmentation.

Category	Explanation
B(eginning)	First line of a cell
I(nside)	Line inside a cell (except first and last)
E(nd)	Last line of a cell
S(ingleton)	Single line of the cell
O(utside)	Outside a table

This BIESO pattern is borrowed from the Natural Language Processing domain, where it is used to recognize entities (sequence of words) in a sentence. Our assumption is that, once properly categorized, it will be easy to finally segment into rows. Figure 6 shows some output of the categorization. Evaluation shows that both CRF and GCN perform very well on our dataset.

Table 3: Accuracy of CRF and GCN for the BIEOS row detection task.

Method	Fold 1	Fold 2	Fold 3	Fold4	Average
CRF	0.938	0.908	0.91	0.865	0.906
GCN	0.945	0.92	0.90	0.89	0.915

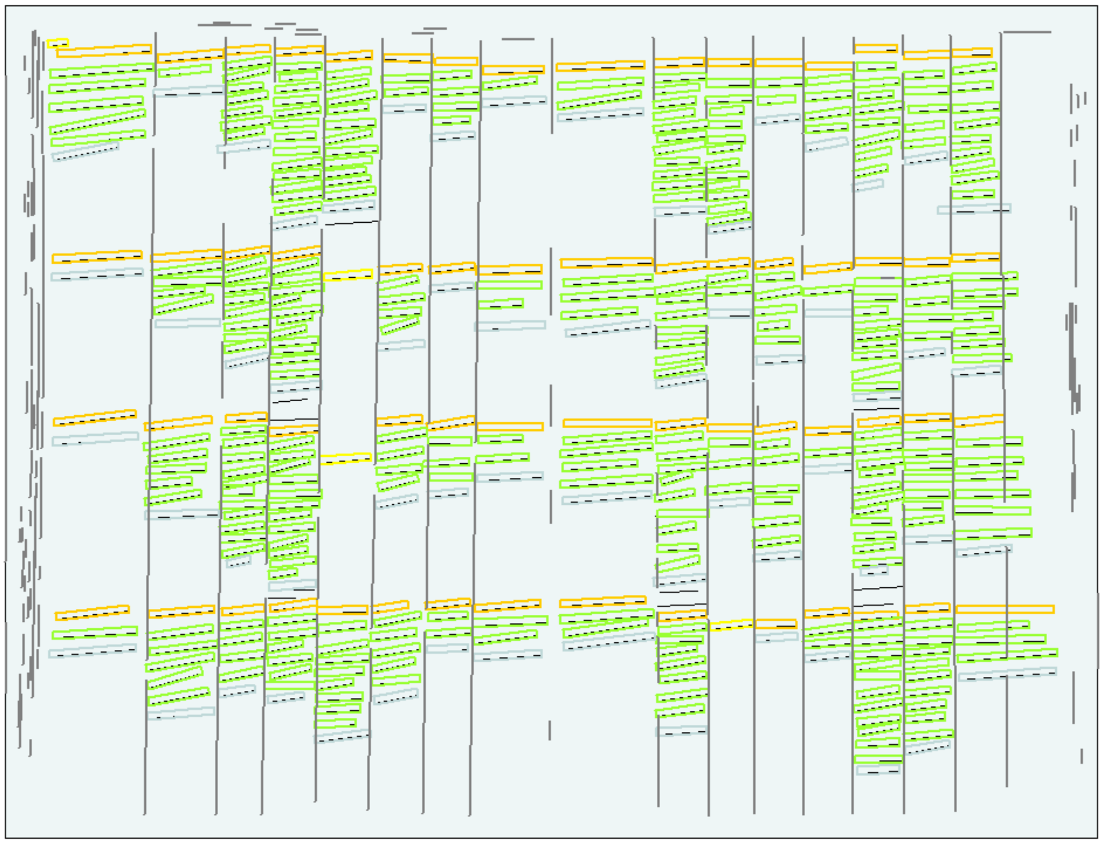


Figure 6. Example of Row detection using the BIEOS model. Orange: Begin of a cell, green: Inside a cell; grey: end of cell.

A full description of this experiment can be found in [6]

3.6. Information Extraction Component

A new component has been added to the TranskribusDU package in order to address Textual Information Extraction (hereafter IE) from table. IE, in our context, aims at tagging some textual elements organized in table cells. In our main use case (see Section 4.2), a record (table row) corresponds to an entry in a death book (first name, last name, family status, location, death date, occupation, death reason, …). A cell can contain various information (death date and location, names and row number for instance), so each word in a cell has to be correctly tagged. Figure 1Figure 7 shows some complex situations where fine tagging is required.

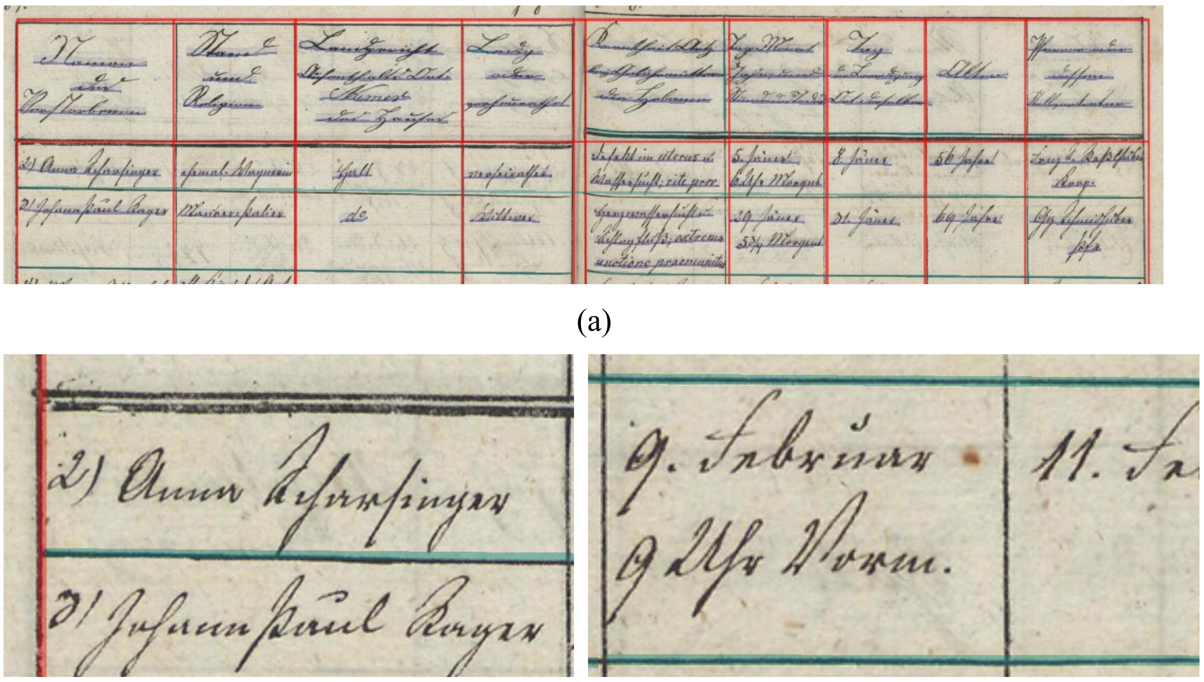


Figure 7. (a) shows the table header and the first two rows corresponding to a record. (b) the name field with a numbering information (second and third item for the given year). (c) The death date field is structured (date and hour), while only the month day and month fit the database schema, and have to be extracted.

In order to tackle this problem, we chose to use a Machine Learning approach: we trained a tagger in order to recognize each field of a record. In order to build the training set, one solution could have been to annotate some pages of the collection. Instead, the solution we chose was to generate a synthetic training set: ABP has already a database with thousands on (partial) entries. The idea is to use these entries (as dictionary) in order to generate a training set. A component was designed to take as entry the dictionaries extracted from the database (one dictionary per record field), or to simply generate small dictionaries for the fields were the values are small and closed (family status). We rely on some Python packages to generate the date fields. Below is an example of training example:

Table 4: Example of synthetic training set. Each word is associated with its label.

Word	Tag
Lambert	firstName
Stadler	lastName

Obersatzbach	Location
$F$lQ	textNoise
Brustwassersucht	Deathreason
48	ageValue
Jahre	ageUnit
ledig	familyStatus
5	weekDayDate
30	monthDayDate
Jul	monthDate
Gütler	profession

Some artificial elements such as noisy text are also generated in order to simulate the possible (probable!) noisy HTR outputs. A more sophisticated labelling system is also used for taking into account multi word elements such as composed first names (Eva Maria), using the BIES labelling. Hyphenation is also taken into account in the generation. We foresee this approach as being generalizable to other use cases (all civil and census records laid out with table whatever the language).

Then our Machine Learning component (based on BiLSTM) can be trained using this synthetic training set. Sections 4.2 explains the full workflow for this use case.

4. Use Cases

We now present the main use cases which were addressed during Year 2. For each of them, a workflow was designed using the Transkribus platform (for layout analysis, HTR model training, HTR), TranskribusPyClient (interaction with the platform), TranskribusDU (document objects categorization, information extraction), and the table matching tool (CVL) locally installed.

4.1. General Document Understanding

This use case aims at designing a general Document Understanding model targeting 'generic' document elements such as heading, page headers, paragraph, caption, … The collection is large and is constituted by 164 books (printed characters), and OCRed with Abbyy FineReader. Several languages are present, often German, but also some in Cyrillic characters. Applying our Conditional Random Field approach, we are able to achieve an overall F1-score of 0.91. Experiments using GCN should be conducted in the future. The next table shows the results for each category.

```
                  Precision  Recall  F1-score   Support


    gtb_TOC-entry     0.85      0.75      0.80      4145
      gtb_caption     0.27      0.36      0.31       398
   gtb_catch-word     0.00      0.00      0.00       188
```

```
              gtb_footer       0.00      0.00      0.00        10
            gtb_footnote       0.83      0.82      0.83     29279
  gtb_footnote-continued       0.27      0.25      0.26      1374
              gtb_header       0.96      0.98      0.97     10822
             gtb_heading       0.79      0.76      0.78     14493
          gtb_marginalia       0.93      0.96      0.94      6134
         gtb_page-number       0.98      0.98      0.98     28976
           gtb_paragraph       0.94      0.94      0.94    136075
      gtb_signature-mark       0.81      0.93      0.86      3689

             avg / total       0.91      0.91      0.91    235583
```

This use-case clearly shows that some general DU model can be learnd. The next challenge is to adapt this model to a specific collection. In the framework of CRF or Deep learning, this generic model trained on a large collection could be used for initialisation, and some small training data could be provided before performing some additional training steps.

More details about this dataset and the evaluation can be found on the READ wiki.

4.2. Information Extraction from Table (ABP collection)

This use case illustrates an Information Extraction workflow for tables. ABP provided us with a collection of death records from the 19th century in German (see WP 8.11, Section 2 [7]). They also have some partial extracted data stored in their database for this collection which allows us to perform some large scale evaluation for some record fields (first name, last name, occupation, location). The workflow uses several components from various partners:

1. Design of the table template (TranskribusX GUI, UIBK)
2. Layout Analysis (RESTful service, URO)
3. HTR (RESTful, URO)
4. Table Matching (locally installed, CVL)
5. Row Detection (TranskribusDU, NLE)
6. Information Extraction (TranskribusDU, NLE)

The first step is to design table templates associated to the collection. 11 main templates were identified by ABP. Once the templates have been designed, text lines (baselines) are identified. Then the table matching tool associates to each page the correct template. The outcome of this step is that the table region is found as well as the column regions. From this, the table row detection is performed (see Section 3.5). Then the table is fully structured into rows, columns, and cells. The Information Extraction tool is applied to each page, extracts and stores the data into an XML file. Below is an excerpt of output for a given record.

```
<PAGE number="151" nbrecords="10">
      …
    < RECORD  lastname="Gerlesberger"
             firstname="Anna"
             occupation="Warthstöchterlin"
             location="Kirchberg"
             situation="Kind"
             deathreason="Halsentzündung"
             deathDate="Nov"
```

```
            burialDate="Nov"/>
         …
  </PAGE>
```

A first evaluation has been performed with a 151-page document (collection 5400, document 27734), corresponding to approx. 1,500 records. The evaluation is performed as follows: we only considered records for which a last name and a first name field are extracted. Then these names are concatenated (first name + last names) and compared to the names in the database entry. An edit distance is used in order to add some fuzzy matching. Several fuzziness values are tested

The current observations we can draw from this small test are as follows:

- Row detection works well, an evaluation on 150 pages from 15 documents shows an accuracy of 90%
- HTR should be improved (new training data will be available). Secondly, the new regions-level HTR, combined with proper dictionaries should increase the HTR quality as well.
- IE tool is currently able to filter out very bad HTR text, and has to be improved regarding some minor points (doctor/nurse is not identified in the death reasons column)

Edit distance	**Precision**	**Recall**	**F-1**
TH=100	37.5	24.7	29.8
TH=80	60.6	40.0	48.2
TH=75	67.1	44.3	53.4
TH=66	76.1	50.2	60.5

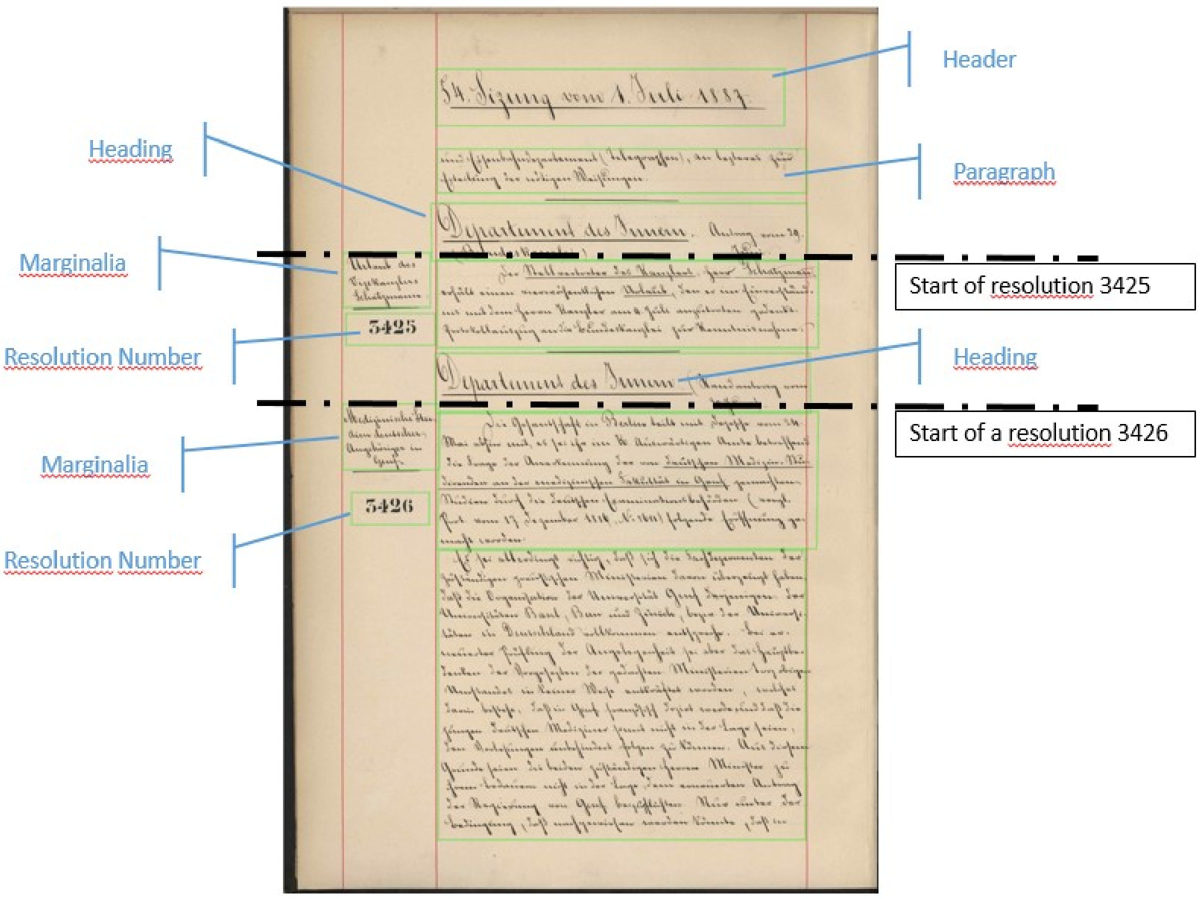
Overall, very encouraging outcome compared to last year. We should be able to show some results for the full 26,000-page collection during the second review.

4.3. Use Case: BAR collection

This use-case aims at recognising the so-called "resolutions" from the 19th century minutes of the Swiss Federal Council at the Swiss Federal Archives, Berne (BAR). This is a joint work with the State Archives of Canton of Zurich [8].

This task is twofold as it involves semantic labelling of the resolution constituents as well as segmenting the document into a series of resolutions, as illustrated on figure 2 below.

We will here only mention the salient aspects of this work and shall refer the reader to the READ Wiki for full details.



2 - a page of the 19th century minutes of the Swiss Federal Council

We jointly designed annotation guidelines and manually selected and annotated 100 pages. We then ran our DU tool for the two tasks, having turned the segmentation task in a labelling task (with the labels *Begin*, *Inside*, *End*). The final output of the task must correspond to a structured file (such as TEI), similar to this example:

```
<body>
 <div type=„volume‟ from=„1903-01-01‟ to=„1903-03-31">
  <head>Protokoll über die Verhandlungen des Schweizerischen Bundesrates
1903</head>
  <div n="1" type=„minutes‟>
   <head>Präsidial-Verfügungen vom 1. Januar 1883</head>
   <div n=„1.1" type=„departement‟>
  <head>Politisches Departement […]</head>
    <div xml:id=„resolution-YYYY-1" n=„1.1.1" type=„resolution‟>
     <head>Minister Kern, Urlaub und Stellvertretung</head>
     <p>Mit […] vom 30. Dezember […] </p>
    </div>
    <div xml:id=„resolution[…]‟ n="1.2" type=„resolution"> ……
    </div>
   </div>
  </div>
</div>
</body>
```

This was one opportunity for comparing a joint classification using the multi-type CRF with two separate CRF-based classifiers (one for the semantic labelling, one for the segmentation labelling). The joint classification is reminiscent of the Factorial CRF model of McCallum et all [10**Error! Reference source not found.**] as each object has a dual type. We report below our findings.

Accuracy	Semantic labelling	Segmentation labelling
Semantic alone	85%	N/A
Segmentation alone	N/A	59%
Joint classification semantic + segmentation	88%	66%

Those results show the interest of joint classification. We expect the accuracy to get a boost from using the text of the document, once an HTR is available.

We performed 7 steps of the workflow we defined and intend to complete this work next year, when the HTR and layout READ tools will be more mature. In particular, we will then be able to report on the performance using a task-oriented metric.

5. Resources:

5.1. 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**
  - **ABP**

5.2. 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)

5.3. Data under Transkribus

Ask permission to access these collections (contact us)

- READDU (collection ID: 3571). StaZH documents annotated with logical labels
- BAR_DU_testcollection (collection 7018). BAR annotated collection (Section 4.3)
- DAS2018 (collection ID 9142). ABP dataset for table (Section 4.2)

6. References

1. J.-L. Meunier, H. Déjean : "Transkribus Python Toolkit". ICDAR-OST, Kyoto, Japan, 10 - 12 November 2017
2. https://transkribus.eu/wiki/index.php/REST_Interface
3. J.-L. Meunier, "Joint Structured Learning and Prediction under Logical Constraints in Conditional Random Fields", CAp 2017
4. Martins, A. F., Figueiredo, M. A., Aguiar, P. M., Smith, N. A., Xing, E. P. "AD3: alternating directions dual decomposition for MAP inference in graphical models", JMLR 2015.
5. T.N. Kipf, M. Welling: Semi-Supervised Classification with Graph Convolutional Networks. CoRR abs/1609.02907, 2016.
6. S. Clinchant, H. Déjean, J.-L. Meunier, Eva Maria Lang, Florain Kleber, Comparing Machine Learning Approaches for Table Recognition in Historical Register Books, submitted.
7. Deliverable 6.8; Table and form analysis tool P2 (CVL)
8. Deliverable 8.11 ; Large Scale Demonstrators. Keyword Spotting in Registry Books P2 (ABP)
9. Deliverable 8.5; Evaluation and Bootstrapping P2 (StAZH)
10. Lafferty, J., McCallum, A., Pereira, F. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", ICML 2001

11. Code

TranskribusPyClient: https://github.com/Transkribus/TranskribusPyClient

TranskribusDU : https://github.com/Transkribus/TranskribusDU

CRF : https://github.com/Transkribus/TranskribusDU/tree/master/src/crf

SPM : https://github.com/Transkribus/TranskribusDU/tree/master/src/spm

GCN: https://github.com/Transkribus/TranskribusDU/tree/master/src/gcn

Row Detection : https://github.com/Transkribus/TranskribusDU/tree/master/src/tasks

Information Extraction :
https://github.com/Transkribus/TranskribusDU/tree/master/usecases/ABP/src

Annex 1: Transkribus Python API

From READ Wiki: Transkribus Python API ; date: 22/11/2017

(We recommend you to click on the link to access an update version; new items are in bold)