Transkribus Python Toolkit

Jean-Luc Meunier, Hervé Déjean
Naver Labs Europe
firstname.lastname@naverlabs.com

Abstract—This paper introduces an open source Python toolkit for the Transkribus platform. One part of the toolkit offers a Python client for the Transkribus RESTful interface. The second part offers various Document Understanding tools. The open-source toolkit is freely available through GitHub.

Keywords—Transkribus platform, RESTful client, Document Understanding, Conditional Random Fields, Sequential Pattern Mining open source, Python.

I. INTRODUCTION

The Transkribus platform [1, 2] is a platform to collaborate, share and benefit from most recent research in Handwritten Text Recognition. Transkribus offers various Document Image Analysis (DIA) and Handwritten Text Recognition (HTR) services besides the storage of digitized documents. Mainly available through a Java desktop application, most services within Transkribus are exposed via RESTful interface. In the framework of the EU-funded READ application [3], we propose a fully open source Python toolkit which fulfills two functionalities: providing a Python client for the RESTful interface (TranskribusPyClient) and a toolkit for performing Document Understanding and Information Extraction (TranskribusDU). The combination of both should provide a complete toolkit for Document Processing and Information Extraction. Both github projects are part of the github Transkribus organization (https://github.com/Transkribus).

Figure 1 illustrates the interactions between the Transkribus platform and both presented Python projects. The interaction with the platform is done through TranskribusPyClient, which is able to locally store data needed by TranskribusDU and launches RESTful services (such as HTR). TranskribusDU offers a set of Document Understanding and Information Extraction tools, and full workflows are built using TranskribusPyClient and TranskribusDU. The call to RESTful services corresponding to Transkribus Layout Analysis and HTR requires upload and download of intermediate transcripts.

The remainder of this article is structured as follows: we describe TranskribusPyClient, the Python module which allows you to interact with Transkribus in Python. Then we present the various tools offered by TranskribusDU, and a complete example is shown in order to illustrate how workflows are built. Eventually the short and long-term extensions are described.

Figure 1: Interactions in the Transkribus environment.

II. TRANSKRIBUSPYCLIENT

TranskribusPyClient is a Python module allowing you to interact with the Transkribus platform through its RESTful interface [4]. It allows you to perform the most important operations:

- Persistent login (you provide your credentials once and persist them using the --persist option. They are stored on disk with appropriate access rights, in a .trnskrbs folder.)
- Downloading data: full collection (image and PageXml); document-wise; page-wise (transcript)
- Performing layout analysis
- Performing HTR training
- Performing HTR analysis
- Uploading PageXml produced locally (by your DIA tool for instance)

The main module (client.py) contains a (currently partial) list of the RESTful operations. Commands are then built on these operations. A command is composed at minimum of the 3 operations: login/operation/logout. Release notes contain the

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE
list of supported operations. You can also have a look at the wiki associated to the TranskribusPyClient project. If this toolkit provides you with a way to interact in Python with Transkribus by using RESTful operations (basic Python client), we also use it in order to connect our internal Document Understanding software with the data and tools available in Transkribus, so that full workflows can be built for specific use-cases. Two main commands (Transkribus_downloader, TranskribusDU_transcriptUploader) have been created to facilitate this connection.

The Transkribus_downloader command creates a local repository structure and locally stores Transkribus documents (image and PageXml files). In order to quickly access all PageXml files of a given document, a ‘multipagexml’ is also created (one single XML file per document aggregating all individual PageXml files). The naming of folders is based on Transkribus collection ID and document ID. The following command downloads the document having the ID Y from the collection having the collection ID X:

```bash
>Python Transkribus_downloader.py X --docid=Y

\t\\nskrbs\X

\tskrbs_X\col[Y].mpxml

\tskrbs_X\col[Y]/[pagexml and image files]

\tskrbs_X\y\run

\tskrbs_X/xml]
```

**Figure 2: Folder organization of TranskribusPyClient**

The col folder contains the files (pageXml and images) organized by document. It also contains the multipagexml for each document (Y.mpxml). The remaining folders (ref, run, xml) are used to store local resources (such as specific ground truth files in the ref folder).

Some of our tools (see next section) directly work with PageXml files as input: they enrich the downloaded PageXml files with new elements or new attributes. Then the enriched XML version is uploaded into Transkribus as a new version using the upload command. When the Transkribus expert User Interface is running, you will be notified that a new transcript is available.

Since some of our other tools also require different format from PageXml, some conversion tools are available which ease PageXml conversion (see Section III.C.1). Once local software has been applied to the documents, their outcome has to be converted to the PageXml format again and uploaded.

III. TRANSKRIBUSDU

TranskribusDU is a Python library allowing you to perform some Document Understanding tasks. This should allow you to build a full 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) and a mining component based on Sequential Pattern Mining (SPM). The CRF component is a modification of the PyStruct [5, 6]. The SPM is based on [7].

The split on two technologies is motivated by the different usages encountered when working on use-cases: On the one hand we see motivated users willing to annotate a couple (or more!) pages of their documents to obtain accurate and/or specific semantic representations of their documents. On the other hand, unsupervised methods such as pattern mining can leverage the large amount of available documents. The pattern mining technology is used for mining a new collection or correcting Layout Analysis tools.

```
src/      Conditional Random Fields tool
  crf/     Sequential Pattern Mining tool
  spm/     specific tools for use-cases
  tasks/   XML conversion tools
  xml_format/  
```

**Figure 3: Folder organization of TranskribusDU/src**

The TranskribusDU/src folder contains two main subfolders: crf and spm, as well as a subfolder (xml_format) containing conversion tools for converting from and to PageXml format. The task subfolder contains examples of scripts using the crf and spm modules for specific purposes (see Section III.C.2).

A. CRF component

In our previous work, we explored several supervised machine learning approaches for solving DU tasks. It turned out that structured machine learning was required to tackle the task. The Conditional Random Fields [8] model gave good results for labelling document elements. 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 [9,10].

The CRF component directly uses the PageXml files as input (specifically the multipagexml file). Annotated data can be manually created through the Transkribus Expert User Interface [2]. This interface allows you to markup generic page elements and you can also use specific tags.

Currently, the CRF component does only work on structural elements (regions, and textlines), but the next release would allow you to automatically tag textual content (such as named entities). The tasks folder contains Python scripts which allow the user to easily describe the training task: how to select the page elements and the list of tags to be considered for training the model (see example Section III.C.2 and Fig. 4).

B. Sequential Pattern Mining-based component

Complementary to the supervised Graph-CRF based approach, we are also working on an unsupervised approach, which does not require annotated documents, and relies on Sequential Pattern Mining algorithm [7]. A high-level presentation of this approach can be found at [11]. One main

---

1 https://github.com/Transkribus/TranskribusPyClient/wiki
use of this tool is to mine collections already processed by Layout Analysis tools, and infer layout patterns such as: this document has a two-column layout, its paragraphs are justified, etc. The method simply mines a document in order to generate document objects templates. Once found, these templates can be used in order to analyze the document (for instance segmenting a set of lines into paragraphs using the template paragraph) or to correct segmentation errors (see example Section III.C.3). Examples of its use is shown in Table 1 at the end of the article.

C. Other functionalities

1) src/xml_format folder

This folder contains a set of scripts in order to convert PageXml files into other data structures and to create or modify an existing PageXml file. Compliance is validated using the official PageXml schema.

2) src/tasks folder

The tasks folder contains specific resources for a given use-case, currently the ones addresses within the READ project. The script DU_CRF Task.py defines the DU_CRF_Task class which allows you to train a CRF with a Transkribus collection (previously downloaded). Some specific instances show you how to define the annotated data (see the DU_StAZH files). Below is an example which defines the elements which will be learnt: the list of tags and the xpath localizing the objects.

```python
DU_GRAPH = Graph_MultiPageXml
nt = NodeTypeInfo_PageXml("TR")  # some short prefix because labels below are prefixed with it
    , ["catch-word", "header", "heading", "marginalia", "page-number"]  # EXACTLY as in GT data!!!!
    , [][1]  # no ignored label
    , True  # no label means OTHER
}
nt.setXPathExpr(".//pc:TextRegion")  # how to find the nodes
    , "/pc:TextEquiv")  # how to get their text
```

**Figure 4: How to specific training elements and tags.**

We assume that the collection contains annotated data for training. This annotation can be done using the Expert User Interface, and pages tagged as GT (groundtruth) can be selected for training. This piece of code shown Fig.3 is the only one to be modified by a user to train our CRF tool with documents stored and annotated in Transkribus.

3) Usecase folder

The TranskribusDU/usecases folder contains several READ usecases (ABP, StaZh) which provides simple examples using the CRF and SPM tools. See the README files. You can also have a look at the referred wiki pages in the readme files. The StaZh use case shows you how to perform the following steps with TranskribusDU:

1. Collection download
2. CRF Training
3. CRF Testing
4. Applying CRF on untagged document
5. Uploading to Transkribus of new tagged documents

See TranskribusDU/usecases/StAZH. Please do not hesitate to ask for access to the test collection.

The ABP folder contains a wiki page which shows you how to use the SPM tool for grouping lines into paragraphs, and also for finding the page layout of a document (identifying the column structure).

IV. A SMALL EXAMPLE

We present here a full sequence of instructions combining TranskribusPyClient and TranskribusDU which illustrate how to download a collection, train a model and upload tagged documents. We use collection 3820 as training data, and the model will be applied to the collection 3829. Command lines with $srcDU refer to TranskribusPyClient client, with $srcTR to TranskribusDU commands.

```
# download of collection 3820
$srcDU/tasks/DU_StAZH_a.py trn3820 mdl-StAZH_a --trn trnskrbs_3820

# got this on disk
ls trnskrbs_3820/col/config.txt out/ref/run/xml/
lst trnskrbs_3820/col/7749/
7749.mpxml
7749_max.ts
7750.mpxml
7750_max.ts
trp.json

#Training!!!
$srcDU/tasks/DU_StAZH_a.py trn3820 mdl-StAZH_a --trn trnskrbs_3820

# model created in mdl-StAZH_a

# Applying the model on collection 3829
$srcDU/tasks/DU_StAZH_a.py trn3820 mdl-StAZH_a --trn trnskrbs_3829

# done
# we produced some _du.mpxml files (tagged)
ls trnskrbs_3829/col/8620/
8620.mpxml
8620_du.mpxml

#now upload _du.mpxml to Transkribus
$srcDU/TranskribusCommands/TranskribusDU_transcriptUploader.py trnskrbs_3829 3829

# DONE, all transcripts were uploaded. See in collection 3829
```

**Figure 5: our use case script.**

The result of this processing can be seen with the Transkribus UI: Fig. 6 shows the final PageXml with the tag occurring in the @custom attribute, and Fig. 7 (last page) shows the automatically added tags through the Expert User Interface: page-number, page header, heading and marginalia (structure column).
Figure 6: The type of the page element is stored in the *custom* attribute.

These command-line based examples are going to be replaced by Python programs very soon.

V. INSTALLATION

Both projects, TranskribusPyClient and TranskribusDU use generic Python 2 libraries. Installation steps are shown in the project readme.md file. We advise future users to install available releases. Do not hesitate to contact us or post an issue in case of problems. Before using these tools, we need to create a Transkribus account [2].

VI. CODE SOURCE AVAILABILITY

The source code of TranskribusPyClient and TranskribusDU as well as some datasets and experiments are available at [https://github.com/Transkribus/Transkribus/](https://github.com/Transkribus/Transkribus/) under a BSD 3-Clause License.

The source code for the modified version of PyStruct is available at [https://github.com/jlmeunier/pystruct](https://github.com/jlmeunier/pystruct)

VII. DOCUMENTATION

Documentation is currently available on the wikis of both github projects:

- [https://github.com/Transkribus/TranskribusPyClient](https://github.com/Transkribus/TranskribusPyClient)
- [https://github.com/Transkribus/TranskribusDU](https://github.com/Transkribus/TranskribusDU)

VIII. EVOLUTION

A. TranskribusPyClient

The short-term objective is to cover all meaningful operations provided by the RESTful API, but also to develop some commands corresponding to common needs for various users: common workflows (layout analysis + HTR training + HTR analysis), or specific workflows not provided by the platform such as a complete experimental framework for training an HTR engine: parameter tuning (à la sklear grid-search) and n-fold validation.

Additional functionalities are added when they correspond to recurrent actions when a workflow or a tool is tested: We are currently developing a command to delete in a convenient way the transcripts which are uploaded during the testing phase.

Contributors are welcome!

B. TranskribusDU

One short-term direction is to be able to provide various CRF models (and their annotated datasets) for some generic tasks (‘standard’ documents, table processing, …). At the workflow level, one medium-term goal is to provide various workflows for some generic and frequent tasks.

A third component, the information Extraction (IE) component, not mentioned in this paper, is at its early stage and dedicated to some READ use-cases. The next step will be to provide a as-generic-as-possible IE component, with potentially some support of the Transkribus Expert User Interface for specifying the information to be extracted.

Last but not least, an integration of the DU tools onto the Transkribus platform is foreseen for 2018.

ACKNOWLEDGMENT

We would like to thank the UIBK team for their support, when developing TranskribusPyClient, especially Philip Kahle.

The READ project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 674943.

REFERENCES

[2] [www.transkribus.eu](https://www.transkribus.eu)
[3] [https://read.transkribus.eu](https://read.transkribus.eu)
Table I: Using the TranskribusDU SPM tool allows you to generate hierarchical structure from a set of lines. Here the line vertical position is used. See https://github.com/Transkribus/TranskribusDU/wiki/TranskribusDU_SPM for more details.