READ

RECOGNITION & ENRICHMENT
OF ARCHIVAL DOCUMENTS

D4.4
Service and Tool Integration

Philip Kahle, Sebastian Colutto, Günter Hackl, Günter Mühlberger
UIBK

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
Contract. date of delivery	31.12.2016
Actual date of delivery	28.12.2016
Date of last update	21.12.2016
Deliverable number	D4.4
Deliverable title	Service and Tool Integration
Type	Report
Status & version	Final
Contributing WP(s)	WP4
Responsible beneficiary	UIBK
Other contributors	All partners
Internal reviewers	Gundram Leifert, Hervé Dejean
Author(s)	Philip Kahle, Sebastian Colutto, Günter Hackl, Günter Mühlberger
EC project officer	Martin Majek
Keywords	Transkribus

Contents

1 Executive Summary

This deliverable outlines the progress of task 4.2, service and tool integration. During the first year of READ, procedures have been defined that allow standardized development and integration of tools from all technical partners.
While most of the currently integrated tools rely on legacy implementations from the TranScriptorium project, efforts have been made in order to add new tools, such as the URO HTR engine, that is already available in an experimental setup.
The following describes the infrastructure that was setup in order to ease deployment of newly available tools and services in the first part. The second part describes already made experiences with this infrastructure and an outlook is given in the end that defines a roadmap for the next steps.

2 Platform Architecture Regarding Tool Integration

During the TranScriptorium project, various tools have been integrated in the READ platform, Transkribus, and UIBK could already gain some experience on how to handle this task. While the integration procedure for each of the tools was then a fitting process with respect to the properties of the specific piece of software, it was clear that a standardized way is needed to cope with a large number of tools as it is the case in READ. Therefore, all technical partners gathered in biweekly conference calls and a set of interfaces was defined that is to be satisfied by any tool developed during the project. The outcome of these efforts can be found on Github within the TranskribusInterfaces project[1].
The main idea is to have predefined Java interfaces, with a respective set of C++ counterparts, which allows to integrate any tool in either of the two languages within Transkribus. Note, that Transkribus is natively written in Java and therefore any Java tool can be more or less integrated seamlessly. For C++ applications, the Java Native Interfaces (JNI) can be utilised to embed such foreign tools within the platform.
A major benefit of this approach is that work can be done concurrently on both the tools and the integrating infrastructure while targeting those interfaces. The integration procedure itself is then merely writing some few lines of custom code, passing the correct, specific parameters, and deploying the tool itself on the server, namely in the TranskribusAppServer, the component that is responsible for executing workflows in the system (see D4.1 for details).
For transferring document image data between the interfacing applications the commonly used OpenCV[2] was employed. This allows to pass data in memory without an additional disk write operation. On the other hand, layout and/or transcription data is passed in the form of PAGE XML files, the format Transkribus natively uses for storing text and layout data.

[1] https://github.com/Transkribus/TranskribusInterfaces
[2] http://opencv.org/

3 Integrated Tools

3.1 URO HTR

The first tool that became available in the first year was the HTR engine provided by URO/Planet. This software is delivered in the form of one propietary jar file (Java archive) and an additionaly package by URO. The latter is made available via Github[3]. As this is Java software, it was easy to integrate within Transkribus, but, due to the complex nature of HTR technology, a lot of features have to be taken into account, especially when it comes to the user interface. The first experimental approach was a simple integration where only a provided network and dictionary could be chosen by filename for recognition of a set of pages (where the layout must already exist). The training of new optical models could be carried out manually by UIBK, i.e. no user interface was given for this approach. That was an important milestone, in order to gain knowledge about the said features of the technology but at the same time making the recognition available to experienced users. To the date of this writing, a new user interface, both for training and recognition, is developed at UIBK which shall summarize all the requirements that could be stated during the experimental phase. For training a new HTR optical model, an arbitrary set of document images can be specified within Transkribus, while a nonintersecting set of images can optionally be specified as test set. Besides the parameters needed by the engine, a user may specify a name for the HTR optical model, the language included and a description which are all stored as metadata in the database of Transkribus once the model has been trained. Besides the optical model itself, a list of known characters and the series of character error rate (CER) values, evaluated on the test set during training, are stored in the platform. All those properties shall outline the performance of the trained model very well and the training set can be viewed by any user, accessing the model, in order to gain an overview of the learned script type.
In a first iteration the new recognition user interface will be made available to users, and, in a second step, during the second year, the training will follow. Due to the vast amount of processing power needed for training, this division is necessary but as the computing resources at UIBK are currently increased (see D4.1), any user will be able to train his/her own optical model in the near future.
HTR somewhat interleaves with language resources and thus this topic is discussed in the following section.

3.2 UPVLC HTR

The HTR engine by UPVLC was integrated in the course of the TranScriptorium project and thus is available as legacy implementation and can be used via the REST API of the Transkribus server. However, the integration of HTR has been restructured during the first year and thus the Transkribus GUI does not offer means to use the UPVLC HTR anymore. However, once the new implementation based on the defined interfaces

[3] https://github.com/Transkribus/CITlabModule

becomes available, it will be reintegrated with Transkribus and then is available to all users.

3.3 ASV Language Resources Toolkit

As HTR with transcription output performs much better when backed with a proper dictionary, this topic was also targeted in the first year. ASV therefore provides a Java library on Github[4] that is able to generate language resources, including a dictionary, from e.g. PDF, TEI, PAGE XML, and HTML. A simple command line tool allows UIBK to generate a dictionary from a Transkribus-compatible document, i.e. PAGE XML. Again, as this library also satisfies the defined interfaces, packaging it with Transkribus components can be done with ease. Most probably, the library will be bundled with Transkribus on client-side and only the generated set of files is transferred to the server for storage. Besides the dictionary, this will include a list of known abbreviations and their expansions and different forms of words. A remaining task for 2016 or early 2017 is implementing the respective user interface dialogs for creating new language resources from the stated input formats.

3.4 NCSR and CVL Layout Analysis Tools

In order to test the integration of native C++ libraries using the interfaces, first experiments have been carried out with line[5] and word[6] segmentation tools provided by NCSR. Although the integration turned out more difficult than Java-born libraries, the deployment is much simpler than the custom approach applied in TranScriptorium. Again, the user interface is a crucial point in this integration task and it is planned to combine efforts on all layout analysis tools, e.g. also the ones from CVL, at once in early 2017.

3.5 XRCE Document Understanding Tools

End-to-end DU workflows require a large combination of tools and resources (such as specific language resources). Since DU is done at the end of the pipeline taking into accout the results of layout analysis, HTR, table recognition, and language operations, XRCE's choice was to design DU workflows by combining Transkribus services through the REST API and local tools and resources. This has several advantages:

- Specific DU workflows may always require dedicated resources not available through Transkribus
- The consistent use of the Transkribus REST API would allow for a quick integration of XRCE's tools into 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.

[4]https://github.com/Transkribus/TranskribusLanguageResources
[5]https://github.com/Transkribus/NCSRTextLineSegmentation
[6]https://github.com/Transkribus/NCSRWordSegmentation

For more details see D6.13, Section 1 (Building Document Understanding Workflows)

3.6 Next Steps

Conclusively, an overview on the various tools shall be given, which become available for integration in 2017 and 2018. This section outlines the current state of work. The items are given in no specific order.

CVL Writer Identification This tool will become very valuable to quickly gain sight on uploaded images regarding changes in writing style. A C++ interface is currently in draft and the tool will be integrated in 2017.

CVL Table and Forms Recognition This tool is also on the top priority list, as it directly affects groundtruth production at partners, e.g. ABP. The interface is yet to be defined, but the subtask will be targeted in 2017.

UPVLC CATTI Engine During this period a first version of the CATTI system has been integrated in the Transkribus platform.

UPVLC Query by String A protoype integration exists but, as the HTR integration will be overhauled, there remains some work to be done on this.

URO Query by String The engine also depends on the HTR integration and thus is targeted for 2017.

NCSR Query by Example The integration of this tool is still pending and will be targeted in 2017/2018.

Text2Image Matching tool The tool will allow to quickly match existing transcriptions with the corresponding image. Interface is defined and integration is targeted for 2017.

Baseline Metric Tool A tool that can be used to measure the error rate of detected baselines against groundtruth data. It is already available[7] and will be integrated in 2017.

Error Rate Tool This tool can compute word and character error rates of recognized text against groundtruth data. It is available on Github[8] and will be integrated in 2017.

4 Conclusion and Outlook

In 2016, the main effort in this task was put into defining the interfaces and setting up the infrastructure that enables the READ consortium to easily add new tools to the platform. This work turned out to be valuable already now, as reliable, standardized

[7] https://github.com/Transkribus/TranskribusBaseLineMetricTool
[8] https://github.com/Transkribus/TranskribusErrorRate

interfaces reduce the amount of work needed vastly on all ends. While the integration process is thus much simplified, the main work in 2017 will focus on testing new software components as they become available and building reasonable user interface components for the several tools.
As already mentioned, the most important next tools to be integrated are the language resources toolkit by ASV and the various layout analysis tools, which will improve the user experience crucially, as users may select the most appropriate layout analysis method for their respective document set.