



Enhancing Usability for Automatically Structuring Digitised Dictionaries

Mohamed Khemakhem, Axel Herold, Laurent Romary

► To cite this version:

Mohamed Khemakhem, Axel Herold, Laurent Romary. Enhancing Usability for Automatically Structuring Digitised Dictionaries. GLOBALEX workshop at LREC 2018, May 2018, Miyazaki, Japan. hal-01708137v1

HAL Id: hal-01708137

<https://hal.science/hal-01708137v1>

Submitted on 13 Feb 2018 (v1), last revised 4 Mar 2018 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Enhancing Usability for Automatically Structuring Digitised Dictionaries

Mohamed Khemakhem^{†*§}, Axel Herold^{†‡¶}, Laurent Romary^{†*‡}

[†] Inria - ALMAAnaCH, Paris

^{*} Centre Marc Bloch, Berlin

[§] Université Paris Diderot

[¶] École Pratique des Hautes Études, Paris

[‡] Berlin-Brandenburgische Akademie der Wissenschaften, Berlin

{mohamed.khemakhem, axel.herold, laurent.romary}@inria.fr

Abstract

The last decade has seen a rapid development of the number of NLP tools which have been made available to the community. The usability of several e-lexicography tools represents a serious obstacle for researchers with little or no background in computer science. We present in this paper our efforts to overcome this issue in the case of a machine learning system for the automatic segmentation and semantic annotation of digitised dictionaries. Our approach is based on limiting the burdens of managing the tool's setup in different execution environments and lightening the complexity of the training process. We illustrate the possibility to reach this goal through the adaptation of existing functionalities and through using out of the box software deployment technology. We also report on the community's feedback after exposing the new setup to real users of different professional backgrounds.

Keywords: electronic lexicography, usability, digitised dictionaries, TEI, Docker

1. Introduction

Web applications have been the main deployment solution for many NLP tool designers to shortcut the need to deal with installation and configuration issues that many desktop applications continue to represent for end users. A web architecture does not rely on the user being familiar with local software tools such as command line shells or software development environments that allow expert and more personalised use of some advanced libraries. A strong current development is the integration of sets of tools into unified web-based working environments for general Humanities research such as the European CLARIN¹ and DARIAH² initiatives. On the more specialised field of lexicography, tools such as the Lexonomy³ dictionary writing system (Měchura, 2017) represent a typical class of web-based applications. While much of this high level way of accessing NLP tools also accounts for desktop applications, locally installed tools and possibly other software they rely on still have to be updated regularly. Different tools may even form a complex “eco-system” with delicate dependencies amongst individual modules. The main concern for users with regard to web-based tools are the security and possibly the confidentiality of their data. Therefore desktop applications still exist after the general turn to web-based solutions.

GROBID-Dictionaries⁴ is a machine learning system which has been developed to serve as a web application for structuring digitised dictionaries (Khemakhem et al., 2017). It also exhibits desktop functionality required for the preprocessing of data during the training process. Although it has a decent documentation, the process of setting up the desktop

version of the tool remains very challenging for users with limited programming knowledge. Annotating the preprocessed XML data also represented a serious challenge in earlier versions of the tool because initially it did not provide mechanism for sanity checks or for visualising annotations for humans.

In this paper we focus on the desktop functionality built into GROBID-Dictionaries. We present new features which have been implemented to enhance the usability of the tool. In section 2. we provide an overview of the architecture and setup of the system. We detail the different stages of the training process in section 3.. We then address the technical challenges related to the installation of the system as well as the annotation process and present our solution to overcome them in section 4. In section 5., we report on first experiences with the new setup and features based on feedback collected from users who were previously not familiar with GROBID-Dictionaries.

2. GROBID-Dictionaries

The work carried out by Khemakhem et al. (2017) resulted in a successful adaptation and extension of GROBID – an existing machine learning platform (Lopez and Romary, 2015) – to be used for the automatic identification of lexical information in digitised lexical resources. The resulting system is called GROBID-Dictionaries to reflect the dependency with the parent project. GROBID-Dictionaries has been tested with several lexical resources with promising results.

2.1. Architecture

The system's architecture is cascaded. Textual and typographical information are processed by means of multi-level classifications performed by machine learning models. Figure 1 sums up the architecture described in Khemakhem et al. (2017). Each blue object represents a *conditional random field* (CRF) model. These models are used to classify

¹<https://www.clarin.eu/>

²<https://www.dariah.eu/>

³<http://www.lexonomy.eu/>

⁴<https://github.com/MedKhem/grobid-dictionaries>

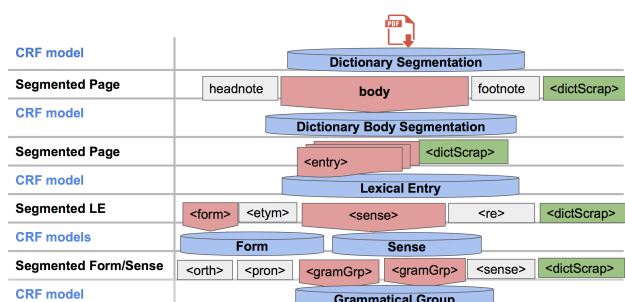


Figure 1: General architecture of GROBID-Dictionaries

the input text together with its typographical features. The other objects represent resulting text clusters to be either directly wrapped into proper TEI elements (elements with angle brackets) or they are temporarily tagged with pivot elements that are transformed into valid TEI constructs only in the final output (e. g., *headnote*, *footnote*, *body*). For the sake of simplicity, figure 1 does not include all possible tags for the *Form* and *Grammatical Group* models. A complete description of all possible TEI structures resulting from these two models can be found under the TEI P5 dictionary chapter⁵⁶ in Budin et al. (2012).

2.2. Configuration

GROBID-Dictionaries depends on core utilities and libraries provided by GROBID⁷. The installation of the system must be preceded by the installation and setup of the parent project. Therefore GROBID-Dictionaries needs to be cloned as an extension module within GROBID's project structure and needs to be built after its parent project.

Due to differences in technical preferences of the project leaders, two different automation build technologies need to be used for building each project: Gradle⁸ for GROBID and Maven⁹ for GROBID-Dictionaries. Successful builds of the system are packaged as Java libraries in two formats:

- a JAR (Java ARchive): this file is required for all processing stages which precede the training of each model, and
- a WAR (Web Application Resource or Web application ARchive): in the case of GROBID-Dictionaries this is not only a standalone web application but also a self-contained one that can be run after the training of the CRF models. It provides a graphical user interface to the existing web services, each corresponding to one or more of the cascading classification models.

GROBID-Dictionaries has been developed, tested and documented for the Linux and Mac operating systems. The behaviour of the resulting libraries is expected to be the

same when run in other operating systems. However, there is no explicit guarantee for such uniform behaviour.

3. MATTER Annotation Workflow

The annotation workflow in GROBID-Dictionaries follows the MATTER methodology (Model–Annotate–Train–Test–Evaluate–Revise, see figure 2) introduced by Pustejovsky and Stubbs (2012). Projected onto GROBID-Dictionaries and the processing of lexical resources, the individual steps are as follows:

Model: define a CRF model for predicting different text structures at one stage and determine the corresponding feature set. This phase requires the involvement of a programmer to create the defined models and integrate them into the cascading architecture.

Annotate: assign a TEI tag to each text block representing a lexical entity defined within a model's scope. This task must be performed on an XML representation of the data and must be strictly synchronised with the corresponding feature set file. The annotation guidelines¹⁰ need to be respected.

Train: use each annotated batch of data to train a corresponding model. The cascading architecture of the models should be respected here.

Test: this step gives just a rough idea about how the trained model behaves on unseen data. There are many ways to accomplish this goal. The easiest one is to run the corresponding web service from the web application on a held-out sample.

Evaluate: a precise evaluation with different measures is enabled at the end of the training process as long as annotated data are provided under the dedicated location in the dataset.

Revise: the last stage is about reviewing the modelling and annotation steps that have been described in the guidelines. Four possible measures are the outcome of this step:

- annotate more data when an improvement of the results was achieved,
- refine the annotation guidelines for new variation noticed in the last training batch
- proof-read the performed annotations when minor anomalies are noticed
- think about redefining the modelling when the results represent unexplainable anomalies. This could be translated either into a simple feature engineering process or into a change of the logic behind and the scope of the models or their architecture.

⁵<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-form.html>

⁶<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-gramGrp.html>

⁷<https://github.com/kermitt2/grobid>

⁸<https://gradle.org>

⁹<https://maven.apache.org>

¹⁰<https://github.com/MedKhem/grobid-dictionaries/wiki/How-to-Annotate%3F>

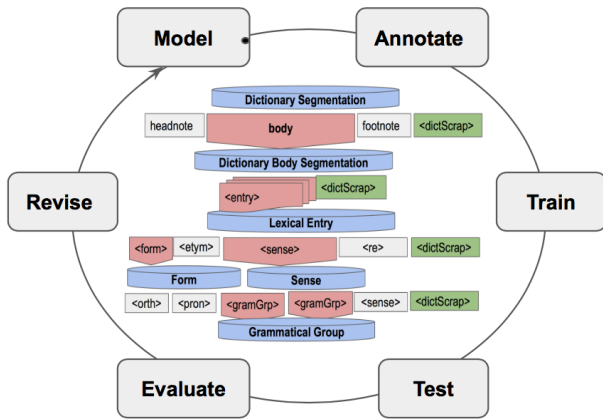


Figure 2: Implemented MATTER Workflow

4. Enhanced Usability

Section 2. has drawn a detailed picture of the technical setup required to install and execute the different parts of the system. Thus it is clear that a certain expertise and understanding of the system architecture is mandatory to successfully install the tool. Section 3. presented the challenges of the iterative training cycle which involves costly manual work by way of carry out data annotation.

Such requirements present a twofold obstacle: on the one hand the tool’s target community mostly consists of users with limited programming skills, such as lexicographers or linguists. If these users are not able to get technical support, the tool will not be usable for an important part of its target community. In the other hand the GROBID-Dictionaries project aims to constantly improve its architecture and to provide more fine-grained lexical information. On the long term, the goal of project is the provision of generic machine learning models which will be able to exploit different types of digitised dictionaries. Collecting and working with different types of lexical data (or at least samples thereof) drawn from a preferably diverse users community is a crucial step in the further development of GROBID-Dictionaries.

The usability of the tool is a vital aspect to it as this enables a broad user community to productively make use of GROBID-Dictionaries. Therefore, issues of usability are of similar importance like the tool’s earlier defined purpose and the research challenges it encounters.

4.1. Unified Execution Environment

As a first measure, we have investigated different ways for streamlining the setup process and for guaranteeing a unique behaviour of the system across different execution environments.

A possible solution would have been using a system image runnable on a virtual machine. Such an image should have a Linux based operating system, a Java development kit (JDK) and the different automated build systems installed. GROBID and GROBID-Dictionaries should also already be cloned and built correctly. This type of solution suffers from two main issues. Firstly, the size of the image would be huge as it would include several unneeded tools and system files that are still part of the operating system. Secondly, the

static nature of such an image would make it complicated to update it after a new version of GROBID-Dictionaries is released. Updates to GROBID-Dictionaries are published frequently since the tool is under continuous development. However, a system image containing the above mentioned components can be built in a more efficient way using a different technique. Docker¹¹ is a state of the art software technology which is also based on the virtualisation of the execution environment. Contrary to the static image approach sketched out initially, Docker allows for the flexible composition of an image. An image is shaped by instructions written in a Docker file¹². These instructions make sure that only the required components are included in the image. Moreover, several alternatives are available to efficiently update a build within an image starting from pushing a newly created image to the online Docker Hub repository¹³, to linking the corresponding GitHub and Docker Hub repositories coupled with activating the automatic build to synchronise the image after each update of the code.

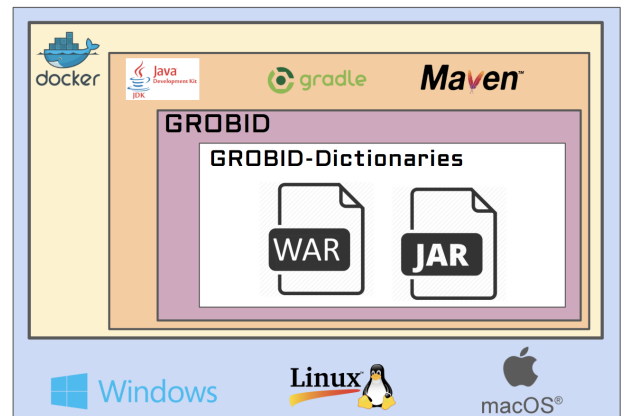


Figure 3: A GROBID-Dictionaries image in a Docker container

To run a Docker image of GROBID-Dictionaries (see figure 3), a user needs to install the version of the Docker software corresponding to his operating system and pull the latest image of the tool from Docker Hub. The pulled image (orange box) will not be run directly on top of the operating system of the host machine but rather inside a Docker controlled container (yellow box). Thus testing the tool on Docker is enough to guarantee a unified behaviour, regardless of the particular system configuration of a user’s computer environment.

It is also possible to synchronise files on the host machine with a running image in the Docker container. This feature allows for the tool hosted inside of a Docker container to directly interact with files stored on the host machine. We profited from this alternative to make the dataset directory shared between the two environments. With this mechanism, the user can exploit the full functionality of the tool living

¹¹<https://www.docker.com>

¹²<https://github.com/MedKhem/grobid-dictionaries/blob/master/Dockerfile>

¹³<https://hub.docker.com/r/medkhem/grobid-dictionaries/>

in the Docker image to train the machine learning models on the data residing locally on his machine.

In addition, thanks to the self contained nature of the tool's web application coupled with its fluid setup and manipulation through the Docker image, using the GROBID-Dictionaries image enables running both of the desktop and web based functionality on the local machine of the user. As previously introduced, such a feature represents an asset for researchers who care about the security of their data and experiments.

4.2. Lightening MATTER Process

The second major category of improvements specifically targets the annotation workflow. Annotating training data involves challenging manual work and demands provisions to ensure data integrity and validity.

4.2.1. Creating Training Data

To train a model in GROBID-Dictionaries based on a PDF file containing the raw text and the typographical features of a lexical resource, two additional files are necessary: a TEI document containing the corresponding reference encoding and a feature file describing textual and typographical information of each printed line or token.

To generate the training files, embedded functionalities of the tool should be used following one of the two options:

- *pre-annotated training data*: it used to be the default mode for creating automatically training data, inherited directly from GROBID's core functionality. This mode is useful when a model was trained on a substantial amount of data. The task of the annotator is then to correct the automatically placed TEI tags by moving, adding or removing them.
- *raw training data*: constitutes new functionality we have implemented to shortcut the checkout and cleaning of the tags automatically generated by using the default mode. The idea is simply creating training data without pre-annotations. Despite being obvious, starting annotating a document from scratch was not possible before integrating this new feature. Such a mode cuts with the old practice of correcting the predictions a model trained on different sample, to make it possible to start annotating totally fresh data. Besides giving more choices to the annotator, such mode saves him time and effort especially if an old model has been trained with multiple TEI elements.

A legitimate question remains yet unanswered: how can a user generate training data based on a selection of specific pages from possibly hundreds of pages a dictionary may comprise? After annotating different lexical samples in PDF format, we could qualify splitting an existing document to separate pages, or sequence of pages, as a very critical step. With some supposedly dedicated PDF manipulation tools producing damaged pages, we found only one tool reliably useful for the purpose of separating PDF pages¹⁴ which seems to produce a quality split as good as the original document. Using workaround solutions for this purpose,

such as print-to-file functionality in web browsers, is also not recommended.

4.2.2. Training Data Annotation

As previously stated, GROBID-Dictionaries generates a pre-processed XML representation from PDF files containing the raw text of a lexical resource. To create training data for the tool the user is then required to introduce semantic mark-up for the different models. Typically, an XML aware editor should be used to perform this task. Some advanced editors such as oXygen¹⁵ allow for visual annotating XML files (see figure 4 for an example).

We aimed to profit from the visual feature to avoid performing inline annotation directly on the text of the XML elements. This is catered for by a new feature in GROBID-Dictionaries that now for each model provides both a schema description (in Relax NG) and a presentational stylesheet (in CSS). The schema description enables the editing software to check or even enforce schema compliance of the training data. The stylesheet can be exploited by the editing software to allow users to mark up the training data semantically by highlighting portions of the text and then enclosing the highlighted portion with a suitable XML tag. The colours attributed to each element can be customised by a simple modification in the stylesheet.

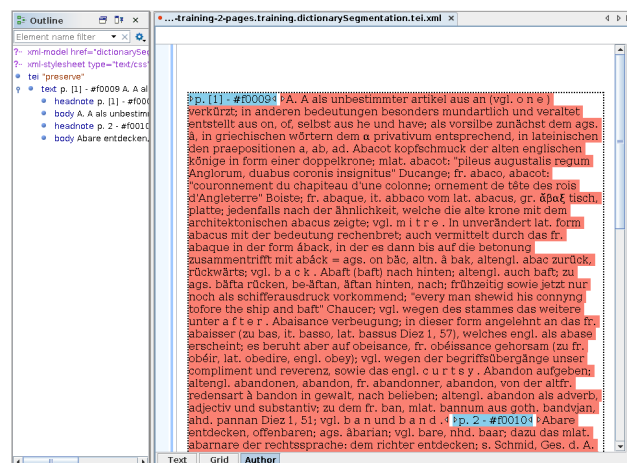


Figure 4: Training data annotation in oXygen author mode for the first model: page headers vs. page body

4.2.3. Train, Test and Evaluate

For this segment of the MATTER workflow, the user is provided with straightforward shell commands to execute, graphical mode to test and varied measures to evaluate and decide whether a model has reached an acceptable level of accuracy. A simple but effective trick could be however employed at this stage to verify the accuracy of the annotations performed in the previous step. Where in a normal case the annotated data should be split between training and evaluation datasets, the training dataset could be used also as evaluation dataset to verify any inconsistencies that might have accrued during the annotation process. In such a setup, a correct annotation should give 100 % accuracy, which

¹⁴<http://community.coherentpdf.com>

¹⁵<http://www.oxygenxml.com/>

means that model could reproduce what it has learnt correctly. Any other result should lead to the last step described in section 2.

5. User Experience

We had the opportunity to expose the system with its new setup and features to a mixed group of users in the course of a winter school on lexicography that was held at the Berlin-Brandenburg Academy of Sciences and Humanities at the end of 2017¹⁶. During this event we collected information about the usability of the tool. Additionally, we asked participants to respond to a questionnaire after the winter school to gain further insights as to their experiences while working with the tool.

5.1. Setup

A group of nine users participated in the experiment which was carried out during three hands-on sessions of four hours length each. The users were free to join one or more sessions of the the tutorial. The goal of the tutorial was to familiarise the participants with the MATTER workflow as implemented in GROBID-Dictionaries, while excluding the first modelling step which requires programming skills. Note that none of the participants was familiar with the tool prior to the tutorial.

After a short introduction to the architecture of the system, the users were guided through the process of installing and running the docker image¹⁷. Running the docker image, the participants were then able to reproduce the results reported in Khemakhem et al. (2017) which are based on a modern English monolingual dictionary. As the next step, several users used the possibility to experiment with their own lexical samples by repeating the learnt workflow and crafting new models for their individual datasets. Two of the participants succeeded in training and using all of the implemented models for their own datasets, thus adapting all of the functionality currently implemented in GROBID-Dictionaries.¹⁸

5.2. Gathered Insights

We asked the participants of our tutorial to respond to a questionnaire that we made available to them after the winter school. The questionnaire was created as a Google Form¹⁹. The results of the inquiry can be summed up in the following points:

Tool setup / user profile The first three questions focus on establishing the professional background of the participants. The tutorial group consisted of lexicographers, linguists, computational linguists, a computer scientist, a web developer and a philologist. Participants were free to name more than one field of expertise. Of

the nine respondents, seven reported previous knowledge of machine learning techniques but only four of them had actually worked with machine learning tools before.

When asked whether they encountered any problems with actually running the tool from the docker image, the majority of the participants (seven) responded that this was not the case. The setup failed once on a Windows based computer with insufficiently sized memory that was running an advanced version of the operating system. Consequently there was not enough memory left to run the Docker software which requires more than the 1 GB of free memory. The participant could still continue the tutorial by sharing a machine with her colleague. Without taking into account the answer of another respondent who has involuntarily reported encountering an installation issue, almost 90% of the users were able to launch the tool without any problem.

Sample data / Initial training The lexical resources brought to the tutorial were considerably varied. They included different types of dictionaries (some digitised, some born digital with no explicit semantic markup) such as general monolingual, bilingual and etymological dictionaries as well as a dictionary from a language documentation field project (see table 1).

We asked the participants whether they successfully trained at least the first two models and thus were able to perform the general dictionary segmentation (page segmentation) and the dictionary segmentation (entry recognition). Despite the variety of their datasets, 100% of the answers were positive. This supports the assumption of the implemented cascading approach to be sample independent.

Type	Language(s)	Size
general, bilingual	Greek, English	≈ 17 000 entries
general, monolingual	Basque	≈ 16 000 pages
etymological, bilingual	Hittite (a languages of the ancient Near East), English	≈ 470 pages
lang. documentation	French, Yemba (an African language family)	≈ 2 1000 entries
lang. documentation	German (Bavarian dialects in Austria)	≈ 75 000 entries
general, monolingual	English	≈ 370 pages

Table 1: Dictionaries experimented with during the tutorial. Note that two participants worked on the same resource and another two used the resource provided by us.

Creating training data Two questions focus on the usability of the graphical annotation of the training data using oXygen’s author mode. None of the participants found

¹⁶<https://lexmc.sciencesconf.org/>

¹⁷see instructions at https://github.com/MedKhem/grobid-dictionaries/wiki/Docker_Instructions

¹⁸A more detailed description of the conditions of the experiment can be found in a blogpost at <https://digilex.hypotheses.org/250> as shared by one of the participants.

¹⁹<https://goo.gl/Zt2gDy>

graphically marking the training data a hard task and six qualified it as a straightforward process. Compared to creating the training data by manipulating the XML structure directly with a text editor, most of the participants (seven) confirmed that the graphical approach was easier.

Training workflow Although just two participants could finish the training for all models of the tool, all those who were not able to train the remaining models during the tutorial expect to be able to complete the training on their own. Moreover, all participants reported to be confident that they were able to re-apply what they learnt on other lexical resources. It's important though to clarify why some users could not successfully train all of the models until the end of the tutorial. This was mainly due to the fact that participants were free to attend only parts of the tutorial sessions and due to the considerable long time spent for downloading the huge Docker image with the available internet connection.

Future use of the tool Based on the apparently successful mastering of the training workflow, all but one participant declared to be willing to continue using GROBID-Dictionaries after the tutorial. It is worth noting that the participant who does not intend to continue using GROBID-Dictionaries is working with non-lexical data and still plans to adapt the parent project GROBID to his type of data.

Having motivated inter-disciplinary experts participating in the tutorial as well as testing the tool on new lexical samples provided us with the opportunity to spot some issues and several possible improvements. We could fix some of the minor triggered implementation issues in the course of the tutorial. Other issues have been filed as new tickets on GitHub, e. g. issues concerning the treatment of lexical entries that stretch over more than two pages in print. Some serious technical issues related to GROBID core still need to be overcome such as the support for some classes of special characters which are wrongly encoded in the preprocessing of the raw input text. The annotation guidelines should also be further refined to provide clearer definitions of constructs to be annotated, such as related entries.

6. Conclusion

Where Khemakhem et al. (2017) presented the basis of the approach to implement GROBID-Dictionaries and first experimental results, this paper aims to provide an in-depth description of the machine learning system with a focus on its architecture, technical setup and the training workflow. Enhancing the usability of the tool has been addressed as a fundamental feature given the fact that the tool is in its early development stage and the involvement of end users is a key factor for the evolution of the tool. Therefore several measures have been implemented to guarantee a straightforward installation and user friendly annotation process. The exposure of the tool to real users has confirmed many of our choices to alleviate the challenges of a complex ML workflow. This experiment also provided us with a possibility to promote the tool as well as to collect in-depth

feedback, which will help us to efficiently set our priorities. The recent version and setup of the tool, presented in this paper, do not only enhance its usability but also support the reproducibility of findings resulting from its use.

7. Acknowledgement

This work has been supported by the “Pooling Activities, Resources and Tools for Heritage E-research Networking, Optimization and Synergies” (PARTHENOS) project.

8. References

- Budin, G., Majewski, S., and Mörtz, K. (2012). Creating lexical resources in TEI P5: a schema for multi-purpose digital dictionaries. *Journal of the Text Encoding Initiative*, (3).
- Khemakhem, M., Foppiano, L., and Romary, L. (2017). Automatic Extraction of TEI Structures in Digitized Lexical Resources using Conditional Random Fields. In *electronic lexicography, eLex 2017*, Leiden, Netherlands, September.
- Lopez, P. and Romary, L. (2015). Grobid - information extraction from scientific publications. *ERCIM News*.
- Měchura, M. B. (2017). Introducing Lexonomy: an open-source dictionary writing and publishing system. In *electronic lexicography, eLex 2017*, Leiden.
- Pustejovsky, J. and Stubbs, A. (2012). *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*. "O'Reilly Media, Inc."