Nasḫī – an Efficient Tool for the OCR-Aided Transcription of Printed Texts

Andreas Büttner
University of Würzburg
Würzburg
andreas.buettner@uni-wuerzburg.de

ABSTRACT

This paper presents a simple yet effective solution for the transcription of printed texts. Our tool consists of a web-based user interface that provides an easy-to-use and ergonomic workflow and a collaborative environment for the philologists while allowing them to profit from machine learning OCR technology. As the targeted use case is not mass digitisation but the creation of accurate citable digital editions, the user interface for ground truth production and post correction is built to provide the means for rapid proofreading while minimising the amount of errors. The productivity of the setup is further improved by enabling progressive OCR training and recognition in the background to constantly increase the accuracy of the predictions.

The advantages of the application are showcased in the second part of the paper by documenting our experiences utilising it for digitising Arabic and Latin texts. Over the course of several months the tool has been used to create transcriptions of a wide range of sources, among them challenging early modern editions and Arabic scripts, producing a large amount of reusable OCR training data as a positive side effect.

Finally, there will be a discussion of possible future extensions of the tool and of how it could be adapted to fit the needs of other digitisation projects.

KEYWORDS

transcription, OCR, post-correction

ACM Reference Format:
Andreas Büttner. 2019. *Nasḫī* – an Efficient Tool for the OCR-Aided Transcription of Printed Texts. In *Proceedings of … (unpublished)*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Why would anyone manually transcribe documents these days when Optical Character Recognition (OCR) and Handwritten Text Recognition (HTR) with constantly shrinking error rates are available? Putting those cases aside where there is clearly no sensible manual way – for example, when creating a searchable version of several years' issues of a daily newspaper or similar huge collections of scans –, there are various technical and philological considerations that might lead to a decision against an automated approach.

From the philological point of view, there is the demand for *at least* 100% accuracy of the result. A proper edition should not only reproduce the underlying material, but should make and document justified editorial decisions, e.g. whenever the material is unclear or does not make sense. As long as a general artificial intelligence is still fiction rather than science, we need to rely on competent human readers to create a reliable resource from the digitised image.

On the more technical side, there are some aspects that tend to discourage especially smaller projects from adopting OCR or HTR tools. One of them is the preconception that OCR does not work for some historic typefaces or scripts. The next step involves the intricacies of installing, using, and training own models for that kind of software, most notably when the amount of text to be taken from a single edition is relatively small.

Considering the fact that the text has to be corrected manually anyway, it doesn't seem unreasonable to avoid the effort of constructing an OCR or HTR workflow and toolchain as well as the often cumbersome task of segmenting regions and lines, and to start transcribing by hand right away. But even then mistakes like typing errors or even skipping lines have to be compensated for, e.g. by having a text transcribed twice (rekeying) and comparing the results.

This shows that deciding whether or not to choose OCR over manual transcription requires weighing multiple factors. The present paper aims to demonstrate how with moderate means and mostly existing software components one can put together a workflow that is focused on accurate manual transcription while still profiting from machine learning. That can help a small team to speed up the production of high-quality digital editions. It is built around a simple yet efficient transcription user interface and allows for the OCR processes to be run independently in the background or remotely.

After briefly surveying existing software solutions for similar purposes, the architecture of our tool, the integration of components, and the user interface will be described. The second part of the paper outlines some experiences gathered while using it for transcribing Latin and Arabic texts. In the end, there will be a discussion of possible additions in functionality and of possible steps needed for using the system in other contexts than the one mentioned.

2 RELATED WORK

As the task of digitally transcribing a scanned document does not, in principle, require more than some kind of text editor and an image viewer, it is not surprising that there are countless solutions available to optimise and facilitate the user experience and the procedures involved in managing the data. Only a selection of the most relevant and prominent options can be discussed here.

The *Wikisource* project[1], for example, builds upon the collaborative features of a Wiki to crowdsource the proofreading of text generated by OCR. The user interface shows a text input field and the page image side by side. There are, however, no special tools for managing or retraining OCR engines mentioned on the website of the project.

If otherwise the page images were split into lines, one could simply rely on the scripts included with open source OCR software (e.g. *ocropus*[2] or *kraken*[3]) to combine these images and the corresponding OCR text into a HTML file and correct the text line by line. The generated ground truth text can later be exported and used for training an OCR model. As all of the interaction with the transcribing user happens inside the browser, these tools could easily be integrated into a web-based application. This has, as of now, not been realised, though.

A more refined approach is implemented by the *PoCoTo* [7] post correction tool. It includes a user interface that allows to correct OCR output word by word, assisted by concordance tools and an error profiler in the background. The program is intended for local installation and a single user, but a web version is currently under development.[4]

The proprietary *Aletheia*[5] (see [1]) annotation software could also be used to transcribe text from scanned images. Components for a web-based version of the tool are available as open source software, but the user interface provides only rudimentary text editing functions.

The most complete solution for the task of OCR or HTR aided transcription is provided by the *Transkribus*[6] (see [3]) platform. It consists of a web service that handles the storage of the documents, carries out training and recognition in the background and provides a complex user interface for the editing via a client software as well as a web based interface for the same purpose. Unfortunately, although the platform can be used free of charge, only some parts of the software are open source, which means it cannot be run on one's own hardware nor easily be customised.

Since there are many existing components – OCR engines, layout recognition, transcription user interfaces – under open source licenses, it seems like a viable aim to try to put together an environment for the collaborative transcription of documents that can be run, used, and customised by a smaller digitisation project.

[1]https://en.wikisource.org/wiki/Wikisource:Transcription_Projects
[2]https://github.com/tmbdev/ocropy
[3][4], http://kraken.re/
[4]https://github.com/cisocrgroup/pocoweb
[5]http://www.primaresearch.org/tools/Aletheia
[6]https://transkribus.eu

3 NASẖĪ

The paramount aim of the ongoing development of the *nasẖī*[7] transcription environment is to create a platform for the digitisation of the Arabic-Latin translations corpus[8] at the University of Würzburg. It is projected for five to ten researchers to be able to simultaneously segment, transcribe, and comment on scans of historical and modern printed editions in Latin, Arabic, and Greek language. The transcription is planned to be based on an adaptive OCR running in the background, continuously improving while the manual correction of errors proceeds.

3.1 Architecture of the System

The decision to split the OCR from the main application was made because of two reasons. To begin with, there is a lot of ongoing research in OCR software, which means that code in this area has to be fixed, updated or improved more often than the data management and user interface components. Since this part needs special treatment anyway, it does not need to be accessible from the graphical user interface until all components are stabilised to a high degree. The other reason is that artificial neural networks greatly profit from special hardware like GPUs or even TPUs that can be found in high performance clusters or, at the least, gaming PCs, but not on the average university web server. The latter, on the other hand, is required to provide stable and continuous access for the contributors to the image data and the user interface.

The back-end of the application is freely available, developed as open-source software mainly in the Python programming language and can be installed either as a Python package[9] or by downloading from GitHub[10]. It stores and updates the texts in PAGE XML format[11] (see [5]) in a database, handles user authentication, and provides a RESTful API for the front-end and OCR processes. In addition, it handles user access to the layout segmentation module and allows to import segmented pages from there in connection with running a line segmentation job on a task queue.

Image files are expected to be stored in a directory on the machine running the application in the PNG image format.[12] This means that converting and pre-processing the scans, deskewing, binarising, and renaming according to actual page

[7]The name is derived from the Arabic script قلم النسخ. To simplify its usage, the author hereby grants the permission to pronounce it analogically to the Japanese pear *Pyrus pyrifolia*.
[8]http://arabic-latin-corpus.philosophie.uni-wuerzburg.de
[9]https://pypi.org/project/nashi/
[10]The source code and documentation is available on https://github.com/andbue/nashi
[11]https://www.primaresearch.org/tools/PAGELibraries
[12]The naming convention for the images, <page number>.{raw|nrm|bin|col}.png was adopted from the *ocropus* OCR program.

or folio numbers is not in the scope of the functionality of *nasḫī*.[13]

3.1.1 Layout Analysis. For the layout analysis task, the *LAREX* [6] application was integrated as an optional module. It provides a semi-automatic method to quickly assign region types to selected areas on the page scan. To keep the *LAREX* code unchanged instead of adopting it to the filesystem and database structure, *nasḫī* provides the means to link only the required binarised images to the input directory of *LAREX* and, after saving the results, to import the generated PAGE XML files.

Since *LAREX* only allows to annotate page regions – e.g., in order to separate content from marginalia, headings, or page numbers –, a second step is required to segment each of these regions into single text lines. While the first task is quite difficult to perform automatically because of the often complex layouts especially of early printed books, the line segmentation can be run with satisfying results without user interaction after regions like columns and drop capitals have been separated.[14]

If *LAREX* is used again later in the process to correct errors, there is a danger of incongruency between the newly generated segmentation and the existing one. At the current stage of development, *nasḫī* avoids this problem by simply discarding the existing document. To prevent the user from accidentally deleting already transcribed text, the pages that have been edited before can be simply excluded from the list of files that are selected for being imported and overwritten.

3.1.2 OCR. As mentioned above, the module for OCR training and recognition has been separated from the main application and can be run on a remote machine. It uses the *Calamari*[15] [8] OCR engine that is developed at the University of Würzburg and can be significantly accelerated through the use of appropriate graphics hardware.

The usual OCR process consist of five steps:

(1) downloading and parsing the PAGE XML for a book from the server running the transcription interface,
(2) extracting the already transcribed text and cutting the line segments from the page images according to the coordinates given in the XML files,
(3) normalising the resulting images and training a model using text-image pairs,
(4) predicting the text on not yet transcribed lines using the model, and
(5) sending the results back to the server.

To improve the accuracy of the predictions, especially when there are still only few transcribed lines available from the same volume, the results can be improved by fine-tuning a model trained on existing ground truth data instead of starting from scratch. This means that the process of the transcription can be expected to accelerate on two levels: While working on a single book or a collection of books printed using the same typeface, the quality of the predicted text improves with each iteration of the training, thereby reducing the amount of user interventions when proofreading the text. On a higher level, the progress of a project transcribing texts written in a certain language or printed at a distinct period can profit from the growing stock of already transcribed material.

[13]There are open source components available for these tasks, e.g. *ScanTailor* (http://scantailor.org/) and the *ocropus-nlbin* script.
[14]The code included in *nasḫī* for this task has been taken over from *ocropus* and *kraken*.
[15]https://github.com/Calamari-OCR/calamari

3.2 Transcription User Interface

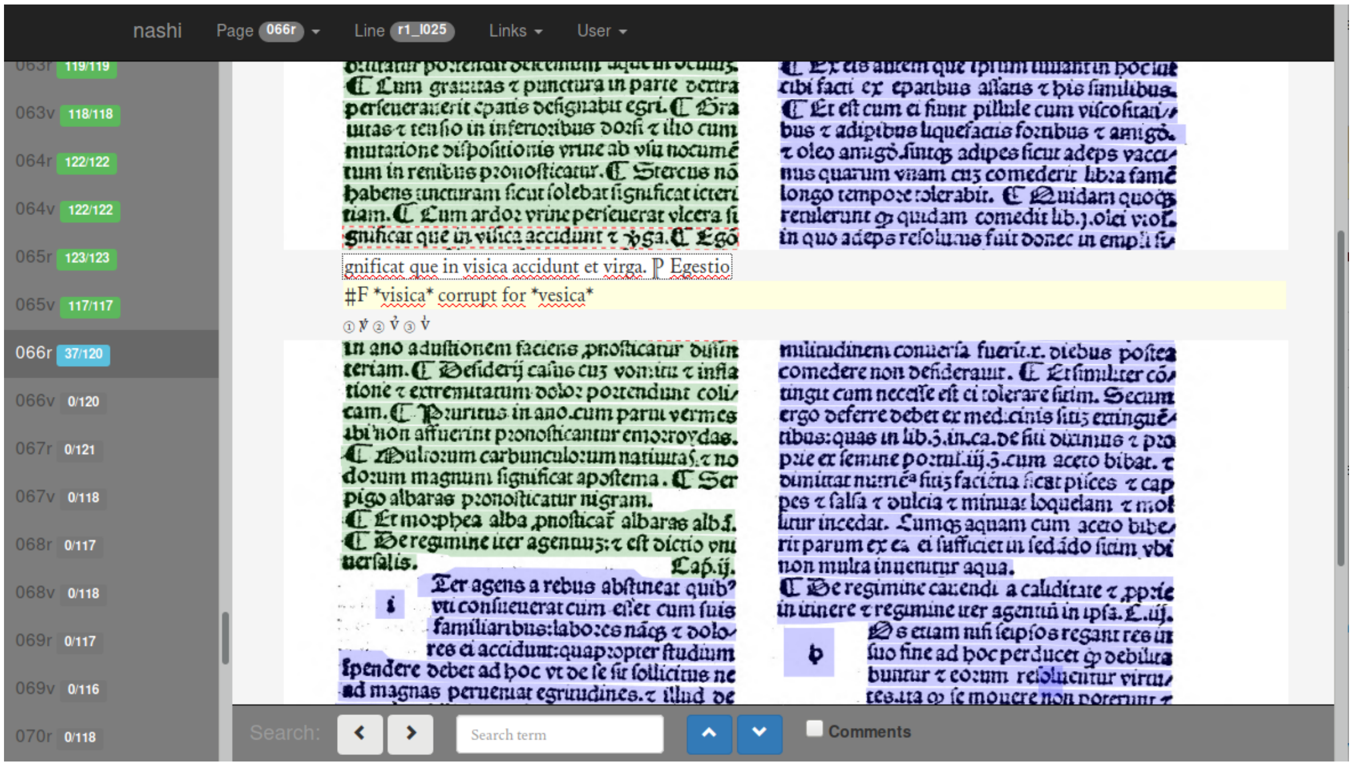
While from a technical perspective the graphical user interface might be a far less complex part against the backdrop of the machine learning techniques deployed in the back-end, it is far more decisive for the user's work then any minor improvement of the character error rate. Accordingly, the development of *nasḫī* is aimed at providing an environment that exhibits as few obstacles as possible.

In general, there are two types of transcription user interfaces: the ones focused on single line images, and the ones that display a larger part of the page. The ergonomic advantages of the former, featuring a minimal distance between image and display, are usually bought at the cost of obfuscating the context of the line. The latter, on the other hand, face the problem that either the user risks losing the line when switching between separate positions on the screen, or has to deal with some kind of editable overlay that partially blocks the image. The idea behind the *nasḫī*-interface (see 1) is to combine both approaches, trying to preserve their advantages while avoiding both their disadvantages. When a line is selected by the user, the page image is split horizontally just below the lowest point of the polygon enclosing that line. An input line containing the predicted text is shown in the space between the two parts of the page and stretched until it fits the length of the printed line of text. Keyboard shortcuts allow the user to select certain alternatives to the standard input character set, zoom in, open an additional input field to add comments, and finally mark the current line as correct and proceed to the next.

Accidentally confirming the correctness of a line would possibly carry OCR errors into the finished text. To prevent this from happening, the user is required to press a certain key combination to save a line to the server. The progress of the transcription is visualised by the colouring of the line regions on any page and a status indicator in the main view of the application.

In some cases, the line recognition fails, produces line segments stretching over more than one line, introduces additional lines, or simply cuts off parts of a line. To manually correct this kind of error, a second input mode has been implemented that, once activated, allows the user to remove, add or edit the points defining the polygons around the lines.

Navigation inside the OCR text, comments, and transcription is facilitated by a simple search function. The user can quickly search for any string contained in the text or comment, and jump to the next or previous line in the document that has been annotated with a comment.



gnificat que in visica accidunt et virga. P Egestio

#F *visica* corrupt for *vesica*

Figure 1: The transcription user interface.

In addition to that, some capabilities for search and replace throughout a book and the administration of different text versions – e.g. produced by different OCR models or engines – have been implemented. The user could, for example, copy and save one transcribed or recognised version of the text by duplicating all the respective nodes and assigning them a new index attribute at once. By that, it is later possible to return to an earlier stage or to evaluate and compare several variants of the text kept together in one document.

Of course, the set of features described here does not constitute a complete graphical editor for every annotation that can be stored in a PAGE document. It can, nonetheless, already be used as a productive tool, as the following paragraphs are going to show.

4 DEVELOPMENT AND USE

Nasẖī's development is part of the Arabic and Latin Corpus project at the University of Würzburg. As a consequence, many aspects of the user interface and the functionality are optimised for the specific requirements and aims of this particular research area. This can be seen as a downside on one perspective: *Nasẖī* does not (yet) provide a solution for the needs of any general digitisation project, as will be discussed below. Nevertheless, it may serve as a blueprint for such kind of projects in the humanities, showing how with reasonable efforts a custom made software solution integrating current OCR research is viable. With this in mind, the following remarks regarding our hands-on experience might be instructive.

4.1 Arabic-Latin Corpus

The Arabic and Latin Corpus project aims to create a digital collection of philosophical and scientific texts translated from Arabic into Latin from the 10th to 14th century. The collection is meant to provide the material for research in the translation movement, e.g. using stylometric analyses, but also to create citable digital versions of texts not available in modern critical editions. While the former method works quite well on messy OCR output (cf. [2]), the latter requires philological scrutiny and manual proofreading, so that the resulting edition is accurate to a level at which it can be cited in lieu of the original source. While a LSTM-based OCR engine reaches character error rates of less than 3% even on Arabic text or early printed materials, the necessary training data, too, has to be prepared by a skilled reader who is able to understand the text and its content.

The user interface in *nasẖī* integrates a basic annotation feature into the transcription and proofreading task. Due to the limitations of the restriction to the line level, this is not as fine-grained as it would be in a TEI annotation environment, but it nevertheless turned out to be one of the most helpful components in everyday work. The possibility to mark and highlight uncertain or illegible parts of the text and thereby exclude them from OCR training or postpone decisions for later discussion is a central part of philological activity and therefore has to be represented in the user interface.

Another aim in developing *nasḫī* was to allow a division of responsibilities while minimising the cost of collaborating, communicating, and exchanging data. The web-based architecture of the application centralises all work necessary for software installation, OCR, and image preparation, which means the contributors working at the text level may solely focus on the task of transcription. In addition, the task of manually segmenting the texts can be completed by others who may not even need to understand the language of the text, without there being a risk of data loss through exchanging and saving results at various stages of the process at different locations.

As the advantages of using *nasḫī* became visible, we started using it to proof-read existing OCR output from earlier stages of the project. Documents created with commercial OCR software could easily be converted from PDF or different XML formats to PAGE XML. This allowed us to skip the manual segmentation steps and quickly increase our collection of ground truth data.

4.2 OCR and Abbreviations

Many of the Latin versions that are to be digitised for the Arabic and Latin Corpus have been immensely influential throughout Europe and were widely read and distributed as printed editions compiled in the 16th century. The style of those prints is still highly reminiscent of manuscripts, in regard to the layout as well as the partially idiosyncratic composition and use of the fonts. Words are shortened and characters left out in ways that are often motivated by the attempt to fit the text in the space available at a particular region of the page. To resolve the resulting ambiguities, one has to rely on contextual information and knowledge about the language – the latter being exacerbated by the lack of orthographic rules in medieval Latin.

Our first attempts aimed to produce a diplomatic transcription by representing all of the abbreviations with appropriate unicode equivalents. This was to ensure that an unambiguous relation between any printed word and its transcription existed. The drawback of this strategy is a much more complex input process which involves many diacritical marks and special characters. Also, it requires another step to resolve all the abbreviations after finishing the transcription. This turned out to be much more time consuming than expected, even after building a custom tool that allows to search and replace many occurrences of a sequence of characters at once, because to avoid erroneous replacements, every single one of these occurrences has to be read again with a sufficient amount of context.

After many hours spent with resolving those abbreviations, we abandoned the diplomatic transcription and started to transcribe the final version straight away. As expected, this impaired the accuracy of the OCR output, since in some cases the transcription can only be guessed from the line image. Nevertheless, the likelihood of this guessing being correct tends to improve slowly as the amount of training material increases, and to correct the remaining mistakes is a task much less tiresome than the combined efforts of first transcribing special characters and later going through all the occurrences again.

Of course, this partially devalues the resulting material as potential ground truth for a more general approach to digitising similar texts that would entail, for example, first creating a diplomatic digital version from the scanned images and then trying to apply language models to resolve the abbreviations and to correct for errors. In this regard, the problem highlights the importance of taking into account the human factor when designing a digitisation workflow.

4.3 Evaluation of Productivity

Over the past year, *nasḫī* has been used to digitise 29 texts of different lengths for the Arabic and Latin Corpus. More than 40,000 lines of Arabic and 20,000 lines of Latin text from modern editions as well as 80,000 lines from early prints have been segmented and transcribed. While the speed-up in comparison to full manual transcription was clearly recognisable, the prognostic value of any further quantitative analysis for the expectable output would be limited by the plethora of factors involved.

Obviously, the speed of transcribing a text depends on the quality of the scan and the scanned material, but also on the complexity and content of the text, the expertise of the transcribers, and the time needed to look up unknown words or abbreviations in special dictionaries. As we have seen in the preceding paragraphs, the level of accuracy or interpretation required for a digital edition also heavily influences the strategy of transcription and its progress.

What can be said, though, is that *nasḫī*, due to its modular architecture and the simple construction of the user interface[16], can be easily adopted to account for the requirements of a particular project and allows for a division of work according to individual skills. In the Arabic and Latin Corpus project, this included keeping pre-processing, segmentation, and transcription tasks apart, handling the OCR components in the background or remotely, and re-configuring the user interface according to the needs of the participating researchers.

5 CONCLUSIONS AND FUTURE WORK

The description of *nasḫī* and its use has shown how an open source, OCR aided, and collaborative transcription workflow can be built and successfully utilised for the digitisation of complex historical sources. An innovative transcription user interface facilitates the task of proof-reading OCR output and thereby helps to produce highly accurate digital versions of printed texts.

Since it has been developed specifically for our project, its general applicability depends on further improvements of

[16]The user interface is built using standard web technology (HTML, CSS, JavaScript and SVG).

the software and on testing in different use cases.[17] This would entail a better and easier control of many parameters – e.g. of the font settings, the line segmentation, or OCR processes – by the user. Also, so far, the functions for manual corrections of the segmentation are far from complete. Further development of the layout recognition module towards a machine-learning-backed approach would improve the overall productivity.

On a more general level, some considerations regarding the utility and accessibility of the data produced by projects transcribing and annotating with tools like *nasḫī* would be fertile. How can training data for OCR be shared and accessed more easily? Which technical and institutional advances would improve the availability and usability of the digitised data for scholars? And, finally, how can the cooperation between computer science and humanities in digitising historical sources be the most fruitful?

REFERENCES

[1] C. Clausner, S. Pletschacher, and A. Antonacopoulos. 2011. Aletheia - An Advanced Document Layout and Text Ground-Truthing System for Production Environments. In *2011 International Conference on Document Analysis and Recognition*. 48–52. https://doi.org/10.1109/ICDAR.2011.19

[2] Dag N. Hasse and Andreas Büttner. 2018. Notes on Anonymous Twelfth-Century Translations of Philosophical Texts from Arabic into Latin on the Iberian Peninsula. In *The Arabic, Hebrew and Latin Reception of Avicenna's Physics and Cosmology*, Dag N. Hasse und Amos Bertolacci (Ed.). de Gruyter, Berlin and Boston, 313–369.

[3] P. Kahle, S. Colutto, G. Hackl, and G. Mühlberger. 2017. Transkribus - A Service Platform for Transcription, Recognition and Retrieval of Historical Documents. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 04. 19–24. https://doi.org/10.1109/ICDAR.2017.307

[4] B. Kiessling, M. Romanov, M. T. Miller, and S. Bowen Savant. 2017. Important New Developments in Arabographic Optical Character Recognition (OCR). *Al-Usur al-Wusta: The Journal of Middle East Medievalists* 25 (2017), 1–13.

[5] S. Pletschacher and A. Antonacopoulos. 2010. The PAGE (Page Analysis and Ground-Truth Elements) Format Framework. In *2010 20th International Conference on Pattern Recognition*. 257–260. https://doi.org/10.1109/ICPR.2010.72

[6] Christian Reul, Uwe Springmann, and Frank Puppe. 2017. LAREX: A Semi-automatic Open-source Tool for Layout Analysis and Region Extraction on Early Printed Books. In *Proceedings of the 2Nd International Conference on Digital Access to Textual Cultural Heritage (DATeCH2017)*. ACM, New York, NY, USA, 137–142. https://doi.org/10.1145/3078081.3078097

[7] Thorsten Vobl, Annette Gotscharek, Uli Reffle, Christoph Ringlstetter, and Klaus U. Schulz. 2014. PoCoTo - an Open Source System for Efficient Interactive Postcorrection of OCRed Historical Texts. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage (DATeCH '14)*. ACM, New York, NY, USA, 57–61. https://doi.org/10.1145/2595188.2595197

[8] Christoph Wick, Christian Reul, and Frank Puppe. 2019. Comparison of OCR Accuracy on Early Printed Books using the Open Source Engines Calamari and OCRopus. *JLCL: Special Issue on Automatic Text and Layout Recognition* (2019).

[17] A *docker* image containing an complete open source OCR workflow and *nasḫī* as a post correction tool will be available soon.