Glyph Miner: A System for Efficiently Extracting Glyphs from Early Prints in the Context of OCR

Benedikt Budig*
Chair for Computer Science I
University of Würzburg
benedikt.budig@uni-wuerzburg.de

Thomas C. van Dijk
Chair for Computer Science I
University of Würzburg
thomas.van.dijk@uni-wuerzburg.de

Felix Kirchner
Digitization Center
University Library Würzburg
felix.kirchner@uni-wuerzburg.de

ABSTRACT

While off-the-shelf OCR systems work well on many modern documents, the heterogeneity of early prints provides a significant challenge. To achieve good recognition quality, existing software must be "trained" specifically to each particular corpus. This is a tedious process that involves significant user effort. In this paper we demonstrate a system that generically replaces a common part of the training pipeline with a more efficient workflow: Given a set of scanned pages of a historical document, our system uses an efficient user interaction to semi-automatically extract large numbers of occurrences of glyphs indicated by the user. In a preliminary case study, we evaluate the effectiveness of our approach by embedding our system into the workflow at the University Library Würzburg.

Keywords

Early Prints; Document Recognition; OCR; Glyph Extraction; Efficient User Interaction

1. INTRODUCTION

Early printed documents are a precious source of information for researchers of various disciplines and a remarkable part of our cultural heritage. Scans of such documents are widely available,[1] but their contents need to be extracted to make the most use of them. Particularly, optical character recognition (OCR) is necessary to make the contained text searchable and available to further analysis. While off-the-shelf OCR systems work well on modern documents, they have trouble with early prints due to lower printing quality, higher visual variance within characters and possibly poor conservation state.[2] Instead, general purpose OCR software (e.g. Tesseract [13]) has to be trained specifically to the early print to be processed. At best, the trained system can then also be applied to other books printed by the same workshop (*Offizin*). The process of training an OCR system for use on early prints is tedious and requires significant manual effort. Typically, a first step is to catalog all different glyphs that occur in the given corpus. Then, several "representative" occurrences of each of these glyphs are located and cropped from the scans. Finally, the OCR engine can be trained on this set of glyph examples. Each step in this process requires a considerable amount of fine-tuning and domain knowledge – both of the OCR system and of the characteristics of the particular historical document.

In this paper, we describe a new system that replaces part of this pipeline with a more efficient workflow: finding (many) examples of the various glyphs. The main feature of this system is an efficient user interaction to learn the parameters of a template matching algorithm. As output, our software provides the detected glyph occurrences in the PAGE XML format, which can then be used to train (for example) the Tesseract OCR engine.

After reviewing related work (Sect. 2), we describe the features and contributions of our system (Sect. 3). We demonstrate the effectiveness of our approach in a small experiment, comparing our tool to the current workflow at Würzburg University Library, which relies on Aletheia [5] (Sect. 4).

2. RELATED WORK

Since the digitization of early prints has seen increasing attention over the last years, efforts have been made to automatically extract the contents of these documents. Off-the-shelf OCR systems like Tesseract [13] and ABBYY FineReader[3] need to be specifically trained to obtain fair results on early prints. This is a nontrivial task [10]. To make this procedure more convenient for users, several tools simplifying the construction and tuning of training sets have been developed. Clausner et al. [5, 6] introduced the ground-truthing system Aletheia, which provides a convenient user interface for annotating documents. It is focused on the annotation of regions, text lines and glyphs in a given page; for this task, some semi-automated tools are provided. The gathered information can be saved in the PAGE XML for-

*Corresponding Author

[1]For example, more than 17 million scanned pages are available through the Early English Books Online (EEBO) project. `http://eebo.chadwyck.com/home`

[2]This was also identified as a key problem by IMPACT, a European research project focused on digitizing historical printed text. `http://www.impact-project.eu/`

[3]`http://www.abbyy.com/finereader/`

JCDL '16, June 19 - 23, 2016, Newark, NJ, USA
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4229-2/16/06...$15.00
DOI: `http://dx.doi.org/10.1145/2910896.2910915`

mat [11] for further processing. The detected occurrences of glyphs can be used to train the Tesseract text recognition system, but first need to be checked for classification mistakes (and converted to appropriate file formats). Torabi et al. [14] introduced Franken+, a tool that aids this process by reading PAGE XML files, building a database of glyphs and creating synthetic training images for Tesseract.

The Gamera framework by Droettboom et al. [9] uses a different approach: It is a generic toolkit for building custom document recognition applications. It implements various image processing algorithms, including template matching and connected component analysis, which can be used to detect glyphs. Gamera also allows the training of classifiers to distinguish the detected glyphs and can thus be used as an OCR system (potentially using additional plug-ins [7]). The software exposes a variety of algorithms and its user interface is firmly aimed at technical experts who develop recognition processes. Off the shelf, it cannot be considered appropriate for users without a technical background in image processing.

There are various ways to arrange workflows for the digitization of early prints; see Pletschacher et al. [12] for a recent evaluation of options. In the current paper, we evaluate our system in the context of the digitization workflow at Würzburg University Library. First, a table of all occurring glyphs is created (using a spreadsheet software). Next, multiple representative occurrences of each glyph are obtained using Aletheia. Finally, this data is used to train Tesseract OCR through the Franken+ toolkit. This workflow is advocated by Torabi et al. [14] in their eMOP project and follows a sample use case for Aletheia;[4] Clausner et al. [6] describe a similar approach to train Gamera's OCR engine. As noted, our system would replace Aletheia (and the spreadsheet) and produce the table of representatives for each glyph. This would also be useful in a workflow based on Gamera.

Note that the problem of locating many occurrences of a given glyph is not limited to the training of OCR on early prints. In fact, the algorithmic underpinnings of our system were originally developed for locating pictographs and characters on early maps [3].

3. GLYPH MINER: SYSTEM DESIGN

The system presented in this paper simplifies the extraction of multiple occurrences of given glyphs from large corpora of printed text. It is based on an earlier system that we designed for metadata extraction from early maps [3]. In this section, we present our extensions to this system to make it suitable for OCR workflows of early typeset prints. The objective for the new system is to quickly detect a large set of samples for a given glyph, employing efficient user interactions. Note that it was specifically designed towards usability, to enable users without technical knowledge of OCR (e.g. researchers in the humanities) to contribute to the digitization workflow.

We start with two remarks. The system assumes that the images have already been binarized, a step that is commonly included in many digitization pipelines. We further note that depending on the particular use case, it might not be necessary to import all pages of a particular corpus, as long as the desired glyphs occur often enough. This can

[4]http://www.prima.cse.salford.ac.uk/tools/Aletheia/Usecases

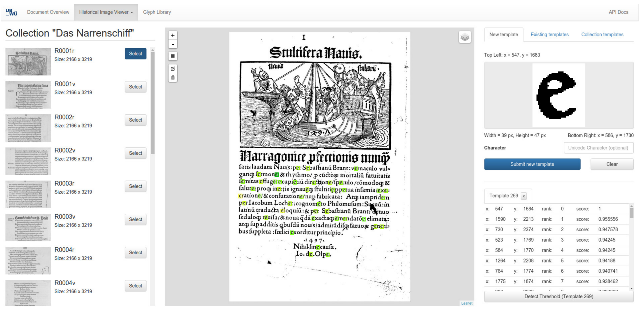

Figure 1: Interface for handling a collection of pages from an early print. In the center, all occurrences of the glyph "e" on a particular page are presented.

increase the system's runtime performance without significantly impacting the quality of the results (see Sect. 4).

Mining Glyphs

In the first step, the user indicates a *template* of the glyph he or she is interested in by drawing a rectangle on one of the pages; the corresponding user interface is shown in Fig. 1. Once a template has been provided, the system runs a template matching algorithm on all images of the collection in parallel. This is based on the assumption that occurrences of the same glyph *look* the same, which is reasonable considering that they were printed using the same types. A renewed investigation of template matching in the context of OCR has also recently been advocated by Caluori and Simon [4]. In contrast to glyph segmentation based on connected components of ink (as used, for instance, in Gamera's classification module), this approach is reasonably stable against smudged printing, suboptimal binarization, and glyphs touching each other.

The template matching yields a set of candidate matches together with respective similarity scores, but we do not know which of these matches are in fact semantically correct (that is, depict the desired character). We model this problem as a classification task and apply active learning: The system iteratively presents carefully selected batches of candidate matches to the user, such that his or her time and effort is spent where it is most useful. Typically, the training process can be completed well within a minute and the learned model is applicable to all pages of the corpus. Fig. 2 shows our classification interface, which can be used on desktop PCs as well as on mobile devices. A complete description of this system's algorithmic and machine-learning aspects as well as its user interfaces is available elsewhere [3].

Glyph Library

All candidate matches and their respective classification models are stored in a database. Our system presents the matches to the user in the *Glyph Library* view: see Fig. 3. Here, the user can inspect and filter glyphs aggregated from all images of the chosen collection; by default, only the matches classified as positive are shown. The glyphs are presented in descending order of their similarity scores, allowing the user to easily find particularly clean occurrences of each glyph (as well as particularly distinct edge cases). The selected glyphs can be exported in various formats, including PAGE XML (which can be processed further by Franken+).

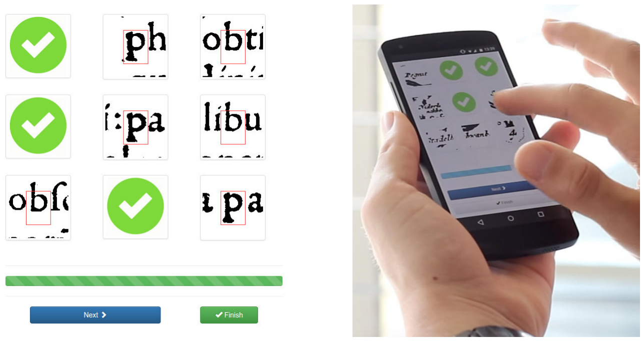

Figure 2: The classification interface as shown on a PC and a smartphone. The user's task is to touch all tiles showing the desired glyph (here: "p", three already checked), thereby training the classifier.

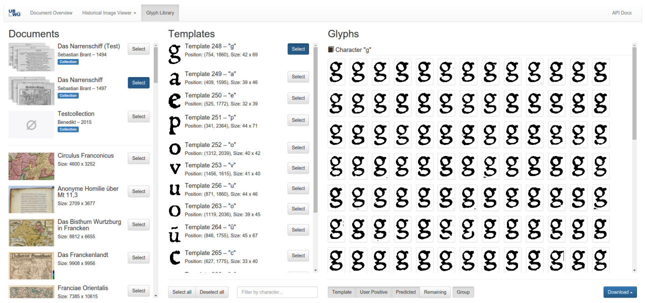

Figure 3: The *Glyph Library* view is used to browse and export detected occurrences of glyphs.

Implementation

The implementation of our system is web-based and we have used it in the experiments of Sect. 4. The client side runs in a web browser, providing cross-platform compatibility. The web application communicates through a REST API with a server that is responsible for image handling and maintaining the database of glyph occurrences. Our modular approach and the API design make it possible to quickly change individual components of the system (e.g. integrate a more sophisticated template matching algorithm).

See our demo video of this system at:
`https://youtu.be/T-p_kIdsn6k`

4. EXPERIMENTS AND CASE STUDY

In this section, we present a preliminary case study where we embed our system into the digitization workflow of the Würzburg University Library. Our experiments were run on an incunable printed in Basel in 1497, containing a Latin translation of Sebastian Brant's *Narrenschiff* (GW 5061). The Narrenschiff ("Ship of Fools") is considered an outstanding work in the history of German literature, being the most successful German book until Goethe's *Werther* almost three centuries later. It spectacularly combined the printing techniques emerging at the time, which now poses a considerable challenge to OCR systems. There exist approximately two dozen other prints using the same type inventory, to which OCR training results could be transfered.

All pages of this work were available as high-resolution grayscale scans, which we binarized with a fixed threshold. Unless otherwise noted, we ran the experiments on the first 20 out of 320 pages.

All parameters for the matching algorithm are learned automatically. This is in contrast to Gamera, which for similar tasks depends on a nontrivial set of parameters to be picked by hand (for filtering connected components by size, similarity thresholds, et cetera). Although Aletheia has the official use case of training OCR through Franken+, it is actually a ground-truthing system (including e.g. annotating the page layout), which hampers usability in our context. In contrast to these two systems, our tool outputs directly to Franken+, which is no longer the case for the latest releases of Aletheia.

Experiment: Glyph Miner on early prints

First we consider the precision and recall of Glyph Miner on two example characters. For both, we used our system for 10 iterations in the active learning interface. This takes approximately 90 seconds per character, including indicating the desired character, computation time, and the active learning procedure. We compare the output of our system to the glyphs actually contained in the pages; this ground truth was created manually.

Using a "g" glyph, the system achieved precision 1, recall 0.991, and F1-Score 0.995. There were only two false negatives, one match being wrongly classified and one occurrence not being detected by the template matching at all. With an "a", we achieved precision 0.941, recall 0.969, F1-Score 0.955. Note that for the latter glyph, the system has some difficulty with similar glyphs such as "ã" and marginalia of slightly smaller font size. The false negatives in this case were mostly caused by glyph occurrences undetected by the template matching algorithm. These quality measures are much higher than reported in previous work [3], where the same algorithms were applied to early printed maps with hand-drawn characters. This experiment shows that our system is suitable for finding repeat occurrences of glyphs in early typeset prints.

A particular class of false positives in our system occurs for characters that are visually contained within one another (for example, the character "m" might contain approximate occurrences of "n"). This is a known issue with glyph detection in general [8], but not an important one, since the user can work around it with clever template selection or in postprocessing. We do not specifically address this problem.

Experiment: Glyph Miner vs. Aletheia

Our second experiment was conducted with the help of an OCR engineer at the Würzburg University Library. In this experiment, we assess how our system could be used as an in-place substitute of part of their current workflow.

As the baseline experiment, the OCR engineer worked for 45 minutes using Aletheia following the existing workflow. In this time, he annotated roughly 1$^1/_3$ pages. His annotation consisted of 1251 occurrences of 65 different glyphs, with a median number of 7 occurrences per glyph. The seemingly slow progress has two main reasons. First, Aletheia's automatic segmentation of glyphs often fails and needs to be corrected manually. Second, the characters suggested by Aletheia are often wrong and must be checked and corrected.

In the contrasting experiment, the OCR engineer worked on the same pages for another 45 minutes, now using Glyph

Miner. In this time, he processed 26 different glyphs, arriving at 17 426 repeat occurrences, with a median of 498 occurrences per glyph. These repeat occurrences were extracted from the 20 pages of input given to the system. As a byproduct, Glyph Miner has trained a classifier which can be used to extract further occurrences from the remaining 300 pages of the corpus. For the glyph “a”, for example, this gives us a total of 21 189 occurrences; for the rarer “ct” ligature we find 648. This extrapolation took 168 seconds per glyph on a single quad-core 3.4 GHz PC, and can be trivially distributed further for additional performance.

In the 45 minutes alloted in the above experiments, Aletheia gets a better coverage of glyphs: 65 versus 26. On the other hand, the Glyph Miner finds a considerably larger number of repeat occurrences: a total of 17 428 versus 1251. The occurrences found with Aletheia are from the first 1⅓ pages; the Glyph Miner covers all of the first 20 pages. Using some additional computational time, the occurrences on the remaining 300 pages can be found fully automatically.

Of the 65 different glyphs covered by Aletheia, only 25 were backed by ten or more occurrences. The limited number of occurrences for the others glyphs may hamper proper training of OCR software. In contrast, the minimum number of occurrences per glyph in Glyph Miner was 29, with many more for most glyphs.

In the current workflow, the OCR engineer goes through a number of pages in full, until sufficiently many occurrences of all common glyphs have been found. Afterward, he or she searches through all other pages (potentially hundreds) to obtain enough examples of the rare glyphs. Our system should provide a particular speed-up for this second stage, because every rare glyph needs to be found only once: The other occurrences will be found by template matching. Evaluating this speed-up in practice would be time-consuming and was beyond the scope of our experiment.

5. CONCLUSION AND OUTLOOK

We have presented a system for efficiently extracting repeat occurrences of given glyphs from scans of early prints. The system can be embedded into a digitization workflow and combines template matching and active learning into an efficient user interaction. We have done a preliminary case study showing that our system has the potential to outperform state-of-the-art approaches in this use case.

The motivating application in this paper is training OCR software. We have shown that we are able to massively increase the number of examples available per glyph. Giving more examples to Franken+ sounds promising, but at present we have not evaluated if this actually leads to an increased OCR performance. Even if not all occurrences are used for training, the large number of glyph examples, sorted by quality, makes it easier for OCR engineers to compose a good training set. Emerging new OCR approaches based on deep learning would certainly profit from the large set of training data. Additionally, we note that a catalog of occurrences of glyphs can in itself be interesting, for example to date or attribute printed works [2]. Such catalogs can also be used to handle particular sets of glyphs in highly specialized digital libraries, for instance handling neumes in medieval chant manuscripts [1].

In future work, we want to further exploit the massive number of detected occurrences. One approach is to use this information to simplify the (manual) search for rare glyphs by fading out everything that has been identified before. Here, a semi-automatic approach to detect unidentified glyphs could also be feasible. Another direction of work is to explore if the high detection accuracy for single glyphs can be expanded into a complete OCR system.

Acknowledgments

We thank H.-G. Schmidt, Head of the Department of Manuscripts and Early Printed Collections and the Digitization Center at Würzburg University Library, and OCR engineer P. Beckenbauer. This research was partially supported by the German Federal Ministry of Education and Research (project KALLIMACHOS, reference ehuman-539-084).

6. REFERENCES

[1] L. W. G. Barton, J. A. Caldwell, and P. G. Jeavons. E-Library of Medieval Chant Manuscript Transcriptions. In *Proc. JCDL’05*, pages 320–329, 2005.

[2] M. Behr. *Buchdruck und Sprachwandel*. De Gruyter, 2014.

[3] B. Budig and T. C. van Dijk. Active Learning for Classifying Template Matches in Historical Maps. In *Proc. DS’14*, pages 33–47, 2015.

[4] U. Caluori and K. Simon. An OCR Concept for Historic Prints. In *Archiving Conf.*, pages 143–147, 2013.

[5] C. Clausner, S. Pletschacher, and A. Antonacopoulos. Aletheia – An Advanced Document Layout and Text Ground-Truthing System for Production Environments. In *ICDAR’11*, pages 48–52, 2011.

[6] C. Clausner, S. Pletschacher, and A. Antonacopoulos. Efficient OCR Training Data Generation with Aletheia. In *Short Paper Booklet of the 11th IAPR Workshop DAS’14*, pages 19–20, 2014.

[7] C. Dalitz and R. Baston. Optical Character Recognition with the Gamera Framework. In *Doc. Image Analysis with the Gamera Framework*, pages 53–65, 2009.

[8] M. P. Deseilligny, H. Le Men, and G. Stamon. Character String Recognition on Maps, a Rotation-invariant Recognition Method. *Pattern Recognition Letters*, 16(12):1297–1310, 1995.

[9] M. Droettboom, I. Fujinaga, K. MacMillan, G. S. Chouhury, T. DiLauro, M. Patton, and T. Anderson. Using the Gamera Framework for the Recognition of Cultural Heritage Materials. In *Proc. JCDL’02*, pages 11–17, 2002.

[10] M. Heliński, M. Kmieciak, and T. Parkoła. Report on the comparison of Tesseract and ABBYY FineReader OCR engines. *Improving Access to Text*, 2012.

[11] S. Pletschacher and A. Antonacopoulos. The PAGE (Page Analysis and Ground-Truth Elements) Format Framework. In *ICPR’10*, pages 257–260, 2010.

[12] S. Pletschacher, C. Clausner, and A. Antonacopoulos. Europeana Newspapers OCR Workflow Evaluation. In *Proc. HIP’15*, pages 39–56, 2015.

[13] R. Smith. An Overview of the Tesseract OCR Engine. In *Proc. ICDAR’07*, pages 629–633, 2007.

[14] K. Torabi, J. Durgan, and B. Tarpley. Early Modern OCR Project (eMOP) at Texas A&M University: Using Aletheia to Train Tesseract. In *Proc. DocEng’13*, pages 23–26, 2013.