Mass Digitization of Early Modern Texts With Optical Character Recognition

MATTHEW CHRISTY, ANSHUL GUPTA, ELIZABETH GRUMBACH, LAURA MANDELL, RICHARD FURUTA, and RICARDO GUTIERREZ-OSUNA, Texas A&M University

Optical character recognition (OCR) engines work poorly on texts published with premodern printing technologies. Engaging the key technological contributors from the IMPACT project, an earlier project attempting to solve the OCR problem for early modern and modern texts, the Early Modern OCR Project (eMOP) of Texas A&M received funding from the Andrew W. Mellon Foundation to improve OCR outputs for early modern texts from the Eighteenth Century Collections Online (ECCO) and Early English Books Online (EEBO) proprietary database products—or some 45 million pages. Added to print problems are the poor quality of the page images in these collections, which would be too time consuming and expensive to reimage. This article describes eMOP's attempts to OCR 307,000 documents digitized from microfilm to make our cultural heritage available for current and future researchers. We describe the reasoning behind our choices as we undertook the project based on other relevant studies; discoveries we made; the data and the system we developed for processing it; the software, algorithms, training procedures, and tools that we developed; and future directions that should be taken for further work in developing OCR engines for cultural heritage materials.

CCS Concepts: • **Applied computing → Optical character recognition**;

Additional Key Words and Phrases: Machine learning, digital humanities

ACM Reference format:
Matthew Christy, Anshul Gupta, Elizabeth Grumbach, Laura Mandell, Richard Furuta, and Ricardo Gutierrez-Osuna. 2017. Mass Digitization of Early Modern Texts with Optical Character Recognition. *ACM J. Comput. Cult. Herit.* 11, 1, Article 6 (December 2017), 25 pages.
https://doi.org/10.1145/3075645

1 INTRODUCTION

Special collections departments in libraries around the world contain documents published in English from 1476 to 1800, both in England and America. More than 300,000 documents—roughly 45 million pages—from this early modern era have been digitized by vendors for these libraries. However, the path to digitization was not ideal: these documents were imaged in the late 1970s, transformed into microfilm during the 1980s, and the microfilms digitized in the 1990s. Because of the state of reproductive technologies during the late 20th century, as well as the circuitous path to digitization (through microfilm), the image quality is very poor and bitonal, with no greyscale

This work was supported by an Andrew W. Mellon Foundation grant (31200645).
Authors' addresses: M. Christy, E. Grumbach, and L. Mandell, English Department, Texas A&M University, College Station, TX 77843; emails: {mchristy, egrumbac, Mandell}@tamu.edu; A. Gupta, R. Furuta, and R. Gutierrez-Osuna, Computer Science and Engineering Department, Texas A&M University, College Station, TX 77843; emails: {anshulg, furuta, rgutier}@tamu.edu.

https://doi.org/10.1145/3075645

images available. Furthermore, the original documents themselves, printed with premodern technologies, pose problems even for human readers of their pages, but much more so for optical character recognition (OCR) engines. For example, printed characters were not perfectly situated on a baseline, blackletter fonts were used, ink bled through the paper, and the typeface was broken and overworn. Moreover, these documents are aged: pages are missing, ripped, or blotted with handwritten marginalia and spilled ink.

For the sake of effective cultural research, the humanities require that these documents are available not as images but as keyed text. If all that is preserved are page images, some of them with very inconsistent and obfuscatory metadata, they will become part of a "dark archive"—preserved but fundamentally undiscoverable by search algorithms. So worrisome is this problem that a report commissioned by the European Commission [1] warned that without adequate preservation of and discovery tools for cultural heritage materials, Europe is in danger of "entering a new Dark Age." Recognizing OCR as central to this task, the European Commission funded the Improving Access to Texts (IMPACT) group for 4 years to improve OCR for cultural heritage materials.[1]

Preliminary research to be completed and published this year indicates that in the best OCR currently available, as many as 57% of search returns are missed when a user searches for a bigram of words that are hard for OCR engines to process because of early modern fonts, and 47% for the easiest bigrams [2]. This means that all work currently being performed in the field of cultural analytics on documents published before 1800—stylometry, word counts, topic modeling, cluster analysis, and feature extraction, as well as simple word search—is producing conclusions that are only about 50% reliable. As IMPACT put it when they started, "[Optical Character] Recognition [results] are poor or even useless. No commercial or other OCR engine is able to cope satisfactorily with the wide range of printed materials published between the start of the Gutenberg age in the 15th century and the start of the industrial production of books in the middle of the 19th century."[1]

Thus, our capacity to mechanically transcribe early modern texts with less than optimal page images must be improved. Engaging the key technological contributors from IMPACT to improve on their contributions (see Section 2), Texas A&M received funding from the Andrew W. Mellon Foundation to run the Early Modern OCR Project (eMOP), a 2-year, multipartner endeavor to improve OCR outputs for early modern texts,[2] specifically those from the Eighteenth Century Collections Online (ECCO) and Early English Books Online (EEBO) proprietary database products—approximately 45 million pages. This article describes the development of eMOP—its computer systems architecture, software tools, and data workflows. We discuss the challenges that we encountered to generate high-quality OCR outputs from the ECCO and EEBO collections, which result from poor image quality, variability in typefaces and page layout, and lack of ground-truth data. We will present computational techniques to address these issues and validation results.[3] The article concludes with a discussion of the lessons learned during the project, the significance of our findings, and ongoing work and future directions.

2 RELATED WORK

The work reported in this article draws from many threads developed over the past 30 years as large-scale digital libraries and high-speed networked computers have been transformed from specialized facilities available only to the few scholars lucky enough to be affiliated with elite universities, high-powered governmental laboratories, and the largest corporations to resources we now expect to be available as a common tool of the trade. Encouraged by these resources, collections of literary source material have been created, both those focused primarily on volume—large collections of materials without a particular thematic focus—but also others focused on specific academic disciplines. In the first category, examples include the ubiquitous Google Books[4] (launched in 2002), the

[1] http://www.impact-project.eu/about-the-project/concept/.
[2] http://emop.tamu.edu.
[3] Sections 2.1 and 4 of this article are derived from a previously published conference paper [3].
[4] https://books.google.com/.

pioneering Project Gutenberg[5] (initiated in the 1970s), the Million Book Collection[6] (also launched in 2002), and the HathiTrust[7] (launched in 2008). Pioneering exemplars of collections focused on a specific academic area of interest are Crane's Perseus Project[8] [4], Landow's Victoria Web,[9] Price's Whitman Archive,[10] McGann's Rossetti Archive,[11] and the William Blake Archive[12] (edited by Eaves, Essick, and Viscomi). The work associated with these pioneering projects naturally separated into acquiring digital representations of books and manuscripts, manually or automatically converting some or all of the textual portions into more computer-readable formats, and creating scaffolding to allow the use of the materials by scholars and other readers. Our project focuses on the critical aspect of the middle step—the conversion from images of text to machine-readable text (e.g., OCR).

As adumbrated earlier, many of the pages that provide the source for our work are bitonal scans from microfilm, created following the standards in use when they were first photographed and digitized but not as rich as would be scans created using more modern standards (e.g., see the conversion trade-offs described in Cornell Library's *Moving Theory Into Practice: Digital Imaging Tutorial*[13]). Thus, our algorithms must operate without information from greyscale or color spectra. The content itself also raises questions about design assumptions embedded in modern OCR programs. Changes in printing technology and standards from the early modern period to the present (printing practices only became "modern" around 1820 in the West) reduce the effectiveness of current OCR engines, which are tuned for the regularity and consistency of today's digitally generated printing rather than the variations in type and composition that result in the analog print process, and OCR engines certainly had not been developed to accommodate the vagaries of premodern print—until the IMPACT and eMOP projects began.

IMPACT focused on (1) producing ground-truth documents, for which they developed the Aletheia tool, discussed in Section 3.2; (2) developing language-specific resources—dictionaries and postprocessing routines containing corrections for each language's most common OCR errors; and (3) training an ABBYY Finereader engine to read as many types of documents as possible. IMPACT has not published the correctness rates that they achieved.[14] Based on personal communications with IMPACT consultants on eMOP, we believe that the historical OCR problem that IMPACT sought to address remained unsolved. Hence, eMOP sought to address the following issues:

(1) IMPACT used an engine (ABBYY Finereader) that is not open source; we chose to use Google's Tesseract, which is open source and built for enhanced line detection [5, 6]; thus, it outperforms ABBYY in finding line bases even when they are uneven.
(2) IMPACT used ABBYY Finereader's built-in image processing—its effectiveness decreasing as image quality decreases. Working on high-quality images provided by European libraries, the IMPACT workflow was not designed to deal with poor images.[15] eMOP was necessary because most of the early modern documents

[5] https://www.gutenberg.org/.
[6] http://www.ulib.org/.
[7] https://www.hathitrust.org/.
[8] http://www.perseus.tufts.edu/hopper/.
[9] http://www.victorianweb.org/misc/vwintro.html.
[10] http://www.whitmanarchive.org/.
[11] http://www.rossettiarchive.org/index.html.
[12] http://www.blakearchive.org/blake/.
[13] https://www.library.cornell.edu/preservation/tutorial/conversion/conversion-01.html.
[14] The ABBYY engine enhanced by IMPACT is available only if one joins the Centre of Competence, paying an annual fee, and arranges for an ABBYY license as well. The "IBM Adaptive OCR Engine" that includes the capacity to train the ABBYY engine that they created is not available for testing and use. It is unknown if it is available for purchase, but this option is not identified on their Web site.
[15] Page images of documents comprising the EEBO and ECCO collections have a history: the early modern holdings at the British Library were scanned and made into microfilm in the 1970s and 1980s, then the microfilm was transformed into digital images in the 1990s.

digitized to date are based on microfilm.[16] Thus, denoising algorithms were needed to handle the most problematic low-quality images (Section 2.1).

(3) IMPACT faced two challenges with ABBYY. First, as their experience demonstrated, training a single engine to handle multiple fonts led to poor recognition results. Instead, we decided to use a triage mechanism to sort documents according to the specific fonts used in printing them and then develop font-specific OCR engines for each type of font (see Section 2.2). Second, IBM Haifa developed for IMPACT the Concert tool, which presented a human being with all candidates that the ABBYY Finereader engine read as a particular letter. The humans were asked to eliminate the engine's incorrect identifications. However, the degree to which this improved OCR outputs was unclear. In our project, we instead focused on human-in-the loop for selecting the best instances of letter images that were properly identified (not vetting all images), as identifying a few example images was more directly related to the input needed to train the Tesseract engine.

The best OCR results available for the low-quality page images digitized from microfilm collections to date have been produced by the company Prime Recognition, which uses six of the top commercial OCR engines and a voting algorithm to choose the best guess for any word. We compare our results using Tesseract to Prime Recognition's results in Section 7.

2.1 Denoising OCR Outputs

Several studies have focused on postprocessing techniques to correct errors in OCR outputs by modeling typographical variations in historical documents (see Reffle and Ringlstetter [7] and Reynaert [8] and references therein). As an example, Alex et al. [9] proposed two OCR postprocessing methods for the problems of end-of-line hyphen removal and substitution of long-"s" (recognized as "f") to letter "s" (e.g., "fenfible" to "sensible"). Using dictionary-based methods, the authors reported a 12.5% reduction in word error rates. For these techniques to be effective, however, noise in page images must be removed in advance.

Certain image features, "noise," correlate with OCR performance. These include global properties, such as the amount of black background speckle, image sharpness, and uniformity, as well as local properties of the text, such as stroke thickness and continuity, and character/word height to width ratio [10]. A few studies have focused on improving OCR performance by preapplying image restoration techniques, such as deblurring, skew removal, and bleed-through removal. However, these techniques should not be blindly applied but should be used selectively based on the type of noise or degradation present in the document. For this purpose, Lins et al. [11] developed a method to identify five types of noise (bleed through, skew, orientation, blur, and framing) based on image features, such as palette, gamut, or number of foreground pixels. The authors found that the overhead of this noise classifier was far lower than running the image through all of the filters, some of which were not necessary for any given image. In related work, Sandhya et al. [12] developed a taxonomy of image noise in historical documents that extends beyond the five categories of Lins et al. [11]. Their taxonomy considered four types of noise sources: aging, digitization and storage, physical factors (e.g., folding, burn, bleed through), and document factors (e.g., varying fonts, mixed alphabets). More recently, Farahmand et al. [13] reviewed image-processing techniques for removing ruled-line noise, marginal noise, clutter noise, stroke-line pattern noise, background noise, and salt-pepper noise.

Novel to eMOP is using the document's structure detected by the OCR engine rather than the images themselves to identify pages that require additional processing and to direct that processing; this includes denoising and possible referral for human assistance.

[16]There was a big microfilming push in the late 20th century that includes EEBO, ECCO, Readex's North American Imprints, the Burney Collection of early newspapers (Gale), and 19th-century journal collections (ProQuest). More than 100 million page images comprising these collections were digitized.

2.2 Typeface Identification

Turning to the second topic, our evaluation of font classification draws from the observation that the fonts and layout of the historical documents vary substantially from one document to other, which makes training a single high-performance OCR engine difficult. Ait-Mohand et al. [14] proposed an HMM-based multifont OCR system that modifies its HMM model for each document to better recognize a specific font class. When tested on a dataset with 17 modern font classes, the authors reported a 98% recognition rate, which is very close to the recognition rates of monofont OCR systems used for single-font texts (99%). An alternative to adapting a multifont OCR model is to train multiple monofont OCR systems and direct documents to the respective OCR engine. However, this requires that the font class of each document be identified in advance.

In an extensive literature survey, Ghosh et al. [15] organize font recognition methodologies into two categories based on the extracted features: structure based, and appearance based. Structure-based methods extract the connected components of characters and analyze their shape and structure to recognize particular fonts. In contrast, appearance-based methods use features that can capture the visual appearance of the individual characters and the way they are grouped into words, lines, and paragraphs. These two categories are further divided according to whether they operate at the document, page, paragraph, or word level. Rani et al. [16] presented a character-level font identifier. Using Gabor and gradient features and an SVM classifier, they reported 99% average accuracy in identifying fonts on a dataset with 19,000 images of characters and numerals from 17 fonts for the English language and 14 fonts for the Gurmukhi language.

Whether they are used to classify documents, words, or characters, font identification systems incur a large cost to generate sufficient labeled data for training. However, recent studies have shown that active learning can be very effective in reducing the large overhead of labeling data [17, 18]. Unsurprisingly, active learning has begun to garner attention in the document analysis community. For example, Bouguelia et al. [19] proposed a semisupervised active learning algorithm for stream-based classification of documents into multiple classes, such as bank checks, medical receipts, invoices, or prescriptions. Compared to a model built with a fully labeled training dataset, active learning provided a 2% to 3% precision boost while using on average only 36% of the labeled data.

Based on these studies and the experience of the IMPACT group, we decided to use the monofont training approach and develop training sets that would be run on documents according to their font. We launched eMOP intending to identify fonts through metadata analysis—correlating early modern print shops with the typeface that they used, then sorting texts according to printer/font. That proved to be a monumental task, and although we are continuing to research this, our Publisher Imprint Database does not yet contain font information. To compensate, we began developing semiautomated methods for font identification, most crucially distinguishing between roman fonts and blackletter, described fully in Section 6. eMOP created upward of 40 font training sets for the Tesseract OCR engine—some of them roman, some blackletter. Particular sets among them work well on most roman fonts and others on most blackletter fonts. Combining the two kinds, however, reduces accuracy. An OCR engine that has been trained for reading both kinds of font will produce poor outputs because the difference between a roman "a," for example, and a blackletter "a" is greater than the difference between a roman "a" and "o" or "a" and "c." Thus, our algorithm for semiautomated identification of fonts is crucial to the project. Again, novel to the eMOP approach for identifying fonts was using hOCR output rather than page images, significantly reducing the amount of time and energy needed to identify typefaces.

2.3 OCR With a Human-in-the-Loop

Human-in-the-loop has been applied to the OCR process in the past. In the early 2000s, Newby and Franks [20] applied what they called *distributed proofreading* to the task of proofreading scanned books for Project Gutenberg. An early instance of crowd sourcing, Distributed Proofreading interface focused completely on the task at hand, providing page scans and editable transcripts in a single window. Von Ahn [21] recognized that

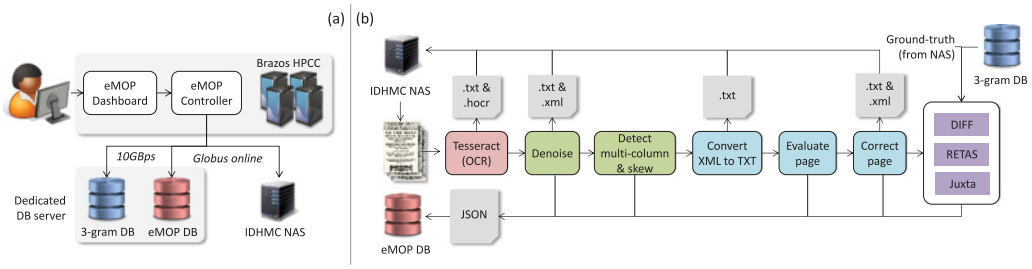

Fig. 1. (a) eMOP system architecture. (b) Steps in the eMOP controller. HPCC, high-performance computing cluster; NAS, network attached storage; DB, database; IDHMC, Initiative for Digital Humanities, Media, and Culture; OCR, optical character recognition.

a wider group of participants could be recruited if the task provided were recast into a game format—in other words, if the task were carried out as a side effect of an activity that the participant wanted to or needed to carry out. He called this *Games With a Purpose* and elucidated guidelines for their design [22]. His reCAPTCHA [23], which controls access to information to exclude bots, has successfully been applied to large-scale character recognition tasks, including the transcription of the *New York Times* back library. The National Library of Finland also has applied the gamification principle to verify the OCR of their archives in the DigitalKoot Project.[17] The project has provided several games using animated cartoon characters that are controlled as OCR verification tasks are carried out, as well as TypeWright, a crowd-sourced correction tool for distributed proofreading (see Section 3.2), but our main focus was developing Franken+ for training Tesseract, which dramatically improved OCR results by allowing us to create training sets that could be applied to documents with a particular typeface (see Section 6).

3 SYSTEM ARCHITECTURE

When work began in October 2013, the eMOP team faced a daunting task. The project had a 2-year lifespan, but assuming a cursory estimate of 1 minute per page, processing 45 million pages would require more than 17 months of constant work. Before processing could start, however, we would need to gather and clean the available metadata from multiple sources, ingest that into a database, test the available open-source OCR engines, develop additional tools and methods to improve OCR results on early modern documents, establish a workflow, learn to exploit a multiprocessing computing infrastructure for our needs, and create software to integrate all of these tools. The result was a workflow that incorporates nine new tools, using training from more than 20 typefaces, powered by the huge (by humanities standards) eMOP database, managed by a task scheduler (the eMOP controller), with human oversight through an online user interface (the eMOP dashboard), all performing OCR (via Google's Tesseract) and postprocessing page images on 128+ processors for more than 3 months.[18]

The overall system architecture is illustrated in Figure 1(a). A dedicated virtual machine (VM) Web server provides online access to the system through the eMOP dashboard, per user authentication. The dashboard displays the contents of the eMOP database, allows users to schedule selected page images for processing (by the eMOP controller), and displays OCR results and accuracy scores. The dashboard provides access to the eMOP database through a Rails-based application program interface (API). Page images are processed by the eMOP controller on the Brazos high-performance computing cluster (HPCC), located in College Station, Texas. To minimize database access hits, data transfers—whether from the database to the HPCC in the form of page images

[17] http://www.digitalkoot.fi/.

[18] With the exception of the eMOP database, which contains proprietary data, all of these tools and font training sets are available open source at the eMOP GitHub page: https://github.com/Early-Modern-OCR/.

selected for processing, or vice versa in the form of results—are done in batches with JSON formatted files. A dedicated SQL database server was created to host both the eMOP database and a database of Google 3-grams for postprocessing OCR correction. The eMOP database consists mainly of a table of metadata describing each document; a table of metadata describing each page image, including file locations, and ground-truth availability and location; and a table of results data for each page. Network connections between the Brazos HPCC, database server, and dashboard server are 10Gbps. A network attached storage (NAS) unit provides connection between the Brazos HPCC and permanent file storage of page images, original OCR, ground-truth files, and results files. File transfers between the NAS and Brazos HPCC are handled by Globus Online, making the eMOP controller more portable.

The main eMOP workflow is executed by the eMOP controller and constitutes work that is performed on every page image. The eMOP controller is a Python framework calling processes, written in various languages, in a pipeline of processing and postprocessing algorithms created for eMOP. Parallelization on the Brazos HPCC is performed by running an instance of the eMOP controller on a single page image, on one processor at a time. The Texas A&M University Initiative for Digital Humanities, Media, and Culture (IDHMC) has 128 processors dedicated to its use and more than 500 more available when not in use by other research groups. The eMOP controller is described more fully in Section 3.1. An extended eMOP workflow contains additional work that does not need to be done for every page. This extended workflow consists of tools that handle the creation of training for Tesseract and postprocessing correction (see Section 3.2).

3.1 The eMOP Controller

The eMOP controller is a pipeline of various software components that turns page images into their text and XML equivalents. The main eMOP workflow is embodied in the eMOP controller and its interactions with the eMOP dashboard, eMOP database server, NAS, and Brazos HPCC. Illustrated in Figure 1(b), the eMOP controller works in five distinct phases.

Phase 1: OCR. An authorized user interacts with the eMOP dashboard to select documents in the collection to be processed via optical character recognition ("OCRd" for short) with Tesseract. A dialogue box allows the user to select the OCR engine and, where applicable, which training set to use. The dashboard also serves as the point of contact between the eMOP controller and the eMOP database via an API.

- —The eMOP controller queries the dashboard for information pertaining to all selected documents' pages (image file location, ground-truth info, current job status). The dashboard returns information from the eMOP database in the form of a JSON response, which the eMOP controller writes as a set of input files to a temporary location on the Brazos HPCC.
- —The scheduler splits pages into jobs with an equal number of pages for each available processor on the Brazos HPCC. These jobs are then assigned to a processor queue for processing, where the eMOP controller is called for each page.
- —Finally, TIF page images are OCRd using the training specified by the user in the dashboard at job submission. Text and hOCR[19] files are produced and saved on the NAS.

Phase 2: Post processing. The hOCR outputs are processed with a series of custom algorithms to remove noise and split pages into multiple columns. These algorithms are described in detail in Section 4, but an overview is included here for completeness:

- —A denoising algorithm analyzes the hOCR output to remove "noise words": page noise and images that Tesseract incorrectly identified as words. Specifically, the algorithm looks at the coordinates of word

[19]hOCR is Tesseract's proprietary XML-like format containing the layout and logical structure of the document, including the coordinates of the BB for each recognized word along with its text transcription and recognition confidence.

bounding boxes (BBs) to identify words whose position and size indicate that they are not part of the page's text block. The denoising algorithm is run on every page that is OCRd and takes the page's hOCR files as input. It produces an XML file with noise words removed, which is written to the NAS. The algorithm also produces a measure of noise for the page that is written to the page's JSON results file for inclusion in the eMOP database.
— A multiple-column and skew detection algorithm analyzes the distribution of BBs in the page to identify when multiple columns are present in a page image and estimate their locations. The algorithm also identifies and measures the amount of skew present. These values are then written to the JSON results file of each page.
— A final algorithm creates a new text version of the output with the noise words removed and writes it to the NAS.

Phase 3: Page evaluation. The page evaluator is the first step in the page correction process. It evaluates the text produced by the denoising algorithm for each page to determine whether it fits the profile expected from a "normal" page of text. The page evaluator examines several document properties, including number of words (tokens) on the page, average word length, occurrence of continuous strings of repeated characters, length of each word compared to page average, interspersion of alphabetic and numeric characters, punctuation in a word, and proportion of words in the dictionary. Using this information, the page evaluator creates a score for estimated correctability (ECORR) and estimated page quality (ECORR normalized by the number of words on the page). These values are then added to each pages' JSON results file for inclusion in the eMOP database.

Phase 4: Page correction. Pages undergo correction based on early modern dictionaries and a database of Google 3-grams collected from early modern documents. The dictionaries include alternate and abbreviated spellings with special characters and ligatures converted to modern, multicharacter equivalents. The system includes multiple English language dictionaries, as well as a French dictionary and a Latin dictionary. The page corrector takes as input a denoised XML file containing the denoising confidence measures. Starting with the first three words on the page, the page corrector performs the following sequence of steps:

(1) Look up each word in the dictionaries for a match, and make character substitutions for each word in search for other possible dictionary matches.
(2) Use all possible matches of each word to look for matches in the Google 3-gram database. Matching 3-grams are weighted based on the number of uses in the original texts. Words that matched in the dictionary without substitution are given more weight. This information is then used to determine the correct "matching" 3-gram.
(3) Shift the analysis window to the next three words in the document (two words from the previous window and the next word in reading order) and repeat the previous two steps.

Once each page has been completed, the page corrector creates an ALTO[20] XML file and a text file containing all corrections.

Phase 5: Post analysis. In a final (optional) step, pages with ground-truth equivalents[21] are scored using Juxta-CL and one of three character distance measurement algorithms (Levenshtein, Jaro-Winkler, and Juxta). The Juxta score is then written to the JSON response file for each page. Each page's job status is updated to "Done." Additional details of this postanalysis are included in Section 5.

[20] ALTO is an open XML schema developed by the Library of Congress for OCR text and metadata (layout) information. We decided to use ALTO (instead of hOCR) since it is compatible with library-quality metadata standards such as Metadata Encoding and Transmission Standard and Dublin Core.
[21] The JSON input file contains a flag about whether ground truth is available for each page and the file path information for any ground-truth files.

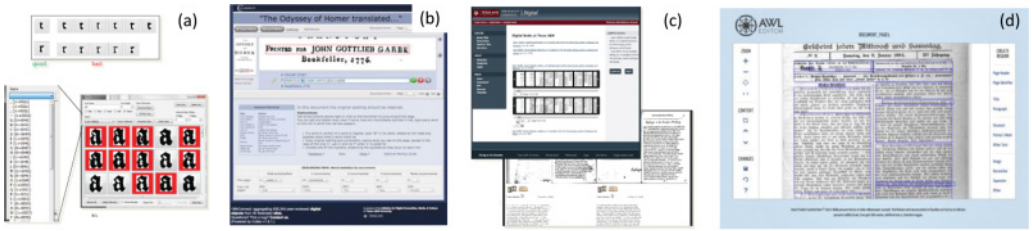

Fig. 2. Screenshots of the additional tools developed for eMOP: Franken + (a), TypeWright (b), Cobre (c), and AWL Editor (d).

When a processor completes every page in its job queue, the dashboard writes all associated JSON response files to the eMOP database. Document and page-level results are then viewable via the dashboard.

3.2 Additional Software Tools

In addition to the eMOP controller, our team has produced several other software tools for an expanded workflow, from producing training for Tesseract to crowd-source correction and typeface identification. Among these tools, four are beneficial to mention here:

— *Franken +* is an IDHMC-created tool that allows the user to create training for early modern, or any nonstandard, typefaces, which greatly improves Tesseract's accuracy. Testing showed that when training Tesseract with typefaces of variable quality, it was important to remove bad exemplars of glyphs from the training set. Glyph exemplars can be of variable quality due the original printing, which could contain bleed through, or over- or underinking, or because of low-quality digitization of the page images. Franken + allows us to easily compare several exemplars of every glyph from several page images and exclude those that were not representative of the typical exemplar. Doing so allows us to greatly improve direct accuracy. A screenshot of Franken + is shown in Figure 2(a).
— *TypeWright* is a crowd-source transcription correction tool created by Performant software for the 18thConnect.org Web site, managed at the IDHMC. Academics and nonacademics alike can view original page images, along with previously created mechanical or human transcriptions, and correct them. When a document is completely corrected, the corrector can then receive the full text and/or XML transcriptions to use as he or she likes. With eMOP complete, the previously un-OCRd EEBO collection has been added to TypeWright along with ECCO.
— *Cobre* is an online typeface identification and document comparison tool available to scholars. Cobre was created by the Texas A&M Libraries to aid scholars with the close examination of page images, enabling per-page typeface identification. In addition, Cobre allows scholars to compare multiple page images, at various magnifications, from various editions or copies of the same text. With access to eMOP-produced transcribed texts from all copies as well, Cobre allows scholars to cut and paste transcribed text between versions to more easily create a corrected transcription of a particular document.
— *Aletheia Web Layout (AWL) Editor* was created by Pattern Recognition and Image Analysis (PRImA) Research Lab as a plugin for TypeWright. With the AWL Editor, users can, on a per-page basis, identify and transcribe paratext (marginalia, headers, stop words, etc.) to be exported as encoded XML with the rest of the document text corrected in TypeWright.

3.3 The eMOP Workflow

The eMOP workflow focuses on postprocessing page images (i.e., after Tesseract OCR transcriptions have been produced) rather than preprocessing. Although most of the page images in our corpus would benefit from

preprocessing (e.g., contrast enhancement, deskewing, dewarping), the size of our corpus made it prohibitive to determine which page requires what type of preprocessing. Rather, we focused our efforts on postprocessing page images based exclusively on information provided by Tesseract. Nonetheless, in addition to creating the best possible text transcriptions, we also gathered additional information from page images that would allow us to add preprocessing algorithms to our workflow at a later time.

The eMOP workflow begins by creating a training set of EM typefaces for Tesseract. Using Aletheia and Franken + , we created training for more than 40 different typefaces in roman, italic, and blackletter families of varying point sizes and moved them to the Brazos Cluster for use by the eMOP controller via the dashboard. The dashboard could then be used to select documents for processing using a given set of training fonts. Because it is not known in advance which typefaces are used in which documents, a master training set was created from most of the EM typeface training produced. The hOCR produced is used by the typeface identification algorithm described in Section 6. To rerun documents that had been classified according to typeface, we developed training sets specifically designed for roman fonts and others for blackletter fonts.

Once a batch of documents or pages is submitted via the dashboard, the eMOP controller retrieves the relevant page images and any available ground-truth files from the NAS. The batch is split into jobs of an equal number of pages and scheduled on a minimum of 128 processors in a queue dedicated for IDHMC use. A greater number of processors are also available for use via background queues that take advantage of unused Brazos Cluster processors. At any one time, eMOP is capable of processing 128 to 512 page images in parallel. The eMOP controller then handles processing each page image according to the steps in Figure 1(b). Tesseract is run on each page image using the training selected in the dashboard, and its hOCR output is passed to the denoising algorithm to remove page noise incorrectly identified as text. This kind of page noise is common in our document corpus due to dirty page images, bad digitization, ink bleed through, and the like, but also, importantly, from the presence of images, maps, and common decorative elements. As such, denoising is an important first step in our postprocessing. Denoising results are then sent to a multicolumn skew detection algorithm to gather data to be used in later preprocessing of the page images: images can be deskewed using image enhancement software such as ImageMagick[22]; columns are run as separate pages so that the OCR engine does not attempt to read the page across. Documents that are noisy beyond recognition, determined by the OCR quality prediction (Section 5), are entered into a database of unreadable documents, designed to give libraries an idea of which particular early modern documents need to be rescanned using better imaging techniques. (That database is not yet complete.)

After running the pages, the page evaluation and correction algorithms are then called on the resulting XML files and utilize a database of period-specific 3-grams culled from Google Books data and a set of word lists employing variant spellings, abbreviations, and other lexical differences. The result is an ALTO-formatted XML file of corrected text including confidence measures gathered from both Tesseract and denoising results and variant possibilities of words. These algorithms also collect data for every page (e.g., number of words corrected, corrections made), which can then be used to further improve the eMOP workflow. The final output is then scored using Juxta-CL, on pages with ground truth available, to produce a score for those pages indicating the correctness of our results at the page level. At this point, the text transcriptions produced are then ingested into TypeWright for further correction through crowd sourcing and into Cobre for typeface identification by scholars.

4 DENOISING OCR OUTPUTS

Page images in the eMOP corpus are generally below the quality expected by OCR engines. They are third-generation copies (original to film, film to microfilm, microfilm to digital file) that had already been binarized with an unknown method—the average file size of our page images is around 50KB. Many images are made from documents that were already in poor shape due to age and damage. Adding to this, the documents had been printed at the beginning of the print era (when the technology was new and of variable quality), so many

[22] https://www.imagemagick.org.

Fig. 3. (a) OCR output for a document in our collection; BBs are shown in green. (b) Segmenting columns by identifying troughs in the horizontal distribution of BBs. Reprinted with permission from "Automatic assessment of OCR quality in historical document" A. Gupta et al., Proc. 29th AAAI Conf. on Artificial Intelligence (AAAI 2015), pp. 1735-1741. Copyright © 2015, Association for the Advancement of Artificial Intelligence.

of the page images exhibit problems such as bleed through, overinking, and underinking. Finally, during the digitization process, many pages became skewed or warped. As a result, when a document has poor quality, the OCR engine generally produces a large number of spurious BBs in addition to those that correspond to words in the document (see Figure 3(a)).

To address these issues, we developed algorithms to identify and filter noisy OCR outputs. As noted earlier, for each document image, Tesseract produces an hOCR data file containing the coordinates of the BB for each recognized word along with its text transcription and recognition confidence. It is the hOCR file, not the underlying image, that we use for denoising. As we will show, it is possible to discriminate between noisy and text BBs by analyzing statistical differences in their shape, size, position, and confidence score. This approach is advantageous for two main reasons. First, it does not require dedicated image processing algorithms [13], which can become prohibitive for large document collections. Second, the approach is language agnostic because it relies exclusively on geometrical properties of BBs rather than the text transcription associated with them. Our approach for discriminating text and noise BBs consists of three steps: prefiltering, column segmentation, and local iterative relabeling.

Step 1: Prefiltering. The denoising process starts by generating initial labels for each of the BBs returned by Tesseract. For this purpose, we use a rule-based classifier that considers three features for each BB: word confidence, height-to-width ratio, and area. The rules are derived as follows:

— *Rule 1: OCR word confidence.* BBs with very low or very high confidence predominantly consist of noise and are flagged accordingly during prefiltering.
— *Rule 2: Height-to-width ratio.* Most words are written horizontally, so the height-to-width ratio is generally lower for word BBs than for noise BBs. Consequently, if this ratio is less than a threshold, we label the BB as text; otherwise, we label it as noise.
— *Rule 3: Area.* Tesseract tends to misidentify speckles as text; fortunately, these areas are small relative to normal text BBs. Accordingly, we label as noise all BBs in the lowest percentiles of the total area for the document.

Individual thresholds are optimized simultaneously with a manually labeled subset of the corpus (see Section 4.1). The final filter is the conjunction of the three rules. BBs classified as text at this stage are used in the next stages to extract column layout and estimate the average font size of each document.

Step 2: Column extraction. Although Tesseract can generally identify multiple columns on a page, the nature of our document corpus presents challenges. Narrow gutters between columns, and the presence of noise, vertical lines, or decorative elements in the gutters, cause Tesseract to misidentify columns and disarrange the reading order of a document that it otherwise may have transcribed accurately. This is especially true of the page images from EEBO, which typically contains two facing pages in each image. When the internal margins of those pages are missing, narrow, noisy, or dark, the text is typically read as one large page, destroying the proper reading order of two pages of text.

For this reason, the second step in the denoising process consists of dividing each image into its constituent pages and columns so that each can be processed individually. First, we identify the leftmost and rightmost text BB from the prefiltering stage; these coordinates define the text boundaries of the image. Then, we perform column segmentation by analyzing the distribution of BBs over the horizontal axis; the dominant troughs in this distribution define the column boundaries. To compute this distribution of BBs, we divide the horizontal axis with 1,000 evenly spaced points. At each point, we trace rays from the top margin to the bottom margin with slopes in the range $90^\circ \pm 3^\circ$in increments of 0.2°, then calculate the number of intersecting BBs for each ray. At each point, we then identify the ray with the fewest intersections, and that becomes the value of the distribution at that point. Since images tend to have a large number of spurious BBs at the margins, any BBs in the top and bottom 20% are discarded. The overall process is illustrated in Figure 3(b).

Step 3: Local iterative relabeling. After each page has been split into columns, we apply an iterative relabeling algorithm to the BBs of each column. The rationale behind this final step is that BBs surrounded by text are more likely to contain text than those surrounded by noise. Accordingly, for each of the four vertices of every BB, we find its nearest neighbors. Then, we calculate a weighted score, S, based on the label of each neighbor penalized by its distance:

$$S(b) = \frac{\sum_{k=1}^{P} w_k L_k}{\sum_{k=1}^{P} w_k}, \quad \text{with } w_k = \frac{1}{dist(b,k)}, \tag{1}$$

where b is the index of the BB, N is the number of BBs within distance D_{max} from the vertices of b, and P is the maximum number of nearest neighbors considered ($P \leq N$). L_k is the predicted label (0: noise; 1: text) for the k-th nearest neighbor, initially taken from the prefiltering step. The distance D_{max} limits the search area for nearest neighbors, preventing text BBs that are far from b to be considered in the computation. The distance D_{max} is computed relative to H_{med}, the median height of text BBs found in the prefiltering stage, plus a tolerance defined by H_{IQR}, their interquartile range; both statistics are computed for each individual column in the image:

$$D_{max} = H_{med} + \alpha \times H_{IQR}, \tag{2}$$

where α defines the tolerance; the larger its value, the more distant neighbors that are allowed in the computation of S of Equation (1). In our implementation, the value of α is optimized to minimize the mean square error between S and the ground-truth label for all BBs in a training set.

The iterative process starts by initializing BB labels with those from the prefiltering stage. From these labels, an initial score S can be computed for each BB. This score is then combined with six additional features and passed as an input to a multilayer perceptron (MLP) previously trained to classify BBs as either text or noise. The additional features include those used in prefiltering (C_{OCR}, H/W, A) and the BB position relative to the document margins, and its height normalized to H_{med} and H_{IQR}. The resulting labels are used to recompute S, and the process is repeated until convergence (i.e., labels no longer change from one iteration to the next).

Table 1. Datasets Used for Training and Validation Purposes

Dataset	Images (#)	Text/Nontext (%)	BBs (#)
1	39	69/31	14,705
2	34	71/29	15,896
3	86	66/34	41,765

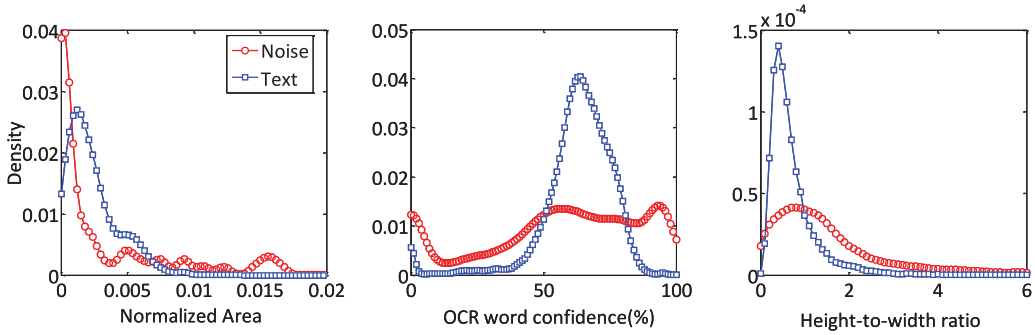

Fig. 4. Probability density function of the three BB features, computed on Dataset 1. Reprinted with permission from "Automatic assessment of OCR quality in historical documents," A. Gupta et al., Proc. 29th AAAI Conf. on Artificial Intelligence (AAAI 2015), pp. 1735-1741. Copyright © 2015, Association for the Advancement of Artificial Intelligence.

4.1 Results

To test the denoising algorithm, we generated three separate datasets (Table 1), which were carefully selected to represent the variety of documents in the eMOP collection. This included single-column, multipage, and multicolumn images, as well as images with artifacts due to ink bleed through, multiple skew angles, warping, printed margins, printed column separators, and pictures. Each BB returned by Tesseract for each document image was then manually labeled (i.e., text/noise) to generate ground-truth data, for a total of 72,366 BBs. As labeling criteria, we considered as noise any BB that spanned more than two lines of text, as well as BBs around pictures, small speckles, and printed margins. The remaining BBs were labeled as text. To guard against differences in image size, the coordinates of BBs for each document were [0,1] normalized. Dataset 1 was used to optimize thresholds in the prefiltering stage, whereas Dataset 2 was used to optimize parameters α and P in the local iterative relabeling stage. Dataset 3 was used to cross validate the MLP and evaluate overall performance.

Figure 4 shows the distribution of features for noise and text BBs in the documents from Dataset 1. The distribution of normalized areas in Figure 4(a) indicates that noise BBs tend to be smaller than text BBs, following our observations that Tesseract has a tendency to generate small spurious BBs whenever speckle noise is present in the image. Shown in Figure 4(b), the distribution of OCR word confidence values for noise BBs is multimodal, with peaks near the extrema (0,1), whereas for text BBs it is normally distributed with a peak around 65% confidence. Finally, the distribution of H/W ratios in Figure 4(c) shows clear differences between the two types of BBs, with text generally having a much lower H/W ratio, as could be anticipated.

To optimize the threshold values for the three rules in the prefiltering stage, we performed a receiver-operating-characteristic (ROC) analysis of the binary classification problem on Dataset 1. Namely, we performed exhaustive search for the word confidence (two thresholds), height-to-width ratio, and area thresholds (a 4D search space) to find the operating point with maximum F1 score on the precision-recall curve. The derived rules were

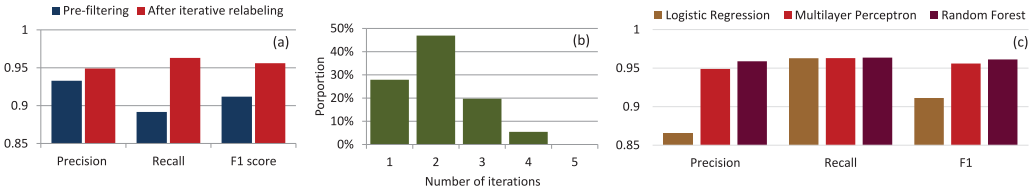

Fig. 5. (a) BB classification rate before and after local iterative relabeling. (b) Number of iterations required for convergence. (c) Comparison between MLPs and two additional classification methods. Parts (a) and (b) are reprinted with permission from "Automatic assessment of OCR quality in historical documents," A. Gupta et al., Proc. 29th AAAI Conf. on Artificial Intelligence (AAAI 2015), pp. 1735-1741. Copyright © 2015, Association for the Advancement of Artificial Intelligence.

—*Rule 1*: If $0 < C_{OCR} < 0.95$, then TEXT,
—*Rule 2*: If $H/W < 2$, then TEXT,
—*Rule 3*: If $A >$ 1st percentile, then TEXT,

which, when used as a conjunction, yield an F1 score of 0.93 (0.94 precision; 0.91 recall). Thus, prefiltering can identify a significant number of noisy BBs, but it also mislabels a large proportion (9%) of text BBs in the documents. This is largely due to the fact that it does not consider information local to each BB, a problem that is handled by the local iterative relabeling step.

The MLP for the iterative process consisted of a hidden layer with eight tangent-sigmoidal neurons and two output neurons (i.e., one per class) with soft-max activation function to allow MLP outputs to be interpreted as probabilities. The number of hidden units ($N_H = 8$) was optimized through threefold cross validation over Dataset 3 with the F1 score as the objective function. Parameter P in Equation (1) (the maximum number of neighbors) was set to 84 (21 per vertex), and parameter α in Equation (2) was set to 10. These optimal values were obtained by minimizing the mean square error between S and the ground-truth label for all BBs in Dataset 2. We also performed threefold cross validation over Dataset 3 to compare model performance before and after iterative relabeling. Results are summarized in Figure 5(a); precision, recall, and the F1 score improve when compared to prefiltering results on Dataset 3, with the largest gains obtained for recall (from 0.89 to 0.96). Figure 5(b) summarizes the convergence rate; in 95% of the cases, the algorithm converges within three iterations. In a final step, we compared the MLP against two alternative classification methods: logistic regression (LR) and random forests (RF). Results are summarized in Figure 5(c); an RF with 150 trees gave the best cross validation result and was used here. The MLP outperforms LR, which indicates that its additional model complexity is beneficial. More interestingly, the MLP provides similar performance as the RF while using a much simpler model (one hidden layer with eight neurons vs. 150 trees for the RF), which is important considering that the model had to be deployed in production. Thus, the remaining results in this manuscript are based on the MLP model.

Figure 6 shows a document overlaid with the BBs returned by Tesseract. The fill color (green vs. red) represents the MLP prediction (text vs. noise, respectively), with higher color saturation denoting higher confidence. Arrows 1 and 2 illustrate two cases for which prediction was correct but the MLP had low confidence, hence the gray tone. Arrow 3 points to a BB that covers graphics and a decorative drop cap, neither of which is likely to lead to a good OCR transcription. Finally, arrow 4 points to a BB that contains two lines of text; as such, the OCR transcriptions are likely to contain garbage.

5 PREDICTING THE QUALITY OF OCR TRANSCRIPTIONS

Thanks to the Text Creation Partnership (TCP), we had approximately 1.8 million pages of ground truth from about 46,000 documents against which we could measure the accuracy of our workflow using Juxta-CL. However, even that impressive number amounts to only 4% of our total page images. In addition, the TCP's efforts have

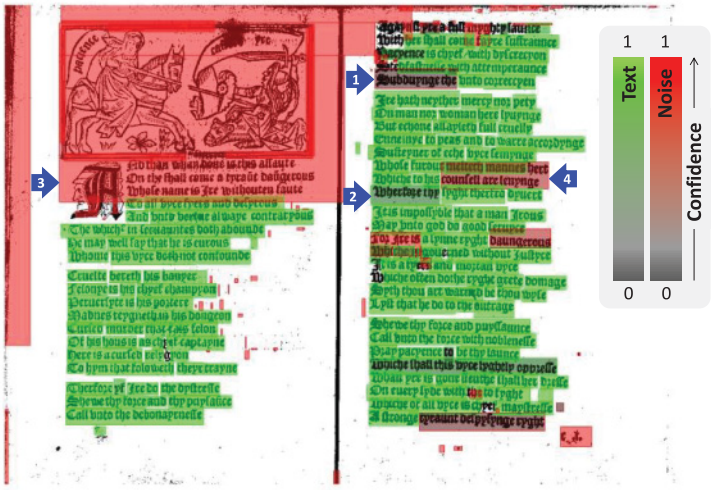

Fig. 6. Iterative relabeling results for the image in Figure 3. Color denotes MLP confidence: the more saturated, the higher the confidence. Red: noise; green: text. Reprinted with permission from "Automatic assessment of OCR quality in historical documents," A. Gupta et al., Proc. 29th AAAI Conf. on Artificial Intelligence (AAAI 2015), pp. 1735-1741. Copyright © 2015, Association for the Advancement of Artificial Intelligence.

historically been focused on EEBO, amounting to 82% of our ground truth. Our ground truth then accounted for about 25% of the entire EEBO collection but less than 1% of the ECCO collection. In general, however, the page image quality of the ECCO collection was much better as a whole than that of the EEBO collection, minimizing that skew somewhat. But we still did not have a large enough sample set of ECCO ground truth upon which to project a true appraisal of our overall accuracy with that collection. We needed a way to estimate the accuracy of our OCR transcriptions for pages for which we did not have ground truth.

Fortunately, and as shown in the previous section, our denoising algorithm can label BBs as text or noise with remarkable accuracy. This suggested that it may be used to estimate the overall quality of each document. Low-quality documents tend to produce a large number of spurious BBs, whereas high-quality documents produce mostly text BBs. Thus, the proportion of noise BBs returned by the OCR engine tends to be representative of the document's quality:

$$BB_{noise} = \#\ noise\ BBs / \#\ BBs. \tag{3}$$

We evaluated this quality measure on a dataset containing 6,775 documents from EEBO that had ground truth. Results in Figure 7(a) show a strong negative correlation ($-0.704; p \ll 0.001$) between the proposed noise measure (BB_{noise}) and the Jaro-Winkler similarity (s_{JW}). Thus, as the proportion of noise BBs in a document increases, differences between OCR and manual transcriptions also increase. The significance of this result is that s_{JW} cannot be computed in practice since it requires the manual transcription, whereas BB_{noise} can be computed directly from the hOCR output. As such, it may be used to automatically triage documents of poor quality and focus computational resources on those whose quality is more likely to generate good OCR transcriptions. In a final step, we tested whether our algorithm could be used to improve the overall OCR performance. For this purpose, we ran the algorithm on the previous dataset (6,775 documents), removed any BBs labeled as noise, and computed s_{JW} between the resulting transcription and the manual transcription. Results are summarized in Figure 7(b). In 85.4% of the documents, the algorithm improved S_{JW} (average: +6.3%), whereas in 10.6% of the documents, it led to a decrease (average: −3.0%). Last, we analyzed the impact of local iterative relabeling

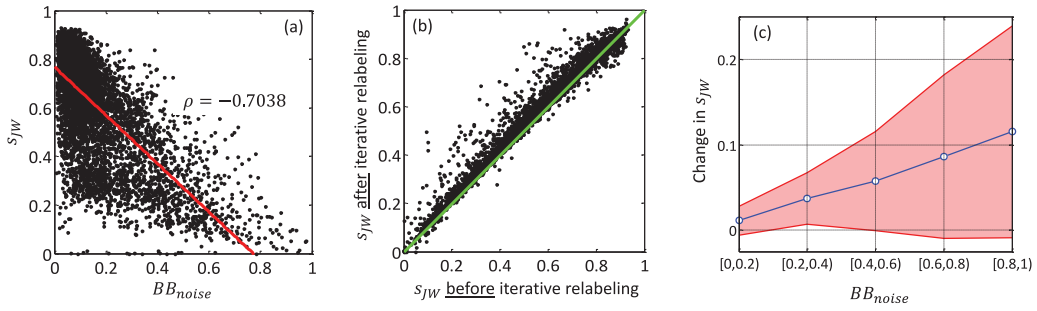

Fig. 7. (a) BB-based quality measure (BB_noise) vs. the Jaro-Winkler similarity (s_JW) for 6,775 documents. (b) s_JW before and after iterative relabeling; for most documents (those above the diagonal line), iterative relabeling improved s_JW. (c) Average change in Jaro-Winkler similarity as a function of document quality (BB_{noise}). The blue line represents the average improvement as a function of the amount of noise in the document, whereas the shaded areas represent variability in the improvement across documents. Reprinted with permission from "Automatic assessment of OCR quality in historical documents," A. Gupta et al., Proc. 29th AAAI Conf. on Artificial Intelligence (AAAI 2015), pp. 1735-1741. Copyright © 2015, Association for the Advancement of Artificial Intelligence.

step of denoising algorithm as a function of document quality; results are shown in Figure 7(c). Regardless of document quality (BB_{noise}), local iterative relabeling increases the Jaro-Winkler similarity. These improvements are modest for high-quality documents (i.e., low BB_{noise}) but become quite significant (up to 0.25) for documents of poor quality,[23] where they are most needed.

6 TYPEFACE IDENTIFICATION

Historical documents in the hand-press period have irregular fonts and show large variations within a single font class since the early printing processes had not been standardized. Blackletter (or Gothic) and roman font classes are the two main font types used in early modern printing, but these two font classes have evolved into multiple subclasses since the first printed book. Knowing the font type and characteristics for each document in a collection can substantially improve the performance of OCR systems [24, 25]. In large collections such as ours, however, hand tagging each individual document, page, and text region according to its font becomes prohibitive.

To address this problem, we developed a typeface identification algorithm to automatically tag individual documents within a large collection according to their fonts. Font identification is best formulated as a supervised classification problem, and as such it requires labeled data. Classification models work best when they have sufficient labeled data that represent the diversity of exemplars in the corpus. However, in our case, obtaining large amounts of labeled data from a corpus of 45 million page images, with varied font types, was a daunting task. For this reason, our typeface identification algorithm operates in an *active learning* fashion to optimize the hand labeling process. Active learning is a mixed-initiative paradigm where a machine learning (ML) algorithm and a human work together during model building: the ML algorithm suggests a few high-value unlabeled exemplars; these are passed to the human to obtain labels; the model is adapted based on these newly labeled exemplars; and the process is repeated until the model converges.

Figure 8(a) illustrates the approach. The process starts by selecting a set of seed training images, which are then passed through the OCR engine and the denoising algorithm described in Section 4. Once text BBs have been identified, they undergo an image preprocessing step that normalizes them by size (to a height of 400 pixels),

[23]For very noisy documents (i.e., $BB_{noise} = [0.8, 1)$), the average improvement is around 0.12 (see the blue line in Figure 7(c)) but can be as high as 0.35 (see the upper boundary of the shaded region).

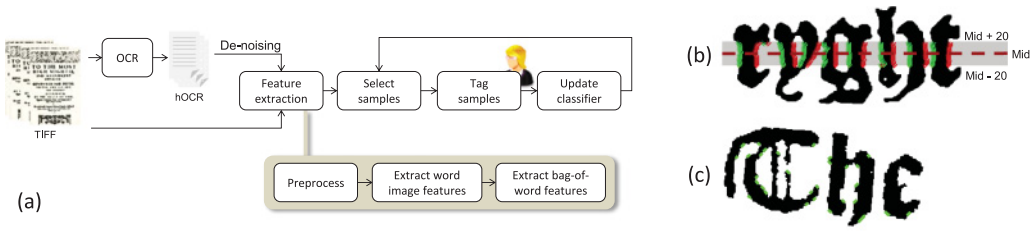

Fig. 8. (a) Steps for active learning–based typeface identification. (b) Calculating mean and IQR character width. (c) Results from the Hough transform; green line segments indicate the detected angled straight lines.

filters out salt-and-pepper noise [26], and removes skew [27]. Preprocessed word images are then passed to a feature extraction module. The resulting feature vectors (from all word images in a page) are then vector quantized and combined into a bag-of-words feature (BoF) representation for the page. These BoFs become the input to the font classifier. Starting with a small set of seed training images, we iteratively train the font classifier and then use it to select the most informative (yet unlabeled) documents for the human analyst to label next. This training-labeling process (active learning) is repeated until a performance criterion (e.g., a target precision/recall rate) is met.

We extract three kinds of features to capture font information at the word-image level:

— *Average and IQR stroke width.* We characterize the stroke width by its trimmed mean and interquartile range (IQR) on each word image. Namely, we scan 10% of the rows (41 rows; middle row $\pm$ 20) of each preprocessed word image, and locate transitions from background to foreground, and transitions from foreground to background—see the green and red points in Figure 8(b), respectively. Difference in their x-coordinates serves as an estimate of the stroke width. Since each row may have multiple such stroke widths, we store the count C_i for each row and then compute the maximum C_{max}. Next, we select rows where $C_i = C_{max}$ and calculate the trimmed mean and IQR over their respective stroke widths. Using these robust statistics and limiting the computation to rows with $C_i = C_{max}$ provides further immunity to outliers.

— *Slant line density.* We estimate the number of slanted lines by applying the Canny edge detector to each word image, followed by the Hough transform [28]. We then compute the number of lines with slope in the range $45° \pm 5°$ and $-45° \pm 5°$, then divide it by the number of recognized characters in the word, which is available from the output of the OCR engine. This results in an estimate of the slant line density for a word image (see Figure 8(c)).

— *Zernike moments.* Finally, we capture the visual appearance of each word using Zernike moments (ZMs) [29]. ZMs are the projection of the image onto an orthogonal basis known as the Zernike polynomials (ZPs). Following Tahmasbi et al. [30], the ZMs for an image of size N $\times$ N can be computed as

$$Z_{m,n} = \frac{n+1}{\lambda_N} \sum_{c=0}^{N-1} \sum_{r=0}^{N-1} I(c,r)\, V^*_{nm}(c,r), \tag{4a}$$

where λ_N is a normalization factor, $I(c,r)$ is the binary image at pixel position (column, row), and V^*_{nm} is the ZP of order m, n, which can be computed as

$$V^*_{nm}(\rho_{cr}, \theta_{cr}) = \mathrm{R_{nm}}(\rho_{cr})\, e^{-jm\theta_{cr}}, \tag{4b}$$

and ρ_{cr} and θ_{cr} are the distance and phase at pixel (*c*,*r*), respectively:

$$\rho_{\mathrm{cr}} = \frac{\sqrt{(2c - N + 1)^2 + (2r - N + 1)^2}}{N},\ \theta_{cr} = \tan^{-1}\left(\frac{N - 1 - 2r}{2c - N + 1}\right). \tag{4c}$$

Finally, R_{nm} is a radial polynomial defined as

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s!\,(((n+|m|)/2) - s)!\,(((n-|m|)/2) - s)!} \rho^{n-2s} \tag{4d}$$

Following Tahmasbi et al. [30], we compute[24] the magnitude of first six ZMs along with their transformations (a total of 15 features).

BoFs [32] are usually extracted using small patches in the image, which in our case correspond to word images. To obtain a BoF for each page, we apply *k*-means (*k*=20) to the distribution of local features (across all training documents), vector quantize each word image, and compute the number of words assigned to each cluster. To achieve word-count invariance, we normalize each BoF by the total number of words on the page.

6.1 Active Learning for Font Identification

The active learning component comprises two parts: a base classifier and a sampling engine. The base classifier is trained on a small amount of labeled data (L), and the sampling engine uses it to select a batch of the most informative instances (X) from an unlabeled set (U) for labeling. A human annotator labels all instances in X, and these are added to L to retrain the classifier. The entire process of training and sampling is repeated until convergence.

6.1.1 Base Classifier. We use a modified label-propagation model proposed by Kobayashi et al. [33] known as logistic label propagation (LLP). In label propagation (LP), all training instances (labeled + unlabeled) are treated as nodes in a fully connected graph, and labels are propagated to unlabeled data points according to their proximity to the labeled data:

$$S_{ij} = \exp\left(-\left|\left|X_i - X_j\right|\right|^2 / \sigma^2\right), \tag{5}$$

where X_i and X_j are the features vectors for document i and j, respectively, and σ is the bandwidth hyperparameter. A major drawback of LP is its computational complexity during recall; LP must reconstruct the whole similarity matrix for new instances and then re-estimate their class posteriors to predict their labels. In contrast, LLP trains a logistic classifier in a semisupervised fashion by adding to the logistic cost function a cost derived from the LP model, which shifts the decision boundary to account for the distribution of unlabeled data. We compare LLP against LR in Section 6.2.

6.1.2 Active Sampling. The performance of active learning depends heavily on the choice of sampling strategy (known as a query function) that selects the most informative instances for labeling. For this reason, our query function considers three separate criteria:

— *Uncertainty.* Following Settles [34], unlabeled instances are selected according to the classifier's uncertainty about their labels. In particular, we compute uncertainty for an unlabeled data point U_k as the entropy of its class-posterior distribution p(y|U_k, L):

$$H(y|U_k, L) = -\sum_y p(y|U_k, L) \log p(y|U_k, L), \tag{6}$$

where L is the labeled data and y is the label, which ranges over all possible labelings of U_k.

[24]For this purpose, we use a Matlab implementation of ZMs developed by Wolf et al. [31] (http://liris.cnrs.fr/christian.wolf/software/zernike/).

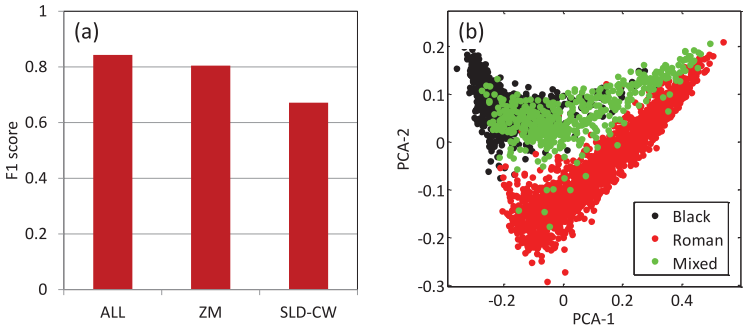

Fig. 9. (a) Cross-validated accuracy for classifiers trained using 18 features (ALL), ZM, and SLD-CW. (b) Principal components analysis of the dataset using page-level features (BoFs); each point represents a document.

—*Dissimilarity to the labeled data.* To promote exploration, we also consider instances that lie in unexplored regions of feature space. For each unlabeled instance (U_k), we find the five most similar labeled instances (L_n) based on LLP's similarity matrix (S_{nk}). Samples with highest dissimilarity D_k are selected for querying:

$$D_k = \frac{1}{5}\sum_{n=1}^{5} 1 - S_{nk}. \tag{7}$$

—*Diversity.* Each iteration, our sampling engine selects a batch of 20 unlabeled instances. To prevent instances within a batch from being too similar to each other, we incorporate a diversity metric that constructs the batch X as follows:

(1) Initialize X with the unlabeled instance that has the largest score: $H(y|U_k) + D_k$.
(2) For each remaining unlabeled instance (U_k), calculate its diversity factor (D'_k) as

$$D'_k = \min_n \left(1 - S_{nk}\right), \tag{8}$$

where S_{nk} is the similarity between U_k and the n^{th} sample already in X(see Equation (7)).
(3) Select sample U_k with the largest combined score ($H(y|U_k) + D_k + D'_k$), and add it to X.
(4) Repeat steps (2) and (3) until 20 unlabeled instances have been selected.

6.2 Results

We devised two experiments to evaluate our feature set and determine the best active learning query function. In the first experiment, we examined whether the proposed features could discriminate between blackletter and roman fonts at the word level. In the second experiment, we evaluated the complete system using six possible active sampling strategies derived from combinations of three query criteria described in the previous section. For these experiments, we created a dataset consisting of 3,272 document images from the ECCO/EEBO databases: 1005 printed in blackletter, 1,768 in roman, and 498 with text in both fonts (mixed). Each of these documents were hand labeled by domain experts on the eMOP team.

6.2.1 Evaluating Word-Image Features. To compare the discriminatory power of the different word-level feature sets, we randomly selected 500 blackletter documents and 500 roman documents and extracted the 18 features described at the start of this section. We then trained an LR classifier to predict the class label for each word. The threshold for the classifier output was selected by maximizing the accuracy through fivefold cross validation. Results are summarized in Figure 9(a). The slant angle density and character width (SLD-CW) feature set achieves 58% classification rate, whereas the ZM feature set achieves 81%. When the two feature sets are

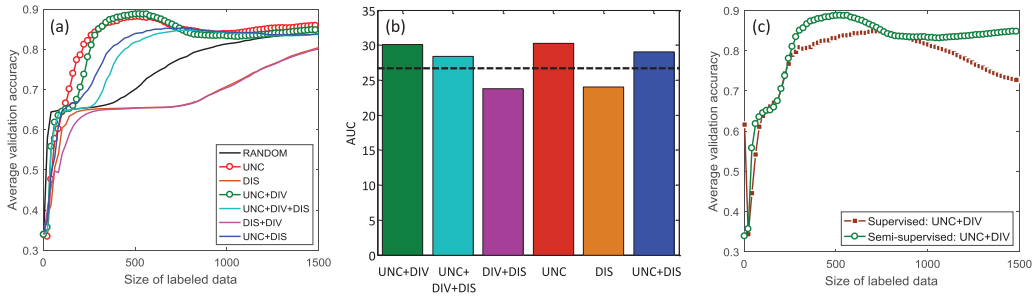

Fig. 10. (a, b) Learning curves (a) and normalized AUC (b) for different sampling techniques. The black line in (b) denotes random sampling. (c) Performance of the entropy + diversity strategy with LLP and LR.

combined (ALL), classification performance improves modestly to 84%. These results indicate that all extracted features are important for font identification and that they provide high between-class separability—even at the word level.

6.3 Evaluating the Active Learning System

To evaluate the overall system, we randomly selected 600 documents (200 per class) as a test set and used the remaining 2,672 documents for training. From these, we randomly selected 3 labeled documents (one per class) as a seed set (X) to train the LLP font classifier; the remaining 2,669 documents became the unlabeled set (U).

After each step of training, the active learner selected a batch with the 20 most informative documents in U (based on the scoring function) and queried their labels from an oracle[25] that played the role of the human labeler. These newly labeled documents were added to the training set L, the font classifier was retrained, and its performance was tested on the 600 instances in the test set. We repeated this process until all unlabeled data was consumed, recording the size of the labeled dataset and classifier accuracy at each step. For comparison, we included a baseline system that iteratively selected a random set of 20 documents. For evaluation purposes, the LLP bandwidth parameter was set to 300. Each experiment (active selection and random selection) was repeated 20 times with a new set of random seeds to get a stable performance estimate.

To ascertain the relative merit and degree of complementarity of the three active selection criteria, we report performance for each of them in isolation[26] and for each of their linear combinations:

$$\begin{aligned} S_1(k) &= H(y|U_k, L) \\ S_2(k) &= D_k \\ S_3(k) &= H(y|U_k, L) + D_k \\ S_4(k) &= D_k + D'_k \\ S_5(k) &= H(y|U_k, L) + D'_k \\ S_6(k) &= H(y|U_k, L) + D'_k + D_k \end{aligned}$$

where $H(y|U_k, L)$ is the uncertainty measure, D'_k is the diversity factor, and D_k is the dissimilarity measure.

Learning curves for these scoring functions and the baseline random selection are shown in Figure 10(a). The best performer is S_5 (uncertainty + diversity) and requires 523 labeled samples (17%) to achieve a maximum average test accuracy of 89%. The uncertainty criterion alone (S_1) also performs well, achieving a maximum test accuracy of 88% using 503 labeled samples.

[25]The oracle in this case was the training dataset itself, which had been fully labeled in advance.
[26]We do not consider diversity in isolation because its primary use is to aid other sampling techniques to pick diverse unlabeled instances.

In a final analysis, we calculated the area under the curve (AUC) of each learning curve as a scalar performance measure for each active sampling approach. Results are summarized in Figure 10(b). Based on the AUC values, the uncertainty criterion performs marginally better than the sampling based on uncertainty + diversity. The remaining sampling techniques, all of which use dissimilarity, perform notably worse. This may be attributed to the characteristics of our dataset. As shown in Figure 9(b), the distribution of BoFs reveals a good degree of separability between roman and blackletter fonts, with some overlap with the mixed class. This arrangement of data makes exploration less useful because most of the information is captured by instances at the class boundaries. As a result, active sampling strategies that involve exploration need more labeled data to reach a desired accuracy compared to a more exploitative technique that samples at the class boundaries.

6.3.1 Choice of Classifier: Supervised Versus Semisupervised. We compared LLP against LR (supervised) as an alternative form of classification. For this purpose, we followed the same experimental procedure as described earlier, except the LR classifier did not use any unlabeled data during training. With LR, the highest validation accuracy (84%) is also achieved when using the entropy + diversity criterion, which requires 23% (723 images) of labeled data (results not shown). Comparing the entropy + diversity strategy for both classifiers (see Figure 10(c)), shows that semisupervised learning achieves higher accuracy with a significantly reduced number of labeled examples. Namely, the LLP classifier achieves 5% better validation accuracy using 27% less labeled data (200 examples) than the LR classifier.

7 DISCUSSION

A goal of eMOP was to determine whether using early modern fonts to train open-source OCR engines would yield better results than training with modern fonts. To fight against the rise of a "dark archive" and the "digital dark age," we chose an initial dataset of the ECCO and EEBO collections (45 million page images) of the first printed books in English—a dataset, curated by ProQuest and Gale, that early modern scholars across the globe use with regularity. Early modern scholars and librarians have lauded collections digitized by Readex, Adam Matthews Digital, ProQuest, and Gale (ECCO, EEBO, and other large collections) for providing "full-text searching" and hence "providing unprecedented access" to key historical documents [35]. The quality of the OCR means, however, that full-text searching is limited, and access to our past cultural heritage is much more limited than it seems.

To date, providing usable transcriptions has required significant effort and expense. In ECCO—a dataset of 182,000 books—machine-generated text runs behind search mechanisms. Scholars conduct research on this text, which was generated by Prime Recognition, who ran six of the top commercial OCR engines on the document images and then used voting algorithms to determine the best results. By our calculations, these transcriptions are 89% correct,[27] whereas our results using eMOP are close to 86% correct using only one OCR engine plus denoising algorithms. Thus, eMOP is significant for having created an OCR workflow that can save significant amounts of time and money by achieving similar correctness rates in an open-source environment.

The worst documents in these collections have correctness rates so low that full-text searching is not possible. To test the performance of eMOP on these difficult cases, our team worked with Readex Newsprint to select a small sample set of documents containing problems that make OCR difficult such as skewing, uneven inking, uneven baseline, and image noise created by print bleed through from the back side of the page—the overleaf; 37 documents were selected (30 chosen by Readex and 7 by the eMOP team), comprising 2,120 page images. eMOP correctly identified 71% of the pages, whereas Readex and Gale identified 64%. These results suggest that the eMOP OCR workflow performs particularly well when page images are poor, showing the strengths of applying denoising and typeface identification algorithms.

[27]Prime Recognition reports a 95% correct OCR figure for their documents after 1721, with a different methodology. To allow comparison to eMOP, we have used our own computations here.

Over the course of the project, we learned a number of valuable lessons, summarized here:

— *Dealing with page images based on their noise characteristics.* We had to reconceptualize what kinds of triage are required after OCR'ing historical documents. In the original grant, the eMOP triage system was designed to identify issues in OCR output and then automatically analyze them for porting to an appropriate correction tool based on issue identification. Once development was under way, we found that the triage system was actually needed to select documents for image preprocessing and rerunning. This led to the development of ML mechanisms to identify large amounts of noise based on patterns in the OCR output. The denoising algorithm also provides several valuable clues: (1) which pages images contain pictures that can trip up an OCR run, causing the engine to time out before it finishes "reading" the page; (2) which page images present layout problems such as columns; (3) which page images need to be preprocessed to remove warping and skew; and (4) which page images will never be readable and need to be rescanned by libraries.
— *How to identify fonts without metadata information.* For more than 6 months, we were delayed by unexpected complications related to data collection and wrangling. Our various partners had diverse metadata and data formats. We now have what we believe to be the most complete and correct database of early modern documents and metadata. This eMOP database has been mined to produce results such as the Imprint Database, a record of every imprint line from books printed between 1,473 and 1,800 with accompanying record identifiers from various bibliographic resources.[28] We had hoped to have this database available at the beginning of eMOP, to use for font identification, rather than at its end: research still needs to be conducted to determine the typeface's associated with each printer named in the database. This issue motivated the development of the typeface identification algorithm (Section 2.2) that allows us to process documents without having any metadata information about printers and hence fonts. The algorithm identifies fonts so that we can rerun these documents with better training sets. We have already tested this process and found significant improvements (results not included).
— *How to train the Tesseract OCR engine for nonstandard fonts.* While dealing with data wrangling, our team was also forced to deconstruct our misconceptions about font training and OCR engines. Whereas OCR engines such as Gamera train on many instances of a specific character, we discovered that Tesseract training requires "ideal" font characters for optimal training, and thus we developed a font training creation tool (Franken +) to semiautomate the Tesseract font training process.[29] It may be that IBM's Concert tool did not significantly improve ABBYY Finereader results for the same reason.

Defending against the danger of the dark archive is not a question of open access to digitized materials but a question of how access to these materials can be responsibly structured, carried out, and *sustained*. From collections of digitized page images that require subscriptions (EEBO, ECCO) to open access collections of digitized materials (e.g., the Digital Walters[30]), scholars and technologists must come together and explore open-source solutions that will make these materials available for scholarly use and analysis. It is not feasible for every collection of digitized materials to organize a massive effort to manually transcribe their holdings, nor is it always possible for institutions to afford to hire companies to mechanically transcribe their holdings. Therefore, the eMOP team partnered with institutions and groups that had employed both of these methods (manual and mechanical transcription) to develop a solution that would combine both: experts would identify early modern fonts,

[28] http://emop.tamu.edu/outcomes/github/ImprintDB.
[29] http://emop.tamu.edu/outcomes/Franken-Plus.
[30] The Digital Walters is a project designed by the Walters Art Museum in Baltimore, Maryland, and funded by the National Endowment for the Humanities. It contains high-resolution archival images of more than 850 illuminated manuscripts and many of the first printed books (http://www.thedigitalwalters.org).

the OCR engines would be trained to "read" them, and then OCR output would be ported into crowd-sourced correction tools for scholars to perfect transcriptions.

It is not satisfying for current scholars to rely on the "dirty OCR" that runs behind many of these early modern databases and collections. Learning from and in collaboration with our eMOP partners, collaborators, and predecessors, our project team will be sustaining development into OCR technologies and correction tools. Immediately after completing the OCR of EEBO and ECCO, eMOP prepared to load the resulting mechanically transcribed texts into various tools that would allow the community to correct any remaining dirty OCR.

— *TypeWright*. In March 2016, we made almost all of the EEBO texts mechanically transcribed through our pipeline, available via 18thConnect's TypeWright tool. TypeWright was designed to allow crowd-sourced corrections for the dirty OCR running behind the 18th-century archive. Originally, TypeWright contained only documents from ECCO in an attempt to enlist experts in the field of 18th-century studies to manually correct the dirty OCR in transcriptions produced by Prime Recognition. eMOP, 18thConnect, and TypeWright are now offering the same opportunity to all early modern scholars. It is our hope that scholars work together to ensure that the untranscribed and poorly OCRd documents (e.g., the 85,200 documents so far not available as TCP-transcribed texts and therefore not searchable) can be fully searchable by future generations of scholars. Importantly, our agreements with Gale-Cengage and ProQuest authorize 18thConnect to release the corrected transcription of a document to the scholar-user who corrects it in TypeWright. This permits the scholarly community to not only improve the searchability of the corpus but also use the corrected text directly in their scholarly endeavors.
— *AWL Editor*. Although still under development, the AWL Editor will be integrated with TypeWright in 18thConnect to allow users to identify, transcribe, and encode paratextual elements on a page. Originally designed to allow users to fix incorrect line designations made by OCR engines (a function that is still available in the codebase), the AWL Editor will allow users to transcribe and encode marginalia, page headers, catch words, and glossed terms. The AWL mechanism will then save the coordinates of these paratextual elements, which are often overlooked or misidentified by OCR engines. Because the AWL Editor will be connected to TypeWright, scholars who transcribe paratextual elements will receive the corrected text and paratext, thanks to our agreements with Gale and ProQuest.
— *Anachronauts*. Developed by a team of undergraduate computer science students at Texas A&M, Anachronauts combines crowd-sourced correction of dirty OCR with gamification. The game entices users on Facebook to both improve the early modern corpus and learn about early modern printing culture. The game combines rewards with the historical context of the texts being corrected, as users take on the role of an early modern "editor" in a printing house. As users "standardize" copy text by identifying the correct transcriptions for single words, they can earn game currency to maintain the printing house's facilities, buy new printing presses or expand their business, and show off their achievements on the game leaderboard. Our next steps for the project will be to find permanent hosting space for the game and then create a mechanism for reconnecting images/words from Anachronauts with the eMOP OCR output automatically.

At present, we are pursuing two complementary directions of future work:

— *Deploying more OCR engines*. As noted, eMOP is significant for having created an OCR workflow that can achieve close to the same correctness rates as were achieved by the company Prime Recognition while using only one OCR engine compared to Prime Recognition's six. However, we might be able to outstrip the best OCR correctness rates for page images digitized from microfilm by adding OCR engines to the workflow. In fall 2015, work began on "Reading the First Books: Multilingual, Early-Modern OCR for Primeros Libros," a project funded by the National Endowment for the Humanities and led by teams at the University of Texas at Austin and Texas A&M. The 2-year First Books project will expand the

functionality of the Ocular OCR engine, originally developed by Taylor Berg-Kirkpatrick at the University of California at Berkeley. First Books deals with a truly multilingual dataset, as it pulls documents from the Primeros Libros project, a digital collection of the first books printed in Mexico before 1601, the first to be printed in North America. Although both Primeros Libros and the eMOP dataset share the challenges of early modern printing practices, the Primeros Libros collection contains documents written in combinations of English, Spanish, and several indigenous Latin American languages, such as Nahuatl. The eMOP team is currently working to incorporate Ocular into the eMOP dashboard and controller to continue investigating early modern printing practices from Europe to the New World.

— *Improving OCR accuracy.* Both Ocular and Tesseract achieve good results, and we are in the process of combining their outputs with that from a third OCR engine that will allow us to use voting schemes to further reduce OCR errors. In addition, Google has recently shifted focus from Tesseract to an OCR engine based on neural networks. Although this engine is not yet ready for open-source release, Google provides an API that can be used for page images and will soon make available the early modern training sets that they have developed for it.[31]

ACKNOWLEDGMENTS

eMOP would not have been possible without generous funding by the Andrew W. Mellon Foundation. Members of the Texas A&M University Cushing Memorial Library and Archives gathered typeface samples and created high-resolution page images for training the Tesseract OCR engine in early modern typefaces. The Koninklijke Bibliotheek (National Library of the Netherlands) provided invaluable guidance and advice based on previous and continuing work on the IMPACT and Europeana Collections projects. The Software Environment for the Advancement of Scholarly Research (SEASR) at the University of Illinois Urbana-Champaign developed the post-OCR portions of the eMOP workflow. The University of Massachusetts Amherst developed a ground-truth comparison algorithm. The PRImA Research Lab at the University of Salford provided us support on their Aletheia tool as we used it to develop training for Tesseract and created the AWL Editor based on this tool. Performant Software Solutions did the ground work development of the eMOP dashboard and controller, and created the Juxta-CL tool for ground-truth comparison. The Academy for Advanced Telecommunications and Learning Technologies developed our system architecture and finalized the eMOP dashboard and controller code. The TCP provided a measure of self-evaluation with their collection of hand-transcribed versions of more than 46,000 documents from our corpus to serve as ground truth. And none of it would have been possible without the partnership of both Gale-Cengage Learning and ProQuest, who provided us with the page images, metadata, and any available previous OCR transcriptions for their ECCO and EEBO proprietary database products, respectively.

REFERENCES

[1] E. Niggemann, J. D. Decker, and M. Lévy. 2011. *The New Renaissance: Report of the "Comité des Sages."* Office of the European Union.

[2] L. Mandell. 2017. What can you do with 'dirty OCR'? Digital literary history beyond the canon. *Presented at Instant History, the Postwar Digital Humanities and Their Legacies: A Day Conference.*

[3] A. Gupta, R. Gutierrez-Osuna, M. Christy, C. Boris, A. Loretta, L. Grumbach, R. Furuta, and L. Mandell. 2015. Automatic assessment of OCR quality in historical documents. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15).* 1735–1741.

[4] G. Crane. 1987. From the old to the new: Integrating hypertext into traditional scholarship. In *Proceedings of the ACM Conference on Hypertext (HYPERTEXT'87).* 51–55.

[5] R. Smith. 1995. A simple and efficient skew detection algorithm via text row accumulation. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition (ICDAR'95).* 1145.

[6] R. Smith. 2007. An overview of the Tesseract OCR engine. In *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR'07).*

[7] U. Reffle and C. Ringlstetter. 2013. Unsupervised profiling of OCRed historical documents. *Pattern Recognition* 46, 5, 1346–1357.

[31] Cloud Vision API: https://cloud.google.com/vision/.

[8] M. Reynaert. 2008. Non-interactive OCR post-correction for giga-scale digitization projects. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing*. 617–630.
[9] B. Alex, C. Grover, E. Klein, and R. Tobin. 2012. Digitised historical text: Does it have to be mediOCRe? In *Proceedings of KONVENS 2012 (LThist 2012 Workshop)*. 401–409.
[10] P. Ye and D. Doermann. 2013. Document image quality assessment: A brief survey. In *Proceedings of the 2013 12th Conference on Document Analysis and Recognition (ICDAR'13)*.
[11] R. D. Lins, S. Banergee, and M. Thielo. 2010. Automatically detecting and classifying noises in document images. In *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC'10)*. 33–39.
[12] N. Sandhya, R. Krishnan, and D. Babu. 2012. A language independent characterization of document image noise in historical scripts. *International Journal of Computer Applications* 50, 11–18.
[13] A. Farahmand, A. Sarrafzadeh, and J. Shanbehzadeh. 2013. Document image noises and removal methods. In *Proceedings of the International Multiconference of Engineers and Computer Scientists*.
[14] K. Ait-Mohand, L. Heutte, T. Paquet, and N. Ragot. 2010. Font adaptation of an HMM-based OCR system. In *Proceedings of SPIE 7534: Document Recognition and Retrieval XVII*.
[15] D. Ghosh, T. Dube, and A. P. Shivaprasad. 2010. Script recognition—a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 12, 2142–2161.
[16] R. Rani, R. Dhir, and G. S. Lehal. 2013. Script identification of pre-segmented multi-font characters and digits. In *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition (ICDAR'13)*. 1150–1154.
[17] G. Schohn and D. Cohn. 2000. Less is more: Active learning with support vector machines. In *Proceedings of the International Conference on Machine Learning*. 839–846.
[18] Y. Fu, X. Zhu, and B. Li. 2013. A survey on instance selection for active learning. *Knowledge and Information Systems* 35, 249–283.
[19] M.-R. Bouguelia, Y. Belaïd, and A. Belaïd. 2013. A stream-based semi-supervised active learning approach for document classification. In *Proceedings of the International Conference on Document Analysis and Recognition*. 611–615.
[20] G. B. Newby and C. Franks. 2003. Distributed proofreading. In *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*.
[21] L. von Ahn. 2006. Games with a purpose. *Computer* 39, 6, 92–94.
[22] L. von Ahn and L. Dabbish. 2008. Designing games with a purpose. *Communications of the ACM* 51, 8, 58–67.
[23] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. 2008. reCAPTCHA: Human-based character recognition via Web security measures. *Science* 321, 5895, 1465–1468.
[24] S. La Manna, A. Colia, and A. Sperduti. 1999. Optical font recognition for multi-font OCR and document processing. In *Proceedings of the 10th International Workshop on Database and Expert Systems Applications*. 549–553.
[25] M. B. Imani, M. R. Keyvanpour, and R. Azmi. 2011. Semi-supervised Persian font recognition. *Procedia Computer Science* 3, 336–342.
[26] R. C. Gonzalez and R. E. Woods. 2007. *Digital Image Processing* (3rd ed.). Prentice Hall.
[27] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis. 2002. Skew angle estimation for printed and handwritten documents using the Wigner–Ville distribution. *Image and Vision Computing* 20, 813–824.
[28] J. Illingworth and J. Kittler. 1988. A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing* 44, 1, 87–116.
[29] A. Khotanzad and Y. H. Hong. 1990. Invariant image recognition by Zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12, 5, 489–497.
[30] A. Tahmasbi, F. Saki, and S. B. Shokouhi. 2011. Classification of benign and malignant masses based on Zernike moments. *Computers in Biology and Medicine* 41, 8, 726–735.
[31] C. Wolf, G. Taylor, and J.-M. Jolion. 2011. *Learning Individual Human Activities From Short Binary Shape Sequences*. Technical Report LIRIS. Available at http://liris.cnrs.fr/Documents/Liris-5294.pdf.
[32] J. Sivic and A. Zisserman. 2003. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the 9th IEEE International Conference on Computer Vision*. 1470–1477.
[33] T. Kobayashi, K. Watanabe, and N. Otsu. 2012. Logistic label propagation. *Pattern Recognition Letters* 33, 5, 580–588.
[34] B. Settles. 2012. *Active Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool.
[35] K. Black. 2004. *Booklist/Reference Books Bulletin*, November 1.

Received April 2016; revised February 2017; accepted March 2017