Avancées en Reconnaissance Optique des Caractères pour les Documents Arabes Historiques

Benjamin Kiessling

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Pratique des Hautes Études

Avancées en Reconnaissance Optique des Caractères pour les Documents Arabes Historiques

Advances in Optical Character Recognition for Historical Arabic Documents

Soutenue par
Benjamin KIESSLING
Le 13 avril 2021

École doctorale n°472
École doctorale de l'École Pratique des Hautes Études

Spécialité
Informatique, mathématique et applications

Composition du jury :

Peter STOKES Directeur d'études École Pratique des Hautes Études	*Président*
Nachum DERSHOWITZ Professor Tel Aviv University	*Rapporteur*
Gregory CRANE Professor Tufts University	*Rapporteur*
Alicia FORNÉS Senior Research Fellow Universitat Autònoma de Barcelona	*Examinateur*
Daniel STÖKL BEN EZRA Directeur d'études École Pratique des Hautes Études	*Examinateur*
Marc BUI Directeur d'études École Pratique des Hautes Études	*Directeur de thèse*

École Pratique des Hautes Études | PSL

Acknowledgements

First, I would like to express my deepest gratitude to Marc Bui, who accepted to supervise this thesis, for very interesting discussions, for his valuable experience and guidance in matters both scientific and administrative.

I am so grateful to the eScripta team at EPHE and everyone at the Chair for Digital Humanities at the University of Leipzig. Both provided an environment were I was free to explore research ideas and their generous financial support made my existence as a doctoral student a comfortable one. It was Maxim Romanov through whom I had the first contact with Arabic text in Leipzig. I am also eternally indebted to Daniel Stökl Ben Ezra and Peter Stokes for fending off every imposition on my time during the last weeks of writing.

I thank the OpenITI team for their incessant lobbying to get institutional support for research into Arabic OCR. Without them, none of this would have been possible. A special thanks goes to Matthew Thomas Miller and Jonathan Allen who responded to my endless question on details of the Arabic script with a celerity unheard of in academia.

The people of ALMAnaCH at INRIA should not be forgotten. They welcomed a clueless foreigner on his arrival in France and integrated me in the lab.

I have wonderful friends who were there even at times when all they heard from me was about this thesis. A huge thanks goes to Louise for helping improve the French summary. Above all I thank Monika for reminding me in the darkest moments that instant ramen is not the foundation of a balanced diet.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation consists of a collection of publications and proposes several methods with which to facilitate the retrodigitization of historical Arabic-script material. While we are only concerned with the Arabic script here, our findings are relevant to the analysis of writings and inscriptions in other writing systems.

1.1 Document Image Analysis and Optical Character Recognition

Document Image Analysis (DIA) is a subfield of Computer Vision (CV). It aims at understanding document content through the processing of its associated digital images. The term "document" is defined loosely as including both handwritten and printed text on paper, as well as writing on other supple supports (e.g. papyrus and palm leafs) or even inscriptions.

Rather than the methods employed, it is the nature of the input images that differentiates Document Image Analysis (DIA) from other fields in Computer Vision (CV). These images are usually obtained through cameras or scanners, often in a professional setting, resulting in source material with minimal noise from non-pertinent elements which are often encountered in the natural scene images treated by other branches of CV. Notwithstanding the cleaner input data, the structured representations desired as output tend to be of higher complexity and quantity in DIA than other applications, requiring detection, classification, and relation of dozens to hundreds of document elements such as lines, characters, illustrations, and tables.

Like other fields of computer science, DIA research can be subdivided into particular tasks, and specific and targeted methods are designed to solve one or more of them. The most prominent task in DIA research is optical character recognition (OCR)[1], although other tasks also exist, whether they be based on OCR or entirely novel (e.g. document classification and dating, or keyword spotting).

OCR is the conversion of printed, written, or inscribed writing into machine-encoded text. It is a well-established process, both as a task in computer vision research as well as for practical day-to-day applications, ranging from address parsing to aids for the blind. As a matter of fact, it is the latter which motivated the first and earliest attempt at creating a document image analysis system, with an 1809 US patent for a non-tactile reading instrument. Early systems were rudimentary and their output required significant human interpretation. Fournier D'Albe's 1914 optophone, for instance, converted strokes into tones and expected the reader to interpret them mentally as character information. Such systems were little more than intellectual curiosities at the time and none of them achieved widespread use.

[1]While in most contexts optical character recognition and handwritten text recognition are treated as distinct, both are subsumed under the term OCR here. A detailed justification is given in 1.5.2

These early explorations preceded the invention of computers by several decades. Their evolution into modern-day DIA techniques has allowed for a broad range of applications in tasks such as address parsing for mail routing, cheque verification and book retrodigitization. It is now a claim largely unchallenged in the field that OCR is fundamentally solved at least for modern, machine-printed documents in English with a reasonable low level of noise, for which modern commercial retrodigitization software achieve character accuracy rates above 99% routinely. Nevertheless, while this holds true for English, we count almost four-thousand other written languages and several hundred associated writing systems or scripts. No practical OCR systems are available for the vast majority of them. Even accounting for the use of purely alphabetic scripts such as Latin and Cyrillic, which present less of a challenge to state-of-the-art OCR when employed accordant with modern western typographic practices, it is clear that a substantial proportion of human literary output is not yet accessible through retrodigitization.

This is all the more true when we consider historical literary output. While large scale digital scanning in rich countries has resulted in the creation of substantial digital libraries, these text are de facto inaccessible to both the public and scholar, even for material as recent as the late nineteenth century. Typographical and orthographical variations degrade digitized texts' quality to a significant extent when transcribed with software geared towards the treatment of modern documents. For most archival material from the Global North, this is most likely a temporary situation as projects such as OCR-D[2] pave the way for greater integration of pure DIA research into library practice. Other collective and more specific efforts include [1], which gathered both humanities scholars engaged in digital methods as well as computer vision experts, with the shared goal of establishing a research program for the digitization of historical and *minority script* material. Nevertheless, these communities of interest remain fractured along geographical, linguistic, and professional boundaries.

Meanwhile, the threat of permanent loss of cultural heritage looms over collections, at risk of permanent deterioration due to political unrest and ill-adapted storage conditions, combined with utter lack of funding and limited interest from parties other than minority populations and a small number of scholars. Even famed collections such as the manuscripts of Timbuktu have barely escaped destruction due to conflict in recent years, and humidity continues to threaten their integrity very much.

Many writings lack a fundamental technological basis to work with: the Berkeley Script Encoding Initiative alone lists over a hundred writing systems that remain to be encoded in Unicode. Circa two-thirds of them are historical and a substantial remainder is used liturgically. Without a standardized way to represent them digitally, their retrodigitization and dissemination becomes infinitely more difficult. A number of these writing systems have substantial scholarly commu-

[2] `http://ocr-d.de`

nities – ranging from Egyptian Hieroglyphs and Demotic, Cuneiform, to a variety of Chinese scripts. Even scripts already encoded in Unicode often lack code points for certain surface forms required for paleographic or epigraphic practice. This is not always an oversight. Rather, it can be the result of the Unicode Consortium's encoding guidelines which largely proscribe inclusion of new allographs and ligatures in the standard. While alternative standards exist (e.g. the code tables as defined by the Medieval Unicode Font Initiative [3]), this situation favors the emergence of ad-hoc standards and thus limits interchangeability and machine-readability of text considerably.

1.1.1 Tasks

As one of the core applications of document image analysis, OCR has dwelt upon a large number of its subproblems. Some of them have become obsolete with time, thanks to the increasing capability of new algorithms. Others are new, resulting from the need to deal with increasingly challenging material. It is not necessary to deal with all of them at once to have a functional OCR system. As a matter of fact, many of them are material-specific, and are of concern only when dealing with specialized applications. A closer analysis of the design requirements for a largely script-independent OCR system, capable of processing Arabic-script text based on a survey of its calligraphic and typographic features, will follow in chapter 2.

A non-exhaustive compilation of generally accepted tasks can be found below.

Binarization Classifying the pixels of an image into two classes: foreground, i.e. text, and background, i.e. everything else.

Denoising Increasing the page image quality for subsequent problems. Denoising includes processes such as background normalization, color space adjustments, deblurring, or stain removal.

Deskewing and Dewarping Correcting both the perspective distortion inherent in camera capture and other degradations introduced commonly in scanning setups such as rotation, warping along the binding, ...

Region Segmentation Subdividing a page image into components such as text, decoration, notes, ...

Text Line Segmentation Extracting the text lines from a page image. Text line segmentation is notable for being a task where not only a large variety of techniques exist but the modellisation of the line itself has been subject to considerable research.

Character Segmentation Segmenting text on a page image down to the glyph or even lower level. While a common operation in traditional OCR systems

[3] http://mufi.info

it is mostly unnecessary with state of the art methods. A related task that is of interest to the humanities, especially paleo- and epigraphers, is character alignment, i.e. the locating individual characters on a document page given the full page text.

Script and Font Recognition Classifying the language, writing system, style, or typeface of the text. This classification can be performed at different levels, such as document-wide or individually for whole or parts of a line.

Table Recognition Inferring the logical structure of table images.

Scene Text Recognition Recognition of text in images taken in the *wild* that contain substantial non-text content.

To solve these problems, the typical optical character recognition pipeline uses specific methods, which operate in three distinct steps:

Preprocessing Denoising, deskewing and dewarping, and binarization

Layout Analysis Extracting structural information from document page images and enriching it with additional semantic information.

Transcription Extracting textual information from all or a subset of objects identified by in the layout analysis step.

This characterization holds true for all but the most esoteric pipelines. However, the exact functional blocks depend heavily on the type and structure of the documents processed. For example, traditionally, binarization is used as a simple process to: enable the use of fast binary morphology in the layout analysis step; reduce the dimensionality of data for classifiers; or, more generally, as a type of basic feature extractor. The relative ease with which accurate binarization for high quality scans of machine-printed text on paper can be computed contrasts with the difficulty of treating documents on other writing surfaces, faded writing, fragmentation, etc. As a result, methods geared towards the treatment of the latter kind often attempt to reduce the reliance on binarization or skip it entirely. Similar to most other preprocessing steps, there is a tendency today to eliminate binarization altogether, due to the increased availability of more advanced techniques. This, however, remains a topic of debate in the research community.

1.2 Motivation

Arabic-script material represents one of the largest literary traditions in human history, both in terms of volume and geographical spread. Examples range from religious texts (most prominently the *qur'ān*, the holy book of Islam) to poetry, and include scientific and legal texts in addition to a large corpus of administrative records. The sheer number of sources and the diversity of domains they

cover make them a prime target for new paradigms in the humanities that employ computational methodology, e.g. *distant reading* and quantitative palaeography. These methods require either large text corpora or accurate DIA methods based on one or more of the abovementioned component tasks of an OCR system. As the vast majority of Arabic texts have never existed in digital form, high quality retrodigitization through OCR favors the development of a substantial number of Arabic Digital Humanities research projects.

When I started working on this thesis, humanities scholars working on Arabic-script texts largely dismissed OCR of machine-printed Arabic text, especially of historical and multilingual material. This was even true for scholars already deeply involved in the digital humanities, as well as funding agencies. At the time, accurate Arabic handwriting recognition was deemed completely out of reach. A long trail of publications on OCR of simple Arabic datasets, such as KHATT [2], existed and a number of open and proprietary OCR software (e.g. Tesseract, Abbyy FineReader, Sakhr) did offer nominal support for the recognition of Arabic-script text. However, it never translated into actual scholarly or large-scale library practice. A multitude of factors accounts for this situation: high error rates caused by classifiers and segmenters ill-suited to the cursive nature of the writing system; a lack of readily available software and technical expertise; and substantial cost and effort required to adapt existing solutions to the material of interest.

It was soon evident that the challenges that prevented scholars working on Arabic-script printed and handwritten texts from relying on OCR mirrored that of many other researchers engaged in retrodigitization of historical and non-Latin script material. Imitating the prevailing opinion on Arabic OCR, [3] claimed medieval (Latin-script) manuscripts to be practically impervious to contemporary OCR. Similar statements can be found elsewhere. This entailed an overall lack of established best practices, data formats, and requirements on software capability and interfaces suited to the workflows of digital humanists. Existing projects like Lace[4] and [4] used OCR technology in an ad-hoc manner. It often resorted to extensive *data carpentry*[5] and incorporated significant domain knowledge to boost accuracy to an acceptable level.

Our goal is therefore twofold. While the research presented here incorporates an understanding of the Arabic writing system and its associated calligraphic practices, we aim at conceiving a largely script- and language-independent *universal* OCR system, one that is useful beyond the immediate community of Arabic-script scholars. It also allows for methods to be evaluated against non-Arabic datasets when Arabic datasets are not available. As such, if the algorithms presented here are particularly suitable to Arabic material, it has been of primary importance in this research to create algorithms that are universally applicable.

[4] `http://heml.mta.ca/lace/index.html`

1.3 Scientific Contributions

Most of the work presented here should be read in relation to the broader field of Digital Humanities. The research presented in the next chapters is not a heterogeneous collection of methods solving single tasks in DIA that are only of use to humanities scholars engaged in retrodigitization. It is part of a coherent ecosystem consisting of two major components: the Kraken OCR engine and the eScriptorium virtual research environment (VRE). As such, not only does this research aim at advancing methods for particular tasks; it also seeks to improve existing scholarly workflows, ones that are more often than not laborious and impractical to use, not to say completely unworkable.

Kraken is a feature-complete, freely-licensed and modular OCR system. It differs from other solutions (both open and proprietary) in multiple and important ways, i.e. target audience, software design, and generalizability. These differentiated features are the result of its roots as a large-scale refactorization of the OCRopus source code, that was performed to enable its integration into a digitization pipeline for scholarly use at the Chair of Digital Humanities at the University of Leipzig. It boasts a stable application programming interface and is also highly modular. Users can create their own workflows or substitute functional blocks with minimal effort. Recognizing the sheer diversity of OCR systems-related needs among humanities scholars, a concerted effort has been made to limit implicit assumptions regarding the functioning of text, and the software accommodates varied material and transcription guidelines as much as possible. As a result, Kraken performs only minimal normalization, is fully compatible with Unicode private use area (PUA) utilization, and supports both horizontal and vertical directions.

A number of case studies were performed as part of the work to enhance Kraken's capacity to support Arabic text work. It led to the first detailed analysis of state-of-the-art OCR methods on machine-printed Arabic text, evaluating their respective weaknesses and strengths. A first preliminary study was conducted on a small number of printed classical Arabic-language books, soon followed by a large-scale retrodigitization feasibility study using a leading Arabic-language journal published by the American University of Beirut.

Partly as a result of these studies, the engine has been extended in multiple ways. This thesis contributes two trainable line segmentation systems, a basic one capable of only detecting baselines, and a more advanced one allowing region and line segmentation in addition to classification. The latter is included in Kraken. Initially, a trainable line segmentation method constructed on top of a U-Net for semantic segmentation was developed. In a second step the training procedure was adapted to allow joint region and line detection and inference of line orientation. This second method has also been optimized for memory usage by employing a ReNet-like stack of separable recurrent layers, reducing memory consumption by circa fifty per cent in comparison to similarly performing fully convolutional semantic segmentation networks.

In addition, a basic script and emphasis detection system, built on Kraken's text transcription module, was devised.

The segmenter in Kraken is currently the only openly available layout analysis module in a complete OCR engine which is able to accurately segment complex curved lines in Arabic manuscripts. In addition, it is the first method following the baseline paradigm for line modellisation incorporating line orientation detection.

A flexible abstraction layer on top of the pytorch neural networking library has been added which allows the flexible reconfiguration of the artificial neural networks (ANN) employed for both layout analysis and text transcription through a lightweight ANN definition language which is able to express many features of common architectures employed in computer vision tasks (see appendix B). This new layer allows the relatively simple addition of new layer types and thus quick prototyping and efficient hyperparameter optimization even for endusers without in-depth machine learning knowledge as has been demonstrated by [6].

In contrast to older open engines such as Tesseract and OCRopus which use custom neural networking backends, a standard library in widespread use in both industry and the machine learning research community offers a multitude of benefits such as easier transfer of development skills and automatic or low-effort inclusion of performance improvements and additional features like GPU acceleration, distributed training, or model quantization.

[1, pg. 19] notes that one of the main obstacles to advancing OCR for historical and non-Latin texts is lack of training and evaluation datasets. During the process of putting together the initial case studies, important efforts were made in this direction. Several thousand lines of training data for text transcription were annotated and made publicly available as freely-licensed datasets. I was involved in the process through technical conceptualization and the creation of transcription guidelines. In addition, to evaluate the proposed layout analysis methods against historical Arabic-script material, another openly licensed dataset comprising of four hundred Arabic-script manuscript pages was annotated with baselines and line orientation. It is purposely diverse in the languages, styles and domains it covered. The composition of this dataset is particularly challenging and it remains the only handwritten non-Latin dataset available for the baseline paradigm for layout analysis.

The second component of the proposed OCR ecosystem is the eScriptorium VRE. eScriptorium is currently under development at Université Paris Sciences et Lettres, under the umbrella of the eScripta Project. While Kraken is designed for maximum flexibility by offering well-defined interfaces at different levels, eScriptorium takes another approach. It is conceived as a complete paleographic research and publication environment for scholarly use, and OCR is but one of its planned features. This is why much effort has been put into designing a user-friendly OCR workflow whose different steps are explained clearly, without scarifying any of the required flexibility in addressing a large number of scripts and languages. While the above-mentioned advances in the Kraken OCR engine make it a versatile tool suitable for a multitude of scholarly purposes, eScripto-

rium cannot possibly expose its full functionality without devolving into a highly specialized tool for OCR exclusively. As a result, eScriptorium is designed to allow manual or semi-automatic intervention at each step of the process, either through the manual manipulation of the interface or via graphical and programmatic data exchange interfaces.

eScriptorium is also an ideal test case for computer vision-assisted research in the humanities, as the platform aims to offer additional scholarly functions that stretch far beyond pure retrodigitization. Potential functions include text reuse detection, automatic sampling of graphemes for palaeographic analysis, document classification, ... In certain cases, advanced functionalities are linked with text transcription. To illustrate this point, a method for deriving grapheme locations from the implicit alignment produced by a line-based text recognition ANN trained with Connectionist Temporal Classification loss and its performance on fragmentary Hebrew material is presented. Depending on the user's choices regarding transcription standards , different palaeographic sampling goals can be met with this method, ranging from semi-automatic allographic inventories to decoration extraction.

1.4 Outline

The remainder of this chapter will be dedicated to a review of computer vision techniques in general and as they pertain to optical character recognition. This includes a summary of the state of the art in research and practical available software packages, in joint with an analysis of general challenges faced by both in a variety of settings.

All subsequent chapters included in this dissertation have been published as scientific articles or have been accepted for publication, except for the conclusion and the presentation of the Arabic script. Since the work presented here is closely linked with the development of the Kraken OCR engine as well as the eScriptorium project, notable differences with current implementations are mentioned in introductory notes to each chapter whenever necessary. This thesis is organized into four parts: introduction to the Arabic script, segmentation and script recognition, character recognition and alignement, and virtual research environments.

Part I focuses on the Arabic script. It is organized into three chapters. Firstly, it proposes a general introduction to the writing system with an emphasis on calligraphic features, and the ensuing specific requirements associated with the development of an Arabic-script OCR system. Secondly, it presents a preliminary study on classical Arabic machine-printed books. Finally, it concludes with an in-depth case study performed on a printed Arabic language journal.

Part II contains three chapters on layout analysis and segmentation of Arabic-script documents. We present a first of its kind, freely-licensed, Arabic-script historical manuscript dataset and a competitive method to perform basic layout analysis on it. In the second chapter a novel, modular method for region and

text line segmentation is presented. Lastly, a method to perform sub-line script classification on printed multi-scriptal text for multi-lingual OCR is shown.

Part III is composed of two articles: a general description of the Kraken OCR engine design and features, and a method to perform character alignment on highly fragmentary Hebrew manuscripts.

The final part IV presents the virtual research environment (VRE) context in which a modern OCR engine like Kraken can be embedded. We do so through two articles: one containing a conceptual description of the eScriptorium VRE and a second one investigating the friction between automatic processing, standardization, user-friendliness and methodological pluralism in the humanities.

1.5 Literature Review

In this literature review, we start with a general review of the existing techniques in Machine Learning or Artificial Intelligence (ML/AI), Computer Vision, and Document Image Analysis, as well as current trends in the research community, with the view of contextualizing the state of the art for the specific task of OCR. This brief survey not only includes the current state of OCR research; it also covers libre and proprietary OCR engines as well as workflow engines, most of which are used primarily for large-scale digitization in a institutional context. Since the present dissertation aims, among other things, at advancing the practical application of OCR for humanities research, this section concludes with a review of the existing virtual research environments targeted at the retrodigitization of historical material.

1.5.1 Computer Vision Techniques

Commercial DIA (including computerized DIA) preceded by several years the creation of both Computer Vision and Artificial Intelligence as academic fields [7, pg. 11-14]. With the establishment of CV and AI as fields however, DIA research progressively came under their larger research umbrella. Today, techniques that became obsolete after this *fusion* remain of interest only to historians and have largely disappeared from academic discussions. This review therefore limits itself to methods established after the late 1960s.

Computer vision processes are generally divided into the following steps:

Image acquisition refers to the capture of an digital image through one or multiple image sensors. Limitations of a particular image acquisition system, such as noise levels and distortion, are often integral in the design of subsequent steps. The most common acquisition systems in DIA are visible light cameras and flatbed scanners, although in some cases multispectral and radiographic sensors are employed.

Preprocessing aims to boost the performance of subsequent steps through normalizing input data. It is frequently targeted at eliminating degradations introduced during image acquisition.

Feature extraction reduces the dimensionality of input data through combination and selection of its characteristics that are deemed pertinent for the desired analysis.

Analysis transforms extracted features into an output representation specific to a particular task, e.g. class probabilities for an image classification task, text for OCR, object locations and labels for object detection, ...

These steps have developed over time and their relative importance have changed with each new paradigm shift in CV and adjacent fields such as machine learning, as well as with increasing computational capabilities. As a matter of fact, advances in research have often emerged across all steps simultaneously. To understand how the current research context has emerged and to best capture the relationships that tie the different methods together, a chronological account of the development of computer vision is presented below.

Early computerized DIA relied on rather rudimentary OCR systems. They barely differed from the early opto-mechanical systems that dominated the first half of the 20th century. Their limitations were similar and severe, and stemmed from using rudimentary template matching, with various attempts at preprocessing to increase accuracy as generalization was generally poor. While at the time of the infamous 1966 CE summer project on pattern recognition [8], the currently most popular machine learning paradigm, artificial neural networks, had existed for more than twenty years they did attract little interest in the field. Minsky and Papert's 1969 CE book on perceptrons [9] relegated research on ANNs to at best secondary role and caused a decade-long stagnation in the field in favor of alternative approaches.

Advances in the 1970s included rule-based expert systems, the popularization of various low-level filters and operators such as gradient approximators, median and gaussian filter smoothing, and morphological operators. The 1970s also saw the first attempts at feature representations such as edges, corners and binarization[10] of images that were suitable to the limited modeling capability of the classification methods of the time. By the early 1990s, a bewildering array of hand-crafted feature representations had been devised, not to mention an ever growing collection of edge and corner detectors, contour and shape descriptors, unsupervised segmentation methods, and other transforms. Assemblies of these carefully selected features were usually coupled with relatively simple unsupervised or supervised classifiers such as k-nearest neighbor, multilayer perceptrons, or decision trees. [11] , a comparative study which focuses on large-scale digitization methods using US census records, is an early example of this trend combining complex feature descriptors and relatively simple neural and non-neural classifiers. It is around this period that it became standard practice in both research

and industry to construct completely hand-crafted heuristic methods based on a combination of low-level instructions to perform high-level CV (i.e. true analysis and interpretation of image data).

Methods that follow this approach work on narrow document domains reasonably well. However, they do not generalize to larger classes of documents. This is due to the fact that the selected features are often relatively specific to the source material. In addition, their adaptation is labor-intensive.

Paralleling the proliferation of feature descriptors in the 1970s, classifiers and training methods gained power throughout the 1970s and 1980s. What would later come to be known as convolutional neural networks started to emerge in the form of learned feature maps and weight sharing in ANNs. Fukushima's 1980 *neocognitron* [12] and LeCun's 1989 CNN [13] are two such examples. Both of them were designed for character recognition purposes. Backpropagation [14], an algorithm allowing for the efficient supervised training of functions through gradient descent (which remains the standard for supervised learning today), enabled, in theory, for the first time, the supervised training of deep neural networks. In practice, the vanishing/exploding gradient problem, i.e. the tendency that cumulative error signals backpropagated through the network either shrink or grow rapidly with each antecedent layer, a phenomenon that was first identified by [15], proved to be a difficulty. As a result, ANNs were limited to shallow problems, for which most computer vision applications still required extensive feature engineering. By the mid-2000s however, the problem was, for the most part, solved. In the case of recurrent neural networks (RNNs), alternative architectures such as Long Short-Term Memory (LSTM) units proved to have more stable gradients. For most other ANN architectures, increased computational power and large datasets proved instrumental in circumventing the issue: it allowed training with small gradients without overfitting and in a reasonable amount of time. The literature from the period is prolific in identifying alternative solutions and circumvention methods, including unsupervised pretraining, hessian-free optimization, gradient-less training, and ensemble methods [16, sec. 5.9]. None of them, however, are currently in widespread use.

Multiple alternative classification methods for computer vision filled the gap that separated the emergence of feature descriptors and the establishment of deep ANNs' predominance in today's landscape. Hidden Markov Models (HMMs), that were already used in the speech recognition field successfully, started to be used for the modelization of sequences in computer vision. One notable application was cursive handwriting recognition [17]. The soft-margin formulation for Support Vector Machines (SVM) and the kernel trick extended the use of linear classifiers to data that is not linearly separable. While either of these methods have largely been surplanted by ANNs in CV, they remain popular in certain parts of the DIA community, most notably in the use of HMMs for Arabic text recognition research.

The major resurgence of ANNs for computer vision can be traced to significant improvement to the overall state of the art shown by deep convolutional

neural networks trained with straightforward backpropagation on a number of image classification contests in 2011 and 2012, often halving the error rate in comparison to previous years and in some cases achieving superhuman performance on constrained domains. While the contests in question, foremost on the *ImageNet* dataset, were limited to image classification, deep convolutional networks disposing completely of hand-crafted features were rapidly adapted to other tasks such as object detection, semantic segmentation, optical flow, image captioning, ...

Advances in neural network design in the following years were, disregarding brief periods of popularity of alternative network architectures and training schemes, mostly driven by attempts to increase the feasibly trainable depth of deep CNNs for image classification. The initial 2011 eight layer AlexNet [18] already contained staples like ReLU activation functions which diminish gradient vanishing in layers, dropout regularization, and data augmentation to inhibit overfitting. The 2014 VGG-Net[19] with up to nineteen layers introduced an architecture made up solely of small 3×3 convolutional filters, increasing the effective receptive field through additional layers. The 2015 Resnet [20] preserves the error signal by introducing shortcut connections skipping one mor more layers effectively training the network as an ensemble of shallower subnetworks [21]. SkipNets [22] and Highway Networks [23] follow the same idea but parametrize the data flow between layers, comparable to the gating mechanism in LSTM RNNs. DenseNets [24] go even further and input the concatenated feature maps of all previous layers directly into the next layer. With ever-increasing model complexity, techniques to reduce both the number of model parameters and operations required were devised: SqueezeNets [25] introduced 1×1 bottleneck filters for dimensionality reduction, neural architecture search was used to find the more efficient AmoebaNet architecture [26], and [27] investigated principled hyperparameter choices to increase computational efficiency.

The deep learning paradigm fundamentally relies on large datasets, *ImageNet* contains fourteen million images organized in more than twenty-thousand classes, which are often not available in fields working with a more constrained data basis such as DIA. This constraint, in addition to computational efficiency, have resulted in the popularization of using all or a part of the convolutional layers in deep networks trained on large image classification datasets as good general purpose features even for vastly different target domains, i.e. performing *transfer learning* for the desired task. Depending on how these layers, the so-called *backbone architecture*, are incorporated, this technique can be seen either as supervised pre-training or as a fixed feature extractor. In the latter case, a shallow ANN is trained on top of the features maps computed by the deep CNN, leaving the convolutional layers untouched. In the pre-training configuration, the last layers trained to produce the representation for the task at hand are trained jointly with the convolutional feature extracting layers in a process called *fine-tuning*. As the weights in the pre-trained layers are already expected to be relatively good, the training hyperparameters are often chosen separately for layers

trained from scratch and ones to be fine-tuned. Hybrid schemes exist as well. These keep some layers fixed and fine-tune others, usually under the assumption that the first layers compute generalized features such as edges, corners, ...while later layers compute more task-specific features that require adaptation.

Pre-training in this vein has generally been seen as an important success, as it broadens the applicability of deep learning in CV tasks significantly [28]. However, its relevance has been put into question recently [29], [30], with some indication that deep CNNs performing better on the image classification task do not necessarily translate into better features for other tasks [31].

Methods that derive in one way or another from discriminative convolutional image classification networks dominate the landscape for the majority of tasks. However, other approaches exist. Recurrent neural networks, i.e. ANNs with connections between layers along a temporal sequence, have been popular for sequence modelling ever since training on arbitrary length sequences became possible after the invention of LSTM units. RNNs have been used in the processing of sequential visual information, e.g. video recognition, natural language description of images, connected writing recognition, robotics, ...They have also been adapted to tasks that are not conventionally sequential such as image classification [32] and [33] segmentation. Nevertheless, RNNs do not rule out the use of convolutional feature extractors. Common constructions such as [34] either combine fixed convolutional feature extractors or train smaller convolutional stacks from scratch. In some cases an additional attention mechanism is inserted, which allows *decoding* recurrent layers to weigh certain areas of the input feature maps dynamically [35].

While LSTMs constitute a powerful and versatile sequence modelling method, one major problem is the lack of a differentiable loss that could allow the supervised training of RNNs without an explicitly given alignment between inputs and output sequences. Traditional losses such as cross-entropy require an explicit output-target pair for each time step. If the network output sequence is of different length than the input sequence, it entails spreading target labels across multiples steps through some mechanism, usually manual annotation. The 2006 Connectionist Temporal Classification (CTC) loss [36] permits alignment-free training, thus solving the issue for many computer vision applications through an efficient dynamic programming based algorithm that sums the probabilities of possible alignments of network output and target. It has some limitations: chiefly a requirement for the target sequence to be of shorter length than the network output, an assumption of conditional independence between target labels, and the complexity of fast and numerically stable implementations. However, these are rarely important in applications of non-linguistic sequence modelling in CV.

Other approaches aimed at solving the alignment problem in sequence modeling exist, although none of them are as prominent as CTC in the DIA domain. Attentional models decouple input and output sequences completely and can thus be trained with standard losses. They are in widespread use in natural language processing due to the limitations of CTC, and have had a brief period of popular-

ity with applications in image [35] and video [37] captioning, sequential object recognition [38], and whole paragraph handwritten text recognition [39], [40]. Transformer networks, on the other hand, are advanced self-attentive networks which have significantly improved the state of the art in NLP, dispensing with both convolutions and recurrent layers. They have also recently made inroads in both OCR [41] and non-sequential CV tasks such as semantic segmentation and image classification [42].

All methods described above are trained in a discriminative manner. In other words, models learn the conditional probability $P(Y|X)$ of a target Y given an observation X. By contrast, generative models learn the joint probability $P(X, Y)$, which can be used for classification using Bayes' rule but also has other applications. It notably allows the generation of realistic synthetic data by sampling the model's latent space. A particular construction to train generative neural models in an unsupervised or semi-supervised manner are generative adversarial networks (GANs) [43], which became popular after 2014. They consist of two adversarial ANNs: a generative model that captures the data distribution and a discriminative model that is tasked with distinguishing real data sourced from the training dataset, using synthetic samples created by the generator. Both are trained simultaneously, usually until the discriminator fails to distinguish generator-produced output. Deep GANs were used to establish a number of CV tasks such as image synthesis from text [44] and conversion from one image domain into another[45]. Their impact on DIA research has been relatively modest, although the literature does contain various image reconstruction and enhancement methods [46], [47] and at least one layout analysis system [48] built upon GANs.

Other deep neural generative models exist but they have generally fallen out of favor. Autoencoders, i.e. networks consisting of a contracting encoding and expanding decoding path trained on reconstruction loss, were widely used for unsupervised pretraining[49], [50] before the advent of supervised ImageNet features. Some newer methods in DIA still use them in tasks such as binarization [51] or page segmentation [52], [53].

1.5.2 State of the Art in OCR

As a subfield of CV, DIA and OCR use similar techniques: for tasks that are most popular among researchers, methods generally resemble that of the wider CV and machine learning community. For other tasks that do not benefit from the same amount of research (e.g. reading order determination), the picture is different: they remain the domain of seemingly obsolete and ineffective algorithms and techniques that do not compare well with modern alternatives.

Traditionally, both the DIA research community and commercial applications make a distinction between methods designed to process machine-printed (OCR) and methods geared towards hand-written text (HTR). In the second case, recognition is further sub-divided into *offline* and *online* text recognition, i.e. between the processing of already completed handwritten text and the recognition of text

using information on the production process such as stroke orders and paths. As our interest is largely limited to historical material, we will ignore online text recognition.

Several reasons explain the traditional distinction made between OCR and HTR. Handwritten material is naturally less uniform and regular than machine-printed texts and is generally considered to be more challenging. In addition, until a few years ago, many printed OCR methods were designed around character classifiers which only processed isolated characters and as such required commensurate layout analysis methods to extract single characters from the page. This paradigm is fundamentally not suitable for connected or cursive text. This fact was recognized as far back as 1973 [54] through Sayre's paradox: to be able to recognize a (cursive) handwritten word it is necessary to segment it into characters but segmentation is not possible without recognizing it first. In OCR research, the Latin script is the writing system which has attracted most interest. When machine-printed, the Latin script appears in a disconnected form, and by contrast its cursive form was until recently almost exclusively handwritten. As such, when describing the fundamental capabilities of a text recognition system, it is appropriate to see the two terms as short-hands for *block letter* and *cursive* text recognition.

As modern text transcription methods are generally able to process both block and cursive text with identical error rates, the two terms are now almost interchangeable if we only take into account the conception of the text transcription method. Certainly, there are differences in capability between text recognition systems that are optimized for handwritten material and those geared towards machine-printed material. However, since the two terms do not convey enough information about the operating principles of the overall recognition system anymore (*hard* segmentation and character classification in OCR *vs* implicit or probabilistic segmentation and sequence modeling in HTR), they will be used interchangeably from here on.

Following the above-mentioned three step model of OCR pipelines, we will now look at the state of the art in both the research community and practically available OCR systems.

Preprocessing

The first step afer digitization of the source material in an OCR pipeline is preprocessing. Preprocessing includes a wide variety of tasks aimed at improving the document quality and in the case of binarization preliminary classification to aid in the subsequent layout analysis and text transcription steps. While for modern machine-printed material the inventory of commonly employed preprocessing methods is fairly static and often rely on hand-crafted algorithms for denoising, deskewing, binarization with a general trend of a reduction in preprocessing over time as later steps in the pipeline have become more powerful, the capabilities of new deep learning techniques have resulted in a renewed interest

in image enhancement and reconstruction for historical and degraded material. Two main operations are performed for preprocessing: document image enhancement/normalization and document binarization.

Image enhancement and normalization encompasses a multitude of specific tasks. Denoising, deskewing and dewarping are the three tasks that have the longest and most extensive history of research. Image denoising is a classic task in low level vision. It aims at recovering a noise-free image from a noisy observation affected by environmental factors such as low light, acquisition method, and transmission channels. To remove this noise while preserving salient image features, classical denoising methods utilize spatial filters such as gaussian or median filters, image transforms, or morphological operations.

Their accuracy is often conditioned to the existence of an accurate model of the type of noise that is present in the digitized material. Elementary denoising algorithms are implemented in OCR engines such as OCRopus [55] and OCR4All [56], digitization workflow engines [57], and dedicated preprocessing utilities [58]. A short review discussing the range of manually constructed denoising algorithms can be found in [59]. Newer methods employing deep learning also exist [60]–[63], although they do not target document images specifically.

Deskewing and dewarping aim at correcting deformations introduced during the digitization process. While deskewing is limited to correcting simple rotation, dewarping includes the correction of perspective and lens distortion, non-planar surfaces, ... Both are primarily intended to support layout analysis. They do so by ensuring that individual lines are horizontal and can thus be modeled accurately by algorithms which assume straight lines and single-orientation writing or printing. Conventional skew detection methods [64]–[66] exploit this assumption mostly by searching for a skew angle that produces a characteristic pattern between dark and light areas in the projection profile. Likewise, dewarping techniques such as [67]–[69] either assume straight lines or that the writing surface in the unwarped state is rectangular [70]. In general, systems as described above are only useful for modern-machine text of decent quality, and are frequently included in OCR pipelines that are optimized for print processing (e.g. Tesseract, OCRopy, or Abbyy FineReader). A few neural dewarping [71], [72] and skew detection [73] have been proposed. However, a large corpus of document images contains lines in multiple orientations, not to mention the advent of LA modules capable of accurately detecting rotated and warped lines. As a result, their relevance to state-of-the-art OCR engines can be put into question.

Other more advanced enhancement methods have also been studied, albeit with much less frequency. They have not found applications in openly available OCR systems as of yet. Neural super-resolution methods for document images as proposed by [74]–[76] result in a circa fifty per cent error rate reduction on the resulting high resolution output, in comparison to the original low resolution images, when using a character segmentation OCR for recognition. Wholesale reconstruction of illegible writing in fragmentary historical handwritten documents using a modified PixelCNN is described in [77].

Document binarization constitutes the other major area of interest in preprocessing. It transforms color- or gray-level images into black-and-white images by classifying each pixel of a given page image as belonging to either a foreground or a background class. It serves multiple purposes, including noise suppression and data domain restriction in subsequent steps. Binarization used to be considered an integral part of any OCR system, and a long tradition of research on the topic testifies of this practice (see, for instance, the 1979 Otsu thresholding[10], the earliest method still in common use). A plethora of hand-crafted methods have been devised over the years. They can be divided into two categories, global ([10]) or local ([78]–[81]) thresholding algorithms, depending on whether they compute a single threshold for the whole image or individual thresholds for smaller patches. A special case concerns algorithms that perform normalization in the grayscale domain and then apply a global threshold. An example of this is [82], which is part of the OCRopus engine. Global thresholding often suffers from significant output degradation if the input image shows inner variations across it, such as uneven illumination or partial fading of ink. As a result, local methods are usually preferred, although they are comparatively slower. In Tesseract, it is global thresholding using Otsu's method which is implemented. On the other hand, the OCR-D workflow engine [57] and the proprietary Transkribus HTR engine [83] prefer more accurate local algorithms. A number of binarization methods based on ANNs exist, and they are usually variants of semantic segmentation methods. However, they suffer from the high cost of producing accurate ground truth. [84] is exemplary of these systems. It uses a Fully Convolutional Network [85] trained in a supervised fashion and fuses lower-level features in the decoding path to improve reconstruction of fine-grained structures in the output image. U-Nets [86] are another popular choice of architecture for similar methods.

Binarization is a complex operation that often fails to achieve acceptable results on historical documents with faded or differently colored ink and degraded or structured writing supports or inscriptions. Its inclusion in OCR systems intended for such material is therefore controversial, especially considering that many recent LA methods do not require binary input data anymore. Nevertheless, to the best of my knowledge, all current OCR systems do include a binarization algorithm. Kraken is one of the few engines where its use is only optional and disabled by default.

Layout Analysis

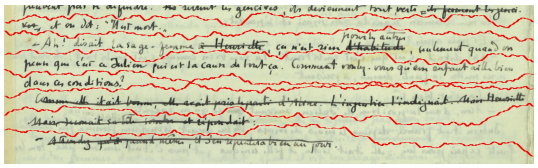
Layout analysis aims to identify and recognize the physical and logical organization of document pages. The exact requirements of an LA system can vary widely, depending on the particular use case and the capabilities of the subsequent text transcription method. The LA module is of critical importance in any OCR system as the transcription method will not be able to recognize any misidentified characters or lines. In fact, LA methods often constitute the limiting factor which determines whether an OCR engine is capable of processing a given document,

and it frequently accounts for a substantial proportion of the system's overall error rate.

Traditionally, LA systems operate as character segmenters, as the first text transcription methods were only capable of recognizing one character at a time. They can operate as single-stage methods directly extracting characters from the page image. However, they are usually designed as a multi-stage process that identifies smaller entities iteratively (e.g. text blocks first, then lines, words, and finally characters). Unfortunately, even the process of segmenting machine-printed block text can be prone to error. The most common approaches to dissection, projection analysis and connected components are indeed susceptible to over- and under-segment due to broken or merged characters [87, sec.2]. This is all the more so as the segmentation of cursive machine-printed or hand-written text requires considerable prior knowledge regarding the structure of the writing system to be segmented. It is common that results are unsatisfactory. According to a survey on Arabic-script character recognition carried on several methods [88], segmentation accuracy lies between seventy-six and ninety-nine percent, with the best-performing algorithms optimized for particular typefaces. Advanced character segmenters (e.g. in Tesseract [89]) perform oversegmentation and use the confidences returned by the text transcription algorithm to determine correct segmentation by testing multiple hypotheses. Due to its significant drawbacks regarding character segmentation, few developed OCR engines still use this paradigm actively. There are, however, two major exceptions: the proprietary Abbyy FineReader and Sakhr engines, both of which are most likely classifying single characters.

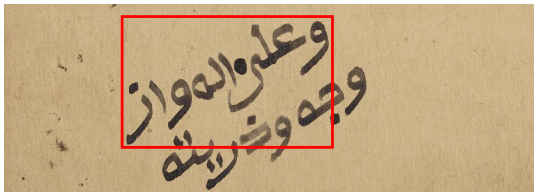
The prevalent paradigm since the development of segmentation-free text transcription that is able to recognize a sequence of characters at once has been text line segmentation. While the *what* is evident, the representation of the text lines is of more interest. The three principal representations with spread in both research and practical applications are paths, bounding boxes, and baselines (see figure 1.1). Axis-aligned bounding boxes, i.e. a bounding box whose edges are parallel to the image boundary, which contains all the textual content of a particular line and have an, often implicit, orientation are the most basic practical way to encode a text line. One of the benefits of this representation is that rectangular boxes are a natural fit to the rectangular line strips ingested by text line transcription methods, and therefore require only minimal processing in the transcription method. There is a wide array of methods which fit machine-printed and handwritten text to bounding boxes such as those described in [91], [92] or the LA module in the OCRopus system [93]. In general, these text line segmenters struggle with slanted or curved texts, either natural or as a result of the digitization process, and require the aforementioned preprocessing to eliminate skew and warping but are inherently unable to accurately segment multi-oriented and highly curved text. A minor extension implemented in the Tesseract LA module which increases the versatility of the text line model is to allow rotation of

the text bounding box in combination with an explicit orientation detection algorithm [89].



(a) Paths dividing a text block into lines. (from [90, figure 5a])

(b) Baselines (detail of Walters W.578 fol. 5a)



(c) Bounding box around a line (detail of Penn Libraries CAJS Rar Ms 132 fol. 53v)

Figure 1.1: Principal representations of lines

Paths are a flexible alternative which allow the segmentation of slanted and curved lines with relative ease. They encode blocks of lines that are roughly in the same orientation through a series of separating linear or non-linear paths. This representation has one drawback: segmenting a page document into blocks can already prove to be a challenge. Early methods utilized projection profiles [94], at times piecewise to improve segmentation of curvature [95]. More recent approaches tend to treat path finding as an optimization problem using the Viterbi algorithm [96] or seam carving [90], [97]. Notwithstanding the fact that this approach is superior to using bounding boxes to ensure tight bounds around curved and/or slanted lines, the above-mentioned block segmentation challenge prevented its dissemination, and it has never been implemented in any mainstream OCR engine. It should be noted, however, that the Aletheia [98] document analysis system implements human-triggered path-based line segmentation.

Baselines are currently the state of the art for text line segmentation in highly challenging historical documents. The baseline is an imaginary line which letters rest on, although letters frequently have parts (called descenders) that dip below it. It is a fairly common concept which exist in a large number of alphabetic writing systems, although certain scripts (e.g. Hebrew, Tibetan, or Bengali) are written with a hanging base or topline instead, and most logographic scripts such as Chinese do not have any in the strict typographic sense. In spite of these variations, approximations can often be systematized well enough to allow the processing of most scripts using a baseline LA method.

Representing text lines by their baseline has one major advantage: the model can accommodate arbitrary curvature by defining the baseline as a simple polyline, i.e. a sequence of straight line segments. Apart from being able to express any text written in linear writing systems compactly, baseline representation has

the added benefit of allowing rectification through projection of the individual polyline segments onto a straight line. However, there is also one drawback. A baseline alone is not sufficient to extract a complete text line as it is not known which points around it belong to the line in question or to adjacent content. As a result, baselines are usually accompanied with additional bounding polygons. Baseline representation appears in the literature as early as 1989 [99]. However, it went largely unnoticed until recently, except for a few segmentation systems [89], [93] that used baselines internally for clustering line blobs. Two factors helped their popularization among researchers. Firstly, [100] showed that bounding polygons had only but a small impact on transcription error rates, although this certainly does not hold true for all historical documents. Secondly, [101] published cBAD, a large dataset annotated with (poly-)baseline, and an evaluation metric for a ICDAR contest. This was followed by a larger and more complex dataset in 2019 [102]. Since then, the literature has produced a number of methods [48], [103]–[108]. They usually combine a deep neural semantic segmentation method (common choices are U-Nets and FCNs) with a postprocessing heuristic of varying complexity. While progress in accuracy levels using the ICDAR contest dataset have been impressive, practical hurdles remain. Since bounding polygons are not part of standard evaluation metric, they do not define an algorithm for computing one. Likewise, the metric does not take line orientation into account and the cBAD dataset is almost exclusively made of upright lines. As a result, there is neither academic incentive nor any datasets available to develop baseline LA methods that are able to determine line orientation effectively. Currently, two OCR systems include trainable baseline LA systems: Transkribus and Kraken (see chapter 8), to which must be added support in the eScriptorium VRE and the OCR-D workflow engine, via the Kraken module.

Other text line representations do exist but they are rarely used in practice. Pixel labelings and bounding polygons are relatively easy to produce using deep convolutional ANNs for semantic [109], [110] or instance segmentation [111]. However, they require tedious training data acquisition. In addition, in the case of semantic segmentation-based methods, separation between close lines can be problematic. Furthermore, without an additional way to determine line orientation or estimate line curvature, text recognition on rotated and highly curved purely polygon-bounded lines affects recognition accuracy negatively. This is due to the fact that lines cannot be effectively normalized to the rectangular line image strips processed by text transcription methods.

Some ancillary tasks are also associated with layout analysis. Region segmentation (also known as page segmentation or zoning) divides a document image into semantic regions such as main text, decoration, illustration, ... Its output has several applications: to improve the OCR engine's final textual output through semantic annotation; to restrict the range of valid output in other tasks, e.g. by only providing regions of interest to the text line detector or by limiting the text transcription method to numerals in a postal code field; or to increase the information available to algorithms for reading order determination.

Region segmentation is well-established as a task. There are hundreds of hand-crafted region segmentation algorithms. They are often optimized for particular use cases and employ various filtering [112], cutting [113], [114], and clustering [115], [116] techniques. As for text line segmentation, the capabilities of deep convolutional ANNs have made them a popular choice, and since 2018 a number of publications have focused on this task [48], [103], [105], [117]–[120]. A number of systems such as [105] (U-Net), [103] (FCN), and [48] (GAN with custom CNN) have implemented an attractive proposition: the possibility to perform both baseline detection and region segmentation with the same network architecture, or even the same model [48], [103]. Certain OCR engines have started to incorporate neural region segmentation, for instance anyOCR [121], Transkribus, or Kraken. Other systems (e.g. Tesseract or OCR4all) retain heuristics for this purpose.

Reading order determination (ROD) is an integral part of layout analysis, albeit it is often overlooked. While systems as described above detect layout structure, they fail to give any information as per how layout elements are related logically. The reading order (i.e. the order in which human readers will read textual and non-textual components) constitutes one of the most important logical structures and is critical to understanding a document. It may sound simple at first, as most modern documents are read following a simple top-to-bottom order. However, there is a large body of documents for which this is no trivial matter. Newspapers, for example, contain articles spanning multiple columns and pages; scholarly editions have critical apparatus which do not obey the normal reading order; and historical manuscripts often display extensive marginal notes, interlinear additions, and parallel texts. The literature on the topic is sparse, in particular when compared to many other DIA tasks. As a matter of fact, the current state of the art has not evolved since the early 2000s in any substantial way. [122]–[124] generate a region tree with X-Y cuts which is subsequently ordered using simple heuristic rules (top-to-bottom, left-to-right). [125] incorporates language modelling to determine likely text block sequences in newspapers. [93] uses topological sorting to sort individual text lines on a page. [126] utilize decision trees that incorporate both spatial and linguistic features, so as to fit to a complex document understanding model. A partially trainable system using linear programming to reconcile user-defined constraints is proposed in [127]. Despite the fact that some of the above methods are indeed intended for highly complex documents, all of them make assumptions regarding the spatial ordering of lines which are ill-suited to many historical documents. However, graph neural networks have shown promise in logical document structure analysis [128] and might be adapted for ROD in the future.

ROD implementations in OCR engines mirror the current state of the art in research. OCRopus and Kraken use [93]. Tesseract employs a simple rule-based algorithm described in [129]. The OCR-D workflow engine does not treat ROD as a separate task, instead relying on the implicit order provided through the respective LA modules.

1.5.3 Transcription

As described in section 1.5.2, conventional OCR systems were constructed around character classifiers. Character classifiers require accurate character segmentations, which are frequently difficult to obtain for historical, degraded, and cursive text. To solve this problem, different segmentation-free methods have been proposed. They recognize one sequence of characters (e.g. a word or a line) at a time. It should be noted that if the term segmentation-free is commonplace in the literature, it only applies to the input data and the training process. This is due to the fact that it is often possible to extract segmentation as well as estimates of character locations from their output, as an implicit segmentation is performed internally.

It was via the adaptation of HMM-based methods used in speech recognition that the earliest approaches, able to both perform sequential classification and to produce an hypothesis for a possible segmentation, were designed [17], [130]. Such approaches can be trained easily using Expectation Maximization. In addition, they allow straightforward incorporation of domain knowledge such as language models, with the goal of increasing their capacity to model long term dependencies. However, most HMM OCR methods operate on feature representations that are calculated on a sliding window over the text line, as is indeed necessary to reduce the number of model parameters to avoid severe overfitting. HMM-based methods may differ widely from one to another, in particular when it comes to the choice of these types of features, and the literature abounds with descriptions of different general-purpose and heuristic features. An outline of different feature extractors, modeling granularity, and language models employed in HMM-based OCR methods is contained in a 2009 survey [131].

The inferiority of HMM-based methods to RNNs trained with CTC loss became blatantly apparent in 2009, when a CTC trained multidimensional LSTM (MDLSTM) [132] won three contests on French, Arabic, and Persian handwriting transcription without any language-specific method adaptation. Because the computational requirements of MDLSTMs for both inference and training made them unsuitable to large scale applications, and also because some doubts existed regarding their assumed superiority over 1D-LSTMs [133], simpler bidirectional LSTMs (BiLSTM) [134] became the basis for numerous derivations of the general BiLSTM+CTC schema. The first hybrid convolutional and recurrent neural network (CRNN) for natural scene text recognition was proposed in [34]. Meanwhile, [135] augmented a basic CRNN with a spatial transformer network block that learned to dewarp input text line images. [136] incorporated lexicon verification to control a cascade of text transcription RNNs. [137] combined novel gated convolutional layers with a multilayer BiLSTM and CTC loss. More complex training procedures are sometimes constructed around these methods, for instance incorporating auxiliary language model losses to adapt a transcription model to an unseen language [138]. Another example is domain adaptation from synthetic machine-printed to handwritten text with virtual adversarial training

[139].

In recent years, various attempts were made to find alternatives to RNNs or CTC for text transcription. Apart from the everlasting search for greater accuracy and generalization, RNNs (and especially LSTMs) are slow and cannot be parallelized easily. CTC has the aforementioned limitations, to which one must add complexity, not to mention that fast implementations suffer from restrictions such as maximum target sequence lengths. Attentional models such as [140]–[142] replace CTC with conventional losses. In general, they consist of an encoder and a decoder, where the encoder produces a feature map. Through an attention mechanism (which can be of different types, including content- and location-aware attention), the decoder can weigh at each decoding step the relative importance of the different parts of the feature map. The number of times the decoder runs steps is arbitrary, and characters are output directly from the weighted feature map. It follows that these methods hardly allow the character sequence to align with the input image. This contrasts with HMMs and CTC-trained ANNs sharply. [143] describes a recurrence-free, CTC-trained, gated convolutional ANN which, if compared to CRNNs, achieves similar (albeit slightly lower) character error rates. Focusing on the recognition of in-air handwritten Chinese text, [144] proposes a system that uses increasingly strided 1D convolutional layers to achieve large receptive fields in the feature extractor, quite similarly to non-causal temporal convolutional networks (TCNs). While the complete ANN architecture includes some final LSTM layers, an ablation study shows that the TCN on its own achieves error rates that are similar to a simple two-layer LSTM.

As state-of-the-art hybrid CRNNs trained with CTC loss largely surpass older HMM-based systems and have existed for a few years, many OCR engines include (C)RNN+CTC transcription modules. The OCRopus system uses a one-layer bidirectional LSTM trained with a particular variant of CTC loss, implemented completely in Python. The latest (fourth) version of Tesseract includes a dynamically configurable neural networking library which is highly optimized for inference on CPU. In addition, Tesseract's default network configurations include unconventional summarizing LSTM layers that compute local features as the last time step output of an LSTM layer ingesting a one-pixel-wide vertical slice of the line, one pixel at a time. The OCR4All system uses the Calamari text classifier, which implements an ensemble method of confidence-weighted voting of cross-fold-trained CRNNs [145]. Kraken implements a configurable neural networking backend with a model specification language B, defaulting to a simple six layer CRNN. anyOCR [121] uses BiLSTMs trained in a conventional supervised fashion with CTC, but includes a procedure to harvest imprecise training data through manually labelled character clusters. It aims to drastically reduce training data requirements.

Specialty tasks

Certain specialty tasks exist without being part of any general-purpose OCR systems. Either the material they are designed for is too uncommon and dissimilar to writing systems for natural languages, or the state of the art is not sufficiently accurate to warrant implementation in practical OCR engines.

Table analysis is a task in DIA whose goal is to detect and recognize both the contents of a table and its logical structure. Despite a long history of research in table processing methods which extends far into the early nineteen-nineties [146], this task is still considered unsolved even for modern printed tables [147]. There is limited research pertaining to the processing of historical tabular material. Even hand-crafted methods that are optimized for a particular table style have high error rates [148]. Since table retrodigitization remains an area of commercial interest in the data entry industry, certain proprietary OCR engines (e.g. Abbyy FineReader) include table detection and structure recognition.

Optical Music Recognition (OMR) aims to transcribe sheet music into a machine-readable representation. The process is analogous to OCR in many ways; yet the term OMR remains rather ill-defined. It encompasses output representations (e.g. MIDI) that one would not traditionally associate with OCR. In addition, musical notation behaves quite differently from most scripts used for natural language, as it is a featural writing system whose information is contained both in the ordered sequences of symbols and their spatial relationships. Some researchers such as [149] reject the categorization of OMR as a sub-type of OCR or *OCR for music* and the field has developed a number of OMR-specific tasks such as staff processing and musical information reconstruction that have no equivalent in the domain of OCR (see [150] for a recent survey of OMR tasks and methods). For practical purposes, the OCR and OMR research fields are distinct. OCR engines are not capable of processing musical notation and vice-versa.

Likewise, digital map processing is usually treated separately even when the goal is limited to textual content extraction. This is due to the fact that non-text elements make up for a larger proportion of the total writing surface than most other documents. This makes text finding harder for most LA methods. However, it should be noted that in comparison to other DIA applications text transcription in maps can better exploit domain knowledge, as the nature of naturally-expressed geographical data allows alignment with other maps and incorporation of toponym dictionaries [151]–[153]. Complete map understanding requires not only text line detection and region segmentation layout analysis but also the extraction of cartographic features such as map symbols and contour lines. Most text LA systems are ill-suited for this purpose, and it is therefore generally performed using dedicated segmentation methods such as [154], [155].

1.5.4 Virtual Research Environments for DIA, OCR, and Digital Humanities

Virtual Research Environments encompass a broad range of research support systems that share common characteristics. They offer a web-based working environment and are tailored to serve the needs of a *community of practice* [5]. They provide a comprehensive catalogue of tools to support the targeted community in accomplishing its goals. They are open and flexible with respect to service offering and lifetime, and promote controlled sharing of both intermediate and final research results by guaranteeing ownership, provenance and attribution[157]. They are not specific to OCR or the humanities and can be designed around all kinds of research activity.

A number of web-based platforms offer manual or computer-assisted segmentation, transcription, and recognition for certain categories of material in the humanities field. However, whether they can be considered actual VREs is open to question. Most platforms are developed within particular institutional frameworks as project-specific tools, without active buy-in from practitioners falling outside well-defined collaborations. This situation explains the lack of platforms that satisfy all criteria for VREs. In this review, we will relax the criteria while still limiting our review to tools whose functionalities at least partially overlap with eScriptorium's.

Most platforms implement specific methods to suit the needs of a particular scientific project, and most research projects are of relatively short duration and seek to solve specific research questions. As a result, in typical settings dedicated platforms are not concerned with implementing a complete catalogue of methods to perform an activity fully. They also rarely create a scholarly community around them, albeit they can be considered as being part of a fabric of loosely-coupled tools that could be made to interoperate through common data interchange formats. Crowdsourcing applications for transcription of historical documents have proved to be a particularly successful type of platforms. Tools like the Bentham Transcription Desk [158] discard any automatic processing to retain simplicity in implementation. Such platforms are rarely openly accessible. In other words, data on which users can perform activities is predetermined by the platform's operators; imports are often already processed extensively through external means. The TypeWright OCR correction and digital edition creation tool [159] is a good example of this. The only raw OCR data that it makes available was created by the eMOP project's internal pipeline. On the other hand, it should be noted that a fairly large number of platforms accepting the use of arbitrary data do exist. They are, however, of very limited capability. For instance, [160] is a platform whose purpose is to create segmentation ground truth data with some low-level com-

[5] Communities of practice as defined in [156] are individuals that are united in action and in the meaning that an activity has for them as individuals and as a collective, i.e. communities are composed of active practitioners who create a community through self-identification with and exchange on a particular, in our case scholarly, activity.

puter vision assistance. Another example is the HInDoLA [161] platform, which is solely intended to aid the layout analysis of palm leafs. LAREX [162] is a semi-automatic tool for layout analysis of early printed books. It can be interfaced to external OCR tools through data exports in PageXML format. PoCoTo [163] is a web service geared towards the postcorrection of OCR output with language modelling assistance.

When it comes to communities of OCR practitioners in the humanities, the Transkribus [164] platform comes closest in meeting the VRE definition. It was designed as a comprehensive platform for computer-aided layout analysis, transcription and information retrieval. It offers all the basic tools necessary to the average scholar interested in retrodigitization of historical handwritten and printed material. Unfortunately, the closed nature of the platform hampers interested scholars's capacity to share their work while being guaranteed ownership of their work. While data can be imported and exported in standard formats freely, artifacts trained in specific OCR tasks (i.e. layout analysis and transcription models) with user-created data are locked inside the platform. Added to the platform's recently adopted pricing model, it presents not only a significant deficit of ownership but also a barrier to the reproducibility of results.

While eScriptorium is not yet feature complete and access to the platform of the public is currently limited, albeit the source code of all components is publicly available and a number of instances are set up at different research groups and institution, it already fulfills the VRE definition. It supports the full gamut of OCR functions needed for the processing of historical material, permits users to selectively share the product of their work, and provides interfaces to import and export all data and workflow-produced artifacts.

References

[1] D. A. Smith and R. Cordell, "A research agenda for historical and multilingual optical character recognition," *NUlab, Northeastern University*, 2018.

[2] S. A. Mahmoud, I. Ahmad, W. G. Al-Khatib, *et al.*, "KHATT: An open Arabic offline handwritten text database," *Pattern Recognition*, vol. 47, no. 3, pp. 1096–1112, 2014. DOI: 10.1016/j.patcog.2013.08.009.

[3] M. Widner, "Toward Text-Mining the Middle Ages: Digital Scriptoria and Networks of Labor," in *The Routledge Research Companion to Digital Medieval Literature*, Routledge, 2017, pp. 131–144.

[4] H. Alpert-Abrams, "Machine Reading the Primeros Libros," *Digital Humanities Quarterly*, vol. 10, no. 4, 2016.

[5] D. Mimno, "Data carpentry is a skilled, hands-on craft which will form a major part of data science in the future," 2014. [Online]. Available: https://blogs.lse.ac.uk/impactofsocialsciences/2014/09/01/data-carpentry-skilled-craft-data-science/.

[6] P. Ströbel, S. Clematide, and M. Volk, "How Much Data Do You Need? About the Creation of a Ground Truth for Black Letter and the Effectiveness of Neural OCR," in *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, N. Calzolari, F. Béchet, P. Blache, *et al.*, Eds., European Language Resources Association, 2020, pp. 3551–3559.

[7] H. Herbert, "The history of ocr, optical character recognition," *Manchester Center, VT: Recognition Technologies Users Association*, 1982.

[8] S. A. Papert, "The summer vision project," 1966.

[9] M. Minsky and P. Seymour, *Perceptrons: an introduction to computational geometry*. The MIT Press, Cambridge MA, 1969, ISBN: 0262130432.

[10] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[11] R. A. Wilkinson, *The first census optical character recognition system conference*. US Department of Commerce, National Institute of Standards and Technology, 1992, vol. 4912.

[12] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*, Springer, 1982, pp. 267–285.

[13] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[15] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen," Ph.D. dissertation, Technische Universität München, 1991.

[16] J. Schmidhuber, "Deep learning in neural networks: An overview," *CoRR*, vol. abs/1404.7828, 2014. arXiv: 1404.7828.

[17] A. Kaltenmeier, T. Caesar, J. M. Gloger, and E. Mandler, "Sophisticated topology of hidden markov models for cursive script recognition," in *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93)*, IEEE, 1993, pp. 139–142.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Y. Bengio and Y. LeCun, Eds., 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," *Advances in neural information processing systems*, vol. 29, pp. 550–558, 2016.

[22] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 409–424.

[23] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," *Advances in neural information processing systems*, vol. 28, pp. 2377–2385, 2015.

[24] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[25] F. N. Iandola, M. W. Moskewicz, K. Ashraf, *et al.*, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 1MB model size," *CoRR*, vol. abs/1602.07360, 2016. arXiv: 1602.07360.

[26] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 4780–4789.

[27] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," Proceedings of Machine Learning Research, vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds., pp. 6105–6114, 2019.

[28] M. Huh, P. Agrawal, and A. A. Efros, "What makes imagenet good for transfer learning?" *CoRR*, vol. abs/1608.08614, 2016. arXiv: 1608.08614.

[29] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 4918–4927.

[30] B. Zoph, G. Ghiasi, T. Lin, *et al.*, "Rethinking pre-training and self-training," H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[31] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?" In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 2661–2671.

[32] F. Visin, K. Kastner, K. Cho, *et al.*, "ReNet: A recurrent neural network based alternative to convolutional networks," *CoRR*, vol. abs/1505.00393, 2015. arXiv: 1505.00393.

[33] M. F. Stollenga, W. Byeon, M. Liwicki, and J. Schmidhuber, "Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation," C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., pp. 2998–3006, 2015.

[34] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.

[35] K. Xu, J. Ba, R. Kiros, *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[36] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 369–376.

[37] J. Song, Z. Guo, L. Gao, *et al.*, "Hierarchical LSTM with adjusted temporal attention for video captioning," *CoRR*, vol. abs/1706.01231, 2017. arXiv: 1706.01231.

[38] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," Y. Bengio and Y. LeCun, Eds., 2015.

[39] T. Bluche, J. Louradour, and R. Messina, "Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, 2017, pp. 1050–1055. DOI: 10.1109/ICDAR.2017.174.

[40] T. Bluche, "Joint line segmentation and transcription for end-to-end handwritten paragraph recognition," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 838–846.

[41] A. Schiller. "Transcribr - a transformer-based handwriting recognition architecture." (2020), [Online]. Available: https://towardsdatascience.com/transcribr-9861c8de2f79 (visited on 12/16/2020).

[42] B. Wu, C. Xu, X. Dai, *et al.*, "Visual Transformers: Token-based Image Representation and Processing for Computer Vision," *CoRR*, vol. abs/2006.03677, 2020. arXiv: 2006.03677.

[43] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative Adversarial Nets," Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., pp. 2672–2680, 2014.

[44] S. E. Reed, Z. Akata, X. Yan, *et al.*, "Generative Adversarial Text to Image Synthesis," JMLR Workshop and Conference Proceedings, vol. 48, M. Balcan and K. Q. Weinberger, Eds., pp. 1060–1069, 2016.

[45] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, IEEE Computer Society, 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244.

[46] K. C. Nguyen, C. T. Nguyen, S. Hotta, and M. Nakagawa, "A character attention generative adversarial network for degraded historical document restoration," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 420–425.

[47] S. Suh, J. Kim, P. Lukowicz, and Y. O. Lee, "Two-stage generative adversarial networks for document image binarization with color noise and background removal," *CoRR*, vol. abs/2010.10103, 2020. arXiv: 2010.10103.

[48] L. Quirós, "Multi-task handwritten document layout analysis," *arXiv arXiv:1806.08852*, 2018.

[49] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[50] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pre-training help deep learning?" In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 201–208.

[51] J. Calvo-Zaragoza and A.-J. Gallego, "A selectional auto-encoder approach for document image binarization," *Pattern Recognition*, vol. 86, pp. 37–47, 2019.

[52] K. Chen, M. Seuret, M. Liwicki, J. Hennebert, and R. Ingold, "Page segmentation of historical document images with convolutional autoencoders," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2015, pp. 1011–1015.

[53] H. Wei, M. Seuret, K. Chen, *et al.*, "Selecting autoencoder features for layout analysis of historical documents," in *Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing*, 2015, pp. 55–62.

[54] K. M. Sayre, "Machine recognition of handwritten words: A project report," *Pattern recognition*, vol. 5, no. 3, pp. 213–228, 1973.

[55] T. M. Breuel, "The ocropus open source ocr system," in *Document recognition and retrieval XV*, International Society for Optics and Photonics, vol. 6815, 2008, 68150F.

[56] C. Reul, D. Christ, A. Hartelt, *et al.*, "Ocr4all—an open-source tool providing a (semi-) automatic ocr workflow for historical printings," *Applied Sciences*, vol. 9, no. 22, p. 4853, 2019.

[57] C. Neudecker, K. Baierer, M. Federbusch, *et al.*, "Ocr-d: An end-to-end open source ocr framework for historical printed documents," in *Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, 2019, pp. 53–58.

[58] *Scantailor*, version 0.9.9, Dec. 16, 2020. [Online]. Available: https://scantailor.org.

[59] L. Fan, F. Zhang, H. Fan, and C. Zhang, "Brief review of image denoising techniques," *Visual Computing for Industry, Biomedicine, and Art*, vol. 2, no. 1, p. 7, 2019.

[60] S. Cha and T. Moon, "Fully convolutional pixel adaptive image denoiser," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4160–4169.

[61] S. Laine, T. Karras, J. Lehtinen, and T. Aila, "High-quality self-supervised deep image denoising," in *Advances in Neural Information Processing Systems*, 2019, pp. 6970–6980.

[62] S. Soltanayev and S. Y. Chun, "Training deep learning based denoisers without ground truth data," in *Advances in neural information processing systems*, 2018, pp. 3257–3267.

[63] J. Chen, J. Chen, H. Chao, and M. Yang, "Image blind denoising with generative adversarial network based noise modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3155–3164.

[64] D. S. Bloomberg, G. E. Kopec, and L. Dasari, "Measuring document image skew and orientation," in *Document Recognition II*, International Society for Optics and Photonics, vol. 2422, 1995, pp. 302–316.

[65] A. Amin and S. Fischer, "A document skew detection method using the hough transform," *Pattern Analysis & Applications*, vol. 3, no. 3, pp. 243–253, 2000.

[66] A. Papandreou and B. Gatos, "A novel skew detection technique based on vertical projections," in *2011 International Conference on Document Analysis and Recognition*, IEEE, 2011, pp. 384–388.

[67] Z. Zhang and C. L. Tan, "Correcting document image warping based on regression of curved text lines," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, IEEE, 2003, pp. 589–593.

[68] A. Ulges, C. H. Lampert, and T. M. Breuel, "Document image dewarping using robust estimation of curled text lines," in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, IEEE, 2005, pp. 1001–1005.

[69] A. Masalovitch and L. Mestetskiy, "Usage of continuous skeletal image representation for document images de-warping," in *Proceedings of International Workshop on Camera-Based Document Analysis and Recognition, Curitiba*, 2007, pp. 45–53.

[70] N. Stamatopoulos, B. Gatos, I. Pratikakis, and S. J. Perantonis, "A two-step dewarping of camera document images," in *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, IEEE, 2008, pp. 209–216.

[71] S. Das, K. Ma, Z. Shu, D. Samaras, and R. Shilkrot, "Dewarpnet: Single-image document unwarping with stacked 3d and 2d regression networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 131–140.

[72] K. Ma, Z. Shu, X. Bai, J. Wang, and D. Samaras, "Docunet: Document image unwarping via a stacked u-net," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4700–4709.

[73] T. Breuel, *Ocropus 3 document skew detection*, May 29, 2018. [Online]. Available: http://github.com/NVlabs/ocropus3-ocrorot.

[74] C. Dong, X. Zhu, Y. Deng, C. C. Loy, and Y. Qiao, "Boosting optical character recognition: A super-resolution approach," *CoRR*, vol. abs/1506.02211, 2015. arXiv: 1506.02211.

[75] A. Lat and C. Jawahar, "Enhancing ocr accuracy with super resolution," in *2018 24th International Conference on Pattern Recognition (ICPR)*, IEEE, 2018, pp. 3162–3167.

[76] Z. Fu, Y. Kong, Y. Zheng, *et al.*, "Cascaded detail-preserving networks for super-resolution of document images," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 240–245.

[77] L. Uzan, N. Dershowitz, and L. Wolf, "Qumran letter restoration by rotation and reflection modified pixelcnn," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, vol. 1, 2017, pp. 23–29.

[78] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[79] W. Niblack, *An introduction to digital image processing (englewood clivs, nj*, 1986.

[80] I.-K. Kim, D.-W. Jung, and R.-H. Park, "Document image binarization based on topographic analysis using a water flow model," *Pattern Recognition*, vol. 35, no. 1, pp. 265–277, 2002.

[81] B. Gatos, I. Pratikakis, and S. J. Perantonis, "An adaptive binarization technique for low quality historical documents," in *International Workshop on Document Analysis Systems*, Springer, 2004, pp. 102–113.

[82] F. Shafait, D. Keysers, and T. M. Breuel, "Efficient implementation of local adaptive thresholding techniques using integral images," in *Document recognition and retrieval XV*, International Society for Optics and Photonics, vol. 6815, 2008, p. 681 510.

[83] K. Ntirogiannis, B. Gatos, and I. Pratikakis, "A combined approach for the binarization of handwritten document images," *Pattern recognition letters*, vol. 35, pp. 3–15, 2014.

[84] C. Tensmeyer and T. R. Martinez, "Document image binarization with fully convolutional neural networks," in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, IEEE, 2017, pp. 99–104. DOI: 10.1109/ICDAR.2017.25.

[85] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[86] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2015, pp. 234–241.

[87] R. G. Casey and E. Lecolinet, "A Survey of Methods and Strategies in Character Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 690–706, 1996. DOI: 10.1109/34.506792.

[88] Y. M. Alginahi, "A survey on Arabic character segmentation," *International Journal of Document Analysis and Recognition*, vol. 16, no. 2, pp. 105–126, 2013. DOI: 10.1007/s10032-012-0188-6.

[89] R. Smith, "An Overview of the Tesseract OCR Engine," in *9th International Conference on Document Analysis and Recognition (ICDAR 2007), 23-26 September, Curitiba, Paraná, Brazil*, IEEE Computer Society, 2007, pp. 629–633. DOI: 10.1109/ICDAR.2007.4376991.

[90] N. Arvanitopoulos and S. Süsstrunk, "Seam carving for text line extraction on color and grayscale historical manuscripts," in *2014 14th International Conference on Frontiers in Handwriting Recognition*, IEEE, 2014, pp. 726–731.

[91] U. Marti and H. Bunke, "On the Influence of Vocabulary Size and Language Models in Unconstrained Handwritten Text Recognition," in *6th International Conference on Document Analysis and Recognition (ICDAR 2001), 10-13 September 2001, Seattle, WA, USA*, IEEE Computer Society, 2001, pp. 260–265. DOI: 10.1109/ICDAR.2001.953795.

[92] V. Papavassiliou, T. Stafylakis, V. Katsouros, and G. Carayannis, "Handwritten document image segmentation into text lines and words," *Pattern Recognition*, vol. 43, no. 1, pp. 369–377, 2010. DOI: 10.1016/j.patcog.2009.05.007.

[93] T. M. Breuel, "High performance document layout analysis," 2003.

[94] A. Antonacopoulos and D. Karatzas, "Document Image Analysis for World War II Personal Records," in *1st International Workshop on Document Image Analysis for Libraries (DIAL 2004), 23-24 January 2004, Palo Alto, CA, USA*, IEEE Computer Society, 2004, pp. 336–341. DOI: 10.1109/DIAL.2004.1263263.

[95] A. Zahour, B. Taconet, P. Mercy, and S. Ramdane, "Arabic Hand-Written Text-Line Extraction," in *6th International Conference on Document Analysis and Recognition (ICDAR 2001), 10-13 September 2001, Seattle, WA, USA*, IEEE Computer Society, 2001, pp. 281–285. DOI: 10.1109/ICDAR.2001.953799.

[96] Y. Tseng and H. Lee, "Recognition-based handwritten Chinese character segmentation using a probabilistic Viterbi algorithm," *Pattern Recognit. Lett.*, vol. 20, no. 8, pp. 791–806, 1999. DOI: 10.1016/S0167-8655(99)00043-4.

[97] X. Zhang and C. L. Tan, "Text Line Segmentation for Handwritten Documents Using Constrained Seam Carving," in *14th International Conference on Frontiers in Handwriting Recognition, ICFHR 2014, Crete, Greece, September 1-4, 2014*, IEEE Computer Society, 2014, pp. 98–103. DOI: 10.1109/ICFHR.2014.24.

[98] C. Clausner, S. Pletschacher, and A. Antonacopoulos, "Aletheia - an advanced document layout and text ground-truthing system for production environments," in *2011 International Conference on Document Analysis and Recognition*, 2011, pp. 48–52. DOI: 10.1109/ICDAR.2011.19.

[99] S. N. Srihari and V. Govindaraju, "Analysis of textual images using the hough transform," *Machine vision and Applications*, vol. 2, no. 3, pp. 141–153, 1989.

[100] V. Romero, J. A. Sanchez, V. Bosch, K. Depuydt, and J. de Does, "Influence of text line segmentation in handwritten text recognition," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, IEEE, 2015, pp. 536–540.

[101] M. Diem, F. Kleber, S. Fiel, T. Grüning, and B. Gatos, "cBAD: ICDAR2017 competition on baseline detection," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, IEEE, vol. 1, 2017, pp. 1355–1360.

[102] D. Markus, K. Florian, and G. Basilis, *ICDAR 2019 Competition on Baseline Detection (cBAD)*, Zenodo, Feb. 2019. DOI: 10.5281/zenodo.3568023.

[103] Y. Xu, F. Yin, Z. Zhang, and C. Liu, "Multi-task Layout Analysis for Historical Handwritten Documents Using Fully Convolutional Networks," J. Lang, Ed., pp. 1057–1063, 2018. DOI: 10.24963/ijcai.2018/147.

[104] O. Mechi, M. Mehri, R. Ingold, and N. E. B. Amara, "Text line segmentation in historical document images using an adaptive u-net architecture," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 369–374.

[105] S. A. Oliveira, B. Seguin, and F. Kaplan, "dhSegment: A Generic Deep-Learning Approach for Document Segmentation," in *16th International Conference on Frontiers in Handwriting Recognition, ICFHR 2018, Niagara Falls, NY, USA, August 5-8, 2018*, IEEE Computer Society, 2018, pp. 7–12. DOI: 10.1109/ICFHR-2018.2018.00011.

[106] K. Romain and B. Abdel, "Semi-supervised learning through adversary networks for baseline detection," in *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, IEEE, vol. 5, 2019, pp. 128–133.

[107] T. Grüning, G. Leifert, T. Strauß, J. Michael, and R. Labahn, "A two-stage method for text line detection in historical documents," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 22, no. 3, pp. 285–302, 2019.

[108] A. Melnikov and I. Zagaynov, "Fast and Lightweight Text Line Detection on Historical Documents," in *International Workshop on Document Analysis Systems*, Springer, 2020, pp. 441–450.

[109] J. Pastor-Pellicer, M. Z. Afzal, M. Liwicki, and M. J. C. Bleda, "Complete system for text line extraction using convolutional neural networks and watershed transform," in *12th IAPR Workshop on Document Analysis Systems, DAS 2016, Santorini, Greece, April 11-14, 2016*, IEEE Computer Society, 2016, pp. 30–35. DOI: 10.1109/DAS.2016.58.

[110] M. Alberti, L. Vögtlin, V. Pondenkandath, *et al.*, "Labeling, Cutting, Grouping: An Efficient Text Line Segmentation Method for Medieval Manuscripts," in *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*, IEEE, 2019, pp. 1200–1206. DOI: 10.1109/ICDAR.2019.00194.

[111] A. Prusty, S. Aitha, A. Trivedi, and R. K. Sarvadevabhatla, "Indiscapes: Instance segmentation networks for layout parsing of historical Indic manuscripts," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 999–1006.

[112] T. Pavlidis and J. Zhou, "Page segmentation and classification," *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 6, pp. 484–496, 1992, ISSN: 1049-9652. DOI: https://doi.org/10.1016/1049-9652(92)90068-9.

[113] J. Ha, R. M. Haralick, and I. T. Phillips, "Recursive xy cut using bounding boxes of connected components," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, IEEE, vol. 2, 1995, pp. 952–955.

[114] B. Kruatrachue, N. Moongfangklang, and K. Siriboon, "Fast document segmentation using contour and xy cut technique.," in *WEC (5)*, 2005, pp. 27–29.

[115] D. Drivas and A. Amin, "Page segmentation and classification utilising a bottom-up approach," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, IEEE, vol. 2, 1995, pp. 610–614.

[116] K. Kise, A. Sato, and M. Iwata, "Segmentation of page images using the area voronoi diagram," *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 370–382, 1998.

[117] C. Wick and F. Puppe, "Fully Convolutional Neural Networks for Page Segmentation of Historical Document Images," in *13th IAPR International Workshop on Document Analysis Systems, DAS 2018, Vienna, Austria, April 24-27, 2018*, IEEE Computer Society, 2018, pp. 287–292. DOI: 10.1109/DAS.2018.39.

[118] D. He, S. Cohen, B. Price, D. Kifer, and C. L. Giles, "Multi-scale multi-task fcn for semantic page segmentation and table detection," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, vol. 1, 2017, pp. 254–261.

[119] K. Chen, M. Seuret, J. Hennebert, and R. Ingold, "Convolutional neural networks for page segmentation of historical document images," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, vol. 1, 2017, pp. 965–970.

[120] T. Monnier and M. Aubry, "Docextractor: An off-the-shelf historical document element extraction," in *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, 2020, pp. 91–96.

[121] S. S. Bukhari, A. Kadi, M. A. Jouneh, F. M. Mir, and A. Dengel, "anyOCR: An Open-Source OCR System for Historical Archives," in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, IEEE, 2017, pp. 305–310. DOI: 10.1109/ICDAR.2017.58.

[122] G. Nagy and S. C. Seth, "Hierarchical representation of optically scanned documents," 1984.

[123] Y. Ishitani, "Document transformation system from papers to xml data based on pivot xml document method," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, IEEE, 2003, pp. 250–255.

[124] J.-L. Meunier, "Optimized xy-cut for determining a page reading order," in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, IEEE, 2005, pp. 347–351.

[125] L. Gao, Z. Tang, X. Lin, and Y. Wang, "A graph-based method of newspaper article reconstruction," 2012.

[126] M. Aiello, C. Monz, L. Todoran, and M. Worring, "Document Understanding for a Broad Class of Documents," *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 1–16, 2002.

[127] D. Malerba, M. Ceci, and M. Berardi, "Machine learning for reading order detection in document image understanding," in *Machine Learning in Document Analysis and Recognition*, Springer, 2008, pp. 45–69.

[128] H. Déjean, J.-L. Meunier, *et al.*, "Versatile layout understanding via conjugate graph," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 287–294.

[129] R. W. Smith, "Hybrid page layout analysis via tab-stop detection," in *2009 10th International Conference on Document Analysis and Recognition*, 2009, pp. 241–245. DOI: 10.1109/ICDAR.2009.257.

[130] G. Rigoll, A. Kosmala, J. Rottland, and C. Neukirchen, "A comparison between continuous and discrete density hidden markov models for cursive handwriting recognition," in *13th International Conference on Pattern Recognition, ICPR 1996, Vienna, Austria, 25-19 August, 1996*, IEEE Computer Society, 1996, pp. 205–209. DOI: 10.1109/ICPR.1996.546818.

[131] T. Plötz and G. A. Fink, "Markov models for offline handwriting recognition: A survey," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, no. 4, p. 269, 2009.

[132] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., pp. 545–552, 2008.

[133] J. Puigcerver, "Are multidimensional recurrent layers really necessary for handwritten text recognition?" In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, vol. 1, 2017, pp. 67–72.

[134] A. Graves, M. Liwicki, S. Fernández, *et al.*, "A Novel Connectionist System for Unconstrained Handwriting Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, 2009. DOI: 10.1109/TPAMI.2008.137.

[135] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar, "Improving CNN-RNN hybrid networks for handwriting recognition," in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, 2018, pp. 80–85.

[136] B. Stuner, C. Chatelain, and T. Paquet, "Cohort of LSTM and lexicon verification for handwriting recognition with gigantic lexicon," *CoRR*, vol. abs/-1612.07528, 2016. arXiv: 1612.07528.

[137] T. Bluche and R. Messina, "Gated convolutional recurrent neural networks for multilingual handwriting recognition," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, vol. 1, 2017, pp. 646–651.

[138] C. Tensmeyer, C. Wigington, B. Davis, *et al.*, "Language model supervision for handwriting recognition model adaptation," in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, 2018, pp. 133–138.

[139] S. Keret, L. Wolf, N. Dershowitz, *et al.*, "Transductive learning for reading handwritten tibetan manuscripts," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 214–221.

[140] J. Sueiras, V. Ruiz, A. Sanchez, and J. F. Velez, "Offline continuous handwriting recognition using sequence to sequence neural networks," *Neurocomputing*, vol. 289, pp. 119–128, 2018.

[141] J. Michael, R. Labahn, T. Grüning, and J. Zöllner, "Evaluating sequence-to-sequence models for handwritten text recognition," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 1286–1293.

[142] L. Kang, J. I. Toledo, P. Riba, *et al.*, "Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition," in *German Conference on Pattern Recognition*, Springer, 2018, pp. 459–472.

[143] D. Coquenet, C. Chatelain, and T. Paquet, "Recurrence-free unconstrained handwritten text recognition using gated fully convolutional network," in *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, 2020, pp. 19–24.

[144] J. Gan, W. Wang, and K. Lu, "In-air handwritten Chinese text recognition with temporal convolutional recurrent network," *Pattern Recognition*, vol. 97, p. 107 025, 2020.

[145] C. Wick, C. Reul, and F. Puppe, "Calamari - A high-performance tensorflow-based deep learning package for optical character recognition," *Digit. Humanit. Q.*, vol. 14, no. 2, 2020.

[146] R. Zanibbi, D. Blostein, and J. R. Cordy, "A survey of table recognition," *International Journal of Document Analysis and Recognition*, vol. 7, no. 1, pp. 1–16, 2004. DOI: 10.1007/s10032-004-0120-9.

[147] L. Gao, Y. Huang, H. Déjean, *et al.*, "ICDAR 2019 Competition on Table Detection and Recognition (cTDaR)," in *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*, IEEE, 2019, pp. 1510–1515. DOI: 10.1109/ICDAR.2019.00243.

[148] C. Lehenmeier, M. Burghardt, and B. Mischka, "Layout Detection and Table Recognition - Recent Challenges in Digitizing Historical Documents and Handwritten Tabular Data," in *Digital Libraries for Open Knowledge - 24th International Conference on Theory and Practice of Digital Libraries, TPDL 2020, Lyon, France, August 25-27, 2020, Proceedings*, M. M. Hall, T. Mercun, T. Risse, and F. Duchateau, Eds., ser. Lecture Notes in Computer Science, vol. 12246, Springer, 2020, pp. 229–242. DOI: 10.1007/978-3-030-54956-5_17.

[149] J. Calvo-Zaragoza, J. H. Jr, and A. Pacha, "Understanding optical music recognition," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–35, 2020.

[150] E. Shatri and G. Fazekas, "Optical Music Recognition: State of the Art and Major Challenges," *CoRR*, vol. abs/2006.07885, 2020. arXiv: 2006.07885.

[151] J. J. Weinman, "Toponym Recognition in Historical Maps by Gazetteer Alignment," in *12th International Conference on Document Analysis and Recognition, ICDAR 2013, Washington, DC, USA, August 25-28, 2013*, IEEE Computer Society, 2013, pp. 1044–1048. DOI: 10.1109/ICDAR.2013.209.

[152] J. Weinman, "Geographic and style models for historical map alignment and toponym recognition," in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, IEEE, 2017, pp. 957–964. DOI: 10.1109/ICDAR.2017.160.

[153] K. Sun, Y. Hu, J. Song, and Y. Zhu, "Aligning geographic entities from historical maps for building knowledge graphs," *International Journal of Geographical Information Science*, pp. 1–30, 2020.

[154] J. H. Uhl, S. Leyk, Y.-Y. Chiang, W. Duan, and C. A. Knoblock, "Spatialising uncertainty in image segmentation using weakly supervised convolutional neural networks: A case study from historical map processing," *IET Image Processing*, vol. 12, no. 11, pp. 2084–2091, 2018.

[155] T. Liu, Q. Miao, P. Xu, and S. Zhang, "Superpixel-Based Shallow Convolutional Neural Network (SSCNN) for Scanned Topographic Map Segmentation," *Remote Sensing*, vol. 12, no. 20, p. 3421, 2020.

[156] E. Wenger, *Communities of practice: Learning, meaning, and identity*. Cambridge university press, 1999.

[157] L. Candela, D. Castelli, and P. Pagano, "Virtual research environments: An overview and a research agenda," *Data Science Journal*, GRDI–013, 2013.

[158] M. Moyle, J. Tonra, and V. Wallace, "Manuscript transcription by crowdsourcing: Transcribe bentham," *Liber Quarterly*, vol. 20, no. 3-4, 2011.

[159] "Typewright, a tool for correcting the text-version of a document made up of page images." (2013), [Online]. Available: https://18thconnect.org/typewright (visited on 01/20/2021).

[160] C. Clausner. "Webaletheia - a web based version of the aletheia ground truthing system." (2014), [Online]. Available: https://www.primaresearch.org/tools/WebAletheia (visited on 01/20/2021).

[161] A. Trivedi and R. K. Sarvadevabhatla, "HInDoLA: A Unified Cloud-Based Platform for Annotation, Visualization and Machine Learning-Based Layout Analysis of Historical Manuscripts," in *2nd International Workshop on Open Services and Tools for Document Analysis, OST@ICDAR 2019, Sydney, Australia, September 22-25, 2019*, IEEE, 2019, pp. 31–35. DOI: 10.1109/ICDARW.2019.10035.

[162] C. Reul, U. Springmann, and F. Puppe, "LAREX: A semi-automatic open-source tool for layout analysis and region extraction on early printed books," in *Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage*, 2017, pp. 137–142.

[163] T. Vobl, A. Gotscharek, U. Reffle, C. Ringlstetter, and K. U. Schulz, "PoCoTo - an open source system for efficient interactive postcorrection of OCRed historical texts," in *Digital Access to Textual Cultural Heritage 2014, DATeCH 2014, Madrid, Spain, May 19-20, 2014*, A. Antonacopoulos and K. U. Schulz, Eds., ACM, 2014, pp. 57–61. DOI: 10.1145/2595188.2595197.

[164] P. Kahle, S. Colutto, G. Hackl, and G. Mühlberger, "Transkribus - A Service Platform for Transcription, Recognition and Retrieval of Historical Documents," in *1st International Workshop on Open Services and Tools for Document Analysis, 14th IAPR International Conference on Document Analysis and Recognition, OST@ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, IEEE, 2017, pp. 19–24. DOI: 10.1109/ICDAR.2017.307.

Part I

The Arabic Writing System

Chapter 2

The Arabic Script

The Arabic writing system is one of the geographically and chronologically most widely used writing system in human history. It is the primary script for the Arabic language, Farsi, Urdu, and multiple others on the Indian subcontinent. Historically, it has been used to produce text in several additional languages, ranging from Spanish to Chinese.

While its exact origins are still a topic of debate among researchers, it is generally accepted that it evolved from either the Nabatean or the Syriac script in the Middle East over the course of several centuries. Its maturation is said to have occurred during the seventh century CE. Its diffusion is closely linked to the spread of Islam, and several alphabetic variants as well as calligraphic regional styles developed over time. Far from being a purely liturgical script however, with a wealth of administrative records, philosophical and scientific treatises, poetry, etc. existing.

From the perspective of DIA research, it would therefore be a misnomer to speak of a single Arabic script. Persian epics in *nasta'līq* style on highly decorated marbled paper have little in common with *naskh* private correspondence or the angular rectilinear writing of early kufic Qur'anic codices. Each style, in combination with regional preferences as well as the particular context of its utilization, presents specific challenges.

2.1 The Principles of the Arabic Writing System

The Arabic script is an *abjad*, i.e. a consonantal writing system. Similar to scripts for other semitic languages, only consonants and long vowels are written. The reader has to infer short vowels from the context. Short vowels and other marks for features such as doubling (gemination) and nunation (adding a final n), can optionally be added (*tashkīl*). However, this is only systematic for transcriptions of the *qur'ān*, and in elementary texts for language learners. Like Syriac and Hebrew, the script is written from right to left with the exception of numbers, which are written from left to right.

The original abjad used for writing Classical Arabic with its twenty-eight distinctive phonemes contains only eighteen graphemes *rasm*, causing the same grapheme to represent up to five different phonemes. bāʾ, tāʾ, thāʾ, nūn, and yāʾ share the same shape and are, depending on their position in the word, written the same way. These glyphs are distinguished by dots placed above or below them: one below for bāʾ, two above for tāʾ, three above for thāʾ, one above for nūn, and two below for yāʾ. Correspondingly, dots are used to differentiate ghayn and ʾayn, ṣād and ḍād, and jīm, ḥāʾ, and khāʾ. In contrast to vocalization dotting is mandatory and is present in all but the earliest Arabic texts. See table 2.1 for an overview of the twenty-eight letters of the standard Arabic abjad.

The Arabic script distinguishes itself from other widely used scripts such as Latin, Cyrillic or Greek in that it does not have a printed form, i.e. a form in which letters can be written separately. The cursive form is the only form that

Table 2.1: The 28 letters of the Arabic abjad

isolated	initial	medial	final	name	transliteration
ا	ا	ـا	ـا	ʾalif	ā
ب	بـ	ـبـ	ـب	bāʾ	b
ت	تـ	ـتـ	ـت	tāʾ	t
ث	ثـ	ـثـ	ـث	thāʾ	th
ج	جـ	ـجـ	ـج	jīm	j
ح	حـ	ـحـ	ـح	ḥāʾ	ḥ
خ	خـ	ـخـ	ـخ	khāʾ	kh
د	د	ـد	ـد	dāl	d
ذ	ذ	ـذ	ـذ	dhāl	dh
ر	ر	ـر	ـر	rāʾ	r
ز	ز	ـز	ـز	zayn	z
س	سـ	ـسـ	ـس	sīn	s
ش	شـ	ـشـ	ـش	shīn	sh
ص	صـ	ـصـ	ـص	ṣād	ṣ
ض	ضـ	ـضـ	ـض	ḍād	ḍ
ط	طـ	ـطـ	ـط	ṭāʾ	ṭ
ظ	ظـ	ـظـ	ـظ	ẓāʾ	ẓ
ع	عـ	ـعـ	ـع	ʿayn	ʿ
غ	غـ	ـغـ	ـغ	ghayn	gh
ف	فـ	ـفـ	ـف	fāʾ	f
ق	قـ	ـقـ	ـق	qāf	q
ك	كـ	ـكـ	ـك	kāf	k
ل	لـ	ـلـ	ـل	lām	l
م	مـ	ـمـ	ـم	mīm	m
ن	نـ	ـنـ	ـن	nūn	n
ه	هـ	ـهـ	ـه	hāʾ	h
و	و	ـو	ـو	wāwʾ	w/ū
ي	يـ	ـيـ	ـي	yāʾ	y/ī

exists, albeit it include a wide variety of styles. Similar to cursive forms in other scripts, in Arabic individual letters will change shape depending on their position within the word. Initial, medial, final and independent shapes exist, to which one must add more complex placement rules along the baseline, which depend on adjacent letters. Contrary to Latin, not every letter in written Arabic can be connected to the letter that precedes or follows it: roughly one fifth actually do not connect to the previous or following letter. As a result, whitespace does not necessarily mark the beginning of a new word, thus providing calligraphers with considerable freedom to vary inter-word, inter-syllable, and inter-stroke spacing as desired. One could go as far as abandoning word separation completely, in a kind of *scripta continua* [1, pg. 15]. However, even conventional calligraphic practice displays spacing variation to some extent, which can prove misleading for optical character recognition systems.

Unlike many other scripts, Arabic does not know capital and lower case letter forms and texts produced before the twentieth century CE do not contain Western-style punctuation or layout such as commas, periods, question marks, paragraphs, ... New sentences and questions are introduced using specific words and phrases. Titles and headings are indicated as such by placing strokes above them. After the ninth century CE, a variety of verse marks and signs appear in Quranic manuscripts to help with recitation [2].

The script has been adapted for a fairly large number of other languages, most prominently Persian and Turkish that do have some additional phonemes. Letter forms are usually adapted by adding dots to graphemes representing similar phonemes in Arabic, e.g. the Persian *pe* is derived from *bā'* by writing two additional dots below. Sometimes graphemes were also modified more directly such as the letter *gāf* which is written by adding another bar above the letter *kāf*. Similar transformations exist for other languages with some such as *xiao'erjing*, the writing of sinitic languages in Arabic script, changing it into a full alphabet by making the marking of short vowels mandatory or creating additional letters for short vowels. The Unicode standard alone lists more than a hundred additional graphemes for local variants of the Arabic script.

2.1.1 Text Justification

As described Arabic does not share many features with other alphabetic scripts but none of these are a fundamental obstacle to a modern OCR pipeline. While the cursive nature was seen as a major hindrance to older methods based on character classifiers which require accurate segmentation to the character level, newer segmentation-less methods are largely unaffected by this. Nevertheless the variability in spacing, even on machine-printed text, is still the primary source of errors (see section 4.4). Unfortunately, the bundle of techniques employed to justify text, i.e. measures ensuring that text ends flush with the left end of the writing space, require methods aware of their function in all parts of the OCR pipeline.

Such techniques followed the proscription of hyphenation (i.e. splitting words

to facilitate line-wrapping) from the tenth century CE onwards. They were created as a mean to avoid justification by whitespace alone, which often produces visually unpleasant results (a phenomenon known as rivers in Western typography). As a matter of fact, Western-style hyphenation is only visible in certain early Koran manuscripts (albeit without explicit splitting markers such as hyphens). It was most probably proscribed in Arabic script due to its significant negative impact on legibility, especially if the second part of the word is continued on a subsequent page.

One technique became prominent among later calligraphers working on hanging styles such as the Persian *nasta'līq*, in particular for poetry written in hemistiches. It consisted in stacking or heaping the last syllable or penstroke above the left end of the line. It was particularly attractive for Persian calligraphers to proceed this way, as many Persian words have similar end letters. As such, the technique has but a negligible impact on legibility. Similar stacking could be performed inside a line by changing the location and size of diacritical marks [1, pg. 14].

The existence of dislocated fragments into the margin represents another challenge for OCR systems. Instead of heaping the last syllable on top of the previous stroke, the fragment is placed in the margins, much like a marginal note. These two practices add to the complexity of the layout analysis component in the OCR pipeline. Firstly, the layout analysis system has to be able to detect this type of fragments accurately. The methods presented in part II have the potential to do so, at least with models specifically trained to that purpose. Secondly, these types of fragments has to be distinguished from bona fide marginalia and interlinear notes or translations in order to be inserted in the textual output smoothly. This has to be done by a reading order determination algorithm which knows of their existence. If the algorithm is naïve and assumes that the text follow a top-to-bottom order, it will insert this type of fragments before the text of the line it is associated with, i.e. in reverse order. This is true even if other marginalia are correctly filtered through previous classification using a segmenter. Unfortunately, reading order determination is a notoriously under-researched task in DIA. ome tentative approaches, such as [3] that could serve as a basis for versatile reading order determination methods, exist, but all practically available OCR system operate on simple heuristics.

Two other practices should have their existence mentioned here, although their impact on OCR performance is likely to be limited. The first practice consists in stretching the letter body or the connections between individual letters. It is interchangeably known as either *taṭwīl* or *kashida*, although certain typography-related publications use *kashida* only to refer to the stretching of the letter body. This practice is visible in most Arabic styles, albeit its exact placement differs among scripts. Not only can it be used for justification but also to highlight the beginning of a verse, a heading, etc.

The second practice consists of the upward curving of the baseline, trading horizontal with vertical space. However, conventional layout analysis methods

tend to model lines as either rectangular bounding boxes or undirected[1] polygons. The presence of an upward curving in the baseline (and more generally the presence of any slanted line) is likely to degrade the results. This is due to the protruding of other lines into the bounding box, or to height normalization, which results in insufficient text size as demonstrated in figure 1.1c. Segmenters founded on the baseline paradigm, like the ones presented in chapters 5 and 6, do not suffer from this problem as they allow the projection of arbitrarily shaped lines onto a straight line.

2.2 Supports and Production

The question of supports is often neglected in the OCR research community. While processing inscriptions in stone or clay tablets present unique and obvious challenges, the difficulties that may arise from subtle differences of supple writing surfaces are usually disregarded. Papyrus and parchment were used as writing material until they were replaced by cheaper paper after the 8th century. Richly adorned specialty papers, which were often used in high status documents, are of particular interest to Computer Vision experts.

We also include in the present dissertation a short discussion on inks, dyes and techniques of illumination, not only because they cause a number of difficulties to OCR systems, but also because of the potential for large scale and automatized analysis across collections that DIA methods may offer in the future.

2.2.1 Supports

Paper, parchment and papyrus are the three main supports used in the Islamic world, although paper largely overshadows parchment or papyrus in terms of usage, whether it be geographically or chronologically.

Papyrus has been a support for writing since at least 3000 BCE. It is a light-colored, smooth, and flexible material which is manufactured using *Cyperus papyrus*, a three-to-six meter high Egyptian water reed. The production process goes as follow: first, the reed stalk is cut into halves, and the pulp is extracted into thin strips. Strips are then arranged into rectangular sheets made of two perpendicular layers, and left to dry under the sun. Once dried, sheets are smoothed with a mallet, prior to their polishing by means of a shell or an ivory. Sheet sizes could vary considerably, although it wasn't uncommon for their width to be around 20-30 cm, and their length around 30-40 cm.

Following polishing, individual sheets are glued end-to-end, while overlapping joints are smoothed again. Sheets are then rolled with the horizontal fibers on the inside. Like in the pre-Islamic period, a roll was composed of twenty sheets,

[1] Undirected in the sense that there is no defined text orientation or direction contained within the polygon. Only a bounding polygon is given.

although papyrus would also be sold in smaller pieces (most commonly as one-sixth of a roll). To protect rolls from wear, a thicker gauge strip of papyrus called a *protokollon* was attached to their top before use.

Scrolls were in use up until the early eighth century CE. However, the majority of the documents that survived are either in codex form or are single sheet documents[4, pg.30]. While some literary papyri are known, most Arabic papyri were documentary in nature, and contained letters, edicts, and contracts. It is likely that papyrus was a rather expensive writing material. They were rapidly phased out after the introduction of cheaper paper, and production ceased in the eleventh century CE [5, pg.193-194].

Papyri raise two problems for DIA methods. They are brittle and fragile, which frequently results in significant degradation (figure 2.1). Besides, with age, they darken to a point where their ink often becomes almost illegible. The ageing process also increases contrast between fibers, which further complicates any attempt at distinguishing the text from its background. Hand-crafted binarization methods have been developed for the highly degraded Dead Sea Scrolls [6], [7] but processing remains difficult. Papyri length in scroll form poses another, perhaps more minor, problem. While many modern neural network-based methods can be adapted to patchwise operation with minimal loss of accuracy, a number of them (in particular layout analysis) perform better with global context.

Figure 2.1: An Arabic papyrus showing both visible fibers and typical deterioration of the writing surface (BnF Arabe 4634).

The second writing surface in widespread use before the introduction of paper in the Islamic world was parchment. Parchment is a carefully processed un- or minimally tanned animal skin; fundamentally it can be procured from a variety of animals and its production is not limited to a particular geographic area as papyrus. Goat, calf, donkey, and gazelle skins, although these have never been corroborated by testing, are mentioned as sources but the most common was sheep skin [1, pg. 44]. Its use in the East extends to time memorial although no dated Arabic manuscripts from before the ninth century CE have survived [4, pg.33].

Parchment is manufactured as follows. First, a basic solution (made from lime

or dates) is used to remove the hair from the hide. Second, residual flesh and fat is scraped from the flesh side of the hide using a blade. It is then placed in a wooden frame to stretch it and dry it. Finally, an abrasive stone is used to smoothen the surface and equalize the texture of both the flesh and hair sides. Chalk is applied to control ink bleeding. Sometimes, parchments could be dyed with blue indigo or yellow saffron. The most famous example of this practice is the Blue *Qur'ān* , dating back from the tenth century CE [5, pg. 195-196].

The ease with which previously used parchments could be reemployed by washing or scraping off existing writing is an important consideration for both humanists and computer vision researchers. Known as palimpsests, this practice is attested both in the literature and by several surviving exemplars. Considerable time could elapse between the moment the first layer was written on and the moment it was covered with a second one. Examples of Arabic text over a lower layer written in another script such as Greek or Syriac exist [4, pg. 43-46]. While it was the complete removal of the initial writing that was intended, oftentimes the lower layer remains visible to a certain extent.

The task of separating the writing to decipher both texts requires immense skill, which can be aided by costly multispectral imaging [8]. However, it has not gathered much research interest in the DIA community. [9] (one of the few publications on the topic) points out that existing methods need substantial improvement. Parchment could also be recycled in other ways, e.g. in bookbinding or as protective covers to loose quires, not quite unlike papyrus *protokollon* [4, pg. 46].

Paper was invented in China in 105 CE by the Han courtier Cai Lun. Its introduction to the Islamic world is attributed traditionally to a Muslim military victory on the river Talas (modern-day Kazakhstan) in July 751 CE. Allegedly, a number of Chinese papermakers were made war prisoners following the battle and dispatched to Samarqand to set up paper mills. Details of this story can certainly be disputed, however there are etymological hints that knowledge of papermaking was received through Central Asia [1, pg. 45]. Paper proved to be significantly cheaper than its alternatives, and rapidly replaced papyrus and parchment as the writing surface of choice. By 794 CE, paper mills were established in Baghdad. Its use by the Abbasid Caliphate administration was mandated in 808 CE. By the ninth century, paper was produced in Egypt. By the tenth century it had reached the Maghreb, and by the twelfth century it was produced in Damascus and in Spain [4, pg. 51].

Apart from economical reasons paper had other benefits: it absorbes ink so writing could not easily be erased [1, pg. 45], it is less brittle than papyrus with a more uniform coloration, and can be easily tinted and decorated.

The availability of inexpensive writing material produced a flurry of literary activity in the ninth century in a broad range of subjects, from theology to the natural sciences. Books were soon copied on paper. Quranic manuscripts, however, continued to be written on parchment until the late tenth century CE, and exemplars could be found in the Maghreb and West Africa up until the fourteenth

century CE.

Descriptions of the papermaking process in the Islamic world are sparse and lack clarity. The general process seems to have included suspension of cellulose fibers for draining through a screen, followed by drying. This resulted in a fiber mat is called paper. Sources of fibers could vary considerably: they could either be liberated from virgin plant material through a combination of heat, beating, and chemical means such as fermentation or acids, or originate from waste material such as rags, old ropes, etc. Primary material was then suspended in water to soak, prior to collection and draining in a mold. An account from modern-day Tunisia describes an example of papermaking from raw flax on a floating screen. On the other hand, analysis of extant specimens shows that waste materials, such as rags from cotton and linen or ropes, were primarily used. Before use, paper has to be sized with a mixture of starches and egg white and burnished to prevent the ink from bleeding excessively [10, pg. 44-45].

(a) Example of marbled paper (Walter W.654, fol. 1b)

(b) A page with an outer border made of blue-tinted paper and an inner rose-tinted paper (Walters W.746).

(c) Illumination in gold on blue-tinted border with text written on orange-tinted paper (Walters W.651, fol. 2a).

Figure 2.2: Decorated papers

A particularly impressive practice that often presents a challenge to modern DIA methods are specially prepared papers used for many fine manuscripts. Premodern papers retained the color of the source of the fibers, a fact that was used in European manufacture to produce colored papers, most often various hues of brown, but could also be tinted or dyed (figure 2.2b). The preference for tinted papers was probably initially fueled by colored imports from China but Iranian artists started developing techniques to make colored, gold-decorated, and marbled papers by the thirteenth century CE. Popular colors were red or orange, and Persian treatises of the time list a plethora of recipes to obtain these colors. Gold-sprinkled paper (figure 2.2c) was utilized throughout the Eastern Islamic lands

by the late fifteenth century CE with even more elaborate gold painting incorporating arabesques coming into fashion from the sixteenth century onward at the Safavid, Mughal, and Uzbek courts. Such gold-decorated papers were mainly used in scrapbook albums of calligraphy and paintings to unify the disparate contents into a single book.

Two other techniques rose in popularity after the fifteenth century CE: marbled papers (figure 2.2a) and paper cuts. While the marbling produced by slipping a sheet of paper over a bath of carefully swirled colorant cause the same issues to DIA methods as other decorated margins, paper cuts are likely more complicated. Cut-out calligraphy, either collage, i.e. placing cut-out letters on a contrasting background, or decoupage, i.e. mount a sheet with cut out letters above a differently colored one, allows larger variation in coloration than purely ink-based writing. Artists were often skilled in more than one of these techniques and they are often employed together. Works such as a collection of forty hadith for the Ottoman prince Mehmed in 1540 CE, contain text in both *taw'qī* and *nasta'līq* script, pasted in gold, white, or light blue on deep-rose or olive grounds. The margins on some pages are kept plain, others gold-sprinkled, others again marbled [1, pg. 52-56]. The sheer variety of text production and decoration techniques a single manuscript can display is an important challenge to DIA systems.

2.2.2 Writing Instruments and Inks

The Arabic script is traditionally written with a reed pen whose front had been trimmed with a special pen knife to create a nib. Depending on the script's desired width, the nib can be slit one or several times at its end, depending on the script's desired width. Multiple nib cuts exist. Straight and oblique cuts change the thickness of strokes at certain angles. Nonetheless, the association of cut angles and scripts seems to have been largely a matter of scribal preference[5, pg. 42].

Brushes were used primarily during the decoration process, for example to paint in the margins or for gilding. In the fourteenth century CE, Chinese Muslims developed the rounded, flowing *ṣīnī* script to adapt Arabic writing to Chinese calligraphy's instruments and conventions. In certain cases, it went as far as changing the directionality of writing to top-to-bottom [11, pg. 29-0].

More than pens and brushes, the type of ink and pigment used for writing is an important element when performing document image analysis. Three kinds of black or brown inks were commonly used: carbon-based inks, so-called mixed inks, and iron gall inks. Compound inks were also common in the medieval period [1, pg. 62-63]. All of them were known since Antiquity in the area that would later be known as the Islamic world. Carbon inks are attested as far back as the second millenium BCE in Egypt. The first recipes for mixed inks date back to the third century BCE and iron gall inks were in use by the fifth century CE [12].

Carbon inks are more commonly known as India inks. They are composed of fine soot or finely ground charcoal mixed with some kind of binder, such as oil,

gums, or shellac. For portability purposes, they often took the form of a solid stick, which was grounded and mixed with water before use. In the Muslim world, the main source of carbon for inks was from the combustion of vegetable matter such as rice, olives, chick-peas or various oils. The most common binders were gum arabic and honey [5, pg. 133]. This type of ink preparation adheres to the writing surface only superficially. It can easily be washed off with water; it smudges easily if kept in humid conditions and can peel off over time. So-called mixed inks represent an improvement. They can be prepared by adding copper, lead or iron salts to carbon ink. The addition of extra components, which acted as drying agents, was intended to increase ink adherence to paper. On the other hand, iron gall inks do not contain any carbon. When applied to paper, the ink undergoes an oxidation process, thus forming an insoluble ferric tannate pigment. It is prepared by mixing tannic acid (extracted from gallnuts), vitriol (ferric sulfate), and a binder such as gum arabic. Unlike other types of ink, it penetrates the writing surface and is indelible [12]. Iron gall inks have a major drawback, however. They are acidic in nature and tend to fade over time. As a result, manuscripts written with iron gall ink are often damaged due to acid burns, to the point of being illegible to human readers and computer vision methods alike [5, pg. 145]

Colored inks were used by scribes and calligraphers as accent colors for rubrics, vocalization, and other decoration. Red, green, and yellow were most common. Pigments used for ink manufacturing have not been studied systematically, but red inks usually contained cochineal, vermillion, and red lead pigments; blue inks contained lapis lazuli and azurite; and verdigris can be found in green inks. There is also substantial regional differences, as pigmentation and coloration traditions differ widely from east to west [1, pg. 63]. This kind of coloration does not pose much of a challenge to modern DIA methods, although pipelines employing binarization might encounter substantial degradation as the most commonplace binarization algorithms perform quite poorly on mixed-color texts. Metallic inks are more troublesome. This is not due to their inherent illegibility; but rather to the low contrast that scanned documents showcase (see figure 2.4c). For metallic inks, fine metals (e.g. gold, silver, copper) were used in at least two different ways. The first way consisted of liquid inks, which were made of metal flakes suspended in a binder [13, pg. 225-227]. Secondly, powdered flakes could be dispersed onto glue, to be subsequently burnished and ringed with other colors. The latter technique produces writing that is prone to degradation, as disintegration of the glue has caused flakes to fall off [1, pg. 63].

2.3 Styles

A large number of both formal and informal calligraphic styles have been devised over the centuries. While informal styles naturally evade uniform classification, formal styles can be divided into two groups: styles that are recognized throughout the Muslim world (such as *naskh*), and styles that are limited to a certain

geographical region or only used to write specific languages, such as the North African and Iberian *maghribī* or the Persian *nasta'līq*.

Arabic styles can be defined by elements such as [5, pg. 242-243]:

Line of writing whether all words sit completely on the baseline, descend onto it as in *nasta'līq*, curve upwards toward the end, or are slanted.

Ascender and descenders vertical, slanted or curved

Nib width especially in relation to script size.

Shading i.e. contrast between thin and thick strokes.

Vocalization Some scripts require a different pen for vocalization.

Ligatures The presence of unauthorized connections between letters.

Contractions The presence of assimilated (omitted) letter forms.

Characteristic letterforms such as straight, wavy, or slanted *'alif*

The two earliest styles that emerged in the seventh century CE are *ḥijāzī* (figure 2.3c) and kufic (figure 2.3a. Both are somewhat confusingly named after early Islamic intellectual centers (Hijaz and the city Kufa in southern Iraq) and a variety of alternative terms such as Early Abbasid have been proposed. As canonicalization was fairly low the terms do not refer singular hands but to families of styles that were used mainly to transcribe copies of the *qur'ān*. One taxonomy divides them into six and four groups for the kufic and *ḥijāzī* family respectively. Either is notably more angular than later round styles. Hijazi being in use from 650 CE it was surplanted by kufic by the beginning of the eighth century CE. While elaborate, highly decorated manuscripts in the kufic style survive, the number of surviving Hijazi fragments is very low and their appearance is utilitarian [5, pg. 98, 124].

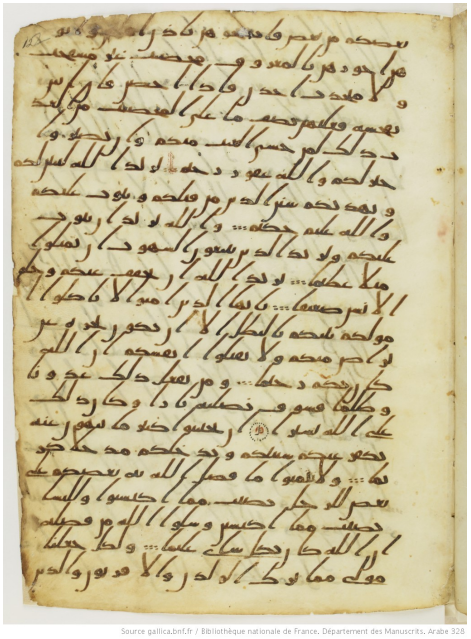
By the tenth century CE the use of kufic in high status manuscripts began to be replaced by a style known under a variety of names such as Eastern kufic, New Abbasid Style, and broken cursive [1, pg. 144] (figure 2.3b). This style was dominant into the thirteenth century CE for copies of the *qur'ān* when it was relegated to ornamental purposes such as titles and headings. Similarly to older styles, it does not represent a single hand but at least two different groups. The main characteristics of this style is the marked difference between thick and thin strokes [5, pg. 167-168]. From this period also dates the practice of using long *taṭwīl* elongation to mark the start of a new section of text [1, pg. 165].

2.3.1 The Six Pens

The greatest impact on the evolution of Arabic calligraphy between the tenth and thirteenth century is usually attributed to three great calligraphers: Ibn Muqlah

(a) Ninth century CE *qur'ān* in Kufic or Early Abbasid script (Walters W.552, fol. 8a)

(b) Twelfth century CE *qur'ān* in broken cursive or New Abbasid script (Walters W.555, fol. 10b)



(c) Seventh century CE *qur'ān* in *ḥijāzī* script (BnF Arabe 328, fol. 12r)

Figure 2.3: Early Arabic styles

(d. 940 CE), Ibn al-Bawwab (d. 1022 CE), and Yaqut (d. 1298 CE). The fundamental development of the period between the tenth and thirteenth century is the system of proportioned scripts and the canonicalization of the various rounded styles under this system. Ibn Muqla is attributed with developing the first proportioned script, *al-khatt al-mansūb*, that defined each letter's dimensions in the unit of rhomboid dots, impressions left by the reed pen on the writing surface, with *alif* spanning a circle circumscribing all other graphemes. While this system is most likely only a thirteenth century CE attempt at reconstructing the hand of the famous calligrapher, as none of his works survived, Ibn al-Bawwab is credited with canonicalizing the round chancery scripts in use at the time using this system [1, pg. 158-160, 213]. Finally, Yaqut popularized the six proportional styles known as the *Six Pens*, which later became the dominant scripts in the East [5, pg. 251].

These six styles are usually paired in three sets of one display (majuscule) and one text (minuscule) script:

- *thuluth* with *naskh*
- *muḥaqqaq* with *rayḥān*
- *tawqīʿ* and *riqāʿ*

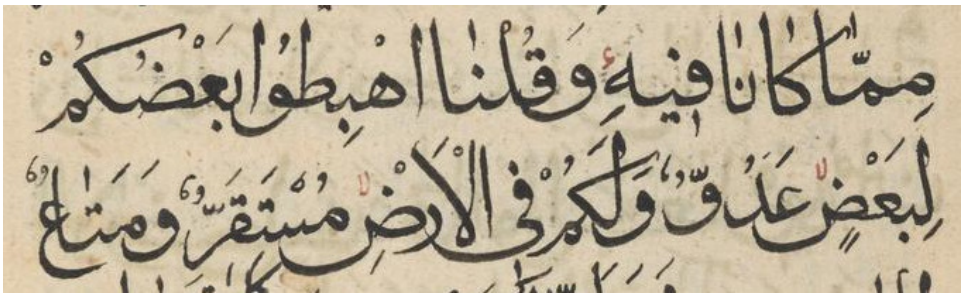
Thuluth (figure 2.4a) is an ancient chancery script although its exact features are unknown prior to its reform to the proportioned script system in the eleventh century CE. Its most notable properties are descenders which fall far below the baseline and curve upwards again for certain letters, hairlines, and many contracted letterforms. As a display script its letter are large but horizontally compact. In administrative use it was utilized for important documents while in codices it was used mainly for titles and chapter headings [5, pg. 275].

Naskh (figure 2.4b) is the most widely used book hand of the Islamic East. It is a serifless script without unauthorized connections between letters, long ascenders, and short descenders. Some Persian and Ottoman variations of the script exist. It is the basis for most modern Arabic typefaces intended for use in prose [5, pg. 162-163].

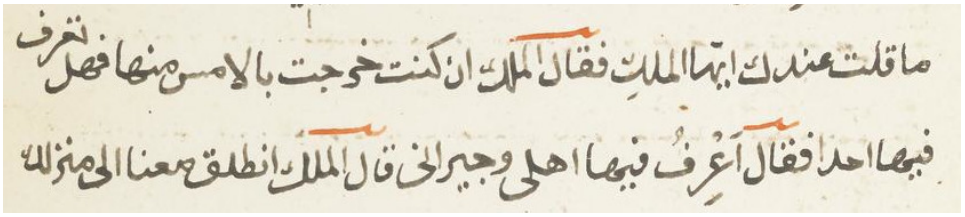
Muḥaqqaq (figure 2.4c) is one of the ancient scripts originally devised before the proportioned scripts system. It is rectilinear, i.e. only a small proportion of the penstrokes are curved or curvilinear. It is a seriffed script with vocalization performed with a different pen, often in a different color. While grouped as the display script to *rayḥān* it also became a bookhand for copies of the *qur'ān* by the thirteenth century CE [5, pg. 160-161].

Rayḥān is the smaller counterpart to *muḥaqqaq*. The letter forms were identical except for their size, the inclusion of serifs, and the execution of vocalization with the same pen as the letters [14, pg. 308].

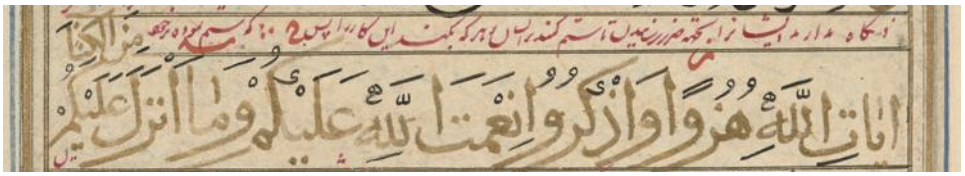
Tawqīʿ (figure 2.4d) is a smaller diplay variant of the *thuluth* chancery script being written with even more hairlines. Like *thuluth* it was rarely used as a bookhand [5, pg. 264-264].



(a) Two lines written in *thuluth* from an undated copy of the *qur'ān* (Columbia University, MS Or 234, fol. 4v).



(b) Two lines written in *naskh* from an undated book of Qur'anic stories (Free Library of Philadelphia, Lewis O 170, fol. 5r).



(c) Central line in *muḥaqqaq* in gold of a bilingual Arabic and Persian copy of the *qur'ān.* The red line above is part of the interlinear Persian translation in *nasta'līq* (Columbia University, Ms Or 222, 23r).

(d) Chapter heading written in *tawqīʿ* script from a fifteenth century CE Timurid history (Walters W.676, fol. Bb).

(e) Heading in *riqāʿ* script of a nineteenth century CE *qur'ān* copy (Walters W.567, fol. 2a).

Figure 2.4: Samples of five of the Six Pens

Riqāʿ (figure 2.4e) is the smaller version of the *tawqīʿ* script. It is optionally seriffed with slightly rightward inclined *alif*. In Iran the differences between it and *tawqīʿ* were minor with one publication using the terms synonymously [5, pg. 224].

2.3.2 Regional Styles

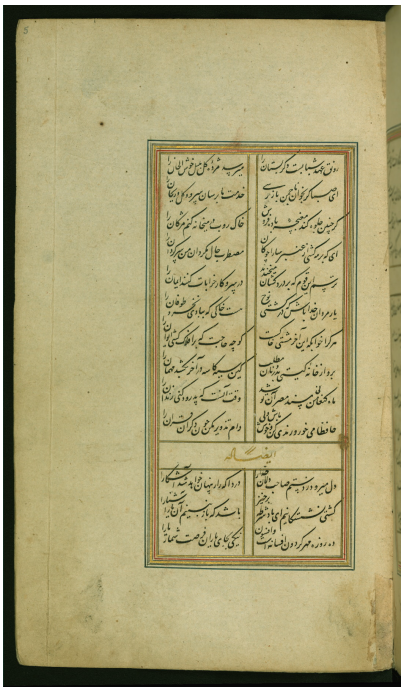
Around the same period a number of regional scripts were developed. In the Maghreb, Muslim Spain, and West Africa a number of scripts emerged that were later summarized under the name *maghribī* (figure 2.5d). These non-proportioned hands are distinctive but no detailled paeleographical analysis has been done to date and there exist a number of sub-types. One shared feature is the use of a rounded nib resulting in even thickness of strokes [5, pg. 147-148]. The earliest manuscripts in these styles date to the mid-tenth century CE with the earliest surviving *qurʾān* copy dated to 1008 CE but little change occured afterwards [1, pg. 566].

Around the same time as Yaqut was refining the Six Pens, two new styles of hanging scripts called *taʿlīq* (figure 2.5a) and *nastaʾlīq* (figure 2.5b) emerged in Iran. These were more suitable for writing languages such as Persian and Turkish that differ from Arabic in their proportion of straight and curved letters. As such they were never popular in the Arabic speaking world but *nastaʾlīq* ended up as the style of choice for a number of other states such as Ottoman Turkey and Mughal India. The highly stylized *taʿlīq* script with its curvilinear elements, extraneous loops, and connected letter was a typical style for official decrees, diplomatic correspondence, sometimes poetry, but almost never codices, from the tenth century CE. The stereotypical *taʿlīq* document contains widely spaced lines with dramatic upward curving at the end of the line. A variant broken *taʿlīq* replaced it from the fourteenth century CE in administrative use [1, pg. 270-273].

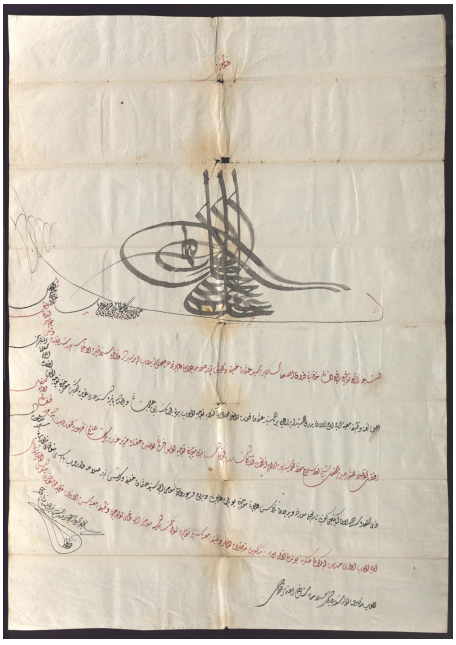
The second major Persian style, *nastaʾlīq*, became the epitome of Persian literature. At first reserved for poetry, it took over the place of *naskh* for prose by the fifteenth century as well. It is characterized by individual words descending onto a common baseline, greatly elongated horizontal lines, and the aforementioned heaping of the last stroke [5, pg. 166-167]. While the aforementioned styles, depending on the preservation of the document, the desired generalization, and accuracy, are not inherently troublesome for a modern OCR engine, the sophistication displayed by the most decorated *nastaʾlīq* epics and calligraphic specimens are highly challenging. Calligraphers of the time were interested in variety and visual excitement, arranging verses on the diagonal, changing direction between successive verses, alternating red, blue, orange, and green inks for headings, and adding lavish illumination and scrollwork. The writing was being subsumed by decoration [1, pg. 436].

Round scripts, primarily *naskh* and *thuluth*, and Persian hanging scripts were common in the Ottoman Empire with their relative proportion in text production

(a) Page from a Persian nineteenth century CE album of calligraphic samples written in *ta'līq* (Walters W.670, fol. 19b)



(b) 1539 CE page from a collection of poems written in *nasta'līq* (Walters W.631, fol. 5a)



(c) 1779 CE Ottoman firman *diwānī* script (American Philosophical Society Ms. Coll. 200, fol. 2)

(d) Nineteenth century CE *qur'ān* in large *maghribī* script (Walters W.568, fol. 31a)

Figure 2.5: Example of regional styles

varying over the centuries[2] In the fifteenth century CE imperial scribes began to develop the *taʿlīq* script into the highly stylized *diwānī* (figure 2.5c) chancery style with its unauthorized connections between letters and curved baselines.

2.4 Printing

Notwithstanding the fact that this dissertation is mostly concerned with handwritten text recognition, printing deserves some consideration. This is especially true given that the development and diffusion of printing is directly linked to the practice of calligraphy. Printing is the mechanical reproduction of text and images. Since Johannes Gutenberg developed his printing system around 1450 CE, the term has been mostly referring to movable type printing, where reverse images of different graphemes (including punctuation marks and decorations) are cast in metal as individual elements (types). Types are arranged into larger units (most often entire lines), which are then assembled into a matrix to create a full page. The matrix thus created is then placed in a printing press. It is inked and brought into contact with the material to be printed, usually paper. The process can be repeated as many times as desired. Nevertheless, it should be noted that other kinds of printing, such as Egyptian and Mesopotamian cylinder seals as well as Chinese woodblock printing, have pre-existed Gutenberg's invention by thousands of years.

Unlike movable type printing, block printing employs a unitary matrix which is carved out of a single piece of wood. The production of block-printed texts is attested throughout the Islamic world since at least 900 CE, although the practice seems to have died out by 1430 CE. The extent to which this technique was actually used and where it exactly originated from is still unknown. Few block-printed documents can be found in existing archives, and there is a lack of archeological evidence related to matrices and printing equipment, which leaves the topic shrouded in obscurity. Amulets and charms are the only documents that have been regularly preserved. They contain Quranic passages printed on paper in a wide array of styles, and generally follow similar calligraphic practices as their handwritten counterparts [3]. They have not attracted much academic interest, and the DIA community has not developed any specific methods to treat them.

Movable type printing reached the Islamic world at the end of the fifteenth century CE with the establishment of printing presses in Constantinople and other cities of the Ottoman Empire. However, they were run by and for the Jewish minority and did not produce books in Arabic script. It is often assumed that the Ottoman Sultan Bayezid II prohibited movable type printing outright, or restricted it to certain scripts or minorities living in the Empire in 1485 CE. However, evidence is slim and contradictory [16]. What is certain is that printing of the Arabic script in the Islamic world was first performed in 1727 CE, in a work-

[2] [1, chapter XI] elaborates the cycles of popularity between round and hanging scripts.

[3] A survey of the current state of scholarship is found in [15]

shop established by imperial printer Ibrahim Müteferrika. It does not mean that Arabic books were not printed, as imports did exist, mostly from Italian workshops. The earliest example of this is a 1513 CE book of hours printed in Venice by Gregorio de Gregori [17]. The *qur'ān*'s earliest printed edition was produced in 1537-1538 CE by two brothers, Paganino and Alessandro Paganini. It proved to be an utter commercial failure because of its odd typeface and of the numerous errors the text contained [10, pg. 219-220]. Later, famous type-founders such as Robert Granjon (1585 CE), William Caslon (1720 CE), or Giambattista Bodoni (1759 CE) made other attempts at cutting typefaces. However, even these proved to be simplistic and of low quality [18].

Although universally derided for their lack of grace, legibility, and orthographical correctness later imports seem to have found more popular acceptance as lamented by Muteferrika in a 1727 CE work on the usefulness of printing. The secular nature of the vast majority of these imports support the theory that religious sentiment was a principal factor hindering the adoption of printing, although the economic interests of a substantial class of scribes and calligraphers certainly played a role, in addition to the inherent difficulties of creating adequate typefaces for a cursive script [1, pg. 605].

Despite this multi-faceted resistance to movable type printing, some refinement took place. While typeface cut in the Christian world were largely modelled on the *maghribī* style, Muteferrika introduced the more legible *naskh* style in printing which remains to this day the most popular style for printed Arabic texts. However, creating a full Arabic font remained a truly staggering task; the Imprimerie Nationale's twenty-four-point nineteenth century CE Arabic typeface contained 710 different types [10, pg. 218]. In addition, actually setting such large typefaces required skill incomparable to the much simpler Latin script. It has been argued that apart from social objections, the sheer laboriousness combined with a lacking industrial base made movable type printing uneconomical in the Islamic world [19]. Indeed, the rise of printing presses in the nineteenth century coincides with the invention of lithography which allowed inexpensive, accurate reproductions of handwritten texts, missionary activity and modernization drives less subject to immediate economic pressures [20]. Printers like Fāris al-Shidyāq, founder of the al-Jawā'ib press in Istanbul, introduced Western-style layouts with punctuation marks and paragraphs but were not able to resolve the laborious typesetting process or aesthetic issues such as lacking overlap of letters, line justification limited to baseline stroke elongation instead of the more aesthetically pleasing letter elongation or stacking, and visible gaps between individual types [1, pg. 605]. The invention of hot type line casting machines such as the 1911 Arabic Linotype eliminated gaps between glyphs but the limits of the machine did not allow proper placement of vowel marks [21, pg. 67] and subsequent iterations economised more and more glyph variants reducing the visual appeal of the final product. Only the advent of software-driven phototypesetters in the 1960s make sufficiently large type repertoires including contextual and elongated forms economically feasible. Even though the last vestiges of the lim-

itations of physical type have disappeared with purely digital typesetters, most common typefacess do not make use of these.

In the next chaper, we take a closer look at the limitations of current OCR technology on modern printed Arabic texts.

2.5 Basic requirements for Arabic OCR systems

To summarize, the recognition of Arabic handwritten texts requires a number of design decisions and features that available OCR engines do not have. This dissertation could not address all of the required decisions and features, chiefly because of the lack of available Arabic-script or substitute training data. The main necessary features are presented below, by order of processing inside a typical pipeline:

Freedom of Binarization A number of reasons makes it impossible to devise a general binarization method that could apply to most Arabic handwriting: the variety of supports used to write Arabic, the often fragmentary nature of writing on papyrus, the presence of faded and acid-damaged writing in iron-gall inks, and the highly decorated nature of pages. While trainable methods using semantic segmentation approaches exist, oftentimes their training data requirements are insurmountable.

Segmentation of curved and slanted lines The frequent use of baseline curvature and slanted lines for both practical and aesthetic purposes requires a layout analysis (LA) method capable of modelling lines in a way that allows normalization of the axis of writing and the effective suppression of non-line content.

Semantic layout analysis Many Arabic manuscripts contain an unusual amount of paratext or textual *noise*, be it marginal notes, parallel texts, interlinear translations, or detached word fragments. Proper treatment of these elements, e.g. separation of commentary from maint text or selection of the appropriate recognition model for a parallel text in another script, requires at least in part awareness of these elements in the LA component. As the types of elements, their presentation, and frequency can vary immensely it is highly desirable for the LA system to be trainable.

Advanced reading order determination Related to the classification of textual components in the LA module is the correct ordering of the texts and the proper linking of paratextual components to the main text. As mentioned before this is a nacent field of research and all OCR systems available infer the order of text using highly flawed heuristics.

Segmentation-free transcription The cursive nature of Arabic writing, particularly the variation of letter connections encountered between different scripts makes reliable segmentation into single graphemes impractical

for handwritten texts, especially so when executed in highly stylized and contracted scripts. While a significant body of literature has focused on this problem, the success of sequence-to-sequence machine learning models that require no or only implicit segmentation/alignment of input image data and characters has by and large solved it.

Data creation and curation tools While they are not directly part of OCR pipelines (especially if employed in large-scale archival or library digitization projects), the lack of adapted open tools to create training data that can capture the pertinent features of Arabic handwritten text; and the subsequent lack of training data sets available for DIA research have stifled the field. Simple transcription tools such as those used in the studies detailed in chapter 4 are inadequate for preparing multi-purpose datasets. Fully featured Virtual Research Environments (VREs) for palaeographic scholarly work are one way of reaching deeply annotated data through flexible non-writing-system-specific data models.

Other minor technical requirements which are not large enough to be the focus of a dedicated research question are sometimes disregarded. The most visible example of this is the lack of support, in older or purely research software, for non-ASCII and bidirectional text requiring manual substitution and reordering of code points. More subtle sources of errors such as overly aggressive normalization of input text, lack of Unicode whitespace normalization, and the treatment of non-printing code points are also prevalent and often difficult to ascertain, especially in proprietary software.

It should be noted that while the underlying motivations to implement the above requirements are specific to the Arabic script, comparable situations exist for documents written in other scripts. De facto, similar motivations for binarization-freedom apply to almost any historical written document; not to mention curved handwriting, marginalia, and parallel texts. Therefore, research which focuses on tailoring an OCR system capable of better processing Arabic texts is also likely to *enhance* the capacity to process other texts as well.

References

[1] S. Blair, *Islamic Calligraphy*. Edinburgh University Press, 2006.

[2] D. Awad, "The Evolution of Arabic Writing Due to European Influence: The case of punctuation," *Journal of Arabic and Islamic Studies*, vol. 15, pp. 117–136, 2015.

[3] H. Déjean, J.-L. Meunier, *et al.*, "Versatile layout understanding via conjugate graph," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 287–294.

[4] F. Déroche, A. Berthier, and M. Waley, "Islamic Codicology: An Introduction to the Study of Manuscripts in Arabic Script," *Muhammad Isa Waley, London, al-Furqān Islamic Heritage Foundation*, 2006.

[5] A. Gacek, *Arabic manuscripts: a vademecum for readers*. Brill, 2009, vol. 98.

[6] M. A. Dhali, S. He, M. Popovic, E. Tigchelaar, and L. Schomaker, "A Digital Palaeographic Approach towards Writer Identification in the Dead Sea Scrolls," in *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2017, Porto, Portugal, February 24-26, 2017*, M. D. Marsico, G. S. di Baja, and A. L. N. Fred, Eds., SciTePress, 2017, pp. 693–702. DOI: 10.5220/0006249706930702.

[7] T. Lavee, "Computer analysis of the dead sea scroll manuscripts," Ph.D. dissertation, Tel Aviv University, 2013.

[8] R. L. E. Jr., K. T. Knox, and W. A. Christens-Barry, "Multispectral Imaging of the Archimedes Palimpsest," in *32nd Applied Image Pattern Recognition Workshop (AIPR 2003), Image Data Fusion, 15-17 October 2003, Washington, DC, USA, Proceedings*, IEEE Computer Society, 2003, pp. 111–118. DOI: 10.1109/AIPR.2003.1284258.

[9] A. Starynska, R. L. Easton Jr, and D. Messinger, "Methods of data augmentation for palimpsest character recognition with deep neural network," in *Proceedings of the 4th International Workshop on Historical Document Imaging and Processing*, 2017, pp. 54–58.

[10] J. Bloom, *Paper Before Print: The History and Impact of Paper in the Islamic world*. Yale University Press, 2001, ISBN: 9780300089554.

[11] H. Ghoname, "Sini Calligraphy: The Preservation of Chinese Muslims' Cultural Heritage," Ph.D. dissertation, University of Hawaii at Manoa, 2012.

[12] T. Christiansen, "Manufacture of black ink in the ancient mediterranean," *Bulletin of the American Society of Papyrologists*, vol. 54, pp. 167–195, 2017.

[13] L. Raggetti, "Inks as instruments of writing: Ibn al-ǧazarī's book on the art of penmanship," *Journal of Islamic Manuscripts*, vol. 10, no. 2, pp. 201–239, 2019.

[14] K. Versteegh, M. Eid, A. Elgibali, M. Woidich, and A. Zaborski, *Encyclopedia of Arabic Language and Linguistics, Volume 3*. Leiden, The Netherlands: Brill, 2007, ISBN: 978-90-04-14475-0.

[15] K. R. Schaefer, *Enigmatic charms: medieval Arabic block printed amulets in American and European libraries and museums*. Handbook of Oriental Studies, 2006, vol. 82.

[16] K. A. Schwartz, "Did Ottoman Sultans Ban Print?" *Book History*, vol. 20, no. 1, pp. 1–39, 2017.

[17] M. Krek, "The enigma of the first Arabic book printed from movable type," *Journal of Near Eastern Studies*, vol. 38, no. 3, pp. 203–212, 1979.

[18] W. Tracy, "Advances in arabic printing," *British Journal of Middle Eastern Studies*, vol. 2, no. 2, pp. 87–93, 1975.

[19] H. Auji, "Neither good, fast, nor cheap: Challenges of early arabic letterpress printing," *American Printing History Association (blog)*, 2017.

[20] J. Turner, *The Dictionary of Art*. 1996.

[21] T. Nemeth, *Arabic type-making in the Machine Age: The influence of technology on the form of Arabic type, 1908–1993*. Brill, 2017.

Chapter 3

Important New Developments in Arabographic Optical Character Recognition (OCR)

This chapter has been published as B. Kiessling, M. T. Miller, G. Maxim, S. B. Savant, *et al.*, "Important New Developments in Arabographic Optical Character Recognition (OCR)," *Al-ʿUṣūr al-Wusṭā*, vol. 25, pp. 1–13, 2017

3.1 Introduction

3.1.1 Summary of Results of OpenITI's OCR

The OpenITI team[1] building on the foundational open-source OCR work of the Leipzig University's (LU) Alexander von Humboldt Chair for Digital Humanities—has achieved Optical Character Recognition (OCR) accuracy rates for classical Arabic-script texts in the high nineties. These numbers are based on our tests of seven different Arabic-script texts of varying quality and typefaces, totaling over 7,000 lines (400 pages, 87,000 words; see table 3.1 for full details). These accuracy rates not only represent a distinct improvement over the actual[2] accuracy rates of the various proprietary OCR options for printed classical Arabic-script texts, but, equally important, they are produced using an open-source OCR software called Kraken (developed by Benjamin Kiessling, LU), thus enabling us to make this Arabic-script OCR technology freely available to the broader Islamic, Persian, and Arabic Studies communities in the near future. In the process we also generated over 7,000 lines of "gold standard" (double-checked) data that can be used by others for Arabic-script OCR training and testing purposes.[3]

3.1.2 OCR and its Importance for Islamicate Studies Fields

Although there is a wealth of digital Persian and Arabic texts currently available in various open-access online repositories,[4] these collections still need to be expanded and supplemented in some important ways. OCR software is critical for this broader task of expanding the range of digital texts available to scholars for computational analysis. OCR programs, in the simplest terms, take an image of a text, such as a scan of a print book, and extract the text, converting the image of the text into a digital text that then can be edited, searched, computationally analyzed, etc.

[1]The co-PIs of the Islamicate Texts Initiative (ITI) are Sarah Bowen Savant (Aga Khan University, London), Maxim G. Romanov (Leipzig University), and Matthew Thomas Miller (Roshan Institute for Persian Studies, University of Maryland, College Park).

[2]Proprietary OCR programs for Persian and Arabic (e.g., Sakhr's Automatic Reader, ABBYY Finereader, Readiris) over-promise the level of accuracy they deliver in practice when used on classical texts. These companies claim that they provide accuracy rates in the high 90 percentages (e.g., Sakhr claims 99.8% accuracy for high-quality documents). This may be the case for texts with simplified typeset and no short vowels; however, our tests of ABBYY Finereader and Readiris on high-quality scans of classical texts turned out accuracy rates of between 65% and 75%. Sakhr software was not available to us, as the developers offer no trial versions and it is the most expensive commercial OCR solution for Arabic. Moreover, since these programs are not open-source and offer only limited trainability (and created training data cannot be reused), their costs are prohibitive for most students and scholars and they cannot be modified according to the interests and needs of the academic community or the public at large. Most importantly, they have no web interfaces that would enable the production of wider, user-generated collections.

[3]This gold standard data is available at: https://github.com/OpenArabic/OCR_GS_Data.

[4]Collecting and rendering these texts useful for computational textual analysis (through, for example, adding scholarly metadata and making them machine-actionable) is a somewhat separate but deeply interrelated project that OpenITI is currently working on as well.

The specific type of OCR software that we employed in our tests is an open-source OCR program called Kraken, which was developed by Benjamin Kiessling at Leipzig University's Alexander von Humboldt Chair for Digital Humanities. Unlike more traditional OCR approaches, Kraken relies on a neural network—which mimics the way we learn—to recognize letters in the images of entire lines of text without trying first to segment lines into words and then words into letters. This segmentation step—a mainstream OCR approach that persistently performs poorly on connected scripts—is thus completely removed from the process, making Kraken uniquely powerful for dealing with the diverse variety of ligatures in connected Arabic script (see section 3.1 for more technical details).

3.2 Initial OCR Tests

We began our experiments by using Kraken to train a model[5] on high-quality[6] scans of ~1,000 lines of Ibn al-Faqīh's al-Buldān (work #1).

We first generated training data (line transcriptions) for all of these lines, double checked them (creating so-called "gold standard" data), trained the model, and, finally, tested its ability to accurately recognize and extract the text. The results were impressive, reaching 97.56% accuracy for the entire text and an even more impressive 99.68% accuracy rate on the Arabic script alone (i.e., when errors related to punctuation and spaces were removed from consideration; such non-script errors are easy to fix in the post-correction phase and, in many cases, this correction process for non-script errors can be automated). See table 3.2, row #1 for full details.[7]

These numbers were so impressive that we decided to expand our study and use the model built on the text of Ibn al-Faqīh's al-Buldān (work #1) to OCR six other texts. We deliberately selected texts that were different from Ibn al-Faqīh's original text in terms of both their Arabic typeface, editorial orthographic conventions, and image quality. These texts represent at least two different typefaces (within which there are noticeable variations of font, spacing, and ligature styles), and four of the texts were high-quality scans while the other two were low-quality scans downloaded from www.archive.org (via http://waqfeya.com/).[8]

[5] "Training a model" is a general term used in machine learning for training a program to recognize certain patterns in data. In the context of OCR work, it refers to teaching the OCR software to recognize a particular script or typeface—a process that only requires time and computing power. In our case, this process required 1 computer core and approximately 24 hours.

[6] "High quality" here means 300 dpi, color or grayscale images. Before the actual process of OCR, these images must be binarized—converted into black-and-white images; if binarization is not performed properly, a lot of information is lost from an image, negatively affecting the accuracy of the OCR process. For this reason, for best results, one should avoid using pre-binarized images.

[7] We have also experimented with the internal configuration of our models: more extensive models, containing 200 nodes in the hidden middle layer, showed slightly better accuracy in most cases (works #4-5 were an exception to this pattern), but it took twice as long to train and the OCR process using the larger model also takes more time.

[8] "Low-quality" here means 200 dpi, black and white, pre-binarized images. In short, the standard

Table 3.1: Description of data

	Book[a]	Quality	Type	Size of data samples			
				Pages	Lines	Words	Chars
1	Ibn al-Faqīh al-Buldān	high[b]	training	79	1466	16 909	92 730
2	Ibn al-Athīr al-Kāmil	high[b]	testing	40	794	12 818	58 481
3	Ibn Qutayba Adab al-kātib	high[b]	testing	55	794	7848	42 230
4	al-Jāḥiẓ al-Ḥayawān	high[b]	testing	65	992	11 870	59 191
5	al-Yaʿqūbī al-Taʿrīkh	high[b]	testing	68	1050	13 487	66 341
6	al-Dhahabī Taʾrīkh al-Islām	low[c]	testing	50	1110	11 045	55 047
7	Ibn al-Jawzī al-Muntaẓam	low[c]	testing	50	938	13 156	62 574
				407	7144	87 133	436 594

[a]For details on the editions, see bibliography

[b]300 dpi, grayscale spanned specifically for the purpose of testing with ideal parameters

[c]200 dpi, black-and-white, pre-binarized, both downloaded from http://www.archive.org (via http://waqfeya.org)

When looking at the results in table 3.2, it is important that the reader notes that works #2-7 are "testing" data. That is, these accuracy results were achieved by utilizing a model built on the text of work #1 to perform OCR on these other texts. For this reason it is not surprising that the accuracy rates for works #2-5 are not as high as the accuracy rates for the training text, work #1. The point that is surprising is that the use of the work #1-based model on the low quality scans of works #6-7 achieved a substantially higher accuracy rate (97.61% and 97.8% respectively on their Arabic script alone) than on the high-quality scans of works #2-5. While these higher accuracy rates for works #6-7 are the result of a closer affinity between their typefaces and that of work #1, it also indicates that the distinction between high- and low-quality images is not as important for achieving high accuracy rates with Kraken as we initially believed. In the future, this will help reduce substantially both the total length of time it takes to OCR a work and the barriers to entry for researchers wanting to OCR the low-quality scans they already possess.

quality of most scans available on the internet, which are the product of scanners that prioritize smaller size and speed of scanning for online sharing (i.e., in contrast to high-quality scans that are produced for long-term preservation). "Pre-binarized" means that the images were converted to black and white during the scanning process, ostensibly for size reduction, resulting in some degradation of quality.

Table 3.2: Accuracy rates in test of our custom model

	Book[a]	Quality	Type	Character accuracy			
				Size 100	Ar[b]	Size 200	Ar[b]
1	Ibn al-Faqīh al-Buldān	high[c]	training	95.88%	99.68%	97.56%	99.68%
2	Ibn al-Athīr al-Kāmil	high[c]	testing	85.78%	90.90%	87.18%	90.56%
3	Ibn Qutayba Adab al-kātib	high[c]	testing	75.28%	87.67%	74.03%	87.90%
4	al-Jāḥiẓ al-Ḥayawān	high[c]	testing	69.03%	72.78%	68.32%	71.87%
5	al-Yaʿqūbī al-Taʿrīkh	high[c]	testing	78.78%	83.42%	78.28%	81.85%
6	al-Dhahabī Taʾrīkh al-Islām	low[d]	testing	92.19%	97.54%	94.42%	97.61%
7	Ibn al-Jawzī al-Muntaẓam	low[d]	testing	90.40%	97.39%	92.26%	97.80%

[a]For details on the editions, see bibliography
[b]Performance on Arabic only (excluding punctuation, spaces and numerals
[c]300 dpi, grayscale spanned specifically for the purpose of testing with ideal parameters
[d]200 dpi, black-and-white, pre-binarized, both downloaded from http://www.archive.org (via http://waqfeya.org)

Table 3.3: Ligature variations in typefaces. The table highlights only a few striking differences and is not meant to be comprehensive; examples similar to those of the main text are "greyed out."

Book									
1	Ibn al-Faqīh al-Buldān	لهم	بها	الملا	نجد	لم	إلىٰ	فيها	نم
2	Ibn al-Athīr al-Kāmil	لهم	بها	الملا	نجد	لم	إلى	فيها	نم
3	Ibn Qutayba Adab al-kātib	لهم	بها	not present	نجد	لم	إلى	فيها	نم
4	al-Jāḥiẓ al-Ḥayawān	لهم	بها	الملا	نجد	لم	إلى	فيها	نم
5	al-Yaʿqūbī al-Taʿrīkh	لهم	بها	الملا	نجد	لم	إلى	فيها	نم
6	al-Dhahabī Taʾrīkh al-Islām	لهم	بها	not present	نجد	لم	إلى	فيها	نم
7	Ibn al-Jawzī al-Muntaẓam	لهم	بها	not present	نجد	لم	إلى	فيها	نم

(the table highlights only a few striking differences and is not meant to be comprehensive; examples similar to those of the main text are "greyed out")

The decreased accuracy results for works #2-5 are explainable by a few factors:

1. The typeface of works #4-5 is different than work #1 and it utilizes a number of ligatures that are not present in the typeface of work #1 (for examples, see table 3.3 above).

2. The typefaces of work #2-3 are very similar to that of #1, but they both have features that interfere with the #1-based model. #2 actually uses two different fonts, and the length of connections—kashīdas—between letters vary dramatically (0.3 kashīda to 2 kashīdas and everything in between), which is not the case with #1, where letter spacing is very consistent.

3. The text of work #3 is highly vocalized—it has more ḥarakāt than any other texts in the sample (and especially in comparison with the model work #1).

4. The text of work #3 also has very complex and overabundant punctuation with highly inconsistent spacing.

Our #1-based model could not completely handle these novel features in the texts of works #2-5 because it was not trained to do so. As the results in table 3.4 of the following section show, new models can be trained to handle these issues successfully.

Table 3.4: Accuracy rates in text-specific models

				Character accuracy	
	Book[a]	**Quality**	**Type**	**Size 100**	**Ar**[b]
2	Ibn al-Athīr al-Kāmil	high[c]	testing	93.79%	97.71%
3	Ibn Qutayba Adab al-kātib	high[c]	testing	89.30%	98.47%
4	al-Jāḥiẓ al-Ḥayawān	high[c]	testing	94.87%	97.59%
5	al-Yaʿqūbī al-Taʿrīkh	high[c]	testing	96.81%	99.18%

[a]For details on the editions, see bibliography
[b]Performance on Arabic only (excluding punctuation, spaces and numerals
[c]300 dpi, grayscale spanned specifically for the purpose of testing with ideal parameters

3.3 Round #2 Tests: Training New Models

The most important advantage of Kraken is that its workflow allows one to train new models relatively easily, including text-specific ones. In a nutshell, the process of training requires a transcription of approximately 800 lines (the number will vary depending on the complexity of the typeface) aligned with images of

these lines as they appear in the printed edition. The training itself takes 12-24 hours and is performed by a machine without human involvement; multiple models can be trained simultaneously. Kraken includes tools for the production of transcription forms (see figure 3.1 below); the data supplied through these forms is then used to train a new model. (Since there are a great number of Arabic-script texts that have already been converted into digital texts, one can use these as the base texts to fill in the forms more quickly—i.e., instead of typing the transcription—and then double-check them for accuracy; this was what we did, and it saved us a lot of time.)

The importance of Kraken's ability to quickly train new models is illustrated clearly by texts such as works #2-5 . When using the model built on work #1 in our initial round of testing, we were only able to achieve accuracy rates ranging from the low seventies to low nineties on these texts (see table 3.2). However, when we trained models on works #2-5 specifically in our second round of testing, the accuracy rates for these texts substantially improved, reaching into the high nineties (see full results in table 3.4 above). The accuracy results for work #5, for example, improved from 83.42% on Arabic script alone in our first work #1-based model tests to 99.18% accuracy when we trained a mode on this text. The accuracy rates for works #2-4 similarly improved, increasing from 90.90% to 97.71% , 87.90% to 98.47%, from 72.78% to 97.59%, respectively (see appendix 3.A for the accuracy rates of these new models on all other texts as well.). These accuracy rates for Arabic-script recognition are already high, but we actually believe that they can be improved further with larger training data sets.

Although the process of training a new model for a new text/typeface does require some effort, the only time-consuming component is the generation of ~800 lines of gold standard training data. As we develop the OpenITI OCR project we will address the issue of the need for multiple models through a two-pronged strategy. First, we will try to train a general model, periodically adding new features that the model has not "seen" before. Secondly, we will train individual models for distinct typefaces and editorial styles (which sometimes vary in their use of vocalization, fonts, spacing, and punctuation), producing a library of OCR models that gradually will cover all major typefaces and editorial styles used in modern Arabic-script printing. There certainly are numerous Arabic-script typefaces and editorial styles that have been used throughout the last century and a half of Arabic-script printing, but ultimately the number is finite and definitely not so numerous as to make it impossible to create models for each over the long term.

3.4 Conclusions and Next Steps for the OpenITI OCR Project

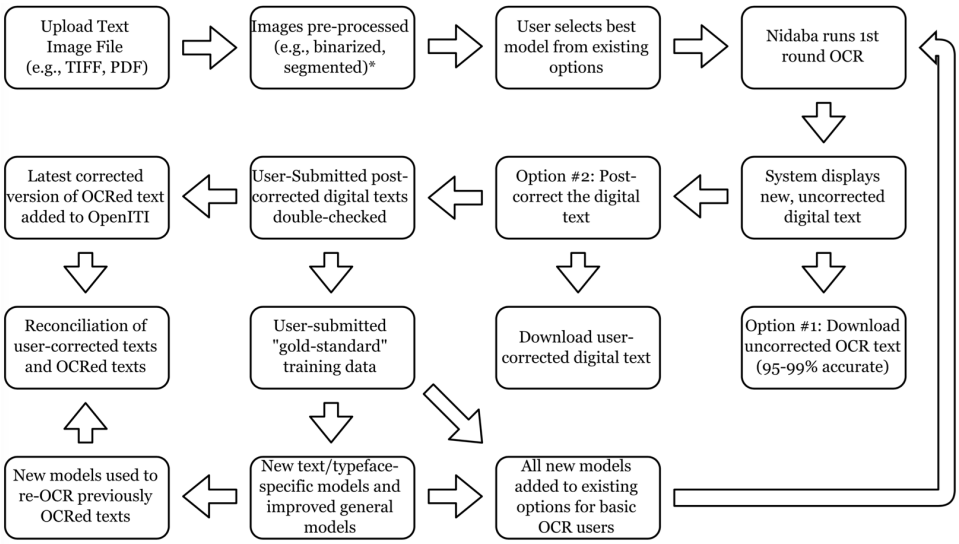
The two rounds of testing presented here indicate that with a fairly modest amount of gold standard training data (~800–1,000 lines) Kraken is consistently able to pro-

Figure 3.1: Kraken's transcription interface

duce OCR results for Arabic-script documents that achieve accuracy rates in the high nineties. In some cases, such as works #6-7, achieving OCR accuracy rates of up to 97.5% does not even require training a new model on that text. However, in other cases, such as works #2-5, achieving high levels of OCR accuracy does require training a model specific to that typeface, and, in some select cases of texts with similar typefaces but different styles of vocalization, font variations, and punctuation patterns (e.g., works #2-3), training a model for the peculiarities of a particular edition.

In the near future we are planning to develop a user-friendly web-interface for the generation of new training data and the post-correction of the OCR output (see figure 3.2 above). Data supplied by users will allow us to train new models. It should be stressed that training edition-specific models is quite valuable, as there is a number of multivolume books—often with over a dozen volumes per text—that need to be converted into proper digital editions. In the long term, we are also planning to train models for other Islamicate languages (Ottoman Turkish, Urdu, Syriac,[9] etc.). Our hope is that an easy-to-use and effective OCR pipeline will allow us, all,collectively to significantly enrich our collection of digital Persian and Arabic texts and thereby enable us to understand better the cultural heritage of the Middle East as reflected in its literary traditions.

[9]Together with George Kiraz (Gorgias Press, Editor-in-Chief; Rutgers University, Visiting Scholar), we have most recently tested Kraken on Syriac texts, achieving comparable accuracy rates of 95.17–99.09%.



* The pre-processing step automatically includes image binarization, conversion to grayscale, conversion to PNG, and page segmentation. De-warping, de-skewing, and de-noising are also possible in Nidaba, but they are not currently built into the full pipeline as automatic features.

Figure 3.2: Web-based OCR pipeline flowchart

3.5 The Technical Details: Kraken and its OCR Method

Kraken is the open-source OCR software that we used in our tests. Developed by Benjamin Kiessling at UL's Alexander von Humboldt Chair for Digital Humanities, Kraken is a "fork"[10] of the unmaintained ocropus package[11] combined with the CLSTM neural network library.[12] Kraken represents a substantial improvement over the ocropus package: its performance is drastically better, it supports right-to-left scripts and combined LTR/RTL (BiDi) texts, and it includes a rudimentary transcription interface for offline use.

The OCR method that powers Kraken is based on a long short-term memory [9] recurrent neural network utilizing the Connectionist Temporal Classification objective function ([10], as elaborated in [11]). In contrast to other systems requiring character level segmentation before classification, it is uniquely suited for the recognition of connected Arabographic scripts because the objective function used during training is geared towards assigning labels—i.e., characters/glyphs—to regions of unsegmented input data.

The system works on unsegmented data both during training and recognition—its base unit is a text line (line recognizer). For training, a number of printed lines have to be transcribed using a simple HTML transcription interface (see figure 3.1

[10] "Fork" is a computer-science term for a new independent development that builds on an existing software.

[11] For details, see: https://github.com/tmbdev/ocropy and https://en.wikipedia.org/wiki/OCRopus.

[12] See: https://github.com/tmbdev/clstm.

above). The total amount of training data (i.e., line image-text pairs) required may vary depending on the complexity of the typeface and number of glyphs used by the script. Acquisition of training data can be optimized by line-wise alignment of existing digital editions with printed lines, although even wholesale transcription is a faster and relatively unskilled task in comparison to training data creation for other systems such as tesseract.[13]

Our current models were trained on ~1,000 pairs each, corresponding to ~50-60 pages of printed text. Models are fairly typography specific, the most important factor being fonts and spacing, although some mismatch does not degrade recognition accuracy substantially (2-5%).[14] Thus new training data for an unknown typeface can be produced by correcting the output from a model for a similar font—in other words, generating training data for every subsequent model will require less and less time. Last but not least, it is also possible to train multi-typeface models by simply combining training data, albeit some parameter tuning is required to account for the richer typographic morphology that the neural network must learn.

3.6 Acknowledgements

We would never have been able to complete this work without the help of our team members at Leipzig University, University of Maryland (College Park), and Aga Khan University, London. We would also like to thank Elijah Cooke (Roshan Institute, UMD) for helping us to process the data, Samar Ata (Roshan Institute, UMD) for generating several sets of high-quality scans for us, and Layal Mohammad (ISMC, AKU), Mohammad Meqdad (ISMC, AKU), and Fatemeh Shams (ISMC, AKU) for helping us to generate and double check the training data. Lastly, we would like to express our gratitude to Gregory Crane (Alexander von Humboldt Chair for Digital Humanities, LU), Fatemeh Keshavarz (Roshan Institute for Persian Studies, UMD), and David Taylor (ISMC, AKU) for their guidance and support of our work.

[13] See: https://github.com/tesseract-ocr and https://en.wikipedia.org/wiki/Tesseract_(software).

[14] For example, if a glyph is in a slightly different font than the one that the model was trained on, it may sometimes be misrecognized as another one (or not at all), thus leading the overall accuracy rate to be slightly lower despite the fact that most of the other text is recognized correctly.

3.A Performance of Text-Specific Models

Table 3.5: Performance of #2-based model on other texts

	Book	Quality	Type	Character accuracy			
				Size 100	Ar	Size 200	Ar
2	**Ibn al-Athīr** **al-Kāmil**	**high**	**training**	**93.79%**	**97.71%**	**93.58%**	**97.59%**
3	Ibn Qutayba Adab al-kātib	high	testing	82.68%	95.72%	80.92%	94.88%
4	al-Jāḥiẓ al-Ḥayawān	high	testing	71.78%	75.16%	70.85%	74.27%
5	al-Yaʿqūbī al-Taʿrīkh	high	testing	79.67%	84.40%	78.12%	82.21%
6	al-Dhahabī Taʾrīkh al-Islām	low	testing	90.68%	95.95%	90.37%	95.78%
7	Ibn al-Jawzī al-Muntaẓam	low	testing	93.33%	98.51%	92.96%	98.22%

Table 3.6: Performance of #3-based model on other texts

	Book	Quality	Type	Character accuracy			
				Size 100	Ar	Size 200	Ar
2	Ibn al-Athīr al-Kāmil	high	testing	83.52%	88.56%	83.55%	88.56%
3	**Ibn Qutayba** **Adab al-kātib**	**high**	**training**	**89.30%**	**98.47%**	**89.42%**	**98.44%**
4	al-Jāḥiẓ al-Ḥayawān	high	testing	74.82%	76.51%	74.87%	76.65%
5	al-Yaʿqūbī al-Taʿrīkh	high	testing	81.50%	84.05%	79.81%	83.67%
6	al-Dhahabī Taʾrīkh al-Islām	low	testing	84.89%	93.19%	83.08%	92.53%
7	Ibn al-Jawzī al-Muntaẓam	low	testing	87.56%	94.21%	86.34%	93.57%

Table 3.7: Performance of #4-based model on other texts

	Book	Quality	Type	Character accuracy			
				Size 100	Ar	Size 200	Ar
2	Ibn al-Athīr al-Kāmil	high	testing	80.23%	86.27%	82.46%	87.48%
3	Ibn Qutayba Adab al-kātib	high	testing	80.90%	91.54%	82.61%	93.24%
4	**al-Jāḥiẓ** **al-Ḥayawān**	**high**	**training**	**94.86%**	**97.59%**	**94.82%**	**97.41%**
5	al-Yaʿqūbī al-Taʿrīkh	high	testing	90.91%	95.13%	91.28%	94.71%
6	al-Dhahabī Taʾrīkh al-Islām	low	testing	81.93%	91.23%	83.03%	92.22%
7	Ibn al-Jawzī al-Muntaẓam	low	testing	84.07%	93.58%	86.26%	94.20%

Table 3.8: Performance of #5-based model on other texts

	Book	Quality	Type	Character accuracy			
				Size 100	Ar	Size 200	Ar
2	Ibn al-Athīr al-Kāmil	high	testing	79.80%	86.35%	N/A	N/A
3	Ibn Qutayba Adab al-kātib	high	testing	72.99%	82.84%	N/A	N/A
4	al-Jāḥiẓ al-Ḥayawān	high	testing	83.38%	87.65%	N/A	N/A
5	**al-Yaʿqūbī** **al-Taʿrīkh**	**high**	**training**	**96.81%**	**99.18%**	**N/A**	**N/A**
6	al-Dhahabī Taʾrīkh al-Islām	low	testing	82.76%	90.65%	N/A	N/A
7	Ibn al-Jawzī al-Muntaẓam	low	testing	87.71%	96.00%	N/A	N/A

Editions of Printed Texts

[1] Ibn al-Faqīh (d. 365/975), *Al-Buldān*, Yūsuf al-Hādī, Ed. ʿĀlam al-Kutub, 1996.

[2] Ibn al-Athīr (d. 630/1232), *Al-Kāmil fī al-taʾrīkh*, ʿAbd Allāh al-Qāḍī, Ed. Dār al-Kutub al-ʿIlmiyya, 1415/1994.

[3] Ibn Qutayba (d. 276/889), *Adab al-kātib*, Muḥammad al-Dālī, Ed. Muʾassasat al-Risāla.

[4] al-Jāḥiẓ (d. 255/868), *Al-Ḥayawān*. Dār al-Kutub al-ʿIlmiyya, 1424/2003.

[5] al-Yaʿqūbī (d. 292/904), *Al-Taʿrīkh*. Dār Ṣādir.

[6] al-Dhahabī (d. 748/1347), *Taʾrīkh al-Islām*. Al-Maktaba al-Tawfīqiyya, Ed.

[7] Ibn al-Jawzī (d. 597/1201), *Al-Muntaẓam*. Muḥammad al-Qādir ʿAṭā, Muṣṭafá al-Qādir ʿAṭā., Ed. Dār al-Kutub al-ʿIlmiyya, 1412/1992.

References

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 369–376.

[11] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait, "High-performance OCR for printed English and Fraktur using LSTM networks," in *2013 12th international conference on document analysis and recognition*, IEEE, 2013, pp. 683–687.

Chapter 4

Advances and Limitations in Open Source Arabic-Script OCR: A Case Study

This chapter has been accepted for publication as B. Kiessling, G. Kurin, M. T. Miller, and K. Smail, "Advances and Limitations in Open Source Arabic-Script OCR: A Case Study," *Digital Studies / Le champ numérique*,

Abstract

This work presents an accuracy study of the open source OCR engine, Kraken, on the leading Arabic scholarly journal, al-Abhath. In contrast with other commercially available OCR engines, Kraken is shown to be capable of producing highly accurate Arabic-script OCR. The study also assesses the relative accuracy of typeface-specific and generalized models on the al-Abhath data and provides a microanalysis of the "error instances" and the contextual features that may have contributed to OCR misrecognition. Building on this analysis, the paper argues that Arabic-script OCR can be significantly improved through (1) a more systematic approach to training data production, and (2) the development of key technological components, especially multi-language models and improved line segmentation and layout analysis.

4.1 Introduction

In late 2017 JSTOR initiated a collaboration with the Open Islamicate Texts Initiative (OpenITI) to run an Optical Character Recognition (OCR) pilot for the JSTOR Arabic digitization feasibility study (funded through a generous grant by the National Endowment for the Humanities).[1] The problem that JSTOR had encountered in their feasibility study was the problem that has long plagued efforts of scholars and librarians to digitize Arabic, Persian, Urdu, and Ottoman Turkish print documents: Arabic-script OCR programs produce notoriously poor results, despite the optimistic claims of some of their marketing materials[2].[2] In addition to these programs' lackluster performance, they also are not ideal systems for academic users for other reasons as well - for example, several are prohibitively expensive (for the average academic) and they offer little out of the box trainability (i.e., they come only with a generic OCR model and they cannot be trained to recognize new typefaces).

[1] OpenITI is a multi-institutional initiative that is focused on building digital infrastructure for the computational study of the texts of the Islamicate world. It is currently led by Dr. Matthew Thomas Miller (Roshan Institute for Persian Studies, University of Maryland, College Park), Dr. Sarah Bowen Savant (Aga Khan University, London), and Dr. Maxim Romanov (University of Vienna). Benjamin Kiessling (University of Leipzig/Université PSL) is one of OpenITI's primary computer science collaborators and he served as the technical lead for the OpenITI JSTOR OCR pilot. More information on the OpenITI project are available on the project's website: https://openiti.org/about. More information on JSTOR's NEH-funded project can be found here: https://about.jstor.org/news/jstor-receives-50000-neh-grant/.

[2] Mansoor Alghamdi and William Teahan open their 2017 study by noting that "although handwritten script is significantly more challenging than printed Arabic text for OCR, Arabic printed text OCR still poses significant challenges." After evaluating Sakhr, Finereader, RDI Clever Page, and Tesseract (3)—the main options for Arabic-script OCR—on Arabic print works they conclude that "all the evaluated Arabic OCR systems have low performance accuracy rates, below 75 per cent, which means that the time which would take to manually correct the OCR output would be prohibitive." These results are consonant with the PIs' own experience using these OCR engines and those of our colleagues in the field of Islamicate Studies.

With these problems in mind, in 2016 OpenITI began working on the development of open source OCR tools for Arabic-script languages (in print form) in collaboration with the computer scientist Benjamin Kiessling.[3]

OpenITI's first OCR study with the new open source OCR engine, Kraken, developed by Kiessling, demonstrated that it was capable of achieving Arabic script-only accuracy rates >97.5% with as little as 800-1,000 lines of training data for that document's typeface [7].[4] OpenITI has also replicated these high accuracy rates on Persian texts, with Perso-Arabic script-only accuracy rates ranging from 96.3% to 98.62% with typeface-specific models.[5]

In this work, we present the results of our OCR study done in collaboration with JSTOR on the al-Abhath Arabic journal (arguably the most important Arabic language scholarly journal in the Middle East). In contrast with many OCR accuracy reports, in this study we performed both detailed manual and automatic Character Error Rate (CER) accuracy checks, which enabled us to develop a much more fine-grained understanding of where the Kraken OCR engine was failing to properly transcribe the Arabic text. These results confirm Kraken's ability to produce highly accurate Arabic-script OCR, but they also provide new insights into the importance of systematic training data production, the relative accuracy of typeface-specific and generalized models, and the key technological improvements needed for improved Arabic-script OCR output.

Section two reviews the open-source software used in this study before the JSTOR Arabic OCR pilot and accuracy study are described in sections three and four, respectively. Section five contains our recommendations for the necessary technological and data improvements needed to improve Arabic OCR in the future.

4.2 OpenITI OCR Software: Kraken

Kraken is an open-source OCR engine that was developed by Benjamin Kiessling. It utilizes a segmentationless sequence-to-sequence approach [8] with a hybrid convolutional and recurrent neural network to perform character recognition [9] which obviates the need for character-level segmentation—i.e. the neural network responsible for text extraction from image data recognizes whole lines of

[3]To date our work has primarily focused on Arabic-script OCR for print documents since handwritten text recognition (HTR) for Arabic-script manuscripts presents a series of additional issues (e.g., even more complex line segmentation and page layout analysis problems, a dizzying array of different script styles and scribal hands). However, we have begun preliminary experiments on Persian and Arabic manuscripts with some promising initial results using distantly supervised methods of training data production and the new line segmentation methods developed by Kiessling [3]. See also the experiments on HTR for Arabic manuscripts led by the British Library [4]–[6]).

[4]Training data, in the context of OCR, consists of pairs of scans of individual lines of text with their digital transcription.

[5]This work has not been published yet, but the full CER reports for these tests can be viewed at OpenITI's GitHub repository: https://github.com/OpenITI/OCR_GS_Data/tree/master/fas.

text without resorting to smaller subunits like words or characters which can be difficult to accurately compute for languages written in connected script.

As an initial preprocessing step page images are converted into black and white through a process called binarization. Layout analysis, i.e. the detection of lines for subsequent steps, is then performed on this binarized image with an algorithm based on traditional computer vision methods. In a final step, the previously detected rectangular lines are fed into the neural network for character recognition.[6]

The benefit of eliminating fine-grained segmentation in comparison to older character segmenting systems such as tesseract 3, Sakhr, and most likely Abbyy FineReader[7] is not only expressed during recognition but also through easier to produce training data for adaptation of the OCR system to new scripts and typefaces. With character-based systems annotators have to manually locate and transcribe single characters while Kraken is trained on full line transcriptions which are faster to annotate and verify, especially in the case of connected scripts.

4.3 The OpenITI JSTOR OCR Pilot

OpenITI began the JSTOR OCR pilot by performing a randomized review of the Arabic typefaces used in each year of the al-Abhath journal. The page images of the journal were obtained from the Arabic and Middle Eastern Electronic Library (Project AMEEL) of Yale University Library.[8] It was determined that there were two basic typefaces in the al-Abhath journal archive, with the first typeface being much more prevalent than the second. Examples of these two typefaces can be seen in figures 1-2 (for comparison sake, the last word in both lines - on the left of the page - is the same word).

Typeface #1 volumes 1-33, 36-39, 48-50

Typeface #2 volumes 34-35, 40-47

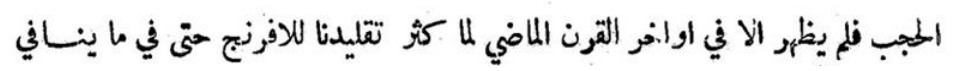
Both typefaces had some internal font differences and other minor character/script variations (e.g., patterns of use of alef hamza, slight shifts in placement of dots, slight differences in the degree of curvature of lines in a couple of instances, and minor ligature differences). This intra-typeface variation was especially apparent in typeface #1, which had a long run as al-Abhath's typeface. To address this issue it was decided that the best approach would be to produce approximately 5,000 lines of training data for the first typeface and 2,000 lines of training data for the second typeface.

After a randomized sample of the pages representing each typeface were selected, research assistants working with the OpenITI team produced the training

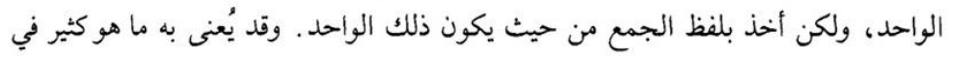
[6] More on Kraken's technical details can be found here: https://github.com/mittagessen/kraken.

[7] As a proprietary software the exact nature of the classifier is unknown.

[8] Project AMEEL's website is available here: http://www.library.yale.edu/ameeljournals/ameel_steps.html.



(a) Typeface #1 sample



(b) Typeface #2 sample

Figure 4.1: Sample of the two typefaces

data for these 7,000 lines using CorpusBuilder 1.0 (a new OCR postcorrection platform produced through the collaboration of OpenITI and Harvard Law School's SHARIAsource project).[9] After these 7,000 lines of training data were double checked for accuracy, a final spot review was conducted. This "gold standard" data was then utilized for model production and OCR.[10]

The first round of OCR accuracy tests were performed by an outside contractor, which JSTOR hired to conduct a ten-page manual accuracy comparison between the Kraken output and the corresponding output for ABBYY (see table 4.1).

With the exception of page #2, OpenITI (Kraken) performed substantially better on the pages the contractor reviewed, achieving >99% accuracy in 4/10 pages, >97% accuracy in 6/10 pages, >95.8% in 8/10 pages. The exception to these generally impressive numbers were pages #2 and #8 in which OpenITI (Kraken) only achieved 27.027% and 93.539% respectively. While the contractor's review was quite useful and generally confirmed OpenITI's results from its previous work (i.e., that Kraken achieves significantly higher accuracy rates on Arabic texts than the commercial OCR solutions for Arabic), the OpenITI team discovered upon further review that there were several problems with the contractor's study.[11]

First, Page #2 - by far the most disappointing result - is a highly atypical page of al-Abhath data. It only contains 37 characters total and much of these are contained in a large header that is in a highly calligraphic script and is heavily vocalized (with diacritics) (see figure 4.2). It is noteworthy that Abbyy performed better on this script, but this page is an extreme outlier in the data.

The second issue that we identified in the contractor's review was that they were marking certain differences between the original scans and OCR output as errors which were not true errors, and, in some cases, even marked some characters in the OCR output as errors that were not errors at all. For example, in the former case, they marked all numbers as errors in the OpenITI OCR output which

[9]For more on CorpusBuilder 1.0, please see: https://www.openiti.org/projects/corpusbuilder.

[10]This training data is available for reuse and can be found in the OCR Gold Standard Training Data repository on OpenITI's Github page: https://github.com/OpenITI/OCR_GS_Data.

[11]For OpenITI previous study on Kraken, please see: [7].

Table 4.1: Contractor's accuracy comparison of Abbyy and OpenITI (Kraken) OCR results

	Total number of characters	Abbyy character errors	OpenITI character errors	Abbyy character accuracy	OpenITI character accuracy
Page #1 (00010004_187997831.tif)	1230	270	38	78.049%	**96.911%**
Page #2 (00010004_187997832.tif)	37	15	27	**59.459%**	27.027%
Page #3 (00010031_187998459.tif)	3182	355	23	88.843%	**99.277%**
Page #4 (00010063_187999338.tif)	3157	327	29	89.642%	**99.081%**
Page #5 (00010129_188001031.tif)	3222	378	16	88.268%	**99.503%**
Page #6 (00010012.tif)	3259	326	75	89.997%	**97.699%**
Page #7 (00010030.tif)	2503	230	17	90.811%	**99.321%**
Page #8 (00010126.tif)	2631	252	170	90.422%	**93.539%**
Page #9 (00010127.tif)	2294	223	35	90.279%	**98.474%**
Page #10 (00010132.tif)	2296	243	96	89.416%	**95.819%**

were rendered as western Arabic numerals (e.g., 1, 2, 3) instead of as eastern Arabic numerals (e.g.,١,٢,٣) - a problem that was particularly prevalent on page #8 (thus at least partially responsible for OpenITI's comparatively lower accuracy rate on page #8). The OCR rendered them as western Arabic numerals instead of eastern Arabic numerals because we decided to merge western and eastern Arabic numerals into their universal numerical values in the OCR process and then represent that value in western Arabic numerals in the OCR output.[12] These differences, thus, are not true errors—their numerical value is correct—and their representation can be changed to eastern Arabic numerals if that is what users prefer. Another similar issue was discovered in the contractor's treatment of diacritics: they routinely marked correctly rendered words as incorrect if the word's original diacritics were not included in the OCR output. However, again, this difference in the original text and the OCR output is not a true error in transcription

[12]This practice of collapsing numeric values to their universal numerical value can be done for multiple reasons, but, in this particular case, one of our primary motivating factors was the fact that there were inconsistencies in the transcription practice of numbers in the training data.

because OpenITI has followed the practice (with one exception discussed further below) of not reproducing diacritics in its training data (for reasons elaborate below) and thus the fact that the diacritics were not rendered in OpenITI's OCR output is actually a sign that the Kraken OCR engine was functioning correctly. (This training data generation practice can be changed if the users desire, and given the results in OpenITI's larger accuracy study described below, this change may be advisable in the future, depending on the requirements of each individual user's use case.)

These problems in the contractor's approach to error designation led them to calculate lower accuracy estimates for OpenITI OCR output than it achieved in actuality - a problem that was particularly accentuated in the case of page #8, which contained a larger amount of numbers than the other pages the contractor reviewed. Due to the problems discovered in the contractor's initial accuracy study, JSTOR requested that OpenITI perform a more detailed accuracy assessment on approximately fifty pages.

Figure 4.2: Header

4.4 OpenITI Accuracy Study

The OpenITI team began by generating automatic character error rate (CER) reports for the al-Abhath data (see table 4.2 for full results).[13] In the first round of experiments, we built two different models—typeface model #1 and #2—based on the two different sets of training data produced for the two typefaces that we identified in the full run of al-Abhath. After extracting 1,000 lines of training data from the original 5,000 lines for typeface #1 and 700 lines of training data from the original 2,000 lines for typeface #2 to use as validation sets, we then trained the model on the remaining lines and tested these models' accuracy using the validation sets.[14] These accuracy results can be found in rows 2-3 in table 4.2. These typeface-specific models were the ones used to produce the OCR output that was transferred to JSTOR and that the contractor reviewed for their accuracy study.

In the time between the delivery of the al-Abhath OCR output to JSTOR and OpenITI's manual accuracy study (discussed below), we began developing a generalized Arabic model from all of the training data that OpenITI has produced

[13]The full CER reports can be found in the following OpenITI Github repository: https://github.com/OpenITI/OCR_GS_Data/tree/master/ara/abhath.

[14]This method of isolating a fixed number of lines of the training data as a validation set for automatic accuracy testing is a standard procedure when evaluating machine learning models.

over the last year two years (circa 15,000 lines).[15] (Generalized OCR models incorporate character features from all of the typefaces represented in the data upon which it is trained and therefore can often achieve higher levels of accuracy on a broader range of typefaces.) We decided to test this model on all of the al-Abhath data to determine if total OCR accuracy could be improved and, if so, by how much. The results, shown in row #4 of table 4.2, were impressive.[16] The generalized model's total character accuracy rate was 97.41% - a 2.57 percentage point improvement over the typeface #2-only model (i.e., a 50% improvement rate) and 1.45 percentage point improvement over the typeface #1-only model - and its Arabic script-only accuracy went up to a respectable 98.46%. The generalized model performed better than the typeface-specific models in all categories, but its most significant gains were in the category of "inherited" characters.

Table 4.2: Overview of OCR accuracy rates (drawn from character error rate (CER) reports

	Total character accuracy	**Arabic script only accuracy**	**Common character accuracy**[a]	**Inherited character accuracy**[b]
Typeface #1 Model	95.96%	97.56%	96.91%	79.67%
Typeface #2 Model	94.84%	97.11%	94.16%	85.18%
Generalized Model	97.41%	98.46%	96.36%	89.44%

[a]"Common characters" are characters shared by multiple scripts, primarily punctuation and other signs and symbols. In Arabic script, the kashīda or tatwīl (elongation character) is included in common script class.

[b]"Inherited characters" are characters, such as diacritics, that can be used on multiple languages and they only come to be defined in reference to the character with which they are combined (i.e., they "inherit" the script of the base character with which it is used).

According to the CER reports, the most significant source of errors in both the typeface #2 model and the generalized model were whitespace (spacing) errors and the Arabic diacritic, fatḥa tanwīn (unicode codepoint: Arabic fatḥatan). In the case of the typeface #1 model, whitespace errors were again the most significant source of errors, followed by kāf (ك), yā' (ي), and then fatḥa tanwīn errors. The hamza above (ء) character ranks as the seventh most common error in typeface #1 model and fifth in typeface #2 model. The mīm (م) character also is a common error in both the typeface #1 and #2 models, ranking as the sixth and fifth most common error in their CER reports respectively.[17]

[15]All of this gold standard training data can be found here: https://github.com/OpenITI/OCR_GS_Data.

[16]For this accuracy assessment, 2,096 lines of the 7,000 lines of training data were isolated as a validation set.

[17] Again, the full CER reports can be found in the following OpenITI Github repository: https://github.com/OpenITI/OCR_GS_Data/tree/master/ara/abhath.

Concurrent with the generation of CER reports, the OpenITI team began a far more expansive manual review of fifty—randomly selected—pages of the original OCR output produced by the typeface #1 and typeface #2-specific models. Each of these fifty pages were reviewed and then their error reports were collated into a master list of 1,096 total error instances.[18] Finally, each error instance was examined with an eye towards identifying possible factors in its adjacent context that may have led to that error and coded the error instances with any of the following categories that were applicable:

1. Poor scan quality: an element in the raw scan is unclear, or extraneous marks are present.
2. Ligature/atypical letter or dot form: connection between letters or placement of dots is in a less common form.
3. Diacritics: diacritics were present in original word.
4. Kashīda/tatwīl (elongation character): error appears in the context of a word that has been elongated.
5. Header/font alteration: bolded, italicized, or enlarged text.
6. Footnote: error appears in the context of a footnote.
7. Format: atypical format of presentation, e.g., table, list.
8. Hamza: mistranscribed character was a hamza or a hamza was present in the original word that was mistranscribed.
9. Doubled character: a single letter or number in the original scan was doubled in the OCR output.
10. Missed fatḥa tanwīn: fatḥa tanwīn in the original text was not transcribed.
11. Punctuation or other symbol: error was a punctuation mark or other symbol.
12. Non-Arabic language: original text was not Arabic.
13. Numbers: error was a number.
14. Superscript numerals: error was a footnote numeral in the body of the text.

[18] We use the term "error instance" here to highlight the fact that we are not exclusively recording individual, one-to-one character errors, but instances in the text in which one or more characters were read incorrectly. In most cases, this is a one-to-one character mistranscription, but in some other cases one character in the original was read as two or more in the OCR output or multiple characters in the original were read as one or none in the OCR output. In a few cases - discussed in more detail below -there are whole sections of text that are severely mistranscribed due to one or another feature in the original text.

This list of error codes is a mixture of error types (#8-14) and the most common recurring contextual features of the errors (#1-7). For categories of the latter type, it is important to emphasize that the presence of any of these contextual features near an error in the original text does not necessarily mean that it caused the error. But their repeated co-occurrence may be related and thus suggest future avenues of research and/or the need to better address this issue in the process of future training data production. We should also point out that in the case of some errors none of the following category codes were applicable, which only means that the reason for their improper rendering was not immediately evident to the human reviewers.

We do want to preface our presentation of the results of this manual review and error coding below with one further cautionary note. Manual evaluations are both essential and problematic: they provide far more detailed data (i.e., "thick data") about the OCR output and where OCR is failing, but they are much more time and labor intensive (and thus more limited in scope) and subject to human error. The results presented in table 4.3should be understood in this light. They should be understood as a snapshot of the human-inferable errors present in the OCR output. Each error type and possible ways to address it will be discussed in more detail in separate sections below table 4.3.

Table 4.3: Error coding for error instances in OpenITI manual OCR output assessment

Error code	**Quantity identified**
Poor scan quality	25
Ligature/atypical letter or dot form	182
Diacritics	90
Kashīda/tatwīl (elongation character)	31
Header/font alteration	113
Footnote	88
Format	14
Hamza	97
Doubled letter	209
Missed fatḥa tanwīn	91
Punctuation or other non-alphanumeric symbols	25
Non-Arabic language	70
Numbers	94
Superscript numerals	26

4.4.1 Doubled Letter

The "doubled letter" error type was the most frequent that we observed in the OCR output data (see example in figure 4.3).[19]

Figure 4.3: "Doubled Letter" errors

At first this error was perplexing. However, it was subsequently discovered that these "doubling" errors were an artifact of the decoding algorithm converting the sequence of confidences for each character produced by the neural network into a series of characters. As the network assigns each character a probability for each pixel-wide vertical slice of the input line image and printed characters are wider than a single pixel an algorithm is needed to extract the actual line text from the longer character probability sequence. Our implementation was based on a thresholding and merging approach which can cause doubled characters when the network assigns a probability below the threshold for a character at a vertical slice between high probability slices for the same character.[20] This error has been effectively addressed by switching to a greedy decoding which always uses the highest probability character at each vertical slice.

4.4.2 Header/Font Alteration, Footnotes, and Superscript Numerals

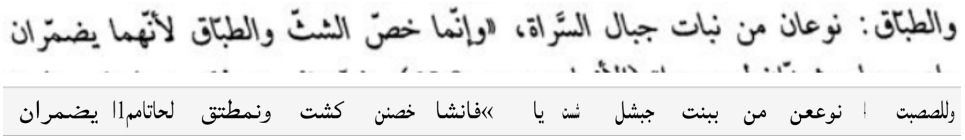


Figure 4.4: Example of poor transcription of italicized passage in figure 4.5

Errors that occured in the context of changes in the font (bolded, italicized, enlarged/decreased text size) represent the largest category of errors in the OCR output. Their total numbers are not even fully reflected in table 4.3 because examples in which whole sections (see examples in figures 4.5,4.4) were severely mistranscribed were not enumerated (character-by-character) in the error totals of the OpenITI manual assessment. Such sections were very rare, and in most other cases the OCR still rendered text with font alterations with a relatively high degree of accuracy, but font alterations do seem to increase error rates.

One of the most common examples of this issue was observed in text headers (including both section headers and chapter titles), which were typically bolded or

[19] In the images in figures 4.3 to 4.16 below the Arabic text at the top of the images is the original scan and the text below is the OCR output

[20] As a simplified example, assume the network assigns a probability for character x at 4 vertical slices: $(0.9, 0.95, 0.6, 0.9)$. Decoding with a threshold of 0.5 will result in an intermediate sequence xxxx that is then merged to x, while selecting a higher threshold of 0.7 will result in a potentially erroneous character sequence of xx merged from xx_x.

القزويني، ص 58).

معنى البيت ومؤدّاه:

حثحثوا: حثُّوا؛ الحصّ: واحدها الأحصّ، وهو الذي تناثر شعره أو ريشه وتكسّر؛ القوادم: من الريش، ما يلي الرأس؛ الخِشف: ولد الظبية؛ الشَّثّ والطبّاق: نوعان من نبات جبال السَّراة، «وإنّما خصّ الشثّ والطبّاق لأنّهما يضمّران راعيهما ويشدّان لحمهما» (الأنباري، ص 12:8). شبّه نفسه بذكر نعام ثم بظبية توافرت لهما دواعي السرعة وعدّتها، فقال: في مطاردة بجيلة إيّاي كنت سريعًا كظليم تكسّر ريش رأسه لشدة سرعته بمواجهة الريح، وكلّما خفّ ريش رأسه خفّ احتكاكه بالريح فكان أسرع له، هذا، أو كظبية مذعورة على ولدها مرعاها الشثّ والطبّاق، يضاعف من سرعتها دافع الخوف على خشفها وما تزوّدت به من غذاء مضمرٍّ.

Figure 4.5: Example of text in italics

bolded and enlarged in al-Abhath (see figure 4.6). In some headers, as mentioned above, an entirely different typeface was employed (see figure 4.2) – although this is a less common practice.

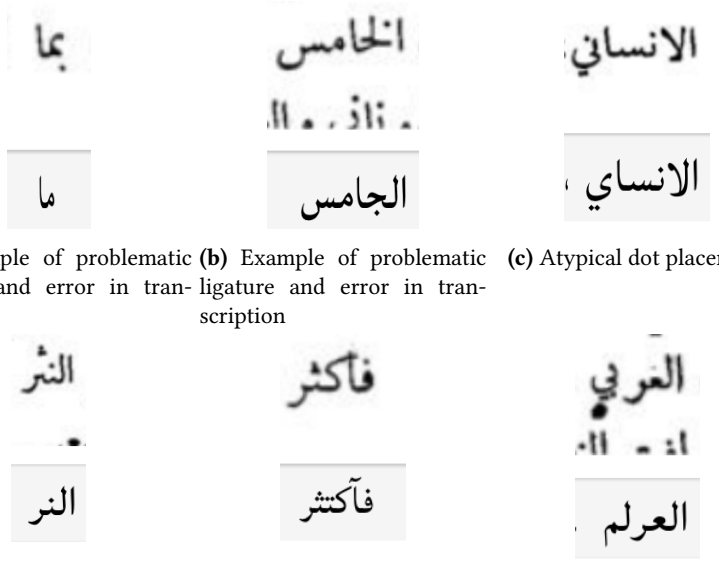
Figure 4.6: Bolded and enlarged text size header and poor transcription

Other modifications to the font of the typeface, e.g., footnotes, superscript (decreased text size) numerals, also seem to be correlated with decreased accuracy rates. In the future, this could be addressed by ensuring that a sufficient number of lines of training data with such font modifications is included.

4.4.3 Ligatures/Atypical Letter or Dot Forms

Not surprisingly, ligatures and other types of less common letter patterns and dot placements led to less accurate transcriptions in the OCR output (see figure 4.7). This same problem has been observed in an earlier study as well [7].

It is nearly impossible to completely avoid this problem, but a more systematic approach to training data generation that selected pages/lines of data with an eye



(a) Example of problematic ligature and error in transcription

(b) Example of problematic ligature and error in transcription

(c) Atypical dot placement

(d) Atypical dot placement

(e) Atypical letter pattern (printing error?)

(f) Atypical dot/letter placement and poor scan quality

Figure 4.7: Ligatures and uncommon letter patterns

towards ensuring sufficient representation of the maximum number of ligatures could improve OCR accuracy on these characters/character combinations.

4.4.4 Diacritics

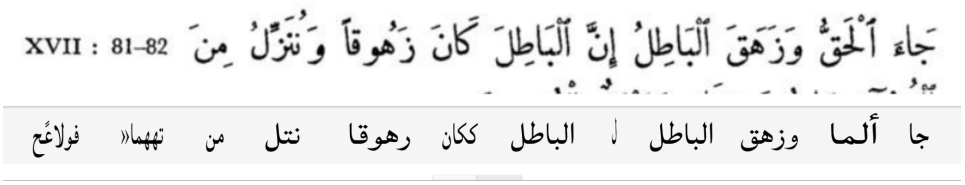
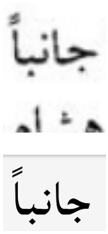


Figure 4.8: Highly vocalized Qur'anic passage that is transcribed poorly due to diacritics

Words that contained diacritics also appear more frequently to have errors in transcription, which leads us to believe that diacritics are interfering with character recognition. This tendency especially can be seen in examples of heavy vocalization, such as the fully vocalized Qur'anic passage seen in figure 4.8, which are poorly transcribed. Figure 4.8 is an extreme case that is an outlier in the al-Abhath data, but it clearly illustrates this problem. Moreover, although al-Abhath journal articles are not heavily vocalized, this could be a significant issue in other Arabic texts that are heavily vocalized.

In general, OpenITI has traditionally followed the practice of not transcribing Arabic diacritics in our training data production (with one exception discussed

(a) Missed fatḥa tanwīn



(b) Correctly transcribed fatḥa tanwīn from same page as figure 4.9a.

Figure 4.9: Transcription of fatḥa tanwīn

below). We have followed this practice for three reasons: (1) vocalization is often inconsistent and sometimes incorrect (so it is better to allow the individual scholar to determine the proper vocalization based upon their reading); (2) vocalization can interfere with computational textual analysis (computational linguists, for example, typically remove it in their normalization of texts in preparation for analysis); and, (3) not all full-text search algorithms support diacritics in a useful way. There is one problem with this approach, however, that we have found in both this study and another concurrent one on Persian OCR. If there is a sufficient amount of diacritics in the original text, the model will "learn" to ignore diacritical marks and it will not interfere with character recognition. However, if the original text is lightly vocalized and not enough examples of diacritics are contained in the training data, then it appears that the model does not "learn" well enough to ignore the diacritics and thus their presence in a word interferes with accurate character recognition. This situation presents us with a dilemma around which we need to develop a set of guidelines: we do not want to include diacritics because of the aforementioned reasons and because including them in the training data will require even more time expenditure in the training data generation process, but by not including them in texts with light vocalization (e.g., al-Abhath, some of the Persian texts in our other tests) character recognition is reduced in words with them.

4.4.5 Missed Fatḥa Tanwīn

The exception to our traditional treatment of diacritics discussed in the previous section is the case of the Arabic diacritic fatḥa tanwīn (ً). As observed in the CER reports, missed fatḥa tanwīns were a significant source of errors. We also observed this in the manual review of the OCR output (see figure 4.9a).

Although in the past we have not transcribed fatḥa tanwīns in the training data production process, we did include fatḥa tanwīns in the JSTOR pilot training data. In many cases the fatḥa tanwīns were transcribed correctly (see figure 4.9b for comparison sake). However, as both the CER reports and manual review showed, they still remained a relatively common source of errors. The

reason(s) that fatḥa tanwīn remained a problem in the transcription process could be related to either (1) its lack of sufficient representation in the training data, or (2) its position in the line segment—i.e., in some cases it might be partially getting cut off since it appears so high in the line segment box. In either case, we are inclined to ignore fatḥa tanwīns in future training data production.

(a) Missed hamza **(b)** Missed hamza on alif **(c)** Inserted extra hamza

Figure 4.10: Misrecognized hamzas

4.4.6 Punctuation Marks, Number, and Other Non-Alphanumeric Symbols

Punctuation marks, numbers, and other non-alphanumeric symbols (e.g., $) - especially representatives of each of these categories that were less commonly used in al-Abhath - were another recurring source of errors. The way to address this problem is by making sure these signs, symbols, and numbers are sufficiently represented in the training data.

4.4.7 Hamzas

The hamza character was another common source of errors in the output, both in the sense that it was misrecognized (see figure 4.10) and inserted in instances in which it was not in the original scan (see figure 4.10c).

Again, this is a case in which more focused training data will improve recognition rates—an intervention we must make at the training data generation phase of the OCR process.

4.4.8 Atypical Text Presentation Format and Kashīda/tatwīl (elongation character)

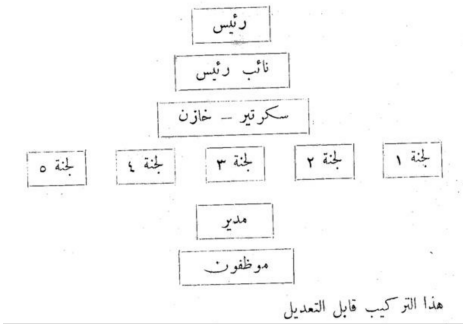
There are a series of errors that occur in the context of atypical presentation formats/atypical character patterns. These range from the use of the Arabic elongation character (kashīda/tatwīl) (see figure 4.11a) to various types of table formats (see figures 4.11b,4.11c,4.11d).

Although the character recognition in these examples is usually not as poor as in figure 4.11c, we still observed that errors seem to appear more frequently in such contexts (see the better recognition in figures 4.11a and 4.11d). More

training data from these atypical presentation formats and character patterns will help improve accuracy, but improvements in line segmentation are also necessary for such examples as figure 4.11c.

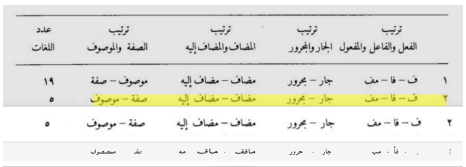
Full view of our OCR post-correction interface is shown in figures 4.11, 4.12 in order to show the broader page context from which the highlighted lines are drawn and displayed in a line pair (image of line and its digital transcription) in the pop-up. Specifically, please note that in figure 4.11d this line is drawn from a larger table and in figures 4.12, 4.13 this line is drawn from a page with significant admixture of both Arabic and Persian in the text.

(a) Read letter 'sin' into word due to kashīda/tatwīl (elongation).



(b) Example of table format

(c) Example of particularly poor transcription on an atypical (table) presentation format.



(d) Example of particularly poor transcription on another atypical (table) presentation format.

Figure 4.11: Atypical presentation forms

4.4.9 Non-Arabic Language

There were two significant types of transcription errors that were related to the presence of non-Arabic language in the original text. The first, seen in figure 4.12, is the poor transcription of non-Arabic characters on a page that predominantly contains Arabic text. (Figure 4.12 represents a particularly poor transcription of the non-Arabic text; most transcriptions in such instances were much more accurate.)

The second type of error that occurred in the context of non-Arabic script was the inverse: that is, poor transcription of Arabic text on a page that is predominantly composed of a non-Arabic language (see figure 4.13).

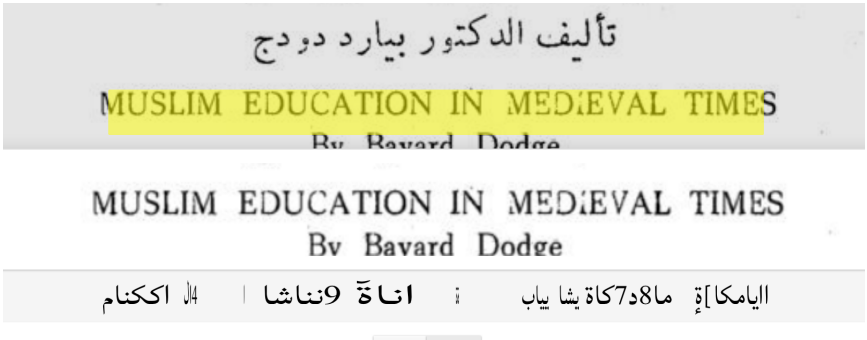


Figure 4.12: Example of particularly poor transcription of non-Arabic language in a page of primarily Arabic text.

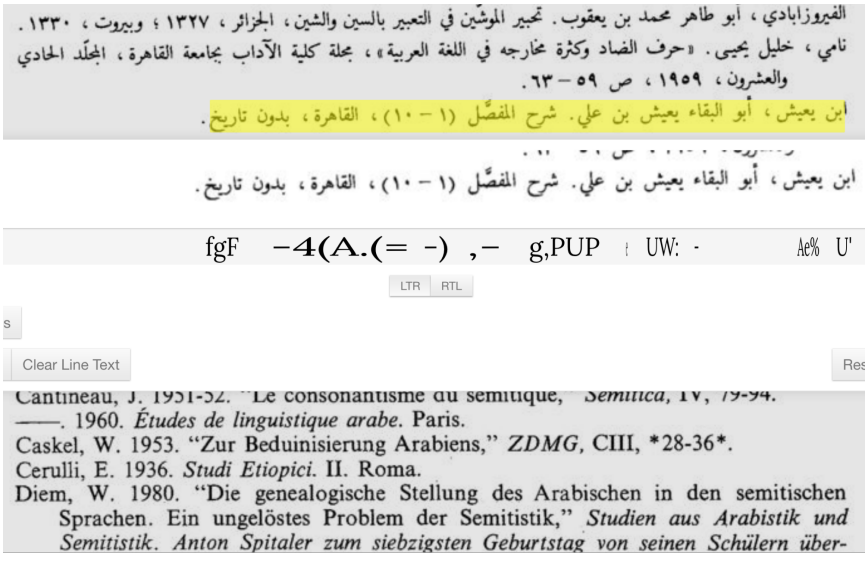
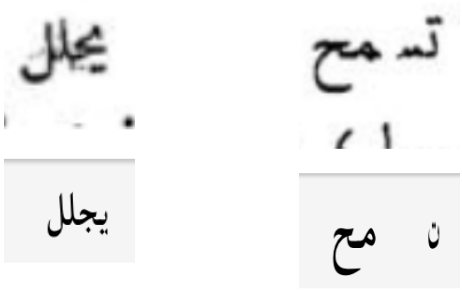


Figure 4.13: Page with substantial non-Arabic language interferes with Arabic OCR.

This is a known problem that can be addressed through the development of multi-language OCR models - a project that OpenITI is currently working on.

4.4.10 Poor Scan Quality



(a) Black shading in background of letters.

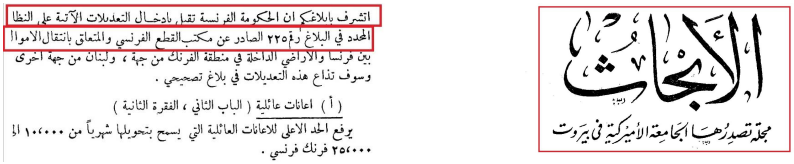
(b) Missing print in letter.

Figure 4.14: Examples of poor scan quality

Poor scan/print quality—including, errant marks (see figure 4.14a), lack of ink (see figure 4.14b), misplaced letters/punctuation (figure 4.7e) - is not a particularly common source of errors in the al-Abhath data, but there is a critical mass of errors caused by this problem.

This problem cannot be addressed in the OCR process. OCR accuracy is (obviously) limited by the quality of the original scans.

4.4.11 Line Segmentation



(a) Missed line segments (from outside contractor accuracy review).

(b) Large header segmented as one line (from outside contractor accuracy review).

Figure 4.15: Examples of missegmentation

One final error type that should be mentioned is line segmentation errors (see figures 4.15,4.16).

This type of error was not commonly found in the OpenITI manual accuracy assessment (figure 4.15 were errors identified in the outside contractor's review of the OCR output), but there were a few cases in which the line segmenter missed a section or a word of a line. Typically this would occur in atypical text presentation formats, such as the table seen in figure 4.16.

Figure 4.16: Line segmenter missed final word in the line.

Truncation of Arabic text lines is a known problem for the Latin-script-optimiz-ed line segmenter in the version of Kraken that was used for this study. The implementation of a novel trainable layout analysis method has largely solved this issue[3].

4.5 Recommendations and Future Avenues of Development for Open Source Arabic-script OCR

The results of this study indicate that work in the following three areas could generate significant improvements for open source Arabic-script OCR:

1. Systematic training data production. Instead of generating training data in a completely randomized (or haphazard) manner (as is often done),[21] future Arabic-script OCR projects need to study the particularities of the documents they plan to OCR and make sure that the pages selected for training data production contain a sufficient number of the less common ligatures, headers, diacritics, footnote text, numbers, and other particularities of the works to be OCR'd. This more systematic approach to training data production will require more time upfront. But the models produced in this manner could potentially achieve much higher baseline accuracy and reduce the burden of postcorrection.

2. Generalized models. One of the most exciting results from this study was the significant improvements in accuracy achieved with the generalized Arabic model. The success of this approach tentatively suggests that if we continue to add training data sets to this generalized model we can anticipate to achieve higher levels of accuracy on both typefaces on which we have already trained models and new typefaces for which we have no training data yet. If this pattern holds true in future studies, we would be able to gradually reduce the time and resources necessary to achieve high level accuracy (>97%) on new typefaces in the future. However, more research on generalized models is needed as both the optimal training data selection, including artificial data produced by methods such as [11], for such models and the actual variance on an open text corpus is currently unknown.

[21] We followed this randomized training data generation approach in the past [7]. See [10] for an example of a dataset resulting from haphazard convenience sampling, i.e. harvesting data from sources on which existing methods already produce near-perfect results.

3. There are a range of technical improvements—e.g., multi-language models, improved line segmentation and layout analysis—that could significantly improve OCR accuracy numbers. Efforts are currently underway in both the eScripta project (of which Kiessling is a team member) and OpenITI's Arabic-script OCR Catalyst Project (AOCP) to address each of these technical issues.[22]

Acknowledgements

We would also like to thank David Smith (Northeastern University) for his suggestions and feedback on this study. This work was supported by a subgrant from JSTOR from their National Endowment for the Humanities-funded feasibility study on high-quality digitization and digital preservation of Arabic scholarly journals (grant number PW-253861-17).

Competing Interests

This work was supported by a subgrant from JSTOR from their National Endowment for the Humanities-funded feasibility study on high-quality digitization and digital preservation of Arabic scholarly journals (grant number PW-253861-17).

Author Roles

Authors are listed in alphabetical order. The corresponding author is mtm.

Conceptualization bk, mtm

Data curation bk, gk, mtm, ks

Formal analysis bk

Funding acquisition mtm

Investigation bk, gk, mtm, ks

Methodology bk, mtm

Project administration mtm

Resources bk, mtm

Software bk

Supervision mtm

[22]For more information on the eScripta project, please see: https://escripta.hypotheses.org/. For more information on OpenITI's AOCP project, please see: https://openiti.org/projects/openitiaocp.

Validation bk

Visualization bk, mtm

Writing - original draft bk, mtm

Writing - review & editing bk, gk, mtm, ks

References

[2] M. Alghamdi and W. Teahan, "Experimental evaluation of arabic ocr systems," *PSU Research Review*, 2017.

[3] B. Kiessling, D. Stökl Ben Ezra, and M. T. Miller, "BADAM: A Public Dataset for Baseline Detection in Arabic-Script Manuscripts," in *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing, HIP@ICDAR 2019, Sydney, NSW, Australia, September 20-21, 2019*, ACM, 2019, pp. 13–18.

[4] C. Clausner, A. Antonacopoulos, N. McGregor, and D. Wilson-Nunn, "ICFHR 2018 competition on recognition of historical arabic scientific manuscripts - RASM2018," in *16th International Conference on Frontiers in Handwriting Recognition, ICFHR 2018, Niagara Falls, NY, USA, August 5-8, 2018*, IEEE Computer Society, 2018, pp. 471–476. DOI: 10.1109/ICFHR-2018.2018.00088.

[5] A. Keinan-Schoonbaert. "Results of the RASM2019 competition on recognition of historical arabic scientific manuscripts." (2019), [Online]. Available: https://blogs.bl.uk/digital-scholarship/2019/09/rasm2019-results.html (visited on 09/13/2019).

[6] ——, "Using transkribus for arabic handwritten text recognition." (2020), [Online]. Available: https://blogs.bl.uk/digital-scholarship/2020/01/using-transkribus-for-arabic-handwritten-text-recognition.html (visited on 01/20/2020).

[7] B. Kiessling, M. T. Miller, G. Maxim, S. B. Savant, *et al.*, "Important New Developments in Arabographic Optical Character Recognition (OCR)," *Al-ʿUṣūr al-Wusṭā*, vol. 25, pp. 1–13, 2017.

[8] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 369–376.

[9] B. Kiessling, "Kraken - A Universal Text Recognizer for the Humanities," *Proceedings of the DH*, 2019.

[10] U. Springmann, C. Reul, S. Dipper, and J. Baiter, "Ground truth for training OCR engines on historical documents in german fraktur and early modern latin," *J. Lang. Technol. Comput. Linguistics*, vol. 33, no. 1, pp. 97–114, 2018.

[11] T. Milo and A. G. Martínez, "A new strategy for arabic ocr: Archigraphemes, letter blocks, script grammar, and shape synthesis," in *Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, 2019, pp. 93–96.

Part II

Layout Analysis and Segmentation

Chapter 5

BADAM: A Public Dataset for Baseline Detection in Arabic-script Manuscripts

An updated version of the dataset that contains annotations for line orientation is available at https://doi.org/10.5281/zenodo.3274427. This chapter has been published as B. Kiessling, D. Stökl Ben Ezra, and M. T. Miller, "BADAM: A Public Dataset for Baseline Detection in Arabic-Script Manuscripts," in *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing, HIP@ICDAR 2019, Sydney, NSW, Australia, September 20-21, 2019*, ACM, 2019, pp. 13–18.

Abstract

The application of handwritten text recognition to historical works is highly dependant on accurate text line retrieval. A number of systems utilizing a robust baseline detection paradigm have emerged recently but the advancement of layout analysis methods for challenging scripts is held back by the lack of well-established datasets including works in non-Latin scripts. We present a dataset of 400 annotated document images from different domains and time periods. A short elaboration on the particular challenges posed by handwriting in Arabic script for layout analysis and subsequent processing steps is given. Lastly, we propose a method based on a fully convolutional encoder-decoder network to extract arbitrarily shaped text line images from manuscripts.

5.1 Introduction

Layout analysis as a major preprocessing step for text recognition is currently considered the limiting factor in the digitization of historical documents both handwritten and printed, especially so for non-Latin writing systems such as Arabic. With the rise of Digital Humanities and large scale institutional digitization projects a significant community of researchers engaged in the improvement of layout analysis on historical material has formed.

The most visible expression of this is a long-standing series of competitions evaluating either layout analysis in isolation [2]–[8] or as part of a larger text recognition task such as [9]. Unfortunately, these competitions concern themselves almost exclusively with Western texts written in Latin script despite some efforts to organize competitions on material that is insufficiently treated by current methods.

This euro- and anglocentric focus in document analysis research has changed to some extent recently. Although not directly connected to layout analysis [10] presented binarization, keyword spotting, and isolated character recognition challenges on Balinese palm leaf manuscripts. [11] included a layout analysis task on Arabic manuscripts but notably lacked a publicly available training dataset, except 15 representative images for informational purposes, and participation remained rather modest.

Recognizing that there is an obvious need for a large dataset of non-Western texts we propose a dataset based on one of the most geographically and chronologically extensive manuscript cultures, the Arabic and Persian one. This choice is motivated by multiple reasons: the exceptional size of the available material covering a wide range of topics and styles, complexity of layout rarely encountered in Latin manuscripts, and a large community of scholars working on Arabic-script manuscripts.

In addition, we strive to provide a dataset sufficient in size to support development of state-of-the-art machine learning approaches to layout analysis which

despite increasing popularity for Latin documents [12]–[14] has seen limited uptake for other writing systems.

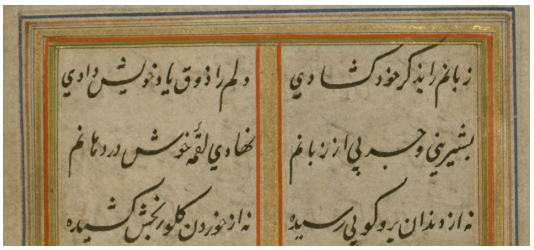
5.1.1 Related work

Existing layout analysis datasets capture text lines in a variety of data models. These range from polygons [11], [15], [16], to sub-word bounding boxes [17], down to explicit pixel labeling [4]. Some others such as [5], [9] also include extensive metadata such as reading order, text order, or full transcriptions.

(a) Expulsion of text into the margin

(b) Per-word slanted baselines

(c) Heaping of words at end of line



(d) Pseudo-columns in Persian poetry

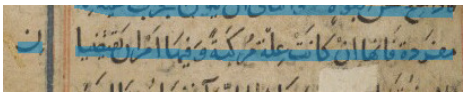
Figure 5.1: Aspects of Arabic-script handwriting

A new paradigm reducing text line segmentation to the successful detection of a continuous sequence of line segments has been established by the ICDAR 2017 Competition on Baseline Detection [8]. There are a plethora of benefits to this minimalistic model: better expression of highly curved baselines in comparison to bounding boxes, lower complexity of training data production than full polygons, easier modelling by semantic segmentation models because of object separability, and the existence of an evaluation scheme [18] that is more directly linked to real world recognition error rates than raw pixel accuracy.

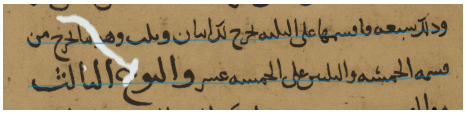
5.2 Dataset

The publicly available and freely licensed BADAM dataset contains 400 annotated scanned page images samples from four digital collections of Arabic and Persian language manuscripts.

5.2.1 Baselines and Arabic Typography

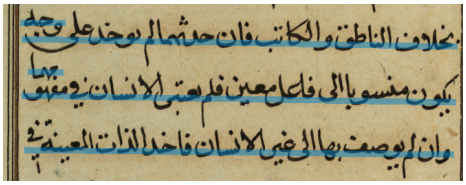


(a) Annotation of dislocated fragments in margin



(b) Holes in writing surface

(c) Per-word baseline annotation through imaginary baseline



(d) Separate annotation of heaped elements with complete overlap vs single baseline for partial overlap

(e) Joint annotation of half-verses as a single baseline

(f) Separated annotation of slanted half-verses

Figure 5.2: Examples of annotation guideline application (baseline indicated with opaque blue polyline)

A term arising chiefly from Western typography, the baseline is defined as the virtual line upon which most characters rest with descenders extending below.

While many Arabic handwritten texts present only a single baseline per logical text line a large number of documents, especially calligraphic works in Thuluth and Nastaliq style, display per word slanted baselines (figure 5.1b), multiple baseline levels, and dislocation of fragments into the margins or above other text in the line (heaping) (figures 5.1c and 5.1a). Most of these cases fulfill the purpose of text justification as hyphenation has been considered unacceptable in Arabic writing for the vast majority of the script's use.

As an additional complication, verses in Arabic poetry almost exclusively consist of two hemistichs, with the half-verse break forming pseudo-columns as shown in figure 5.1d. In some cases there is a combination of pseudo-columns and true multi-column text.

We therefore adopt a modified baseline definition that is oriented towards the current capabilities of text line recognition and reading order determination systems. Text lines are annotated with a single baseline extended through the majority of the line text, except in the cases of majority-overlap heaping (figure 5.2d) and dislocation into

the margin (figure 5.2a). In the case of slanted per-word baselines without horizontal overlap a baseline is drawn through an imaginary rotation point at each word (figure 5.2c). A baseline is split in multi-column text and at marginalia/main body boundaires. The hemistichs of poems are annotated as a single baseline per verse (figure 5.2e), except in the case of 45 degree slanted half-verses (figure 5.2f) that cannot easily be connected. In fragmentary material the baseline is continued through faded ink and split at holes in the writing surface (figure 5.2b).

These annotation guidelines amount to a conservative estimation of the capabilities of layout analysis systems, specifically their capacity to associate disconnected elements on the page belonging to the same logical line. It is relatively easy to extend the dataset with a more abstract data model that groups multiple baselines into a logical text line and we expect to do so in the future.

5.2.2 Data

42 manuscripts were randomly sampled from the collections of the Qatar Digital Library (15), the digital collection of the Walters Art Museum (13), the Beinecke Rare Book and Manuscript Library (6), and University of Pennsylvania Libraries manuscript collection (8). 10 single page images chosen were annotated for each manuscript with the labelme[1] image annotation tool with the exception of 4 shorter manuscripts from the Beinecke Library containing only 3 to 7 pages. Pages were selected manually for being representative of each work. Overall, there are 10770 lines in the corpus with a range of 3 to 176 lines per manuscript page ($\mu = 30.3, \sigma = 22.1$). The majority of the corpus is written in the Naskh style with the remainder being split between Thuluth, Nastaliq, and Kufic. Other regional styles such as ones used in Ottoman writing are currently absent.

A variety of writings is represented in the corpus:

1. Medical treatises including poetry with extensive marginalia
2. Works on logic, commentary on astronomy and arithmetic
3. Illuminated prayer books and religious texts
4. Texts on law such as legal glossaries
5. Illuminated poetic works in Persian and Arabic
6. Treatises on the legality and rules of chess including extensive diagrams and marginalia

The scan quality of the material varies according to the collection it was sourced from. While all are produced to a professional standard, the resolution varies considerably from 200dpi in the QDL, to 300 dpi in material from the Walters and Beinecke, and 500dpi at the University of Pennsylvania.

[1] https://github.com/wkentaro/labelme

A predefined random split into a 320 page training set and a 80 page test set is provided. The annotation is available in both PAGE XML and bit mask image formats. The corpus including both sampled images and ground truth is publicly released under CC-BY-SA 2.0 and available for download on the Zenodo[2] research data archive.

5.3 Baseline Method

Our method consists of two main stages: a pixel level classification of baselines followed by a lightweight baseline extraction step. We call this approach convolutional baseline layout analysis (C-BLLA).

In the first stage a fully convolutional encoder-decoder neural network is used to assign each pixel to a either background or baseline. The second stage is a script- and layout-agnostic postprocessing step operating on the heatmap produced by the neural network. Baselines are vectorized into polylines which are then used to extract rectified rectangular line image suitable for processing by an HTR line recognition system.

5.3.1 Pixel Labeling

The dense pixel-labelling of baselines is performed with a modified U-Net architecture [19]. U-Nets and similar fully convolutional networks [20] are state-of-the-art for general semantic segmentation tasks and have achieved excellent results on the cBAD dataset [8].

The backbone model consists of the first 3 blocks of a 34-layer ResNet in the contracting path followed by 4 3×3 convolution-transposed convolution blocks in the expanding paths with group normalization [21] ($G = 32$) and dropout ($p = 0.1$) employed after each layer and block respectively. A final 1×1 convolutional layer reduces the dimensionality of the input-sized 64-channel feature map to 1, followed by a sigmoid activation. An overall diagram of the network is shown in figure 5.4.

In order to improve generalization, the contracting path is pretrained on ImageNet classification and kept fixed during training of the upsampling blocks. Trainable layers are initialized using the He scheme [22]. We use the Adam optimizer with moderate weight decay ($\alpha = 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.999, w = 10^{-6}$) and early stopping on the binarized F1 score of the validation set. The network is trained on whole color images with the inputs being scaled to a size of 1200 pixels on the shortest edge to limit memory usage.

5.3.2 Baseline Estimation

The final sigmoid activation map has to be binarized prior to baseline vectorization. To suppress noise resulting in a higher number of skeleton branches caus-

[2] https://doi.org/10.5281/zenodo.3274428

Figure 5.3: 4 sample pages from the corpus

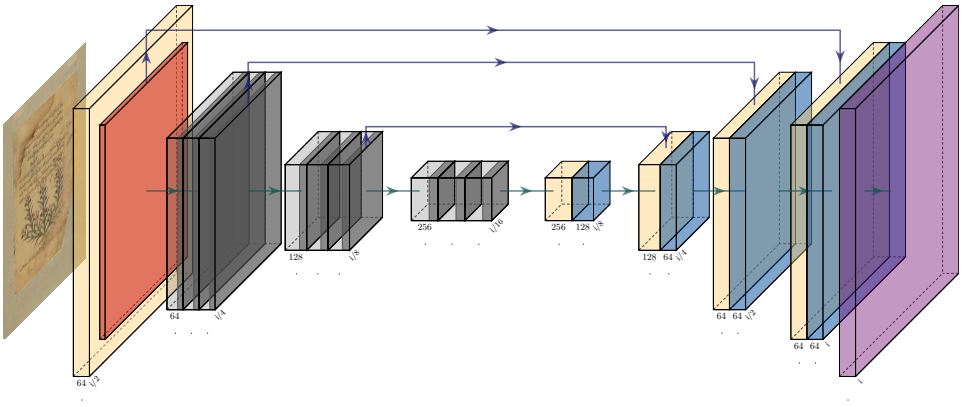


Figure 5.4: Architecture of the baseline labelling network. Dropout and batch/group normalization layers are omitted. (beige: convolutional layers + ReLU, red: max pooling, grey: ResNet blocks, blue: transposed convolutions, purple: convolution + sigmoid)

ing a slow down of end point calculation in the next step, the raw heatmap is smoothed with a gaussian filter ($\sigma = 1.5$) first, followed by binarization with hysteresis thresholding ($t_{low} = 0.3, t_{high} = 0.5$)

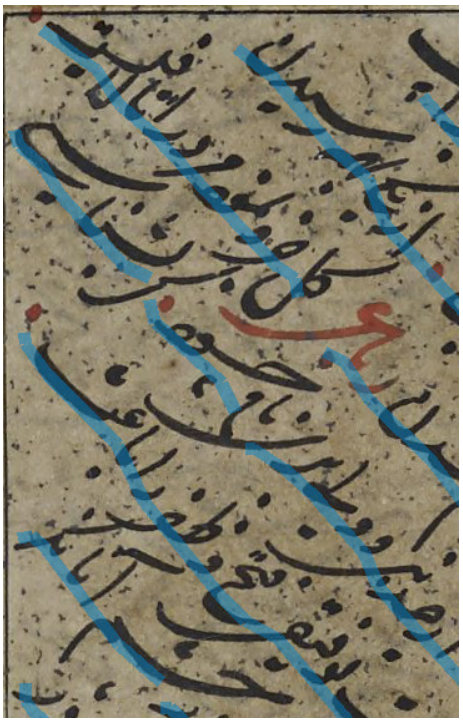
The binarized image is then skeletonized [23] and 1-connected end point candidates are extracted with a discrete convolution. As the skeleton often contains small branches, determining the actual end points of the centerline skeleton can be challenging. We treat all points along the skeleton as nodes in a graph and assume the true end points are the ones furthest apart on the skeleton. The actual baseline is thus the path of the maximum graph diameter of all possible candidate combinations. This path is then vectorized into a polyline with the Douglas-Peucker algorithm [24].

5.3.3 Line extraction

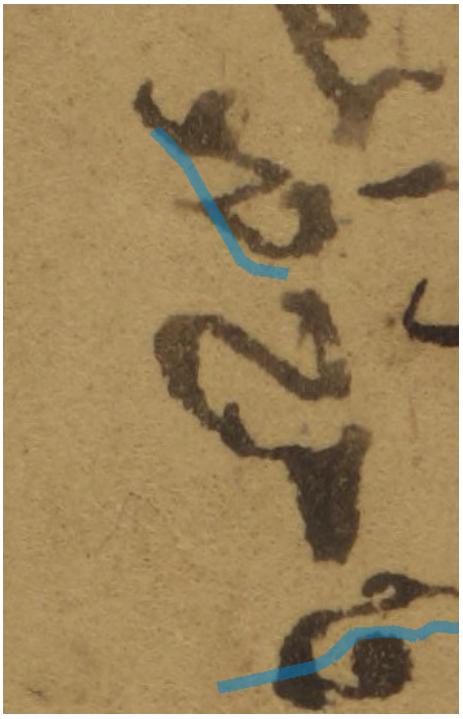
Vectorized baselines have to be converted into rectangular line images for classification by HTR recognition systems. Given that the baselines found by the system can be highly curved, even circular or spiral-formed, each polyline should be rectified by projecting its line segments and their respective environment consecutively onto a straight baseline.

For each line segment we compute an orthogonal vector of appropriate length including the desired area around the baseline determining the control points above and below the segment at each step. The rasterizations produced by Bresenham's line between both control points at each step are then appended to the rectified line.

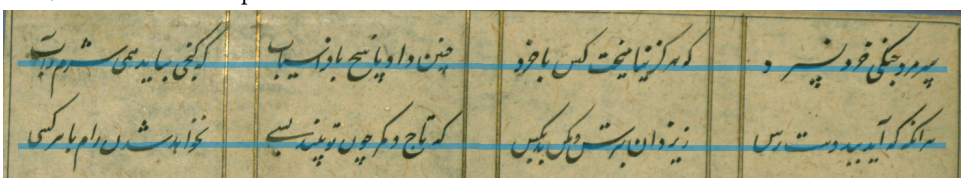
According to the results reported in [25] and our own verification on a typeset synthetic dataset the size of the environment extracted around the baseline is not crucial to recognition accuracy as long as the line contents are contained in the



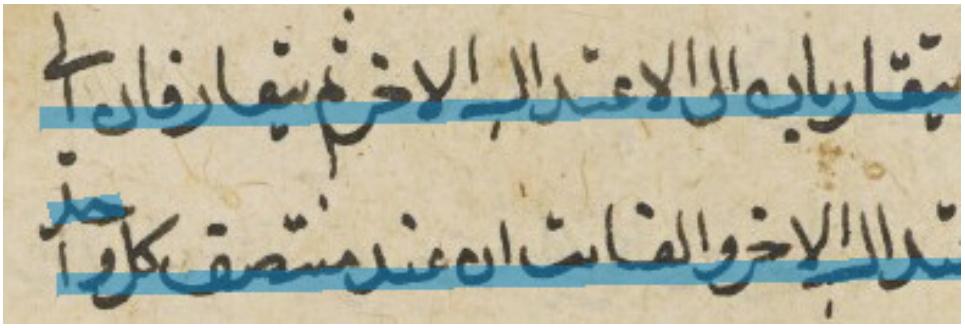
(a) Splitting of calligraphic writing in third/fourth line from top.



(b) Misrecognition of vertical text



(c) Incorrect splitting of logical 2-column poetry



(d) Missed heaped letter in top line, correct example on the bottom

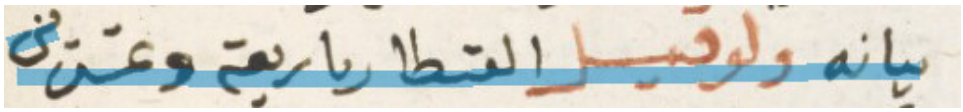
Figure 5.5: Common error modes of the LA system

rectified line image. We estimate the per-line environment by thresholding the input image with [26], calculating connected components under each baseline, and finding the maximum orthogonal distances of their edges above and below

the baseline.

5.4 Evaluation

We evaluated the proposed method on the 80 page test set using the method described in [18]. The results are shown in table 5.1. The metrics are slightly lower for our dataset than on the Latin cBAD dataset with a large gap in recall caused by a failure to extract heaped fragments (figure 5.5d) and vertical writing (figure 5.5b). On the other hand many missegmented lines are ornate or slanted (figure 5.5a), poetry (figure 5.5c) indicating that the network has not been able to learn a coherent model for these features on the dataset.



(a) Detection of lines containing differently colored keywords

(b) Detection of lines illuminated in gold

(c) Extraction of labels in a diagram

Figure 5.6: Strengths of the C-BLLA method

Apart from the higher flexibility of the baseline paradigm in comparison to older text line modelling approaches, C-BLLA has a number of other strengths.

The method is largely robust against changes in text coloration such as red keywords (figure 5.6b) or lines illuminated in gold (figure 5.6b) without the need for binarization of input images. Further it is able to ignore both border elements (support platforms, scales, and holding clips) and illustrations without an explicit text content extraction step. As illustration are processed jointly, textual labels can be detected albeit as a unstructured collection of lines (figure 5.6c).

There are some limitations to semantic segmentation in the context of layout analysis. These methods are inherently incapable of extracting overlapping and crossing text lines. This is exacerbated by the downsampling performed before inference which can cause inadvertent line merging in documents with closely written interlinear notes or commentary directly adjacent to main text.

The overall agreement in accuracy between the different datasets indicates that modern semantic segmentation methods can be employed for a wide variety of scripts when coupled with appropriate script-agnostic postprocessing. It remains to be seen if the accuracy gap between both datasets can be closed with general purpose systems that are not optimized for a particular set or if script-specific adaptations, such as specialized postprocessing, will be necessary.

Table 5.1: Results for the cBAD 2017 dataset and BADAM

	P-val	**R-val**	**F-val**
cBAD Simple Track			
BYU	0.878	0.907	0.892
dhSegment	0.943	0.939	0.941
ARU-Net	0.977	0.980	0.978
C-BLLA	0.944	0.966	0.954
BADAM			
C-BLLA	0.941	0.901	0.924

5.5 Conclusion

We presented a new dataset consisting of 400 annotated page scans of Arabic and Persian manuscripts spanning a wide range of topics and dates of production. Documents in the dataset present various degradations and large differences in the complexity of layout and writing styles. Many of the difficulties posed by them are specific to the Arabic script and should challenge the generalization power of even up-to-date layout analysis methods optimized for Latin script historical documents. While acknowledging that the annotation guidelines oriented on capabilities of current recognition algorithms will likely evolve in the future, our work contributes a solid foundation for comparable evaluation for document

analysis researchers.

In addition we describe a baseline system for line extraction from the corpus and evaluate its results, showing that even state-of-the-art methods have difficulties segmenting challenging Arabic handwriting as accurately as Latin manuscripts.

References

[2] B. Gatos, N. Stamatopoulos, and G. Louloudis, "ICDAR2009 handwriting segmentation contest," *International Journal on Document Analysis and Recognition*, vol. 14, no. 1, pp. 25–33, 2011. DOI: 10.1007/s10032-010-0122-8.

[3] A. Antonacopoulos, S. Pletschacher, D. Bridson, and C. Papadopoulos, "ICDAR 2009 Page Segmentation Competition," in *10th International Conference on Document Analysis and Recognition, ICDAR 2009, Barcelona, Spain, 26-29 July 2009*, IEEE Computer Society, 2009, pp. 1370–1374. DOI: 10.1109/ICDAR.2009.275.

[4] B. Gatos, N. Stamatopoulos, and G. Louloudis, "ICFHR 2010 Handwriting Segmentation Contest," in *International Conference on Frontiers in Handwriting Recognition, ICFHR 2010, Kolkata, India, 16-18 November 2010*, IEEE Computer Society, 2010, pp. 737–742. DOI: 10.1109/ICFHR.2010.120.

[5] A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher, "Historical document layout analysis competition," in *Document Analysis and Recognition (ICDAR), 2011 11th International Conference on*, IEEE, 2011, pp. 1516–1520.

[6] ——, "Icdar 2013 competition on historical newspaper layout analysis (hnla 2013)," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, IEEE, 2013, pp. 1454–1458.

[7] M. Murdock, S. Reid, B. Hamilton, and J. Reese, "Icdar 2015 competition on text line detection in historical documents," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, IEEE, 2015, pp. 1171–1175.

[8] M. Diem, F. Kleber, S. Fiel, T. Grüning, and B. Gatos, "cBAD: ICDAR2017 competition on baseline detection," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, IEEE, vol. 1, 2017, pp. 1355–1360.

[9] A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher, "ICDAR 2015 competition on recognition of documents with complex layouts - RDCL2015," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, IEEE, 2015, pp. 1151–1155.

[10] J.-C. Burie, M. Coustaty, S. Hadi, *et al.*, "ICFHR 2016 competition on the analysis of handwritten text in images of balinese palm leaf manuscripts," in *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, IEEE, 2016, pp. 596–601.

[11] C. Clausner, A. Antonacopoulos, N. McGregor, and D. Wilson-Nunn, "ICFHR 2018 competition on recognition of historical arabic scientific manuscripts - RASM2018," in *16th International Conference on Frontiers in Handwriting Recognition, ICFHR 2018, Niagara Falls, NY, USA, August 5-8, 2018*, IEEE Computer Society, 2018, pp. 471–476. DOI: 10.1109/ICFHR-2018.2018.00088.

[12] B. K. Barakat, A. Droby, M. Kassis, and J. El-Sana, "Text Line Segmentation for Challenging Handwritten Document Images using Fully Convolutional Network," in *16th International Conference on Frontiers in Handwriting Recognition, ICFHR 2018, Niagara Falls, NY, USA, August 5-8, 2018*, IEEE Computer Society, 2018, pp. 374–379. DOI: 10.1109/ICFHR-2018.2018.00072.

[13] L. Quirós, "Multi-task handwritten document layout analysis," *arXiv arXiv:1806.08852*, 2018.

[14] M. Fink, T. Layer, G. Mackenbrock, and M. Sprinzl, "Baseline Detection in Historical Documents Using Convolutional U-Nets," in *13th IAPR International Workshop on Document Analysis Systems, DAS 2018, Vienna, Austria, April 24-27, 2018*, IEEE Computer Society, 2018, pp. 37–42. DOI: 10.1109/DAS.2018.34.

[15] A. Fischer, V. Frinken, A. Fornés, and H. Bunke, "Transcription alignment of latin manuscripts using hidden markov models," in *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, ACM, 2011, pp. 29–36.

[16] F. Simistira, M. Seuret, N. Eichenberger, *et al.*, "DIVA-HisDB: A precisely annotated large dataset of challenging medieval manuscripts," in *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, IEEE, 2016, pp. 471–476.

[17] M. Kassis, A. Abdalhaleem, A. Droby, R. Alaasam, and J. El-Sana, "VML-HD: The historical arabic documents dataset for recognition systems," in *Arabic Script Analysis and Recognition (ASAR), 2017 1st International Workshop on*, IEEE, 2017, pp. 11–14.

[18] T. Grüning, R. Labahn, M. Diem, F. Kleber, and S. Fiel, "READ-BAD: A new dataset and evaluation scheme for baseline detection in archival documents," in *13th IAPR International Workshop on Document Analysis Systems (DAS)*, IEEE, 2018, pp. 351–356.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2015, pp. 234–241.

[20] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[21] Y. Wu and K. He, "Group normalization," *CoRR*, vol. abs/1803.08494, 2018. arXiv: 1803.08494.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, IEEE Computer Society, 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123.

[23] T.-C. Lee, R. L. Kashyap, and C.-N. Chu, "Building skeleton models via 3-D medial surface axis thinning algorithms," *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462–478, 1994.

[24] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[25] V. Romero, J. A. Sanchez, V. Bosch, K. Depuydt, and J. de Does, "Influence of text line segmentation in handwritten text recognition," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, IEEE, 2015, pp. 536–540.

[26] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.

Chapter 6

A Modular Region and Text Line Layout Analysis System

This chapter has been published as B. Kiessling, "A Modular Region and Text Line Layout Analysis System," in *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, 2020, pp. 313–318. The layout analysis method presented herein corresponds to the current implementation in the Kraken engine apart from slight changes in line end marker shapes and postprocessing.

Abstract

High quality document layout analysis is fundamental to the accurate processing of handwritten textual material, on both the level of individual lines and higher order zones demarking textual and non-textual content. We present an artificial neural network based approach to prediction of either that is implemented as part of a libre optical character recognition package and highly reconfigurable for a variety of tasks. Experiments on different openly available datasets show competitive results to state-of-the-art methods.

6.1 Introduction

Over the last decades tremendous amounts of historical handwritten documents have been digitized by archives, libraries, and other institutions engaging in the preservation of cultural heritage. Nevertheless the vast volume of scanned images, often with lack of metadata, results in the majority of this material being inaccessible in any meaningful way to scholars and the wider public. Optical Character Recognition[1] and Keyword Spotting aim to be technical solutions to the exploitation of large amounts of scanned textual data.

Current OCR systems operate largely on *line*-level data, i.e. the module in the OCR pipeline performing conversion into text does so one line image at a time. Therefore, a prior method is needed to extract these line images from whole document images. In addition, many documents require higher level understanding of how lines relate to each other for meaningful interaction. The usual way these higher level relations are modelled is through zoning, i.e. splitting a page into regions such as main text, marginalia, headings, illustrations, etc. Importantly, the nature of those regions can vary considerably between applications and material; they can often overlap, lines might extend across them or not be in any region at all. Consequently, text line extraction and region detection are arguably the most important part of an OCR system apart from the actual text recognizer.

As such, robust and accurate historical and handwritten document image analysis remains an open issue despite the recent advances facilitated by deep learning methods. Highly curved lines, variable orientation, interlinear notes, and multiple texts on the same page remain challenging to even state of the art layout analysis systems. Further, cultural bias in the conception of methods and data models continues to be a persistent problem: [2] shows the large amount of adaptation necessary to apply a seemingly script-neutral line model to Arabic manuscripts.

For our purposes we consider layout analysis along two principal axes. The *geometric axis* deals with the location, shape, and relations of found entitites, e.g. the by now obsolete character segmentation, text line extraction, and region de-

[1] As methods have converged considerably we do not distinguish between recognition of printed (OCR) and handwritten text (HTR)

tection. Text line extraction refers to the locating of individual text lines in the document images. In most modern LA systems text lines are the smallest unit of output, albeit for specialized tasks like scene text recognition subdivisions into words is also widespread. Region detection aims to find higher level, almost exclusively structural, zones, both textual and not, in document images.

The *semantic axis* concerns itself with the functional nature of detected entities, such as titles, illustrations, apparatus criticus, ... While not strictly necessary for most applications and often neglected outside of tools tailored for specific input data, enriching with semantic information can both boost raw metrics through allowing better incorporation of domain knowledge and aid in human understanding by improving output structuring, such as suppressing certain ancillary textual components.

Of note is that the focus of most methods is limited to a single or a subset of the tasks and axes. For example, no method could be found that allows semantic classification of both region and text line detection output simultaneously. In contrast, our method admits geometric and semantic classification on both text lines and regions while not requiring either.

Our method is implemented as part of a free OCR engine[2] which exposes the full customizability of the method's layout analysis features to end users. Hence, we are referring to the system as modular; it is possible to perform a wide range of tasks, ranging from simple text line extraction to highly specialized analysis like writing surface defect detection in a unified software package.

6.1.1 Related work

As a well established task in computer vision research a number of comprehensive surveys of document layout analysis exist [3]–[5].

6.1.2 Text line extraction

The capabilities of text line extraction methods in the literature is to some extent driven by existing datasets. A variety of formulations for line extraction can be found in published datasets. These range from polygons [6]–[8], to sub-word bounding boxes [9], down to explicit pixel labeling [10]. Some others such as [11], [12] also include extensive metadata like reading order, text order, or full transcriptions. A recent model [13] reduces text line detection to the extraction of baselines, i.e. imaginary polylines upon which the text rests or hangs from. These polylines in combination with a bounding polygon can be ingested by line-based text recognizers with minimal adaptation while at the same time requiring only modest effort for manual annotation, encouraging the creation of substantial training datasets for machine learning based methods.

The methods employed for text line extraction are just as varied as the the data models employed. [14], [15] use connected components combined with filtering

[2] `http://kraken.re`

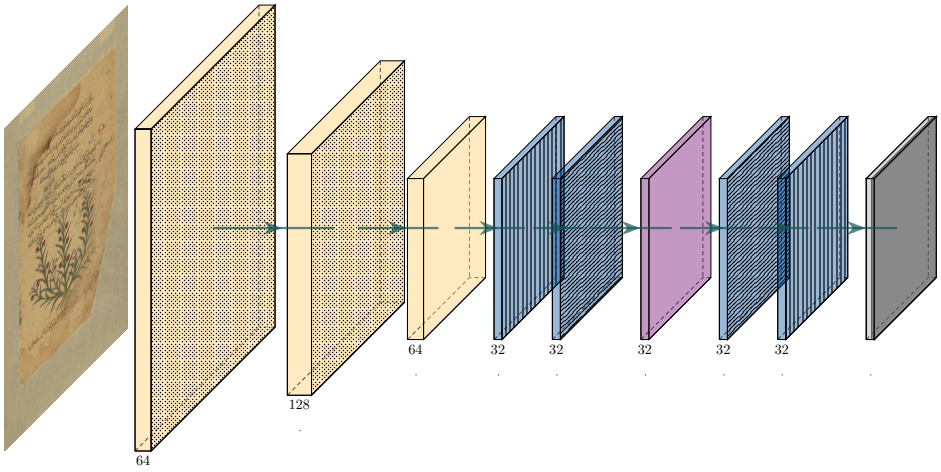


Figure 6.1: Architecture of the pixel labelling network. Group normalization layers are omitted. (salmon: 3x3 convolutional layers, dotting indicates dilation by 2x2; purple: 1x1 convolution, blue: bidirectional LSTM blocks, striping indicates row/column time axis; grey: 1x1 convolution with $|\tau|$ filters + sigmoid)

to perform pixel labeling. A common paradigm utilizes projection profiles in one way or another such as [16] for bounding box extraction, [17], [18] in combination with seamcarving for polygonal output, or RNN-based artificial projection profile generation for in-paragraph line splitting in [19]. A common drawback of the previously mentioned methods is that they operate on binarized input images which can be difficult to obtain for degraded historical material. [20]–[22] bypass this requirement through clustering of superpixels that can be obtained directly from color or grayscale image data to calculate polygons and baselines respectively. A number of deep learning based schemes have been proposed as well: [23]–[25] apply variants of the U-Net architecture for semantic segmentation.

6.1.3 Region detection

Region detection is almost always performed across both the geometric and semantic axis although they vary in the variety of zone labels they can yield. The most basic methods such as [26], [27] only distinguish between text and non-text regions while [28] can in principle be extended to all textual regions determinable solely by layout relations, and [24], [25], [29], [30] are able to distinguish arbitrary, non-overlapping regions with appropriate training data.

Like for text line extraction [24], [25], [29] variants of convolutional encoder-decoder networks are popular albeit pixel classifiers on handcrafted features [26], [27] exist. [28] performs clustering of text lines with convolutional conjugate graph networks. Definite clause grammars on a feature vocabulary as part of a

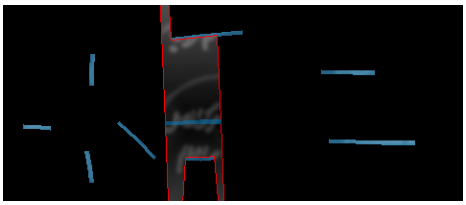
user-driven interactive segmentation system are shown in [31].

6.2 Method

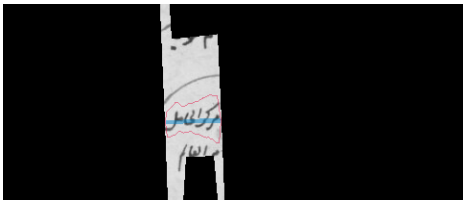
This section describes the proposed method for joint text line and region layout analysis. Our method can be divided into three main stages: multi-label pixel classification, baseline extraction and polygonization, and region extraction.

The first stage comprises of an Artifical Neural Network which outputs the probability of one or more classes (baselines, regions, and auxiliary classes) being present for each pixel of the input image. The second stage consists of the postprocessing extracting baselines from the auxiliary and baseline classes heatmaps, followed by a seam-carving step incorporating the original image to compute the bounding polygons required for inclusion of our method in a fully functional OCR pipeline. The final step extracts the regions from their respective class heatmaps through a contour finding algorithm. Notably, baselines are not restricted to regions, i.e. they can occur outside of regions and cross region boundaries.

(a) Ground truth overlay for a page with different line orientations (blue: baseline class, red: start_sep auxiliary class, green: end_sep auxiliary class)



(b) Region of Interests of the distance biased energy map for a sample line of the same image. (red: border of RoIs, blue: baselines)



(c) Computed seam/bounding polygon on the masked input image crop. (red: combined upper and lower half-seams, blue: baselines)

Figure 6.2: Examples of the data model and intermediate representations for a page from the BADAM [2] dataset

6.2.1 R-BLLA - Architecture

The overall pixel labeling network neural network is described in figure 6.1. Instead of conventional semantic segmentation encoder-decoder networks whose output is at the same scale as the input, our architecture decodes the learned representations at the downsampled scale of the last layer as the spatial information of regions and baselines can be recovered with sufficient

accuracy at this reduced resolution. This architecture roughly halves the memory requirements in comparison to an equivalent U-Net with a Resnet-50 backbone.

Our network is composed of a convolutional feature extractor, utilizing atrous convolutions (3×3 kernel size with 2×2 dilation, ReLU activation) to increase receptive field without increasing filter size or a more memory intensive deeper decoding network. This convolutional stack is followed by consecutive unidimensional LSTM layers as proposed for the ReNet architecture [32]. In this configuration the feature maps from the previous layers are swept by a bidirectional 1D LSTM layer in one direction (vertical or horizontal), followed by a second sweep over the output by a LSTM layer in the other direction, attaining similar performance to more complex multidimensional RNNs. The final decoding layer is a 1×1 convolution with $|\tau|$ filters and a sigmoid activation function that results in per class probability maps. Regularization is performed with group normalization ($G = 32$) [33] after each convolutional layer.

The output of the network is a stack of probability maps $\hat{y} \in \mathbb{R}^{w/n \times h/n \times |\tau|}$ for an input image $I \in \mathbb{R}^{w \times h \times c}$ with height h, width w, c channels, a downsampling factor n, and $|\tau|$ different classes $\{\text{start\_sep}, \text{end\_sep}, bl_0, \ldots, bl_k, reg_0, \ldots, reg_l\}$ for k and l different baseline and region types. The special classes start_sep and end_sep are placed at the beginning and end of each baseline respectively and serve two purposes. First, by explicitly encoding line bounds at locations where lines can be minimally separated such as multi-column texts we avoid inadvertent baseline merging during postprocessing. Second, introducing separate indicator classes for the beginning and end of a line allows the system to determine the orientation of lines. These auxiliary classes are shared across all possible baseline classes $\{bl_0, \ldots, bl_k\}$. As our method is intended to work with most scripts, including multi-script documents, the beginning and end of each line is not determined by the reading direction of the script. Instead we treat all scripts as canonically left-to-right, i.e when following the baseline from the start marker the upper part of the text line is always on the left-hand side (figure 6.2a). This scheme makes processing of arbitrarily oriented lines and mixed script pages with different reading direction without additional domain knowledge possible. The generated ground truth for the baselines classes are simple polylines drawn with a width. Regions are encoded as filled polygons. Thus the ground truth y is a multi-hot encoded tensor.

6.2.2 Training

The network is trained in a supervised manner with binary cross-entropy loss $L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} (y \cdot \log(\hat{y}_i) + (1 - y) \cdot \log(1 - \hat{y}_i))$. We adopt the Adam optimizer with moderate weight decay ($\alpha = 20^{-5}, \beta_1 = 0.9, \beta_2 = 0.999, w = 10^{-6}$). Input data are whole RGB color images scaled to a height of 1200 pixels.

In line with conventional practice, data augmentation is applied to the training set. With a probability of 0.5 a set of randomly parametrized transformations such as rotation, flipping, blurring, shifting, elastic transformations and hue changes

are applied to each image [34].

We train for a fixed number of epochs, per default 50.

6.2.3 Baseline vectorization

Baseline vectorization refers to the extraction of baselines from the probability map output of the model. This task consists of multiple substeps: superpixel calculation, triangulation filtering, and interpolation.

As the process is identical for each baseline type we define the output of the neural network for an arbitrary baseline type $H = \hat{y}_{:,:,n}, n \in \{bl_0, \dots, bl_k\}$. $P = \hat{y}_{:,:,\text{start\_sep}}, Q = \hat{y}_{:,:,\text{end\_sep}}$. Further we define a combined separator map $C = P + Q$.

In a first step we reduce the number of pixels to be considered for baseline clustering through calculating a subset T of all image pixels. Elements of this subset are called superpixels (SPs). Determining SPs is largely identical to the algorithm proposed in [35]. For an arbitrary probability map H the map is binarized with a threshold 0.2 producing H_b and skeletonized with a medial axis transformation that also returns a distance transform of H_b, resulting in the skeleton H_s and the average diameter d_{cc} of each uneroded baseline. All foreground pixels in H_s are projected onto H and sorted in descending order by their probability (S). T is iteratively filled by removing elements from T as long as their distance exceed a minimum ($d_{min} = 10$) from all other pixels in S.

Algorithm 1 Triangulation filter

Input: $DT(T), H, C$

$E = \emptyset$
for $e_{p,q} \in DT(T)$ **do**
  if $\mu(H, e_{p,q}) \geq 0.4 \wedge \sigma^2(H, e_{p,q}) \leq 0.05$ **then**
    if $\mu(C, e_{p,q}) \leq 0.125 \wedge \max(C, e_{p,q}) \leq 0.25$ **then**
      $E \leftarrow e_{p,q}$
    end if
  end if
end for
return E

The following step of the vectorization algorithm filters the Delaunay triangulation $DT(T)$ of T to subdivide it into a set of baseline clusters. An edge between two SPs $p, q \in DT(T)$ is denoted by $e_{p,q}$. As a prerequisite of the filtering algorithm we also define a number of edge metrics. Given the discrete line coordinates produced by a line drawing method $l(e_{p,q})$ between the SPs p, q we define for an arbitrary map $I \in \{H, P, Q\}$:

$$\begin{aligned}\mu(I, e_{p,q}) &= \frac{1}{|p-q|_2} \sum I[l(e_{p,q})] \\ \sigma^2(I, e_{p,q}) &= \frac{1}{|p-q|_2} \sum (I[l(e_{p,q})] - \mu(I, e_{p,q}))^2 \\ \max(I, e_{p,q}) &= \max(I[l(e_{p,q})])\end{aligned}$$

The output of the filtering algorithm Alg. 1 is a set of edges E defining a euclidean distance-weighted graph graph $G_E = G(V, E, w)$ where $V = \bigcup\{p, q\}, \forall e_{p,q} \in E, w(e_{p,q}) = |p-q|_2$, with a set of components C^{G_E}. Each component $C_n^{G_E}$ is treated as a separate baseline cluster. The remaining task is to create a directed polyline representation of each cluster. For each cluster we calculate the pairwise distances of all vertices and select the two most distant nodes a, b as the extrema of the baseline. The polyline approximation of each cluster is the shortest path $\gamma_{a,b}$ between the extrema in $C_n^{G_E}$. A slight correction of the line coordinates is necessary to compensate for the erosion incurred through the skeletonization prior to superpixel selection. The adjusted polyline path $\gamma_{a',b'}$ of $\gamma_{a,b}$ is obtained by elongating the initial and last edges by d_{cc}.

Due to the unknown orientation of each line we inspect each line end's affinity to the difference between the separator classes. As the separators are placed beyond the end of the line, the values of the separator maps at those points are commonly close to 0. By preprocessing P, Q with a maximum filter of size $2 \cdot d_{min}$ resulting in maps P', Q' containing sufficiently dilated separators the correct line orientation is such that:

$$L(\gamma_{a',b'}, P', Q') = \begin{cases} \gamma_{a',b'} & \text{if}(P' - Q')_p > 0.2 \wedge \\ & (Q' - P')_q > 0.2 \\ rev(\gamma_{a',b'}) & \text{if}(P' - Q')_p > 0.2 \wedge \\ & (Q' - P')_q > 0.2 \end{cases}$$

otherwise

$$L(\gamma_{a',b'}, P', Q') = \begin{cases} \gamma_{a',b'} & a'_x \leq b'_x \\ rev(\gamma_{a',b'}) & a'_x > b'_x \end{cases}$$

The final baselines for each baseline class is the set of all paths $\Gamma_m = \{\gamma_1, \ldots, \gamma_o\}, m \in \{bl_0, \ldots, bl_k\}, o \in \mathbb{N}$ determined as above.

6.2.4 Polygonization

For recognition by an HTR engine the vectorized baselines have to be supplemented by full polygons. A baseline with polygon can then rectified to produce a normalized line image with suppression of non-line content by projection onto a straight baseline through a piecewise affine transformation, allowing recognition of even highly curved lines by text recognition models.

Our polygonization algorithm consists of a line-wise seam carving [36] biased by distance from the baseline. The initial energy map is the derivative of the smoothed grayscale input image I^σ:

$$E(I^\sigma) = \left| \frac{\partial(I^\sigma)}{\partial x} + \frac{\partial(I^\sigma)}{\partial(y)} \right|$$

The primary purpose of the smoothing is to prevent the seam from crossing below disconnected line components such as diacritics and tonal marks. A gaussian filter with $\sigma = 2.5$ is sufficient for this purpose. Our implementation estimates the gradient with the Sobel operator.

Let $\{\Gamma_0, \dots, \Gamma_n, \dots, \Gamma_m\}, 0 < n < m, m \in \{bl_0, \dots, bl_k\}$ be the baselines of all classes and $\gamma \in \Gamma_n$ be an arbitrary baseline. To calculate the bounding polygon we first extract two regions of interest (RoI) $E(I^\sigma)[r_{\text{left}}]$ and $E(I^\sigma)[r_{\text{right}}]$ around γ; these RoIs contain the energy map area between the left- and righthand side of γ and $\{\Gamma_0, \dots, \Gamma_n \setminus \gamma, \dots, \Gamma_m\}$ as shown in figure 6.2b. The seams through r_{left} and r_{right} will form the respective halves of the bounding polygon.

Depending on the layout of the document the RoIs can vary considerably in size. Especially for baselines bordered only by the energy map boundaries, a distance bias has to be added to the energy map to ensure sufficiently tight boundary polygons. The biased energy map $E'(I^\sigma)[r_l], l \in \{\text{left}, \text{right}\}$ is computed through a euclidean distance transform D from the baseline with a scaling factor: $E'(I^\sigma)[r_l] = E(I^\sigma)[r_l] + D \cdot \overline{E(I^\sigma)[r_l]} \cdot 0.01$.

Requiring a rectangular area and principal direction for seam calculation, the RoIs need to be rotated. We rotate each RoI patch by the magnitude-weighted average direction. The energy-minimizing seam for each patch is then calculated using dynamic programming as described in [36]. Afterwards, the seams are rotated back into the original image coordinate system and concatenated to form the final bounding polygon for a line. Figure 6.2c shows the result for a single line.

6.2.5 Region extraction

Regions are extracted from the network output for each region type separately by thresholding at 0.5 and then extracting the contours around high-valued regions using the marching squares algorithm [37].

6.3 Evaluation

We evaluate the performance of the proposed method on 4 publicly available datasets: cBAD 2019[38], Bozen [39], OHG [40], and BADAM [2]. Bozen and OHG are Latin script datasets with both region and baseline annotations, cBAD consists largely of Latin script annotated on the baseline line, while BADAM is an exclusively Arabic script baseline dataset.

Table 6.1: Baseline recognition metrics on cBAD 2019, BADAM, OHG, and Bozen

	P-val	R-val	F-val
cBAD			
Planet	0.937	0.926	0.931
DMRZ	0.925	0.905	0.915
UPVLC	0.911	0.902	0.907
TJNU	0.852	0.885	0.868
DMRZ-2017	0.773	0.743	0.758
proposed	0.867	0.945	0.904
BADAM			
[2]	0.941	0.901	0.924
proposed	0.932	0.957	0.944
OHG			
[24]	0.962	0.971	0.966
[24][a]	0.984	0.977	0.980
proposed	0.978	0.973	0.975
proposed[a]	0.909	0.919	0.914
Bozen			
[24]	0.958	0.991	0.974
[24][a]	0.945	0.989	0.966
proposed	0.972	0.982	0.977
proposed[a]	0.936	0.949	0.942

[a]Combined region and baseline model

For datasets providing both region and line data models we evaluate models trained solely on baselines and combined baseline and region detection models.

6.3.1 Metrics

Baseline measurements for precision, recall, and F1-score are calculated as defined in [41] with the default tolerance parameters. For region segmentation the standard metrics mean accuracy, mean intersection-over-union, and frequency-weighted intersection over union are reported:

mean accuracy $(1/n_{cl}) \sum_i n_{ii}/t_i$

mean IU $(1/n_{cl}) \sum_i n_{ii}/(t_i + \sum_j n_{ji} - n_{ii})$

frequency weighted IU $\sum_k t_k)^{-1} \sum_i t_i n_{ii}/(t_i + \sum_j n_{ji} - n_{ii})$

Table 6.2: Metrics for the region detection task of the OHG and Bozen datasets

	Mean accuracy	**Mean Intersection-over-Union**	**frequency-weighted Intersection-over-Union**
OHG			
[24]	0.789	0.727	0.872
proposed	0.988	0.486	0.912
Bozen			
[24]	0.933	0.827	0.913
proposed	0.988	0.81	0.915

where n_{ij} is the number of pixels of class i predicted to belong to class j, where there are n_{cl} different region classes and $t_i = \sum_j n_{ij}$ is the total number of pixels of class i [42].

Two aspects of the proposed method are not evaluated as there are no available datasets or widely accepted metrics: the orientation of the baseline (orientation is disregarded by [41] and only one dataset contains rotated lines) and the polygonization. According to [43] the size of the environment extracted around the baseline is not crucial to recognition accuracy as long as the line contents are contained in the rectified line image.

Results are reported in table 6.1 and 6.2 for baseline and region detection respectively.

6.4 Conclusion

In this work we presented a flexible machine learning based method for text line and region layout analysis for historical documents including procedures for post-processing which enable its use in a typical OCR workflow without further adaptation. The experimental results show its competitiveness with the current state of the art on a number of historical document layout analysis benchmarks.

References

[2] B. Kiessling, D. Stökl Ben Ezra, and M. T. Miller, "BADAM: A Public Dataset for Baseline Detection in Arabic-Script Manuscripts," in *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing, HIP@ICDAR 2019, Sydney, NSW, Australia, September 20-21, 2019*, ACM, 2019, pp. 13–18.

[3] G. M. BinMakhashen and S. A. Mahmoud, "Document Layout Analysis: A Comprehensive Survey," *ACM Computing Survey*, vol. 52, no. 6, 109:1–109:36, 2020. DOI: 10.1145/3355610.

[4] S. Eskenazi, P. Gomez-Krämer, and J.-M. Ogier, "A comprehensive survey of mostly textual document segmentation algorithms since 2008," *Pattern Recognition*, vol. 64, pp. 1–14, Apr. 2017. DOI: 10.1016/j.patcog.2016.10.023.

[5] G. Nagy, "Twenty Years of Document Image Analysis in PAMI," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 38–62, 2000. DOI: 10.1109/34.824820.

[6] A. Fischer, V. Frinken, A. Fornés, and H. Bunke, "Transcription alignment of latin manuscripts using hidden markov models," in *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, ACM, 2011, pp. 29–36.

[7] F. Simistira, M. Seuret, N. Eichenberger, *et al.*, "DIVA-HisDB: A precisely annotated large dataset of challenging medieval manuscripts," in *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, IEEE, 2016, pp. 471–476.

[8] C. Clausner, A. Antonacopoulos, N. McGregor, and D. Wilson-Nunn, "ICFHR 2018 competition on recognition of historical arabic scientific manuscripts - RASM2018," in *16th International Conference on Frontiers in Handwriting Recognition, ICFHR 2018, Niagara Falls, NY, USA, August 5-8, 2018*, IEEE Computer Society, 2018, pp. 471–476. DOI: 10.1109/ICFHR-2018.2018.00088.

[9] M. Kassis, A. Abdalhaleem, A. Droby, R. Alaasam, and J. El-Sana, "VML-HD: The historical arabic documents dataset for recognition systems," in *Arabic Script Analysis and Recognition (ASAR), 2017 1st International Workshop on*, IEEE, 2017, pp. 11–14.

[10] B. Gatos, N. Stamatopoulos, and G. Louloudis, "ICFHR 2010 Handwriting Segmentation Contest," in *International Conference on Frontiers in Handwriting Recognition, ICFHR 2010, Kolkata, India, 16-18 November 2010*, IEEE Computer Society, 2010, pp. 737–742. DOI: 10.1109/ICFHR.2010.120.

[11] A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher, "ICDAR 2015 competition on recognition of documents with complex layouts - RDCL2015," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, IEEE, 2015, pp. 1151–1155.

[12] ——, "Historical document layout analysis competition," in *Document Analysis and Recognition (ICDAR), 2011 11th International Conference on*, IEEE, 2011, pp. 1516–1520.

[13] M. Diem, F. Kleber, S. Fiel, T. Grüning, and B. Gatos, "cBAD: ICDAR2017 competition on baseline detection," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, IEEE, vol. 1, 2017, pp. 1355–1360.

[14] N. Ouwayed and A. Belaïd, "A general approach for multi-oriented text line extraction of handwritten documents," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 15, no. 4, pp. 297–314, 2012.

[15] N. V. Borse and I. R. Shaikh, "Language independent text-line extraction algorithm for handwritten documents," *International Journal*, vol. 4, no. 11, 2014.

[16] M. Diem, F. Kleber, and R. Sablatnig, "Text line detection for heterogeneous documents," in *2013 12th International Conference on Document Analysis and Recognition*, IEEE, 2013, pp. 743–747.

[17] N. Arvanitopoulos and S. Süsstrunk, "Seam carving for text line extraction on color and grayscale historical manuscripts," in *2014 14th International Conference on Frontiers in Handwriting Recognition*, IEEE, 2014, pp. 726–731.

[18] R. Saabni, A. Asi, and J. El-Sana, "Text line extraction for historical document images," *Pattern Recognition Letters*, vol. 35, pp. 23–33, 2014.

[19] B. Moysset, C. Kermorvant, C. Wolf, and J. Louradour, "Paragraph text segmentation into lines with recurrent neural networks," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2015, pp. 456–460.

[20] A. Garz, A. Fischer, R. Sablatnig, and H. Bunke, "Binarization-free text line segmentation for historical documents based on interest point clustering," in *2012 10th IAPR International Workshop on Document Analysis Systems*, IEEE, 2012, pp. 95–99.

[21] B. Ahn, J. Ryu, H. I. Koo, and N. I. Cho, "Textline detection in degraded historical document images," *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, p. 82, 2017.

[22] T. Gruuening, G. Leifert, T. Strauss, and R. Labahn, "A robust and binarization-free approach for text line detection in historical documents," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, vol. 1, 2017, pp. 236–241.

[23] O. Mechi, M. Mehri, R. Ingold, and N. E. B. Amara, "Text line segmentation in historical document images using an adaptive u-net architecture," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 369–374.

[24] L. Quirós, "Multi-task handwritten document layout analysis," *arXiv arXiv:1806.08852*, 2018.

[25] S. A. Oliveira, B. Seguin, and F. Kaplan, "dhSegment: A Generic Deep-Learning Approach for Document Segmentation," in *16th International Conference on Frontiers in Handwriting Recognition, ICFHR 2018, Niagara Falls, NY, USA, August 5-8, 2018*, IEEE Computer Society, 2018, pp. 7–12. DOI: 10.1109/ICFHR-2018.2018.00011.

[26] M. Baechler and R. Ingold, "Multi resolution layout analysis of medieval manuscripts using dynamic mlp," in *2011 International Conference on Document Analysis and Recognition*, IEEE, 2011, pp. 1185–1189.

[27] K. Chen, H. Wei, J. Hennebert, R. Ingold, and M. Liwicki, "Page segmentation for historical handwritten document images using color and texture features," in *2014 14th International Conference on Frontiers in Handwriting Recognition*, IEEE, 2014, pp. 488–493.

[28] H. Déjean, J.-L. Meunier, *et al.*, "Versatile layout understanding via conjugate graph," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 287–294.

[29] Y. Soullard, P. Tranouez, C. Chatelain, S. Nicolas, and T. Paquet, "Multi-scale gated fully convolutional densenets for semantic labeling of historical newspaper images," *Pattern Recognition Letters*, vol. 131, pp. 435–441, 2020.

[30] P. Kaddas and B. Gatos, "A deep convolutional encoder-decoder network for page segmentation of historical handwritten documents into text zones," in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, 2018, pp. 259–264.

[31] A. Lemaitre, J. Camillerapp, and B. Coüasnon, "Multiresolution cooperation makes easier document structure recognition," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 11, no. 2, pp. 97–109, 2008.

[32] F. Visin, K. Kastner, K. Cho, *et al.*, "ReNet: A recurrent neural network based alternative to convolutional networks," *CoRR*, vol. abs/1505.00393, 2015. arXiv: 1505.00393.

[33] Y. Wu and K. He, "Group normalization," *CoRR*, vol. abs/1803.08494, 2018. arXiv: 1803.08494.

[34] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, *et al.*, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020, ISSN: 2078-2489. DOI: 10.3390/info11020125.

[35] T. Grüning, G. Leifert, T. Strauß, J. Michael, and R. Labahn, "A two-stage method for text line detection in historical documents," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 22, no. 3, pp. 285–302, 2019.

[36] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," in *ACM SIGGRAPH 2007 papers*, 2007, 10–es.

[37] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[38] D. Markus, K. Florian, and G. Basilis, *ICDAR 2019 Competition on Baseline Detection (cBAD)*, Zenodo, Feb. 2019. DOI: 10.5281/zenodo.3568023.

[39] A. Toselli, V. Romero, M. Villegas, E. Vidal, and J. Sánchez, *Htr dataset icfhr 2016*, Zenodo, Feb. 2018. DOI: 10.5281/zenodo.1164045.

[40] L. Quirós, V. Bosch, L. Serrano, A. H. Toselli, and E. Vidal, "From hmms to rnns: Computer-assisted transcription of a handwritten notarial records collection," in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, 2018, pp. 116–121.

[41] T. Grüning, R. Labahn, M. Diem, F. Kleber, and S. Fiel, "READ-BAD: A new dataset and evaluation scheme for baseline detection in archival documents," in *13th IAPR International Workshop on Document Analysis Systems (DAS)*, IEEE, 2018, pp. 351–356.

[42] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[43] V. Romero, J. A. Sanchez, V. Bosch, K. Depuydt, and J. de Does, "Influence of text line segmentation in handwritten text recognition," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, IEEE, 2015, pp. 536–540.

Chapter 7

Script and Emphasis Detection using Recurrent Neural Networks

This chapter has been published as B. Kiessling, D. Kinitz, C. Gümmer, and P. Mashhadi, "Script and Emphasis Detection using Recurrent Neural Networks," READ2018: International Interdisciplinary Symposium on Reading Experience and Analysis of Documents, 2018

7.1 Introduction

In Digital Humanities research documents containing multiple scripts and extensive text emphasis for semantic purposes are common, ranging from relatively simple parallel texts, to mixed Fraktur-Antiqua printing, dictionaries, and library catalogs. With the increased availability of Optical Character Recognition software at least in part accessible to the determined DH scholar robust script and text emphasis detection methods are of special importance for effective digitization of these works.

State of the art neural sequence-to-sequence models have largely supplanted older character-based methods for Optical Character Recognition. While neural methods have generally higher accuracy and decreased requirements on training data annotation depth, some earlier approaches, most notably the tesseract OCR engine [2], featured seamless classifier combination and common text emphasis detection. Neither are available in any freely licensed OCR package utilizing the advances of machine learning in the last decade.

7.1.1 Related work

Past approaches to segmentation-less multilingual OCR have focused on building combined models capable of recognizing multiple scripts [3]. Combined models are undesirable for multiple reasons. The irregularity of early modern printing and large number of typefaces result in character accuracy below 95% for mixed-font models even on mono-script texts [4] necessitating time consuming training data acquisition and retraining of these large models. In addition, reusing training data is regularly prevented by being embedded in other non-target scripts or typefaces and legal restrictions imposed by digitization agents..

A second direction labels OCR input images, most often lines, to be able to select appropriate monolingual recognition models.

The method described in [5] labeling whole lines using a recurrent neural network is inappropriate for many humanities texts because of extensive intra-line script switching. A recurrent neural script classifier based on overlapping sliding window profile feature sequences is shown in [6]. [7] published a conceptually simpler approach without feature extraction directly classifying character script using an LSTM network. A refined version of the latter method is the basis of our script detection system.

7.2 RNNs for Script and Emphasis Detection

7.2.1 Script Detection

The system treats script detection as a segmentation-less sequence classification problem, similar to text recognition. Instead of assigning a unique label to each code point or grapheme cluster we assign all code points of a particular script the

same label (figure 7.1), the network is trained to output the correct sequence of script labels using the CTC loss function [8]. It should be noted that CTC is on the face unsuitable for this task, as it includes no mechanism to ensure temporal alignment between graphemes in the input sequence and output activations; fortunately the LSTM network's activations are fairly close to their corresponding location in the input sequence. The output sequence is then used to split the line into single-script runs that can be classified with monoscriptual recognition models.

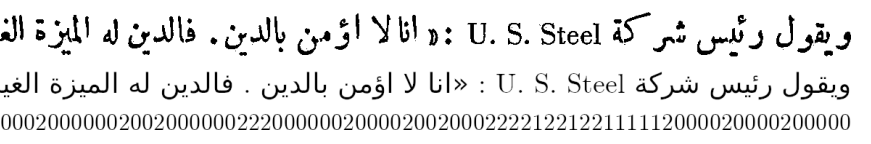


Figure 7.1: Modified ground truth (top: original line, middle: transcription, bottom: assigned script classes)

Script classes are ISO 15924 codes determined through each code point's Unicode script property[1] As there are graphemes that occur in multiple scripts, chiefly numerals and punctuation, we retain both the common and inherited properties. Merging these during post-processing based on their surrounding script increases robustness when classifying non-body text such as page numbers and tables, compared to fusing them beforehand. Bidirectional text is dealt with by rearranging the target sequence into display order using the Unicode BiDi algorithm before script assignment.

Apart from the mentioned merging step, two additional post-processing steps are performed. The first substitutes all individual runs of a line with the whole line if only a single script remains after common/inherited merging. The second stems from the observation that often only a subset of scripts the detection network is trained on occur in any document. A whitelist is added, merging runs of non-included scripts into the surrounding context after filtering for common confusions (Arabic-Syriac and Latin-Fraktur).

7.2.2 Emphasis Recognition

We evaluated two methods of encoding two common text emphasis methods for recognition by a RNN. Initially, italicized and text components with increased letter spacing were marked up with special start and stop markers and the model was trained to produce these markers. While the results of the training were promising, obtaining the amount of training data needed to reliably place both markers was infeasible for our target documents. Creating separate labels for italicized/spaced graphemes and training for these, remedied the marker placement issue with a sufficiently small amount of training data.

[1] ISO 15924 includes separate identifiers for Antiqua and Fraktur texts and similarly visually distinct calligraphic hands for Syriac which are subsumed as Latin and Syriac in the Unicode database.

Separate alphabets for emphasized text components increase model size and execution time, tripling the size of the final fully connected layer. This large increase in possible output labels also seems to preclude fine-tuning base models by resizing the final linear projection of the network.

7.2.3 Architecture

Both the script detection and emphasis recognition share a common network architecture of bidirectional Long short-term memory RNN blocks trained with Connectionist Temporal Classification loss and single-sample stochastic gradient descent with momentum (learning rate: 0.0001, momentum: 0.9). Early stopping is used to terminate training. The system is implemented as part of the kraken OCR engine.

The networks operate on binarized whole lines. Baselines and line height are normalized using a slightly modified version of the centerline normalizer implemented in the OCRopus system.

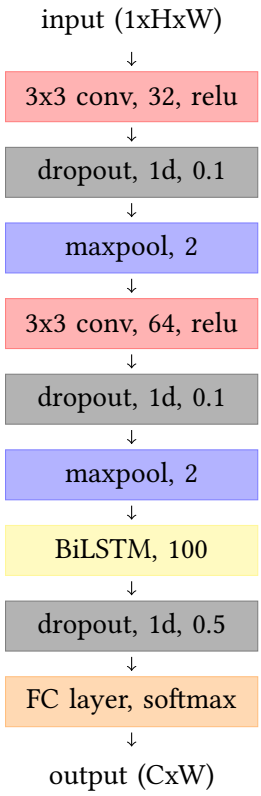


Figure 7.2: Network architecture (H: sequence height, W: sequence length, C: alphabet size)

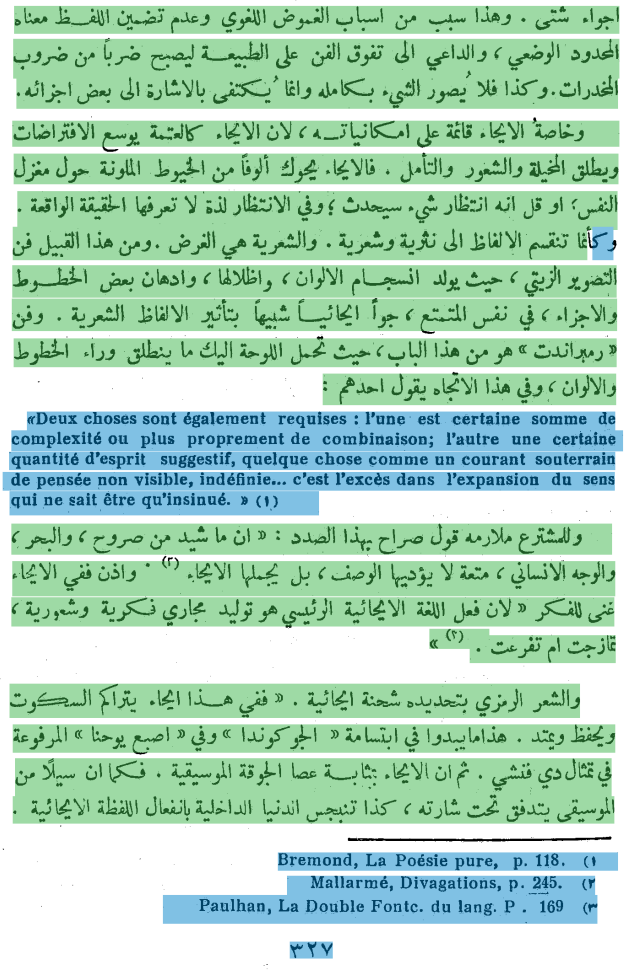
7.3 Results

The script detection and emphasis recognition were evaluated as part of Bibliotheca Arabica which aims to gain new insights into Arabic literature from 1150 to 1850 CE by analysing the ways of production, transmission, and reception of texts. The basis of this research are ~500 library manuscript catalogs which are usually multilingual and employ structured text emphasis as semantic markup.

7.3.1 Dataset

We repurposed publicly available non-synthetic training data for recognition models to build a corpus of 76000 script-annotated line images containing Arabic, Cyrillic, polytonic Greek, Hebrew, Latin, Fraktur, and (western) Syriac text. The majority of text lines contain only a single non-common script although there are mixed lines for all scripts in the corpus. The exact distribution of code points is shown in figure 7.4. 760 randomly selected lines are separated from the corpus as a test set.

Emphasis recognition was evaluated on an English and romanized Arabic catalog using emphasis described in 7.2.2 with 350 transcribed lines. An additional fifty line transcriptions were used as a test set. Overall 220 lines contain some kind of text emphasis. It is representative of a large number of catalogs in purview of Bibliotheca Arabica.



«Deux choses sont également requises : l'une est certaine somme de complexité ou plus proprement de combinaison; l'autre une certaine quantité d'esprit suggestif, quelque chose comme un courant souterrain de pensée non visible, indéfinie... c'est l'excès dans l'expansion du sens qui ne sait être qu'insinué. » (١)

Bremond, La Poésie pure, p. 118. (١
Mallarmé, Divagations, p. 245. (٢
Paulhan, La Double Fontc. du lang. P . 169 (٣

٣٢٧

Figure 7.3: Script recognition on French-Arabic sample page

7.3.2 Script Detection

The fully trained network yielded a character accuracy of 94.62% on the test set. Output for a French-Arabic bilingual sample page can be seen in 7.3. The misclassification of Eastern Arabic numerals as Latin text is caused by the transcription as Latin Arabic numerals in the ground truth.

7.3.3 Emphasis Recognition

The average character accuracy of the trained model over 10 runs is 99.3% (σ = 0.16) with 95.38% on cursive and text with increased spacing (σ = 1.46). When using only emphasized text accuracy as the stopping criterium mean accuracy rises to 99.03% (σ = 0.28).

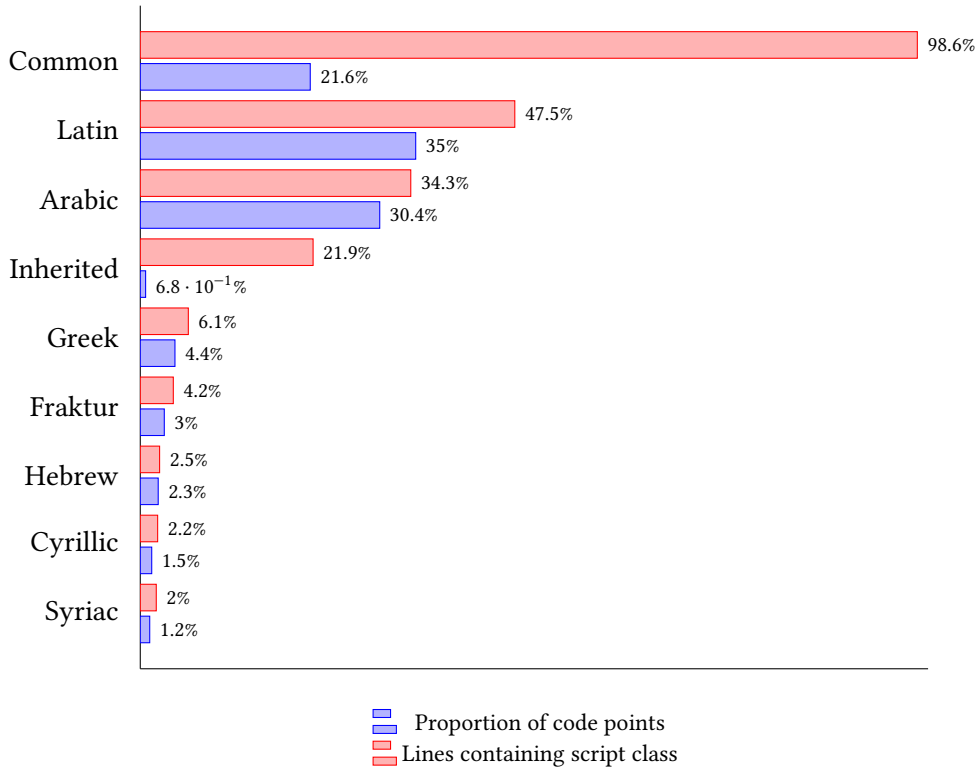


Figure 7.4: Script detection training data distribution

References

[2] R. Smith, D. Antonova, and D.-S. Lee, "Adapting the tesseract open source OCR engine for multilingual OCR," in *Proceedings of the International Workshop on Multilingual OCR*, ACM, 2009, p. 1.

[3] A. Ul-Hasan and T. M. Breuel, "Can we build language-independent OCR using LSTM networks?" In *Proceedings of the 4th International Workshop on Multilingual OCR*, ACM, 2013, p. 9.

[4] U. Springmann, F. Fink, and K. U. Schulz, "Automatic quality evaluation and (semi-) automatic improvement of mixed models for OCR on historical documents," *CoRR*, vol. abs/1606.05157, 2016. arXiv: 1606.05157.

[5] Y. Fujii, K. Driesen, J. Baccash, A. Hurst, and A. C. Popat, "Sequence-to-label script identification for multilingual ocr," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, IEEE, vol. 1, 2017, pp. 161–168.

[6] A. K. Singh and C. Jawahar, "Can RNNs reliably separate script and language at word and line level?" In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, IEEE, 2015, pp. 976–980.

[7] A. Ul-Hasan, M. Z. Afzal, F. Shafait, M. Liwicki, and T. M. Breuel, "A sequence learning approach for multiple script identification," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, IEEE, 2015, pp. 1046–1050.

[8] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 369–376.

Part III

Transcription and Character Alignment

Chapter 8

Kraken - a Universal Text Recognizer for the Humanities

This chapter has been published as B. Kiessling, "Kraken - A Universal Text Recognizer for the Humanities," *Proceedings of the DH*, 2019. This high level overview of Kraken's state in mid-2019 is complemented by technical documentation on it current state in appendix B.

8.1 Introduction

Retrodigitization of both printed and handwritten material is a common prerequisite for a diverse range of research questions in the humanities. While optical character recognition on printed texts is widely considered to be fundamentally solved in academia, with the most commonly used paradigm [2] dating back to 2006, this hasn't translated into increased availability of adaptable, libre-licensed OCR engines to the technically inclined humanities scholar.

The nature of the material of interest commands a platform that can be altered with minimum effort to achieve optimal recognition accuracy; uncommon scripts, historical languages, complex or archaic page layout, and non-paper writing surfaces are rarily satisfactorily addressed by off-the-shelf commercial solutions. In addition, an open system ameliorates the severe resource constraints of humanities research by enabling sharing of artifacts, such as training data and recognition models, inaccessible with proprietary OCR technology.

8.2 Kraken

The Kraken text recognition engine is an extensively rewritten fork of the OCRopus system. It can be used both for handwriting and printed text recognition, is easily (re-)trainable, and great care has been taken to eliminate implicit assumptions on content and layout that complicate the processing of non-Latin and non-modern works.

Thus Kraken has been extended with features and interfaces enabling the processing of most scripts, among them full Unicode right-to-left, bidirectional, and vertical writing support, script detection, and multiscript recognition. Processing of scripts not included in Unicode is also possible through a simple JSON interface to the codec mapping numerical model outputs to characters. The same interface provides facilities for efficient recognition of large logographic scripts.

Output includes fine-grained bounding boxes down to the character level that may be used to quickly acquire a large number of samples from a corpus to assist in paleographic research. Kraken implements a flexible output serialization scheme utilizing a simple templating language. Templates are available for the most commonly used formats ALTO, hOCR, TEI, and abbyyXML.

While including implementations of all the subprocesses needed in a text recognition pipeline, most functional blocks can be accessed separately on the command line, allowing flexible substitution of specially optimized methods. A stable programming interface allows total customization and integration into other software packages.

8.2.1 Recognition

The recognition engine operates as a segmentation-less sequence classifier using an artificial neural network to map an image of a single line of text, the input sequence, into a sequence of characters, the output sequence. The artificial neural network employed is a combination convolutional and recurrent neural network trained with the CTC loss function [2] that reduces training data requirements to line-level transcriptions (figure 8.4). Regularization is mainly provided by dropout [3] after both convolutional and recurrent layers. User intervention in determining training duration and model selection is largely eliminated through early stopping.

Specialized networks, e.g. for particularly complex scripts, can be assembled from building blocks with a simple network specification language although the default architecture shown in figure 8.1 is suitable for the vast majority of applications.

Processing of dictionaries and library catalogues with extensive semantic markup such as italic, underlining, and bolding, is also possible through specially prepared training data.

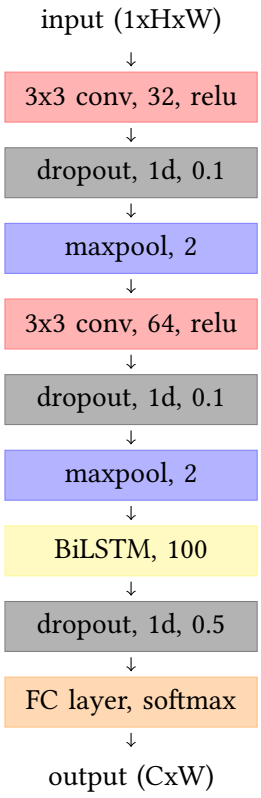


Figure 8.1: Network architecture (H: sequence height, W: sequence length, C: alphabet size)

8.2.2 Layout Analysis and Script Detection

Kraken's layout analysis extracts text lines from an input image for later processing by the recognition engine. Apart from a basic segmenter taken from OCRopus a trainable line extractor is in the process of being implemented. Full trainability of layout analysis is of utmost importance to a truly universal OCR system, as text layout and its semantics varies widely across time and space, e.g. hand-crafted methods for printed Latin text are unlikely to work reliably on Arabic text or manuscripts with extensive interlinear annotation.

The trainable layout analysis module consists of a two-step instance segmentation method: an initial seed-labelling network operates on the whole page labelling the area between baseline and mean of each line. As the output of the network is a probability of each pixel belonging to a baseline it is binarized using hysteresis thresholding after smoothing with a gaussian filter. The binarized im-

age is then skeletonized and end point are extracted with a discrete convolution. Finally, the vectorized baseline between the endpoints is rectified and a variable environment calculated based on the distance of connected components from the labelled area is extracted.

Table 8.1: Mean character accuracy and standard deviation on the validation set across 10 training runs on each training set

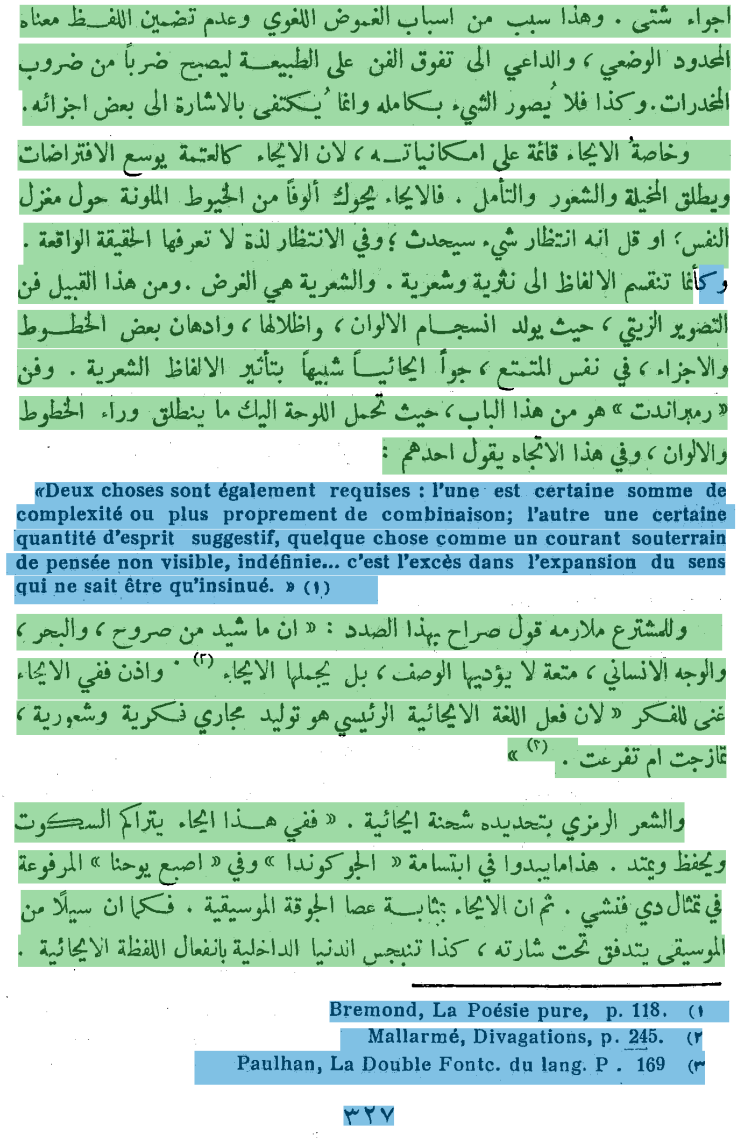
	Mean character accuracy	**Standard deviation**	**Maximum accuracy**
Prints			
Arabic [4]	99.5%	0.05	99.6%
Persian[a]	98.3%	0.33	98.7%
Syriac[b]	98.7%	0.38	99.2%
Polytonic Greek[c]	99.2%	0.26	99.6%
Latin [5]	98.8%	0.09	99.3%
Latin incunabula [5]	99.0%	0.11	99.2%
Fraktur [5]	99.0%	0.31	99.3%
Cyrillic[d]	99.3%	0.15	99.6%
Manuscripts			
Hebrew[e]	96.9%	-	-
Medieval Latin[f]	98.2%	-	-

[a]Mid-20th century printing
[b]Late-19th century printing in Serṭā form
[c]Late-19th century printing
[d]1923 Russian print
[e]Medieval Midrash Tanhuma
[f]Mid-9th century Carolingian of Josephus Latinus

The seed-labelling network is a modified U-net [6] on the basis of a 34-layer residual network [7] pretrained on ImageNet.

Preliminary results on a page from a publicly available dataset of Arabic and Persian manuscripts [8] can be seen in Figure 8.2.

Script detection, the basis for multi-script support in the recognizer, is implemented as a segmentation-less sequence classification problem, similar to text recognition. Instead of assigning a unique label to each code point or grapheme cluster we assign all code points of a particular script the same label. The network is then trained to output the correct sequence of script labels (figure 8.4). The output sequence is then used to split the line into single-script runs that can be classified with monolingual recognition models (figure 8.3).



اجواء شتى . وهذا سبب من اسباب الغموض اللغوي وعدم تضمين اللفظ معناه المحدود الوضعي ، والداعي الى تفوق الفن على الطبيعة ليصبح ضرباً من ضروب المخدرات . وكذا فلا يُصور الشيء بكامله وانما يُكتفى بالاشارة الى بعض اجزائه .

وخاصة الايحاء قائمة على امكانياته ، لان الايحاء كالعتمة يوسع الافتراضات ويطلق المخيلة والشعور والتأمل . فالايحاء يحوك ألوفاً من الخيوط الملونة حول مغزل النفس ، او قل انه انتظار شيء سيحدث ؛ وفي الانتظار لذة لا تعرفها الحقيقة الواقعة . وكأنما تنقسم الالفاظ الى نثرية وشعرية . والشعرية هي الغرض . ومن هذا القبيل فن التصوير الزيتي ، حيث يولد انسجام الالوان ، واظلالها ، وادهان بعض الخطوط والاجزاء ، في نفس المتمتع ، جواً ايحائياً شبيهاً بتأثير الالفاظ الشعرية . وفن « رمبراندت » هو من هذا الباب ، حيث تحمل اللوحة اليك ما ينطلق وراء الخطوط والالوان ، وفي هذا الاتجاه يقول احدهم :

«Deux choses sont également requises : l'une est certaine somme de complexité ou plus proprement de combinaison; l'autre une certaine quantité d'esprit suggestif, quelque chose comme un courant souterrain de pensée non visible, indéfinie... c'est l'excès dans l'expansion du sens qui ne sait être qu'insinué. » (١)

وللمشترع ملارمه قول صراح بهذا الصدد : « ان ما شيد من صروح ، والبحر ، والوجه الانساني ، متعة لا يؤديها الوصف ، بل يجملها الايحاء (٢) . واذن ففي الايحاء غنى للفكر « لان فعل اللغة الايحائية الرئيسي هو توليد مجاري فكرية وشعورية ، تازجت ام تفرعت . (٣) »

والشعر الرمزي بتحديده شحنة ايحائية . « ففي هذا الايحاء يتراكم السكوت ويحفظ ويمتد . هذا مايبدوا في ابتسامة « الجوكوندا » وفي « اصبع يوحنا » المرفوعة في تمثال دي فنشي . ثم ان الايحاء بثابة عصا الجوقة الموسيقية . فكما ان سيالاً من الموسيقى يتدفق تحت شارته ، كذا تنبجس اندنيا الداخلية بانفعال اللفظة الايحائية .

(١) Bremond, La Poésie pure, p. 118.
(٢) Mallarmé, Divagations, p. 245.
(٣) Paulhan, La Double Fontc. du lang. P . 169

٣٢٧

Figure 8.3: Sample output of the script detection on a bilingual French/Arabic page. Note that Eastern Arabic are always classified as Latin text

ويقول رئيس شركة U. S. Steel: «انا لا اؤمن بالدين. فالدين له الميزة الغير المسرة

ويقول رئيس شركة U. S. Steel : «انا لا اؤمن بالدين . فالدين له الميزة الغير المسرة

000000200000200000020020000002220000002000020020002222122122111112000020000200000

Figure 8.4: Original and modified ground truth (top: original line, middle: transcription, bottom: assigned script classes)

8.3 Results

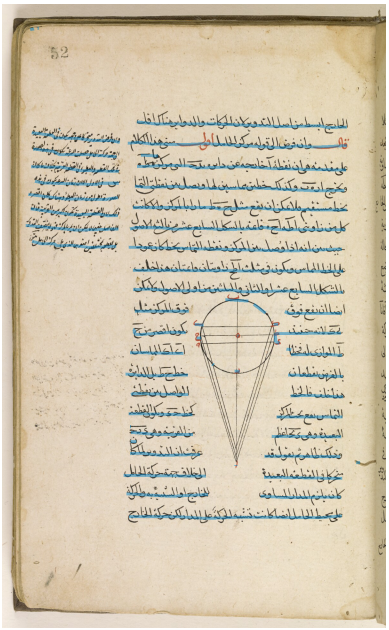


Figure 8.2: Sample output of the trainable segmentation method

Kraken has been used on a wide variety of writing systems, achieving uniformly high character accuracy (CER). Sample accuracies for a diverse set of scripts spanning across multiple centuries of printing are shown in table 8.1. It should be noted that recent improvements in the text recognition engine result in significantly diminished character error rates in comparison to earlier versions such as those evaluated in [4].

As a special use case we evaluated recognition of text and emphasis in a mixed English and romanized Arabic library catalog on a training set of 350 lines (50 lines in the validation set) resulting in an averaged CER of 99.3% ($\sigma = 0.16$) over 10 runs with 95.38% CER on cursive and text with increased spacing ($\sigma = 1.46$). When using only emphasized text accuracy as the stopping criterium mean accuracy rises to 99.03% ($\sigma = 0.28$) [9].

References

[2] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 369–376.

[3] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.

[4] B. Kiessling, M. T. Miller, G. Maxim, S. B. Savant, *et al.*, "Important New Developments in Arabographic Optical Character Recognition (OCR)," *Al-ʿUṣūr al-Wusṭā*, vol. 25, pp. 1–13, 2017.

[5] U. Springmann, C. Reul, S. Dipper, and J. Baiter, "Ground truth for training OCR engines on historical documents in german fraktur and early modern latin," *J. Lang. Technol. Comput. Linguistics*, vol. 33, no. 1, pp. 97–114, 2018.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2015, pp. 234–241.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] B. Kiessling, D. Stökl Ben Ezra, and M. T. Miller, "BADAM: A Public Dataset for Baseline Detection in Arabic-Script Manuscripts," in *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing, HIP@ICDAR 2019, Sydney, NSW, Australia, September 20-21, 2019*, ACM, 2019, pp. 13–18.

[9] B. Kiessling, D. Kinitz, C. Gümmer, and P. Mashhadi, "Script and Emphasis Detection using Recurrent Neural Networks," READ2018: International Interdisciplinary Symposium on Reading Experience and Analysis of Documents, 2018.

Chapter 9

Transcription Alignment for Highly Fragmentary Historical Manuscripts: The Dead Sea Scrolls

Dedicated to the late Yaacov Choueka (1936–2020), pioneer in natural language processing and historical manuscript analysis. May his memory be blessed.

This chapter has been published as D. S. B. Ezra, B. Brown-DeVost, N. Dershowitz, A. Pechorin, and B. Kiessling, "Transcription Alignment for Highly Fragmentary Historical Manuscripts: The Dead Sea Scrolls," in *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, 2020, pp. 361–366

Abstract

Most of the Dead Sea Scrolls have now been digitally transcribed and imaged to very high standards. Our goal is to align the transcriptions with the text visible in the image, glyph by (often fragmentary) glyph. This involves several tasks, normally considered in isolation: (A) Baseline segmentation. (B) Line polygon extraction. (C) Automated transcription by handwritten character recognition, to aid in alignment. (D) Alignment of the Unicode characters in a line transcription with the characters in the image of that line. The task is frustrated by the degraded nature of the frequently very small and/or warped fragments with many broken letters, substantially different allographs, ligatures, and scribal idiosyncrasies. Furthermore, a great number of inconsistencies between current cataloguing systems for the data need to be resolved. For each task, we apply state-of-the-art machine-learning methods in addition to more traditional techniques, each presenting significant difficulties on account of the poor state of most fragments' preservation.

We have built ground-truth datasets and have managed to achieve good results with well-preserved fragments by leveraging heavily augmented transfer learning from prior work with medieval manuscripts.

9.1 Background

The Dead Sea Scrolls (dated to the turn of the Common Era) are of enormous historical significance. They are the oldest witnesses of the biblical books and contain a treasure-trove of texts that have shed light on ancient Judaism shortly before and up into the time of Jesus and Paul. Their study continues to revolutionize our understanding of the evolution of Judaism and the emergence of Christianity. Unfortunately, the scrolls, or rather the fragments, were discovered in the 20th century CE in very poor condition, having deteriorated over the millennia. Few are large enough to contain even several columns.[1] The vast majority show only a low number of words or even just a few letters, many of which are only partially visible. Over the past decades, all texts have been painstakingly transcribed by scholars. The texts are so fragmentary that the editions have developed systems to distinguish between certain, probable, possible, and entirely restored letters. Taking only the certain and probable letters into consideration, the average fragment contains about 53 letters. However, the few scrolls with almost entirely preserved columns skew the mean as the median fragment contains only 13 letters. The extant fragments have recently been digitized by the Israel Antiquities Authority (IAA) using state-of-the-art multispectral imaging.[2] Older infrared images, photographed under the auspices of the Palestine Archaeological Museum (PAM) in the 1950s, are likewise available in retrodigitized form.

[1] For an example of a fragment from the book of Leviticus, see figure 9.1.

[2] https://www.deadseascrolls.org.il/explore-the-archive

Virtually all the texts have been transcribed and most appeared in the *Discoveries in the Judaean Desert (DJD)* series,[3] in Qimron's edition,[4] and in the *Qumran-Wörterbuch (QWB)* database of the Akademie der Wissenschaften zu Göttingen.[5] As the only resource that was truly computationally accessible at the start of this project, we based our work on the latter. See figure 9.2.

9.2 Introduction

Our objective is to develop an automated system to align transcriptions of the texts of the scroll fragments with the visible glyphs on the scroll images on the individual glyph level. Achieving this end requires isolating the fragments in an image from the background so that its text lines can be identified. The alignment of transcribed letters with glyphs appearing in the images of each line is then aided by recognizing at least some of the letters and spaces. The processes developed here are closely related to the ongoing work of several major research projects.

The DIP *Scripta Qumranica Electronica*[6] (SQE) aims to provide the scholarly community with an open source web-based portal for the purposes of material analysis of the scrolls. It combines the high-resolution image database of the IAA with the QWB lexical database. A feature-rich suite of digital tools and computational methods are brought together to create an infrastructure for the production of digital editions [2].

The University of Groningen project, *The Hands that Wrote the Bible*,[7] is dedicated to investigating the paleography of the Dead Sea Scrolls. It has so far produced a benchmark study in writer identification [3] and advances in dating [4] and in binarization techniques of these manuscripts [5].

Nearly all of the Dead Sea Scrolls are written in Hebrew letters on animal skin, i.e. parchment, but phenomenologically they are very close to papyrus, which is mostly from ancient Egypt and written in Greek. The University of Basel project, *Reuniting fragments, identifying scribes and characterizing scripts: the Digital palaeography of Greek and Coptic papyri*,[8] organized a binarization competition [6] and published a dataset on writer identification [7]. In addition, the Wuerzburg-Heidelberg-Paris project, PapyroLogos, works on text-image alignment of literary and documentary Greek papyri [8].

Other major projects on Hebrew manuscript material include the Friedberg Genizah Project, which digitized hundreds of thousands of fragments of medieval manuscripts, mostly in Hebrew, Judeo-Arabic, and Aramaic [9].[9] State-of-the-art computational tools were developed for segmentation [10], paleography [11],

[3] http://orion.mscc.huji.ac.il/resources/djd.shtml
[4] https://zenodo.org/record/3737950\#.XoXRs6gzaiM
[5] http://www.qwb.adw-goettingen.gwdg.de
[6] http://qumranica.org
[7] https://cordis.europa.eu/project/id/640497
[8] https://altegeschichte.philhist.unibas.ch/de/digpaleo
[9] https://fgp.genizah.org

matching fragments by handwriting and codicological features [12], and word spotting [13].

The Sofer Mahir project strives to create open source transcriptions of ca. 6000 pages of 18 substantial manuscripts of the earliest Rabbinic literature (Mishnah, Tosefta and Midreshei Halakhah).[10] In the Tikkoun Sofrim project, crowdsourcing and machine learning is used to correct errors in the automatic transcriptions of manuscripts of medieval exegetical literature [14].[11]

9.3 Methods and Related Work

Different infrastructures allow automatic interaction with historical manuscripts (a brief overview is given in [15]). Most notable are Transkribus [16] and MONK [17], which are however not open source and, at least in the case of Transkribus also commercial,[12] and therefore much more difficult or even impossible to include in a full treatment pipeline. The only cutting edge and fully open-source infrastructure for historical document analysis we know of is eScriptorium [18].Accordingly, we have made use of its tools and have performed the following procedures.

9.3.1 Line Segmentation

After a long predominance of methods relying on traditional computer vision approaches to perform text line extraction from handwritten documents, machine learning based systems have seen wider use recently [19]–[23]. The majority of these methods utilize combinations of CNNs and LSTMs. Still, traditional methods from computer vision can have advantages for certain tasks or types of manuscripts [24]–[26]. We tested several hand-crafted line segmentation algorithms without success, settling on a trainable method described in [8], [19], as implemented in kraken [27] and eScriptorium. Layout analysis is independent of binarization and works very well even on highly fragmentary and damaged material (such as the Dead Sea Scrolls and the Genizah); see figure 9.4.

9.3.2 Automated Transcription

In line with the state of the art in text image classification we utilize a hybrid CNN-RNN trained in a supervised manner to classify sequences of characters on whole text lines using the connectionist temporal classification loss [28]. The kraken OCR engine's recognizer with default parameters is used instead of a custom implementation.

Due to the challenging nature of the material such as high script variability and extensive degradation, even the best modern OCR engines perform quite

[10] https://sofermahir.hypotheses.org

[11] https://tikkunsofrim.hypotheses.org

[12] https://readcoop.eu/transkribus-pricing

Figure 9.1: Manuscript fragment (Leviticus 3) after imperfect foreground segmentation. All images of fragments are courtesy of the Leon Levy Dead Sea Scrolls Digital Library, Israel Antiquities Authority. Photos: Shai Halevi.

[...]על המזבח סביב והקריב מזבח השלמים [...]	1
[...]הָעצה יסירנה ואתָ[ה]חלב המכסה את הָ[...]	2
[...] הכליות ואת החלֶב אשר עליהן אשר עָ[...]	3
[...] נָה והקָ[ט]יר הכהן המזבחה לָ[...]	4
[...]נָי יהוה וסמך את [...]	5
[...]לָ[המז]בָח סביב וָהָ[...]	6
[...]לָ[...]	7

Figure 9.2: Scholarly transcription of the fragment (4Q24 fr. 8) in figure 9.1.

poorly (< 90% CER). While unsuitable for close reading, even poor-quality OCR output can be serviceable for novel applications like intertextuality and search. In contrast to exact search as implemented in standard search engines, which yields very limited results, approximate search can be both applied to finding individual phrases [29], [30] and to matching against an existing corpus [31]. Our method for text identification based on approximate search is detailed in Section 9.4.3.

9.3.3 Transcription Alignment

Early work on aligning OCR text with ground truth is presented in [32]. More recent work includes [33]–[40]. We experimented with (a) optical SIFT-flow [36], (b) alignment with OCR results – by means of minimal edit distance, and (c) a combination – using anchors obtained from the OCR to constrain the optical flow.

For optical SIFT-flow we first render the known Unicode transcription as a line image in a manner and font that is similar to the manuscript. Next, a visual alignment is made between the synthetic transcription image and the original

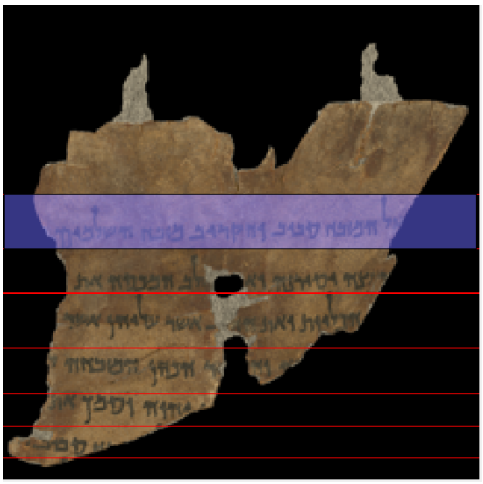


Figure 9.3: Line segmentation of the fragment in figure 9.1.

manuscript image by the SIFT flow image matching algorithm introduced in [41]. Since we have information regarding the letter boundaries in the rendered image, these boundaries translated into the manuscript image by the retrieved optical alignment result in an approximation of the letter boundaries in the manuscript image, thus resulting in a glyph alignment. To enhance the visual alignment, we may use previously discovered correspondences between the rendered image and the manuscript image, which we call anchors for the optical alignment, in order to align the images more precisely. These anchors might be gained, for instance, from character or inter-word bounding boxes found by the OCR algorithm.

In the OCR based method, we first train a recognition model on the known transcription-line pairs with kraken until the system overfits the data. We then apply the same model on the data on which it was trained. We can extrapolate the approximate x-coordinate of the character boundaries based on the highest activation time-step returned by the system for a given character. The y coordinates can be estimated from the line polygon.

9.4 Experimental Results

9.4.1 Corpus Sample

The base data for the following analyses are the images from the Leon Levy Dead Sea Scrolls Digital Library and the text transcription of the Qumran Wörterbuch Project. These projects made use of two different and only sometimes overlapping cataloguing systems, which complicated the correlation of image to a specific set of transcriptions. After aligning the two systems by applying various adaptive rules for entry matching and some manually specified correlations, images were

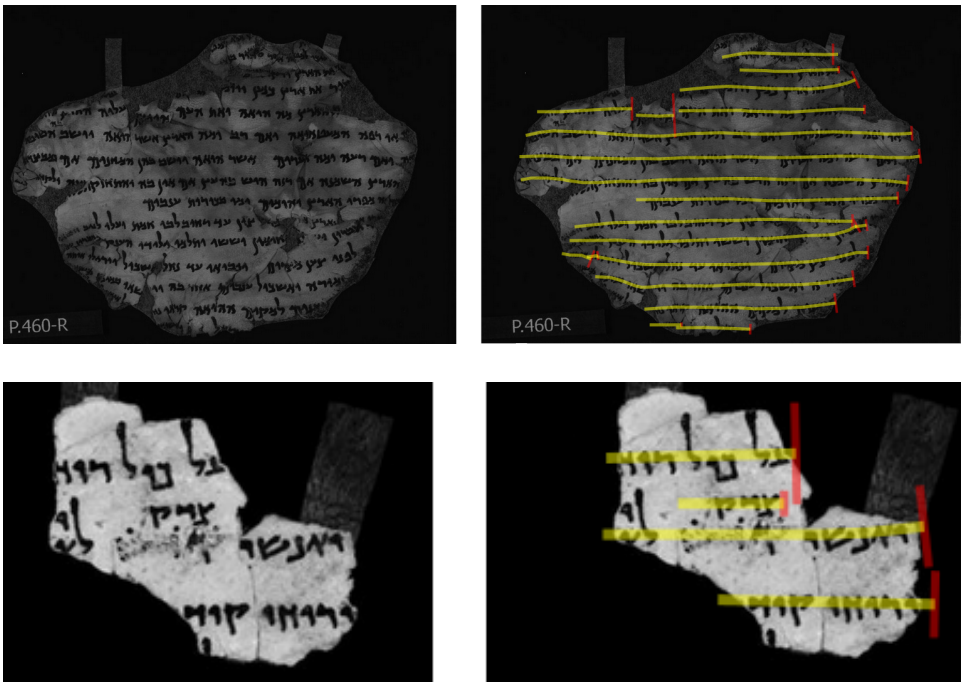


Figure 9.4: Automatic segmentation result (left without, right with baselines marked in yellow and an additional right vertical bar marking the beginning) of a large (top) medium (bottom) size fragment.

selected for which reliable matches to the textual transcriptions were available.[13]

For many reasons, the catalogue remains perfectible. The definition of what is a fragment is not straightforward and the fragments continued to "live" and change after their publication. In the new photographs, a fragment is a physical unit that can be lifted in one piece from its archival plate. However, such a unit may constitute several different fragments in the editions that have been joined, for instance, with Japanese rice paper in a later conservation process. In other cases, the editor gave a single identifier to what constitute several distinct physical units depicted on distinct images. Other fragments, still in one piece in the edition, have since broken up or disintegrated into several pieces. Some images in the image database are identified as representing a specific fragment while in fact the current fragment only contains a fraction of the published text. Several identifications were incorrect, and a few imaged fragments were not identified at all. Therefore, we had to verify the identification of each image with its corresponding transcription from the QWB database.

[13]The results of the merging of these two catalogues can be accessed through the SQE web API https://api.qumranica.org/swagger.

9.4.2 Line Segmentation

As a first step, we transfer-learned a new baseline segmentation model on top of models trained initially on medieval manuscripts and Greek papyri. We bootstrapped the training material following a common procedure: Firstly, we manually annotated 100 images of Qumran fragments, used them as training material, and afterwards applied the results to 300 more images of Qumran fragments. We then manually corrected the automatic results in the eScriptorium web interface and used that larger corpus to train another model to apply to ca. 500 more images. The ergonomic interface of eScriptorium makes this usually cumbersome process very easy. While manually annotating the baselines of an image from scratch takes approximately 90 seconds, the average manual correction time for an automatically segmented image is less than 30 seconds for the first stage and less than 15 for the second stage. However, depending on the complexity of the layout, the time needed for an image can differ markedly. Many images require few or no corrections. See figure 9.5 for an example.

Some fragments have been imaged at a rotation angle other than upright. Consequently, we determine the correct reading order based on the median principal writing angle of the baselines, taking into consideration the writing direction. In the near future, we will add the new kraken and escriptorium feature for automatic segmentation of regions to the pipeline to improve the results for multi-column fragments, especially regarding the reading order[42].

9.4.3 Automated Transcription

In a second step, we extracted textual data from the QWB database and matched it linewise to the segmented lines on the images. We retained only fragments written in Judean square script leaving out any fragment written in paleo-Hebrew, Cryptic C, Greek, or Nabatean. Still the hands of the fragments vary widely in register, formality, and period and represent many different scribal habits. The dates the scrolls were written could vary by 300 years in a very "hot" period, that is, a period with massive changes in ductus according to local schools after the disintegration of the relatively unified Imperial writing system of the Persian Empire. We discarded all letters marked as restored or as merely possible readings, keeping only the probable and certain instances. Due to the aforementioned complications inherent to the fragments, editions, and the database, the "zipping" together of the image and the textual data is not a trivial process. Therefore, the rough OCR of the extant text in the next step, provided a welcome check, (1) whether the identification of the fragment image with the corresponding text was correct, (2) whether the fragment was still complete, and/or (3) whether it had been joined with other fragments.

The third step consisted of training a transcription model on the selected ground truth with a 90/10 training/testing split on the grayscale images (without binarization). Discarding misidentified items and fragments in other scripts, the

Figure 9.5: Imageline to textline alignment result as displayed in eScriptorium. Baselines are depicted in yellow, boundary polygons in alternating red and blue.

final training material comprised 33075 characters on 2474 lines from 440 images. The testing material read 3403 characters on 247 lines from 44 images.

On the average, we can count 5–6 lines and ca. 75 characters (including spaces) per fragment. These are thus relatively large fragments. New models were trained on top of the models previously trained on medieval Hebrew manuscripts.

The best model reached an accuracy of 67.9% on the test material after 21 epochs. While this may seem very low, we applied the trained models to fragments outside of the training and test corpora, and the automatic transcriptions were extremely convincing for most fragments. The results are in fact better than the numbers indicate because frequently the transcriptions include very partial letters of which sometimes only scant remains are visible, in particular in the top and the bottom row of fragments, but not only. Even experts would typically have to expend significant effort evaluating the best reading possibilities.

In particular, the OCR results are sufficient to identify the fragments. With a bag of words approach for identification and with rotations every 90° to choose the best angle for recognition, the system was able to identify 22 out of 24 available fragments comprising more than 100 characters. In other words, given the imperfect OCR of each fragment and searching for the words among all 5756 transcriptions in QWB, the best match was indeed the actual scholarly transcription of the fragment in question. The two exceptions were fragments for which the system preferred a fragment of the same composition but from a different manuscript.

9.4.4 Transcript Alignment

To evaluate the various transcription alignment algorithms' performance, we compared the automatic alignment with the bounding boxes from a set consisting of 1278 letters that had been expertly segmented by hand using the Scripta Qumranica Electronica website. We denote a glyph as correctly aligned if the correct glyph in the manuscript is the closest one to the computed glyph location. To measure the distance between letters, we use Euclidean distance between the centers of the bounding boxes of the glyphs.

To further examine the performance of our leading transcription alignment

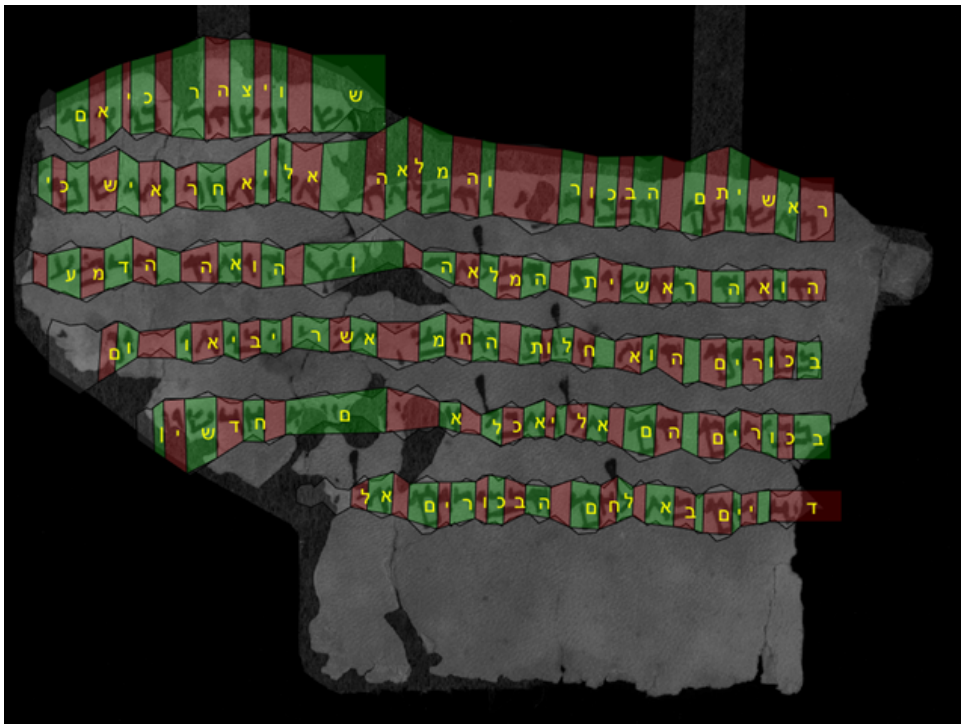


Figure 9.6: Aligned glyphs of a whole fragment. Alternating red and green polygons indicate areas. Yellow overlay indicates identified letter.

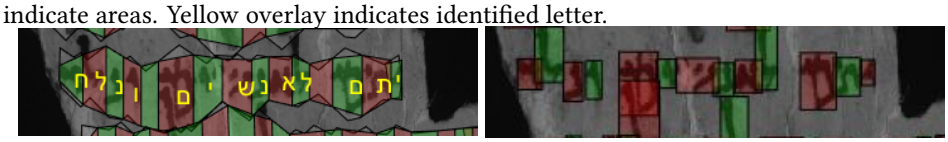


Figure 9.7: Aligned glyphs of a single line. Left: Automatic alignment with alternating red and green polygons indicate areas. Yellow overlay indicates identified letter. Right: Corresponding human annotated ground truth (no interword spaces, no letter overlay).

method, we measured the proportion of the intersection area of the bounding polygon of the glyph of the OCR system and the human annotated ground truth:

As can be seen in table 9.1, the accuracy of the OCR based transcription alignment method is the highest among the methods we've used, and the intersection of the recognized bounding polygon with the original glyph bounding box is high as well (table 9.2). An alignment example is displayed in figure 9.6. The interword space in line 2, for instance, has been well detected and shows that our alignment method provides excellent results on the word level. Figure 9.7 shows aligned glyphs of a single line compared to the ground truth. Finally, the ground truth allows for overlapping glyph bounding boxes, a feature impossible for the current and all other known algorithms.

An analysis of the errors shows that the results can be further improved as some letters and some positions quite consistently reveal a higher error proportion. The letter lamed, which has a high ascender, is frequently cut below its top by the seamcarve algorithm, often because of the deterioration of the writing ma-

Table 9.1: Transcription alignment accuracy

Method	Accuracy
Optical flow without anchors	48.1%
Optical flow with added anchors	74.0%
OCR derived alignment	**90.3**%

Table 9.2: OCR bounding box overlap with ground truth

	Area	Area percentage
Mean	31.0	81.0%
Median	23.8	87.1%
Standard deviation	30.8	20.5%

terial. Similarly final mem has a long descender and can be cut too high. Finally, the seamcarve algorithm uses the neighboring lines to limit the height of rows. This is impossible for the upper boundary of the top and the lower boundary of the bottom rows. For all of these problems with y coordinates, obvious solutions tailored to the type of script and material are available. Otherwise, the method use should be able to be applied to other sequential scripts.

9.5 Discussion

We have put together an end-to-end automated pipeline for processing images and transcriptions of the very fragmentary Dead Sea Scroll manuscripts. Despite the many difficulties posed by the often seriously degraded material, the quality of segmentation and character recognition were sufficient to allow a glyph-by-glyph alignment of existing transcriptions to the new, high-quality images. The successful identification of fragments based on automatic transcriptions holds promise of helping to identify some of the remaining unidentified fragments of the image database with their counterparts in the text database. Each of the stages of the pipeline, viz. (A) baseline layout analysis of the fragment and (B) segmentation into line polygons, (C) rough automated transcription of the text in each of the fragment lines, and (D) alignment of the rough automatic transcription to the scholarly transcriptions to the image of the fragment, can be improved further.

The successful automated alignment of transcriptions to images will allow a textual layer to be added to the IAA images. This means that scholars and laypersons alike will be able to enter search terms and retrieve images containing them. It will also supply additional training data for improved character recognition and future paleographical analyses.

Acknowledgments

This research was supported in part by Grant BE 5916/1-1 KR 1473/8-1 from the Deutsch-Israelische Projektkooperation (DIP) and by Grant Agreement No. 871127 from the European Union's Horizon 2020 Research and Innovation Programme. It was made possible thanks to images taken by Shay Halevi and provided by the Leon Levy Dead Sea Scrolls Digital Library of the Israel Antiquities Authority, all rights reserved. We thank the SQE project members, especially Oren Ableman, Adiel Ben-Shalom, and Lior Wolf.

References

[2] B. Brown deVost, "Scripta Qumranica Electronica (2016–2021)," *Hebrew Bible and Ancient Israel*, vol. 5, pp. 307–315, 2016.

[3] M. A. Dhali, S. He, M. Popovic, E. Tigchelaar, and L. Schomaker, "A digital palaeographic approach towards writer identification in the Dead Sea scrolls," in *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2017, Porto, Portugal, February 24-26, 2017*, M. D. Marsico, G. S. di Baja, and A. L. N. Fred, Eds., SciTePress, 2017, pp. 693–702.

[4] M. A. Dhali, C. N. Jansen, J. W. de Wit, and L. Schomaker, "Feature-extraction methods for historical manuscript dating based on writing style development," *Pattern Recognition Letters*, vol. 131, pp. 413–420, 2020.

[5] M. A. Dhali, J. W. de Wit, and L. Schomaker, "Binet: Degraded-manuscript binarization in diverse document textures and layouts using deep encoder-decoder networks," *CoRR*, vol. abs/1911.07930, 2019. arXiv: 1911.07930.

[6] I. Pratikakis, K. Zagoris, X. Karagiannis, *et al.*, "ICDAR 2019 competition on document image binarization (DIBCO 2019)," in *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*, IEEE, 2019, pp. 1547–1556.

[7] H. A. Mohammed, I. Marthot-Santaniello, and V. Märgner, "Grk-papyri: A dataset of Greek handwriting on papyri for the task of writer identification," in *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*, 2019, pp. 726–731.

[8] B. Kiessling, D. Stökl Ben Ezra, R. Ast, and H. Essler, *Aligning extant transcriptions of documentary and literary papyri with their glyphs*, 29th International Congress of Papyrology (Lecce), 2019.

[9] Y. Choueka, "Computerizing the Cairo Genizah: Aims, methodologies and achievements," *Ginzei Qedem*, vol. 8, 9*–30*, 2012.

[10] R. Shweka, Y. Choueka, L. Wolf, and N. Dershowitz, "Automatic extraction of catalog data from digital images of historical manuscripts," *Literary and Linguistic Computing*, vol. 28, no. 2, pp. 315–330, Feb. 2013.

[11] L. Wolf, N. Dershowitz, L. Potikha, *et al.*, "Automatic paleographic exploration of Genizah manuscripts," in *Kodikologie und Paläographie im Digitalen Zeitalter 2 – Codicology and Palaeography in the Digital Age 2*, ser. Schriften des Instituts für Dokumentologie und Editorik, F. Fischer, C. Fritze, and G. Vogeler, Eds., vol. 3, Germany: Norderstedt: Books on Demand, 2011, pp. 157–179.

[12] L. Wolf, R. Littman, N. Mayer, *et al.*, "Identifying join candidates in the Cairo Genizah," *International Journal of Computer Vision*, vol. 94, no. 1, pp. 118–135, Aug. 2011.

[13] A. Ben-Shalom, Y. Choueka, N. Dershowitz, and L. Wolf, "Querying the Cairo Genizah images with word-spotting algorithm," in *The Twelfth Annual Jerusalem Conference on the Digitisation of Cultural Heritage*, (Abstract), Jerusalem, Israel, Nov. 2015.

[14] T. Kuflik, M. Lavee, D. S. B. Ezra, *et al.*, "Tikkoun Sofrim – combining HTR and crowdsourcing for automated transcription of Hebrew medieval manuscripts," *Digital Humanities (DH2019)*, 2019.

[15] B. Kiessling, R. Tissot, P. Stokes, and D. Stökl Ben Ezra, "eScriptorium: An open source platform for historical document analysis," in *2nd International Workshop on Open Services and Tools for Document Analysis, OST@ICDAR 2019, Sydney, Australia, September 22–25, 2019*, 2019, pp. 19–24.

[16] P. Kahle, S. Colutto, G. Hackl, and G. Mühlberger, "Transkribus – a service platform for transcription, recognition and retrieval of historical documents," in *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[17] L. Schomaker, "Lifelong learning for text retrieval and recognition in historical handwritten document collection," in *Handwritten Historical Document Analysis, Recognition, and Retrieval – State of the Art and Future Trends*, ser. Machine Perception and Artificial Intelligence, A. Fischer, M. Liwicki, and R. Ingold, Eds., World Scientific, 2020. DOI: 10.1142/11353.

[18] B. Kiessling, R. Tissot, P. Stokes, and D. Stökl Ben Ezra, "eScriptorium: An open source platform for historical document analysis," in *International Conference on Document Analysis and Recognition Workshops (ICDARW)*, vol. 2, Sydney, Australia, Sep. 2019, p. 19. DOI: 10.1109/ICDARW.2019.10032.

[19] B. Kiessling, D. Stökl Ben Ezra, and M. T. Miller, "BADAM: A Public Dataset for Baseline Detection in Arabic-Script Manuscripts," in *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing, HIP@ICDAR 2019, Sydney, NSW, Australia, September 20-21, 2019*, ACM, 2019, pp. 13–18.

[20] T. Grüning, R. Labahn, M. Diem, F. Kleber, and S. Fiel, "READ-BAD: A new dataset and evaluation scheme for baseline detection in archival documents," in *13th IAPR International Workshop on Document Analysis Systems (DAS)*, IEEE, 2018, pp. 351–356.

[21] M. Fink, T. Layer, G. Mackenbrock, and M. Sprinzl, "Baseline Detection in Historical Documents Using Convolutional U-Nets," in *13th IAPR International Workshop on Document Analysis Systems, DAS 2018, Vienna, Austria, April 24-27, 2018*, IEEE Computer Society, 2018, pp. 37–42. DOI: 10.1109/DAS.2018.34.

[22] M. Diem, F. Kleber, S. Fiel, T. Grüning, and B. Gatos, "cBAD: ICDAR2017 competition on baseline detection," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, IEEE, vol. 1, 2017, pp. 1355–1360.

[23] B. Barakat, A. Droby, M. Kassis, and J. El-Sana, "Text line segmentation for challenging handwritten document images using fully convolutional network," in *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018, pp. 374–379.

[24] G. Sadeh, L. Wolf, T. Hassner, N. Dershowitz, and D. Stökl Ben Ezra, "Viral transcript alignment," in *ICDAR*, 2015, pp. 711–715.

[25] M. Seuret, D. Stökl Ben Ezra, and M. Liwicki, "Robust heartbeat-based line segmentation methods for regular texts and paratextual elements," in *HIP@ICDAR*, 2017, pp. 71–76.

[26] D. Stökl Ben Ezra and H. Lapin, "Z-profile: Holistic preprocessing applied to Hebrew manuscripts for HTR with Ocropy and Kraken," *Manuscript Cultures*,

[27] B. Kiessling, "Kraken - A Universal Text Recognizer for the Humanities," *Proceedings of the DH*, 2019.

[28] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 369–376.

[29] M. Christodoulakis, G. Brey, Uppal, and R. Ahmed, "Evaluation of approximate pattern matching algorithms for OCR texts," in *Proceedings of the 4th Annual Conference on Advances in Computing and Technology (AC&T)*, The School of Computing and Technology, University of East London, 2009, pp. 35–42.

[30] T. Badamdorj, A. Ben-Shalom, N. Dershowitz, and L. Wolf, "Fast search with poor OCR," *CoRR*, vol. abs/1909.07899, 2019. arXiv: 1909.07899.

[31] A. Zhicharevich, "Tools to aid OCR of Hebrew character manuscripts," M.S. thesis, Tel Aviv University, 2012.

[32] J. D. Hobby, "Matching document images with ground truth," *International Journal on Document Analysis and Recognition*, vol. 1, pp. 52–61, 1998.

[33] S. Feng and R. Manmatha, "A hierarchical, HMM-based automatic evaluation of OCR accuracy for a digital library of books," in *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2006, Chapel Hill, NC, USA, June 11-15, 2006, Proceedings*, G. Marchionini, M. L. Nelson, and C. C. Marshall, Eds., ACM, 2006, pp. 109–118.

[34] I. Z. Yalniz and R. Manmatha, "A fast alignment scheme for automatic OCR evaluation of books," in *2011 International Conference on Document Analysis and Recognition, ICDAR 2011, Beijing, China, September 18-21, 2011*, IEEE, 2011, pp. 754–758.

[35] A. Fischer, V. Frinken, A. Fornés, and H. Bunke, "Transcription alignment of latin manuscripts using hidden markov models," in *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, ACM, 2011, pp. 29–36.

[36] T. Hassner, L. Wolf, and N. Dershowitz, "OCR-free transcript alignment," in *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR 2013, Washington, DC)*, Aug. 2013, pp. 1310–1314.

[37] Y. Leydier, V. Eglin, S. Bres, and D. Stutzmann, "Learning-free text-image alignment for medieval manuscripts," in *14th International Conference on Frontiers in Handwriting Recognition (ICFHR 2014, Crete, Greece, September 1–4, 2014)*, IEEE Computer Society, 2014, pp. 363–368.

[38] T. Hassner, L. Wolf, N. Dershowitz, G. Sadeh, and D. Stökl Ben Ezra, "Dense correspondences and ancient texts," in *Dense Image Correspondences for Computer Vision*, T. Hassner and C. Liu, Eds., Switzerland: Springer-Verlag, 2016, pp. 279–295.

[39] Y. Leydier, V. Eglin, S. Bres, and D. Stutzmann, "Alignement texte-image sans apprentissage pour les manuscrits médiévaux," in *CORIA 2016 – Conférence en Recherche d'Informations et Applications – 13th French Information Retrieval Conference. CIFED 2016 Colloque International Francophone sur l'Ecrit et le Document, Toulouse, France, March 9–11, 2016*, S. Calabretto, B. Coüasnon, L. Goeuriot, and S. Barrat, Eds., ARIA-GRCE, 2016, pp. 481–496.

[40] M. Boillet, M. Bonhomme, D. Stutzmann, and C. Kermorvant, "HORAE: an annotated dataset of books of hours," in *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing, HIP@ICDAR 2019, Sydney, NSW, Australia, September 20–21, 2019*, ACM, 2019, pp. 7–12.

[41] C. Liu, J. Yuen, and A. Torralba, "SIFT Flow: Dense correspondence across scenes and its applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 978–994, 2011.

[42] B. Kiessling, "A Modular Region and Text Line Layout Analysis System," in *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, 2020, pp. 313–318.

Part IV

The Escriptorium VRE

Chapter 10

eScriptorium: An Open Source Platform for Historical Document Analysis

This chapter has been published as B. Kiessling, R. Tissot, P. Stokes, and D. Stökl Ben Ezra, "eScriptorium: An open source platform for historical document analysis," in *2nd International Workshop on Open Services and Tools for Document Analysis, OST@ICDAR 2019, Sydney, Australia, September 22–25, 2019*, 2019, pp. 19–24

Abstract

We describe the new open source document analysis and annotation platform eScriptorium. It allows to upload document collections, transcribe and segment them manually or automatically with the help of the Kraken OCR engine.

10.1 Introduction

eScriptorium[1] is developed in the frame of the Scripta PSL programme[2] at PSL University[3], a recent foundation grouping prestigious institutions such as the ENS[4], EPHE[5], ENC[6], and as associated members the EHESS[7], EFEO[8], and Collège de France and in addition the IRHT[9]. The Scripta programme groups around 100 researchers from the humanities, social sciences, digital humanities and computer sciences that work on written objects from a huge variety of regions, cultures, scripts and languages covering 5500 years with a focus on the premodern time. The aim of eScriptorium is to combine computational tools with manual digital tools for transcription and deep annotation of texts and images (paleographic, philological, historical, and linguistic). As well as scholars from the humanities, the target groups encompass computer scientists, librarians and archivists, students and, potentially, the general public. Development of the platform started in November 2018, although Kraken, one of its key constituent parts, has been under development since 2017 [2].

10.2 Previous Work

We decided to embark on this project because we did not know of another open source web-based infrastructure capable of dealing sufficiently well with historical manuscripts of various writing systems. Transkribus [3] and Monk [4], [5] are state-of-the-art transcription HTR-platforms, yet, their transcription system is not open source and both have turned into commercial products. Aletheia [6], [7] has no web interface and its full version has become commercial as well. The OCR engines Tesseract 4 [8], anyOCR [9], OCRopus [10], [11] have command line interfaces or primitive desktop interfaces only, even though Google is working to extend its web-based OCR service also to handwritten material [12], [13].

[1] Research blog at https://escripta.hypotheses.org. Brief videos are accessible at https://escripta.hypotheses.org/escriptorium-video-gallery.

[2] https://www.psl.eu/en/scripta

[3] https://www.psl.eu

[4] École normale supérieure

[5] École pratique des hautes études

[6] École nationale des chartes

[7] École des hautes études en sciences sociales

[8] École française d l'extrême orient

[9] Institut de recherche d'histoire de textes, an independent CNRS research unit.

Corpusbuilder [14] recently built on Kraken and tesseract 4 with a professional web-based interface, has been developed for print, not manuscripts. OCR4all [15] is webbased, but cannot handle fully automatic treatment of manuscripts.

10.3 Frontend

Users interact with the platform via a graphical user web interface written in Django and Javascript. No downloading of an applet is required. The interface is optimized for Chrome and Firefox. In the current state of affairs it handles:

1. Project and user management
2. Creation of documents and their metadata
3. Import of images, metadata and text
4. Export of segmentation and text
5. Manual interaction via zooming, panning, creating regions and lines or manual transcription
6. Computational image treatment via binarization, layout segmentation, training and automatic transcription or alignment.

10.3.1 Import/Export

Document images can be uploaded via single or bulk HTTPS protocol from the user's computer. A second option is to use a IIIF manifest[10] to import directly from a library or archive. The import via IIIF manifest also imports the metadata included in this manifest as exemplified in figure 10.1. Images can be uploaded in TIFF, PNG, or JPG formats. After uploading to the server they are converted into PNG and compressed losslessly. In addition to color images, users can import binarized images associated with their color image, as well as page segmentation and text files in ALTO[11] format. PageXML[12] and TEI-XML (Text Encoding Initiative) import and export are in preparation. A document overview (see figure 10.2) enables users to browse through the document and to select images for manual or automatic treatment, to change the order of images, and to add further images or delete existing ones. Each document image is represented by a thumbnail image with icons that represent different procedures; the icons blink when the procedures are running and change their color once done. After selecting images, users can click buttons in order to binarize or segment images, to train new transcription models or to apply trained transcription models.

[10] International Image Interoperability Framework: https://iiif.io

[11] Analyzed Layout and Text Object (ALTO) XML Schema https://www.loc.gov/standards/alto

[12] PAGE (Page Analysis and Ground truth Elements XML format. https://github.com/PRImA-Research-Lab/PAGE-XML

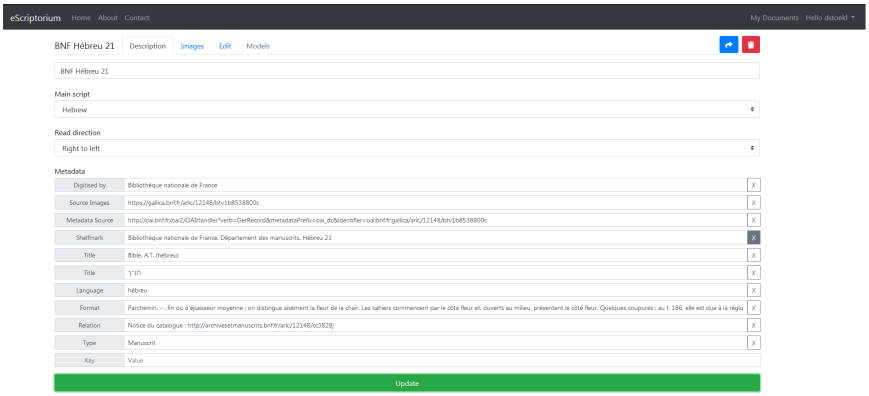


Figure 10.1: Metadata (and images) imported directly from the Bibliothèque nationale de France via a IIIF manifest

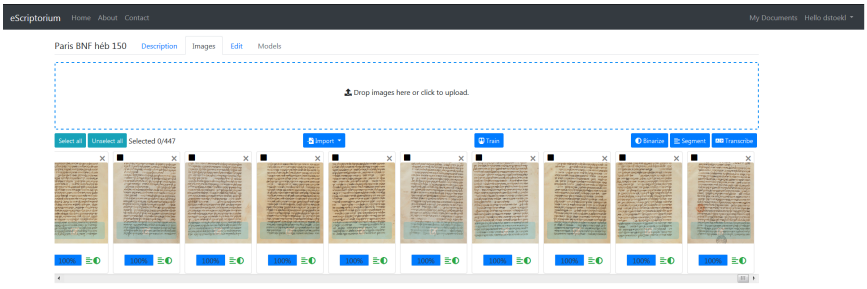


Figure 10.2: Document overview

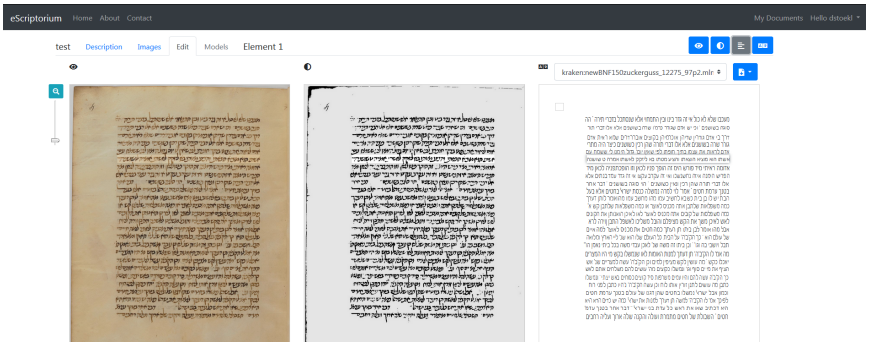


Figure 10.3: Lightbox showing color and binarized images with transcription

10.3.2 Manual layout analysis and transcription

Users can interact with each image either manually or automatically. They can create regions or lines, move them around, adjust their size or delete them. Currently the system is still limited to horizontal rectangles in order to get a first basic system functioning, but the transition to polygons is already underway and will be available soon. Users can transcribe manually or, once a model has been trained, automatically. Corresponding lines in the transcription interface and manuscript image or binarization are visualized by a lightbox (see figure 10.3). Manual transcription can be an end in itself or for Ground Truth generation. Automatic transcription can also be corrected through the same manual interface. Users can save transcriptions and go back in history to see or restore previous versions. Clicking on a line opens an enlarged image of the line and a box for transcription (see figure 10.4).

The platform has been planned from the outset for multiple writing systems and directions (left-to-right, right-to-left, soon also top-to-bottom). In principle, users have complete freedom in setting their own transcription guidelines up to full transliteration E.g., objects inscribed in a Semitic script can be transcribed into Latin (as is usually the case in Semitic epigraphy) or into Arabic or Hebrew. Ottoman Turkish documents can be transcribed into Latin as modern Turkish, or in the type of Arabic script used in the Ottoman Empire. Whole scripts and certain graphs not represented in Unicode can be encoded through use of the Private Use Area. We have had good experience with training specific ligatures or allographs, for example. Users can also choose to resolve abbreviations, or not. One of the project aims is to make hyperdiplomatic transcriptions possible, which can then enable the combination of quantitative and qualitative paleographical analysis, for instance [16].

The transcription and visualization panels are fully synchronized so that any zoom and panning of the image is also applied to the transcription so that users can always see the transcription corresponding to the image zone, as shown in figure 10.5. This enables ergonomic transcription and proofreading, and also enables users to deal with sources in any format (e.g. landscape or portrait) and size (e.g. large tables) far beyond the real estate of a regular computer screen.

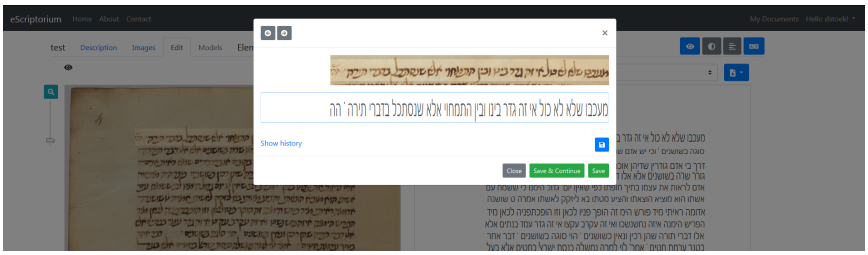


Figure 10.4: Line transcription window

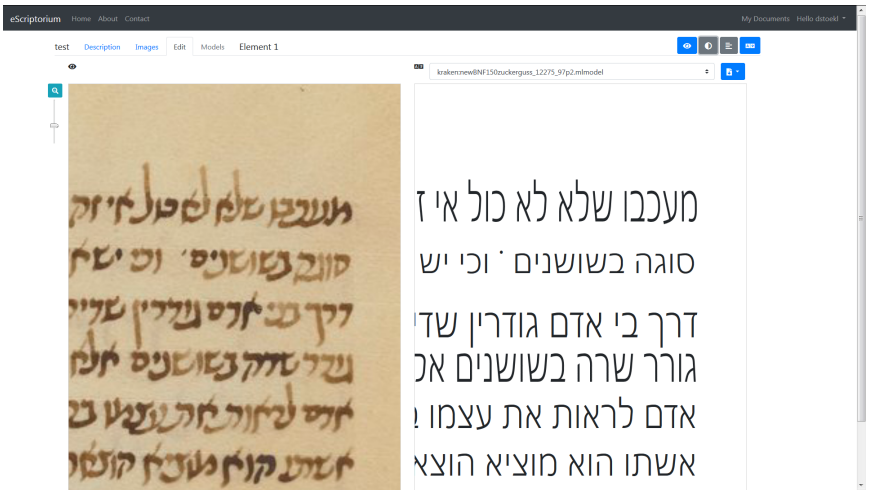


Figure 10.5: Image and transcription panel are synchronized

10.3.3 Automatic layout segmentation

Current layout segmentation requires binarization and allows only for horizontal bounding boxes around regions and/or lines (see figure 10.8). Users can design regions around writing blocks which can help to improve automatic segmentation with the current algorithm, and they can also correct line segmentation via the GUI. However, we have also developed a trainable baseline segmenter that does not require binarization and is able to deal with curved lines and highly complex layout as well as deteriorated material (see figure 10.9 and [17]. We are currently working on its integration into the platform and the transition from rectangular to polygonal zones. A sample page of the ground truth creation module that can also serve to correct automatic annotation can be seen in figure 10.7. Any writing direction will be possible. Layout segmentation ground truth has been created for Arabic [17], and other scripts, notably Hebrew, Greek and Latin, are in preparation.[13] Users will soon be able also to create their own ground truth for layout segmentation via a simple user interface allowing them to draw baselines. Subsequently they will be able to train layout segmentation models based on their own data or that shared by others.

10.3.4 Automatic transcription

Once the layout has been segmented automatically or manually, users can then work on automatic transcription. In order to train a transcription model, users must either transcribe some pages manually or upload an existing transcription in ALTO format (PageXML or TEI XML will also be available soon). They can then choose those pages they would like to use as input for the training process.

[13] Some of this is prepared in the French-Israeli project Tikkoun Sofrim (https://tikkunsofrim.hypotheses.org) [18], [19] and in the project Sofer Mahir (https://sofermahir.hypotheses.org).

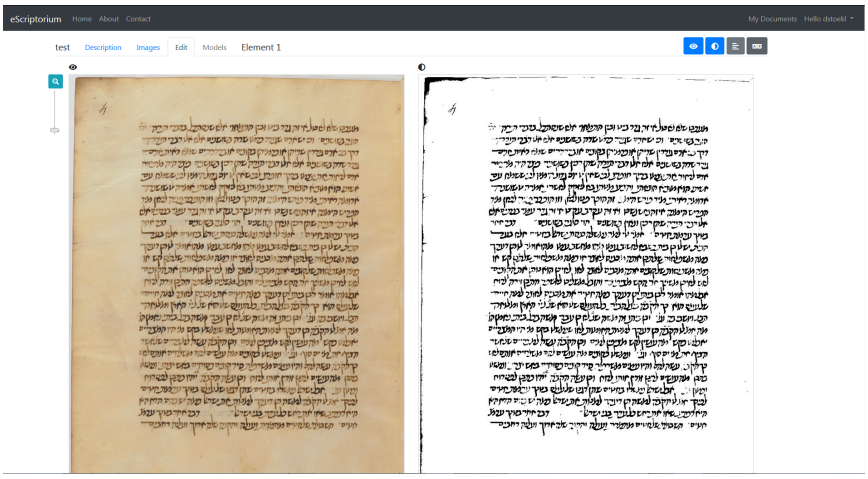


Figure 10.6: Binarization result

Once the training has finished, they can apply the trained model to pages with segmented layout. Another possibility is to upload a Kraken model that has been trained outside eScriptorium. A result of such an automatic transcription is shown in figure 10.8. In the Tikkoun Sofrim and Sofer Mahir projects, we have been able to reach character error ratios between 2% and 8.9%.[14]

10.4 Backend

10.4.1 Architecture

eScriptorium's stack is that of an industry grade web application. All of it's components have been chosen for being open source battle tested software and libraries and seen as the current industry standards. Those services are all exposed through an easily scalable docker-compose configuration.[15] The HTTP server is comprised of nginx+uwsgi[16] for serving regular http(s) requests and daphne[17] behind it for serving web sockets which are used for real time results of long-running asynchronous tasks like image processing and training. Since Kraken[18] is the angular stone of the application and is written in python, python has been chosen as the main backend language to leverage Kraken's internals and not only rely on its public command line API. Python is also gaining a lot of traction at the moment, especially in the scientific community which is important for open source software to engage contributions coming from other teams. The version in

[14]The Geneva 146 manuscript displayed on tikkoun sofrim.firebaseapp.com had a CER of 8.9%. The BNF 150 manuscript displayed there as well had a CER of 2.8%.

[15]https://docker.io

[16]https://nginx.org, https://uwsgi-docs.readthedocs.io/en/latest/

[17]https://github.com/django/daphne

[18]http://kraken.re

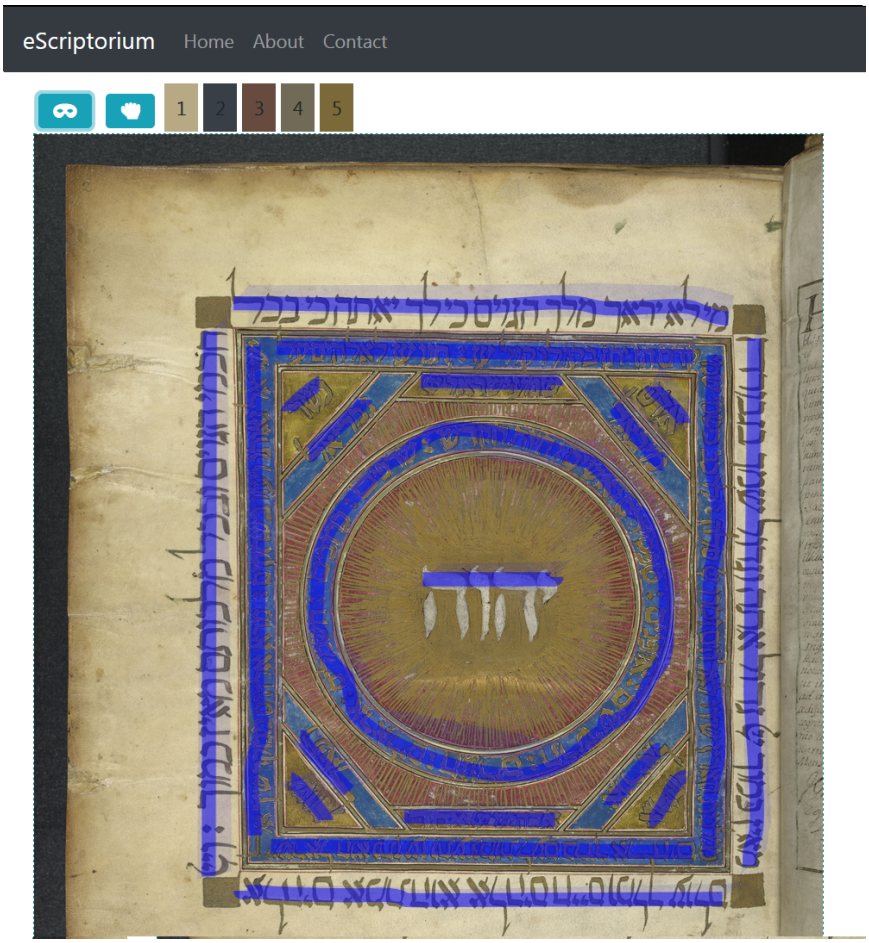


Figure 10.7: Module for the creation of ground truth for the line segmenter. British Library King's MS I fol. 2r

use is python3.6 because both Kraken and eScriptorium require it. The choice for the second most important part, the application server, was between Django[19], Flask and Pyramid. Django made the most sense since we can use a large part of its 'batteries included' and since it is also the framework of Archetype. The queue manager is Celery[20] for simplicity's and portability's sake, but could change depending on the constraints imposed by the hosting solution.

For storing data we use PostgreSQL[21] as a main database. MySQL is often preferred for it's simplicity but postgres is much more efficient in reading when the indexes are done right, and has much more features, for example PostGIS and jsonb. We couple it with redis[22] for caching, as a broker for Celery and gener-

[19]https://www.djangoproject.com
[20]http://www.celeryproject.org
[21]https://www.postgresql.org
[22]https://redis.io

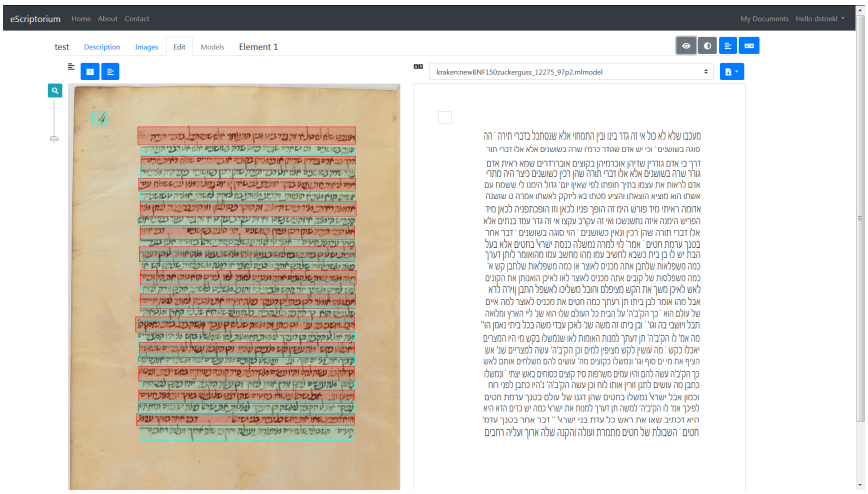


Figure 10.8: Automatic Line Segmentation and Transcription of a page from BNF Heb. 150 produced with eScriptorium

ally to store any temporary data. Elasticsearch [23] is our search engine of choice, other options comprised web services like Algolia, which is more designed for quick plug and play, so not suitable for storing trillions of characters; and non-Lucene based PostgreSQL fulltext search, but it is relatively new and still lacks some important features.

10.4.2 Database design

The modelization is permissive by design, leaving users the freedom to input and gather metadata, annotations and transcriptions according to the needs of their specific work. The goal is to provide sane settings to encourage users to make use of the default ontology but not force them to. This puts some pressure on the frontend code, since data validation is less strict and content search will require a very strong indexer, but it allows us to open the platform to a wide number of use cases.

10.4.3 Code architecture

The application exposes a partially writable REST API to the data, but the web application itself only uses it when forced to by the constraints of building a modern UI. This mix of monolithic and services approach allows us to maintain the logic close to the data while keeping external data integration a major concern. Distributed services were considered but past experiences have proven that these are impossible to maintain for projects with very limited resources such as this one. This should make for reasonable development time when introducing new features across the platform. The principal component of the backend is the Kraken

[23] https://elastic.com/products/elasticsearch

engine for handwritten text recognition, developed by Ben Kiessling. At present, very few Kraken options are exposed through the frontend in an attempt not to overwhelm the user, but these will be added cautiously when the need arises.

10.5 Open-Source Licence

An important aspect of eScriptorium is that the software itself is fully Free and Open Source, and the trained models and training data are not locked into the system and can be freely shared. The software for the framework is available under an MIT licence.[24] Kraken is released under an Apache 2.0 licence.[25] Kraken includes an open archive of models which operates via zenodo. Accordingly, users of eScriptorium are free to publish to and/or download from this (or indeed any other repository of their choice). Users therefore have control of their data and trained models, and can (but do not have to) open it to others inside or outside the platform which aids sustainability and transparency. In this way, others can profit from the effort spent on training and annotation for similar scripts and hands, and so on. This also gives users of eScriptorium control over access to the complete pipeline of their data and models. For instance, they can choose to host the platform on their own infrastructure, where they can customize it to their needs, or they can use their data and/or models in other implementations entirely.

10.6 Future Plans

10.6.1 Computational Extensions

On the computational side, we plan to include keyword spotting as well as automatic classification of manuscripts for dating and provenancing purposes, scribe distinction and tools for corpus linguistics and named entity recognition. We hope to be able to create an API communicating with DivaDIA for some of the methods [20].

10.6.2 Deep Annotation

On the digital side, the immediate next step in eScriptorium is adding the facility for "deep" structured annotations of images and texts. The objective here is for users to be able to add specific information to images and texts, for instance linguistic information or named entities in the text, or details about the handwriting in the images, or information about the document biography and so on. "Shallow" annotations of plain text and images are of course very widely available, and this can easily be used for instance for searches for all occurrences of

[24] https://gitlab.inria.fr/scripta/escriptorium
[25] https://github.com/mittagessen/kraken

a given letter. Structured "deep" annotations are already relatively well established for text, for instance in TEI XML which is the de facto standard for many of our users and is to be incorporated here. The difference with eScriptorium is that image annotations are also associated with a structured model about what is annotated, the model (ideally) being customizable according to the needs of the research project. For instance, researchers in handwriting may develop a model of script such that (for instance) the Latin alphabet has letters such as "a", and each letter may have different forms (allographs) such as **a** and *a*; these allographs may in turn have components such as a stem and a body, and so forth [21]. Embedding such a model into image annotations allows much more powerful searches, such as showing images of any letter with ascenders in order to compare how one or more scribes constructed this part of the letter. Such "deep" annotations have already been used in a group of projects built on the Archetype infrastructure 26 and have proven very effective in addressing needs of researchers in the humanities.

10.6.3 Publication Platform

By the end of the project, we expect users also to publish their work online publication through this infrastructure, in the form of digital editions of text, analysis of the historical handwriting linked to images, and so on. This is intended to meet a clear and well-known need for a relatively low-cost and sustainable framework for the publication of texts but also for other forms of related research in ways that are accessible and transparent to enable knowledge creation and exploration. For digital editions, we currently envisage TEI Publisher,[26] but an API should also enable many other publishing possibilities.

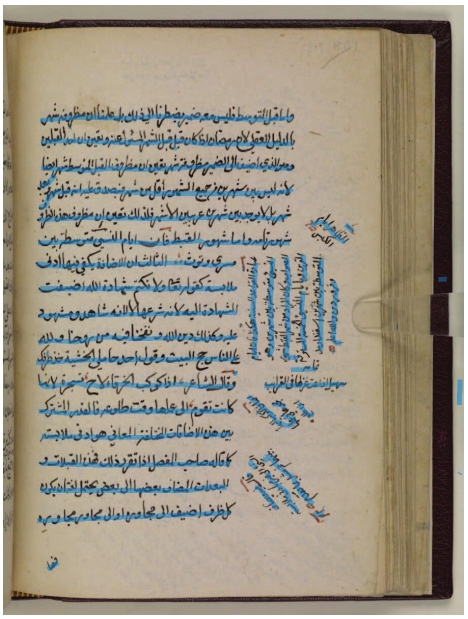


Figure 10.9: Automatic segmentation of an Arabic manuscript

10.6.4 Outreach

We are actively reaching out to other digital, computational and humanities teams that would like to join our efforts in order to assure the sustainabil-

[26] https://teipublisher.com

ity of the architecture as well as its breadth and flexibility to cater to a large audience.

10.6.5 Videos

Brief videos are accessible at https://escripta.hypotheses.org/escriptorium-video-gallery.

Acknowledgments

This work is part of the Scripta-PSL project financed by the Agence Nationale de la Recherche via the Initiative d'excellence PSL (n. 10-IDEX-0001). Annotation of ground truth thanks to the PHC Maimonide France-Israel project Tikkoun Sofrim between the EPHE, PSL, the University of Haifa and the National Library of Israel. Images of BNF Cod. Heb. 150 courtesy of the National Library of France, Paris (BNF). Image of BL King's MS I courtesy of the British Library, London, UK.

References

[2] B. Kiessling, "Kraken - A Universal Text Recognizer for the Humanities," *Proceedings of the DH*, 2019.

[3] P. Kahle, S. Colutto, G. Hackl, and G. Mühlberger, "Transkribus – a service platform for transcription, recognition and retrieval of historical documents," in *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[4] L. Schmaker. "Monk." (), [Online]. Available: https://monkweb.nl.

[5] L. Schomaker, "Design considerations for a large-scale image-based text search engine in historical manuscript collections," *it Information Technology*, vol. 58, no. 2, pp. 80–88, 2016.

[6] "Aletheia." (2014), [Online]. Available: https://www.primaresearch.org/tools/Aletheia.

[7] C. Clausner, S. Pletschacher, and A. Antonacopoulos, "Aletheia - an advanced document layout and text ground-truthing system for production environments," in *2011 International Conference on Document Analysis and Recognition*, 2011, pp. 48–52. DOI: 10.1109/ICDAR.2011.19.

[8] "Tesseract 4.0." (), [Online]. Available: https://github.com/tesseract-ocr/tesseract.

[9] S. S. Bukhari, A. Kadi, M. A. Jouneh, F. M. Mir, and A. Dengel, "anyOCR: An Open-Source OCR System for Historical Archives," in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, IEEE, 2017, pp. 305–310. DOI: 10.1109/ICDAR.2017.58.

[10] T. Breuel. "OCRopy." (), [Online]. Available: https://github.com/tmbdev/ocropy.

[11] ——, "OCRopus 3." (), [Online]. Available: https://github.com/tmbdev/ocropy3-docker.

[12] J. Walker, Y. Fujii, and A. C. Popat, "A web-based ocr service for documents," in *Proceedings of the 13th IAPR International Workshop on Document Analysis Systems (DAS), Vienna, Austria*, vol. 1, 2018.

[13] R. R. Ingle, Y. Fujii, T. Deselaers, J. Baccash, and A. C. Popat, "A Scalable Handwritten Text Recognition System," in *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*, IEEE, 2019, pp. 17–24. DOI: 10.1109/ICDAR.2019.00013.

[14] "corpusbuilder." (), [Online]. Available: https://github.com/berkmancenter/corpusbuilder.

[15] C. Reul, D. Christ, A. Hartelt, *et al.*, "Ocr4all—an open-source tool providing a (semi-) automatic ocr workflow for historical printings," *Applied Sciences*, vol. 9, no. 22, p. 4853, 2019.

[16] B. Kiessling, R. Tissot, D. S. B. Ezra, and P. Stokes, "Escripta: A new digital platform for the study of historical texts and writing," in *Digital Humanities 2019*, 2019.

[17] B. Kiessling, D. Stökl Ben Ezra, and M. T. Miller, "BADAM: A Public Dataset for Baseline Detection in Arabic-Script Manuscripts," in *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing, HIP@ICDAR 2019, Sydney, NSW, Australia, September 20-21, 2019*, ACM, 2019, pp. 13–18.

[18] T. Kuflik, M. Lavee, D. S. B. Ezra, *et al.*, "Tikkoun Sofrim – combining HTR and crowdsourcing for automated transcription of Hebrew medieval manuscripts," *Digital Humanities (DH2019)*, 2019.

[19] A. J. Wecker, U. Schor, D. Elovits, *et al.*, "Tikkoun sofrim: A webapp for personalization and adaptation of crowdsourcing transcriptions," in *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, 2019, pp. 109–110.

[20] M. Würsch, R. Ingold, and M. Liwicki, "DivaServices - a RESTful web service for Document Image Analysis methods," *Digit. Scholarsh. Humanit.*, vol. 32, no. suppl_1, pp. i150–i156, 2017. DOI: 10.1093/llc/fqw051.

[21] P. A. Stokes, "Modelling multigraphism: The digital representation of multiple scripts and alphabets," *Digital Humanities 2018: Book of Abstracts/Libro de resúmenes.*, 2018.

Chapter 11

The eScriptorium VRE for Manuscript Cultures

This chapter has been accepted for publication in P. Stokes, B. Kiessling, D. Stökl Ben Ezra, R. Tissot, and E. Gargem, "The eScriptorium VRE for Manuscript Cultures," *Classics@*,

11.1 Introduction: What is eScriptorium

The eScriptorium VRE is software being developed at the École Pratique des Hautes Études, Université Paris Science et Lettres (EPHE – PSL), with the immediate goal of developing a web interface to an engine for the automatic transcription of written sources, both printed and handwritten, in principle in any current or historical system of writing.[1] The software is intended to be a core component in a larger VRE which provides the key steps in the normal editorial chain. The assumption here is that the researcher has images of the text, and wishes first to obtain a transcription of the text in these images, then (most likely) add markup to the text and potentially also to the images in order to encode the various editorial judgments that are a core part of any edition, perhaps then also or instead to apply techniques such as Natural Language Processing (NLP) or Named Entity Recognition (NER), and finally to publish the text and image, along with accompanying data and metadata. To date, the focus has been entirely on the first of these steps, that is, allowing the manual, semi-automatic or automatic transcription of texts, and the user can then export the results to be marked up, analyzed and/or published in other frameworks.

The development of eScriptorium began as part of a larger project at PSL called Scripta, which is studying the history and practice of writing in almost all its forms across most of human history, and it has since been continued most notably in the Resilience project, which is a European infrastructure project looking to develop a long-term (35-year) research infrastructure for religious studies. The range of languages and writing-systems being studied in the Scripta project is enormous, covering the Ancient Near East (e.g. Ancient Aramaic, Ptolemaic Egyptian, Ugaritic), Iran and Central Asia (including Elamite, Sogdhian, Middle Iranian), India and South-East Asia (such as Sanskrit, Classical Tamil, Old Javanese, Tai-Lue), and East Asia (Tibetan, Classical Chinese, Old and medieval Japanese), as well as the Classical and Medieval West (including Greek, Umbrian, and Old Slavonic), among others. The scope of Resilience is in principle even bigger, as it should cover all languages relevant to any aspect of religious studies in Europe for the next thirty-five years. It has therefore been a crucial element of the project that the software must avoid as far as possible all assumptions about the nature of the writing and language that is in the system. The writing may be left to right, right to left, top to bottom or even bottom to top; the support may be paper, parchment, but also stone, palm leaf, clay, wood, or many others; it may be written with a pen, painted with a brush, inscribed with a chisel; the writing system may be alphabetic, logographic, hieroglyphic; and so on. This variety means that almost all levels of the software are very much more complex than they would be for a single type of script, as will be discussed below.

[1] Further discussions and papers on the eScriptorium project include [2]–[5]. This work has received funding from the European Union's Horizon 2020 Research and Innovation program under Grant Agreement No. 871127 (RESILIENCE), and from the Initiatives de Recherches Interdisciplinaires et Stratégiques of Université PSL (Scripta-PSL).

The eScriptorium VRE is designed to interface with the Kraken engine for OCR/HTR.[2] This engine has been developed by Benjamin Kiessling, who is also from EPHE-PSL and is part of the eScriptorium team. Written in Python, the Kraken engine is designed from the beginning to embed as few pre-assumptions about the writing-systems as possible, and so to work with a very wide range of different scripts. It is highly modular, and each module has a large number of parameters that the user can set to accommodate the specific needs of the case in question. Furthermore, if the existing parameters are not sufficient, it is entirely possible for a sufficiently-skilled user to replace any given module of the Kraken engine with a custom-made one. This flexibility is extremely important, particularly for the very diverse needs of the Scripta and Resilience projects. However, it also means that Kraken requires a relatively good understanding of OCR/HTR software and processes, as well as being comfortable in installing modules and dependencies, and running processes directly from the command line. For this reason it is not very appropriate for the majority of users in the Humanities, for whom the time and effort that must be invested in learning these techniques may well seem too much for a technology that is still relatively new and for which the benefits may be in doubt. The eScriptorium interface therefore serves as a user-friendly way into the Kraken engine, providing a system that functions well for the majority of users, while those with more specialized needs will still be able to judge the system and its likely value and therefore be better placed to make an informed decision whether to invest further in the details.

11.2 The eScriptorium Workflow

In order to understand the current state of the art in OCR/HTR systems, it is necessary first to understand the basic workflow of eScriptorium and other similar systems. In general, going from images to transcription requires the following basic steps:

1. Importing the images into the system, including preprocessing such as PDF import and other format conversions.

2. Finding the lines of text and other significant elements (columns, glosses, initials, etc.) on the images: that is, subdividing an image into sets of shapes on that image that correspond to different region types.

3. Transcribing the lines of text: that is, converting images of lines of text into the corresponding text.

[2] Some writers distinguish Optical Character Recognition (OCR) as the automatic transcription of printed text and Handwritten Text Recognition (HTR) as that of handwritten text, whereas others reserve OCR for character-based approaches to recognition and HTR for line-based. As a result, the difference between OCR and HTR is often blurred in the current literature, and so we use the two terms together as interchangeable unless clearly stated otherwise.

4. Compiling the lines of text into a coherent document and exporting the result for markup, publication, etc.

Steps 1 and 4 are relatively straightforward and are largely a question of interface.[3] In eScriptorium, the import can be done directly from a user's hard drive, or by simply giving the URL of the IIIF manifest file and leaving the machine to import the images automatically. Steps 2 and 3 are generally much more complex, since, as we have seen, the computer needs to be able to treat a very wide range of different documents, supports, and layouts.[4] It is important to note that each of these basic steps comprises multiple sub-tasks with sometimes unexpected ways that they can fail. Step 2 not only finds lines but also has to sort them into the same order a human would read them in, a process which is by no means easy given the many possible layouts across different writing systems. This step is crucial, though, because the transcription of a document produced in step 3 will be completely unreadable, even if perfect on a line level, if this ordering operation has failed. In Kraken, and therefore eScriptorium, both of these two steps are handled by trainable computer vision algorithms, that is, by machine learning. Very broadly, this means that one must have example documents which have already been prepared: that is, images of pages annotated with columns, lines and so on for step 2, and transcriptions of texts matching the images for step 3. One then submits these example cases to the computer, and the machine "learns" from them, creating a statistical model of the images which it has already seen, and this model then allows it either to segment new unseen images into regions, or to produce a transcription of the text from the unseen images. As for transcription, the eScriptorium interface also allows for adding this information directly in order to compile ground truth material for training, and it also provides mechanisms for the user to correct any errors in the automatic or indeed manual results (as show in Figure 11.1 and [6, n. 3]).

In practice, these two steps for machine learning often comprise several substeps. For instance, one may begin by preparing a certain number of pages by hand, most often by typing them out manually (Figure 11.2 and [6, n. 4]). It may then be helpful to train the machine based on this relatively small sample and then automatically transcribe some more pages: the results may have numerous errors, but correcting these errors may be faster than typing out the whole page. After manual correction, the machine can be re-trained with this additional material, and then the subsequent pages will have fewer errors and will therefore be faster to correct. This process can then be repeated until the results are good enough to be useful. How good is "good enough" depends very much on the use-case: it is generally not possible to get perfect results, but it is often possible to get over 99% character accuracy, meaning correcting perhaps one or two errors per page.

[3] For videos showing these steps in an early version of eScriptorium, see [5] and [6, n. 1 and 5]

[4] or videos showing these steps see [5] and [6, n. 2-4 and 6]. For convenience, the term "document" is used in this article as a short-hand to refer to any instance of writing, whether printed or handwritten, without reference to any specific form, support, or writing instrument.

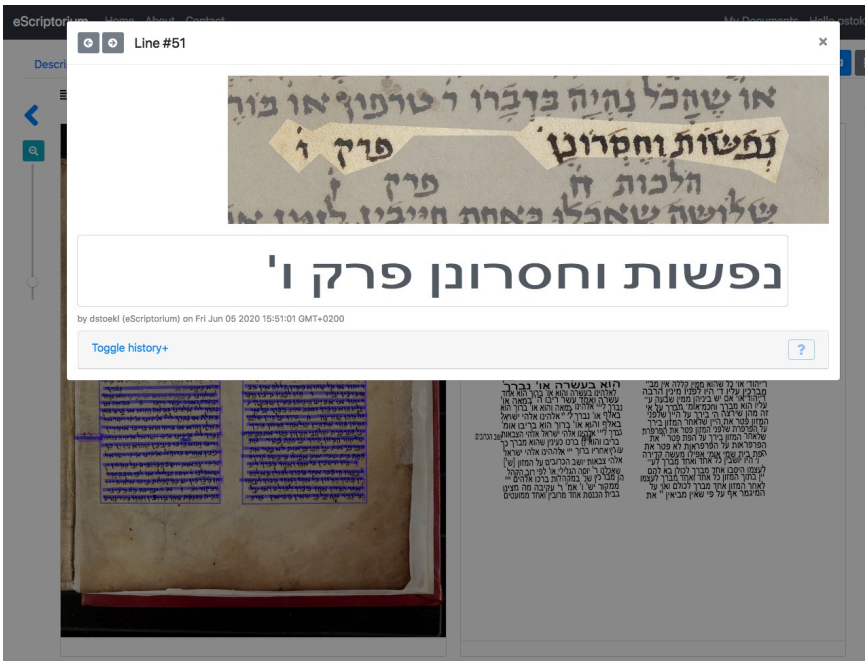


Figure 11.1: Entering and correcting lines of text in eScriptorium.

Correcting this number of errors is very much faster than typing by hand, and the correction may well be unnecessary anyway, since 99% character accuracy is more than enough for many purposes of distant reading and other large-scale analyses, such as automatically identifying uncatalogued texts, for most forms of NLP and NER, and so on. There are also ways of speeding up this process, for instance by taking an existing transcription of the same text elsewhere, importing it into eScriptorium, and then adjusting it to match this particular exemplar, or taking examples of other transcribed texts that were written in scripts very similar to the new text, and training on those.

11.3 State of the Art in Current OCR/HTR

For the more technically-minded readers, it is helpful at this point to understand the current state of the art in systems for manuscript OCR/HTR. Feature-complete systems at the time of writing include Kraken but also Tesseract 4 and Transkribus. Broadly speaking, these systems work in similar ways for automatic transcription, and they give similar results in terms of accuracy, but they differ significantly in areas such as their ability to cope with complex "non-standard" page layout or "unusual" script (as seen from a modern Western point of view), and the degree of openness in terms of Open Source software but also in the ability to import and export data and trained models, points which are discussed in the following section.

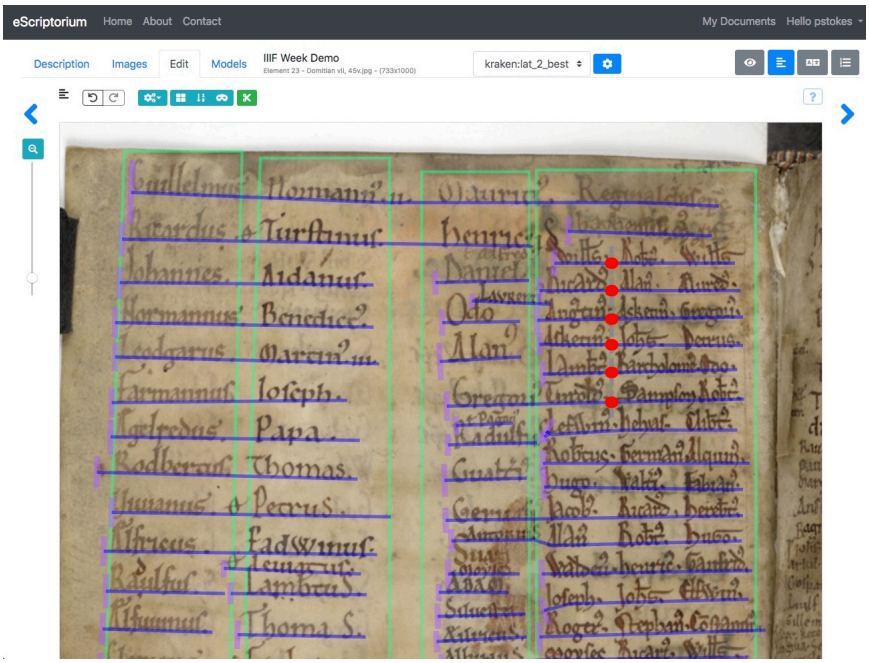


Figure 11.2: Correcting automatic segmentation in eScriptorium. The manuscript image used in this screen-shot is a detail from London, British Library, Cotton MS Domitian A.vii, 45v.

These current systems are generally line-wise text recognizers: that is, text images are transcribed a whole line at a time in contrast to the character-by-character approach which is employed in traditional OCR for printed text (such as in Tesseract 3 and Sakhr). The line-wise transcription is usually performed by a recurrent or hybrid convolutional neural network which has been trained specifically for a particular script or even a specific hand, depending on the complexity and variability of the writing. All modern systems are trained with an implicit alignment between the desired textual output and the input images, most often through what is known as the CTC loss function[7]. The primary benefit of this alignment is easier acquisition of data for training, as the laborious labelling of single characters is replaced with the much simpler transcription of whole lines.

Before transcribing the text, a modern system needs first to identify the lines on the page, and this is handled by a layout analysis (LA) module, the exact functionality of which depends on the nature of the text transcription module. Character classifiers require the extraction of single glyphs from the page by the LA system, a process which can be difficult for cursive scripts, while line-based systems can use whole line extraction techniques which are more versatile and script-independent. The major technical difference between different HTR packages lies in this LA module: how it models lines and if it can be adapted to new kinds of documents. Tesseract and Ocropus retain hand-crafted, non-adaptable computer vision methods that output rectangular boxes around the lines. These

work reasonably well for printed documents and clean handwriting but cannot reliably process complex manuscripts, especially if the lines of text are curved or otherwise do not fit naturally into these boxes. Recently, new forms of LA use the baselines of the text instead of boxes, and this has been successful in dealing with highly complex material containing slanted, curved, and rotated lines[8]. Methods following this paradigm are popular in the research community but actual implementations are currently limited to Transkribus and Kraken/eScriptorium.

In addition, a wide variety of research algorithms can also be found in the literature but which have not yet been implemented in OCR systems. These include systems that merge the steps for layout analysis and transcription [9], or methods that are optimized to extract text from noisy environments such as natural scenes[10]. Another active field of research in computer science is in methods to decrease the manual labor required to successfully train machine learning algorithms through approaches such as domain adaptation (transforming models trained on one kind of document to another), semi-supervised learning (training on partially labelled examples), and wholly synthetic manuscript pages (training the machine on "artificial" images so that it can learn to read the real ones)[5] Nevertheless, it remains that the creation of these example cases or "ground truth" for the computer is the longest and most laborious part of the process for the end user, and it may even seem contradictory that one must manually transcribe many pages in order for the computer to transcribe automatically. Indeed, if one only has a very small corpus, or if the range of different scribal hands or styles of writing in that corpus is large, then it may not be worth the effort to use these automatic methods. However, if the corpus is large and homogeneous enough that the computer can train to a sufficiently high level for your needs, then, once this initial groundwork is done, the results afterwards can be spectacular, with thousands or even millions of words being transcribed automatically at literally the click of a button. Nevertheless, it should not be surprising that there is no "magic solution" that can instantly solve all cases. As we know very well, texts are extremely complex objects, with a great deal of variety in terms of layout, format, structure, style of script, and so on, and it takes us human beings many years of specialized training to learn to read them. This complexity makes them interesting and worthy of years of study, but it should come as no surprise that it also makes them difficult to treat with a machine.

11.4 Openness and Import/Export of Images, Texts and Models

In addition to the flexibility and adaptability to different writing systems, another of the core principles of both Kraken and eScriptorium is that of openness. The software for both Kraken and eScriptorium is open-source and free for anyone

[5] One recent example among others is [11]

to download, use, and modify. More significantly, though, the framework is designed to allow for the easy import and export of images, transcriptions and trained models, and this is particularly important for a number of reasons. As we now know very well from experience, a closed system is extremely risky in terms of sustainability, since if you are locked into a given system then you become entirely dependent on it: if it ceases to function then your project is potentially in jeopardy, and one can easily become hostage to any future developments, such as if a free service becomes paid-for. It is therefore always of the utmost importance that one uses standards-compliant data, and that this data can be freely imported into and exported from different pieces of software, in order to avoid dependance on any one piece and thereby help ensure the longevity of the process as a whole. Equally if not more important in the case of OCR/HTR is the ability to import and export trained models in particular which is important on both scholarly and practical terms. From a scholarly point of view, there are very real questions around scholarly transparency and accountability in the use of machine learning. If we are preparing transcriptions automatically in this way then naturally the results will be influenced according to the training data that we have provided to the machine. As discussed above, there are different standards and practices for transcription, and the types of document and script that are used in the ground truth will inevitably influence the results. Other scholars will therefore need access to the ground truth and trained models in order to understand exactly how the text was obtained, to evaluate if it was appropriate or not, and to anticipate potential biases and errors.[6] From a practical point of view, the process of compiling ground truth is often laborious as we have seen, and in fact on a commercial level, very large-scale datasets of high-quality ground truth are extremely valuable, which is part of the reason why multi-billion dollar corporations are so keen to access our e-mails, labelled photographs, and so on. In addition, the process of training a model is also relatively slow and intensive. This is of less concern to the average user in the Humanities, since we can simply leave the computer to "do its thing" while we get on with something else. Nevertheless, training Deep Learning systems like Kraken is very intensive for the machine, and it can take weeks on a normal home computer. The process is very much faster if one has access to a High Performance Computing (HPC) center with specialized hardware, but very few scholars in the Humanities have this access, and in any case the process uses a relatively high amount of electricity with financial and ecological implications.[7] Fortunately, the training is the intensive and slow part, and once this is done then the model can be used relatively quickly and easily for the segmentation and transcription. However, this again illustrates the benefits of exporting

[6] A simple example of an error resulting from this point was a project which attempted to automatically identify authorship in vernacular Old English texts, but without controlling for different editorial practices in the sources: as a result, the project was successful in identifying editors but not authors. See further [12, pg. 54-56]

[7] As just one example, the GPU units that the eScriptorium team are currently putting in place will have an estimated running cost of approximately €10,000 per year.

and sharing models. If I can train my model in an HPC center, and then download it and send it to you, or - even better - publish it on an open repository, then you and anyone else can take my model, upload it to your instance of eScriptorium (or Kraken, or some other system), and use it from there. You may need to retrain it to fit your specific documents, but as long as our documents are sufficiently close then the training that you need to do can be significantly reduced both in terms of time and the amount of ground truth. You would then ideally also publish your re-trained model in a public repository, and in this way we can build up a shared collection of trained models, thereby reducing significantly the computing time and energy that is currently being wasted on the redundant training of many different models on what is essentially the same script. More specifically, Kraken and eScriptorium both allow users to export and import models, for instance downloading them to their personal computers to do with as they wish. Kraken is also directly linked to Zenodo, which is a large-scale public international repository for research data. This means that one can decide at any point to publish a model to Zenodo, and the system will then take care of the publishing meaning that the model will be saved for the long term according to best practices in data archiving including the automatic assignment of a persistent identifier (a DOI) for future reference. Managing this successfully requires significant care in documentation and metadata, as future users of an existing model will need to know which standards of transcription were used, along with which sample images and so on, and this in turn requires that the entire ground truth also be published along with the model.[8]

11.5 Some Challenges for a Multi-Script VRE

The discussion so far presents eScriptorium as it stands at the time of writing. Although very much still in development, it is already being used by numerous teams in several different instances across Europe and the United States.[9] There are, however, numerous challenges that remain if the project is to achieve its goals. Aside from technical details of implementation, it seems at this point that the most significant challenges lie in the goal of being as close as possible to working with any script. Indeed, it is already clear that this is not truly possible: for instance, as mentioned above, the automatic transcription module of Kraken applies a line-by-line approach, but this assumes that the text is in fact written in lines (or that it can be approximated as such), an assumption that does not hold for hieroglyphic scripts like Mayan or ancient Egyptian. Indeed, the question of text direction is more complex than one might first imagine. On the face of it, the situation is simple enough: most scripts read in lines from left to right and

[8] For further discussion of this and other related problems, see (for example) OCR-D [13].

[9] As well as Scripta and RESILIENCE, other example projects include OpenITI AOCP (https://www.openiti.org), LectAuRep, Sofer Mahir (https://sofermahir.hypotheses.org), DIM STCN (http://www.dim-humanites-numeriques.fr), and CREMMA (https://www.dim-map.fr/projets-soutenus/cremma/), among others.

then top to bottom (such as Greek and Latin), or in lines from right to left and then top to bottom (such as Hebrew or Arabic), or in columns from top to bottom and then left to right (as is often the case for Chinese and Japanese). There are, however, other cases, such as lines read from top to bottom and then right to left (such as Mongolian), or columns read from bottom to top (such as some inscriptions in Old Javanese). Furthermore, some scripts like Latin are (usually) written on a baseline, while others like Hebrew are (usually) written from a top-line, and Arabic can be written along short diagonal segments which form a line overall. Writing can also be circular, for instance on coins, or spiral-shaped, for instance on prayer-bowls, or radiating out from a central point, for instance in Arabic marginal glosses (Figure 11.3), or in very complex shapes that form pictures, for instance in micrography or calligrammes. The situation is even more complex in multigraphic contexts, that is, where different writing-systems are mixed in a single document, such as seventeenth-century liturgical manuscript written in Chinese and Hebrew, to take just one example from many thousands[10]. The international Unicode standard for representing the world's writing systems in computers describes an algorithm for treating bidirectional documents (such as mixing English and Arabic); this standard is in its forty-second revision at the time of writing and currently extends to nearly eighteen thousand words, or around forty-five pages if printed. This illustrates the complexity of simply displaying a typed document with different directions of script, let alone that of automatically transcribing such a document from an image and then presenting it in an interface where the user can correct the lines, regions and transcription, with all the text being displayed in the correct directions as required.

This variety of writing-systems, and particularly the need to cater for so called "rare" and historical scripts, introduces further challenges than directionality. By definition, "rare" languages and scripts do not have large corpora and are not already well catered for by existing software and methods. Indeed, the very nature of Deep Learning is that, as we have already seen, it requires large amounts of pre-existing material, and in general the more such material the better (provided that the data is sufficiently representative). This means that, almost by definition, methods that rely on "big data" are not appropriate for "rare" languages and scripts. In practice these methods can usually be used anyway, to some extent, as long as the corpus is not very small or very heterogeneous, but they will almost always be small "boutique" projects that will not have the support of large companies, for both better and worse, or reusability across domains. Furthermore, some of the basic techniques for improving the results of OCR/HTR will not work in these cases. For instance, it is very common to use some sort of statistical language processing to correct errors in the OCR of modern texts: a very simple example is to run a spell-checker on the result, but more complex examples attempt to automatically analyze the language and attempt to correct errors based on what is or is not linguistically possible. Such an approach can improve the transcrip-

[10]Hebrew Union College MS 926, available at https://mss.huc.edu/manuscripts/ms-926/

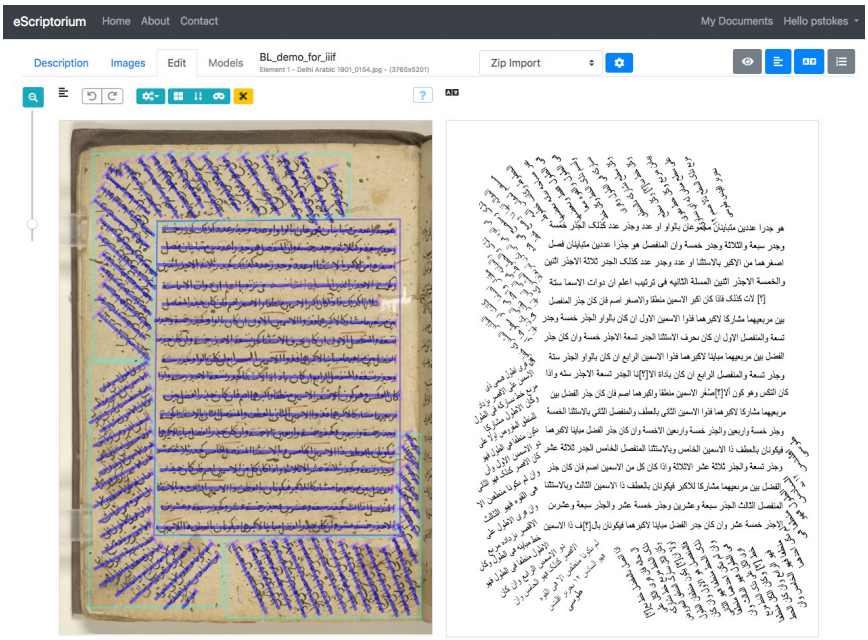


Figure 11.3: Visualizing complex page layouts in eScriptorium. The manuscript image used in this screen-shot is out of copyright and is from British Library and [14].

tion considerably, but it requires a pre-existing model of the language, so that the computer can recognize what is and is not likely to be an error. However, searches for a spell-checker for (say) Old Vietnamese is very unlikely to bear fruit any time soon, if ever, and indeed the same holds for accurate statistical models of orthographically varied historical writing in general.

11.6 Different Points of View

There is, however, a further aspect of this: as those of us in the Humanities know very well, there are many different conventions and possibilities when preparing editions. Despite the claims of some that a transcription must be a simple reproduction of "what is on the page", it is nevertheless clear that a written text contains an effectively infinite amount of information, and any transcription is necessarily an active selection and, in a sense, translation from one system of writing to another.[11] Even Latin texts have different conventions for transcription, depending on whether the context is Classical or Medieval, whether paleographical, epigraphical or papyrological, and so on, and the complexity multiplies enormously when considering practices for languages such as those of South-East Asia or even the Ancient Near East.[12] It is therefore impossible and indeed undesirable that the

[11] Discussions of this include [15, pg. 464-472], [16, pg. 85-101], [17], [18], and [19, pg. 20-29].
[12] One example of a transcription guide for such languages is [20]

computer automatically produce a transcription without any guidance from the user, since it is impossible to know a priori which standards of transcription the computer should follow. One can certainly imagine pre-preparing a list of common cases, following conventions established by significant scholarly bodies, and this would indeed be very useful and desirable, but it still seems certain that many other cases would remain. This need to accommodate different standards makes the reuse of models more difficult, since it increases the degree to which models must be retrained for specific cases. The challenge goes much further than this, however, and extends to the basis of any VRE for manuscript studies or textual editing, since it also means that any VRE which will be used by a wide range of people must be able to accommodate these different standards. Extending the workflow from transcription to edition introduces further complexities, as there are (also) many different types of edition, and the variety in editions is (probably) greater than that of transcriptions. Ideally, a single VRE should accommodate all standards and types of edition, as well as all standards of transcription, for all writing-systems written on all supports. Such a flexibility is possible, but it comes at a cost: either the interface must be extremely complex to account for all the different options, or it requires some level of programming in order to produce a customized interface which is specific to a given project. Indeed, this is perhaps the biggest challenge faced by the challenge of the Text Encoding Initiative (TEI). Many have complained that the TEI guidelines for text encoding are extremely complex and unwieldy, and that they do not proscribe a single convention for transcription meaning that, in effect, they are not a true standard. However, if the TEI Guidelines did impose a single convention then they would immediately become unusable for all those who want or indeed need to use other conventions. Texts are different, and editions and research projects have different goals and therefore different needs, even those projects that are studying the same texts.[13] In practice, then, the TEI is a sort of "meta-standard", from which one can then specify more restricted and proscriptive standards for particular contexts, with one of the more successful examples being Epidoc for editions of inscriptions.[22] VREs and other tools for the preparation and publication of digital editions face a similar challenge: it is certainly within the bounds of technology to develop a simple process whereby one can produce a transcript or even publish an edition very easily in a relatively small number of clicks, but this necessarily means that most of the decisions will be taken from the researcher and put into the hands of the tool-developers. Conversely, processes can also be developed which give manuscript specialists control over fine details according to their own needs, but this means complex interfaces and/or the need to actively write code at some level to customize the process.

This need for different standards, methods and points of view extends well beyond transcription and editions, and indeed goes to all forms of manuscript studies and indeed to all scholarly research in the humanities. Armando Petrucci

[13] An example of this is described by [21]

and Collette Sirat have both made similar observations for palaeography[14]:

> Infatti ogni terminologia paleografica è legata ad una particolare visione storica del fenomeno scrittorio ...ma legittimamente utilizzabili risulteranno comunque tutte quelle fondate su premesse metodologiche valide e su rigorose analisi grafiche. ([24, pg. 70-71][15])

> Two things which are similar are always similar in certain respects. ...Generally, similarity, and with it repetition, always presupposes the adoption of a point of view: some similarities or repetitions will strike us if we are interested in one problem or another. ([25, pg. 310])

This may seem obvious, but in fact it raises a fundamental epistemological question: annotation and comparison are core tasks of scholarship, and have been identified as two of the six "scholarly primitives" which underly all of our work.[16] However, annotation, or description, depend on terminology, as indeed does discovery, another of Unsworth's primitives, but as Petrucci has noted, there is no single terminology which can be claimed as "the" valid one over all others. Similarly, Sirat and Popper have noted that comparison also requires a point of view, and different problems require different comparisons and therefore different points of view. This poses a very significant challenge to VREs, and indeed to the ideal of interoperability and related areas such as Linked Open Data. In addition, the fundamental principles of machine learning itself, as a mere kind of statistical inference, can cast doubt on the scholarly value of the automatic productions of these methods when "hidden" inside VREs, with their design assumptions and limitations remaining relatively opaque to the humanities users even in open systems. In principle, different terminologies can be related through ontologies and other tools, such that one can record for both human and computer use that two given terms are close in meaning, exact matches, related, broader or narrower in scope, and so on, and in this way one can link different terminologies, at least in principle.[17] However, this is a complex and laborious process, and it also requires a very deep understanding of both terminologies as the scope for misunderstanding and error is enormous. We must therefore consider how to design VREs to enable different points of view, which includes allowing for different terminologies, different interfaces, different ways of presenting, annotating, comparing, searching and otherwise working with the data. Indeed, as Elena Pierazzo has observed, the text itself is only one part of the scholarly work and interpretation in an edition, and Johanna Drucker and others have shown that the interface generally is

[14] Some concrete examples of the impact of transcription on interpretation are given by [23, pg. 50-54]

[15] "In fact, every palaeographical terminology is connected to a particular historical vision of the phenomenon of handwriting ... but all terminologies prove legitimately useful nonetheless, if they are founded on valid methodological premises and rigorous graphical analyses" (our translation).

[16] These "scholarly primitives" are from an influential talk by [26].

[17] One important example of such a schema is the Simple Knowledge Organisation System (SKOS), for which see [27], esp. §10 Mapping Properties.

interpretative and embeds specific methods and viewpoints, and it therefore necessarily excludes others[15], [28]. For this reason, it seems unlikely that there can ever be a single VRE or other system which responds to all requirements, but instead perhaps we must accept that there will always be many different tools and frameworks. Certainly we must follow existing standards where these are available: otherwise there is no hope for interchange or data sharing, and our material is certain to be lost very quickly once our custom tool is no longer maintained and our custom data is therefore unreadable. Similarly, the best option remains to follow standards and to use tools that allow for the ready import and export of data, including trained models in the case of machine learning. In this way we have some chance of moving between different tools, frameworks and VREs as necessary, taking advantage of those that best respond to the point of view that we need for a given problem at a given moment. This also suggests favouring smaller modules that can be pieced together into different workflows as required, rather than large, centralised, monolithic VREs, and the piecing together in turn requires at least some understanding of data and (probably) basic programming. This should not be of concern to Classicists or others in the Humanities: certainly software development is a highly skilled profession that requires specialised training, but the basics of Python and XSLT are very much easier to learn than the complexities of Greek or Latin.

References

[2] B. Kiessling, "Kraken - A Universal Text Recognizer for the Humanities," *Proceedings of the DH*, 2019.

[3] B. Kiessling, R. Tissot, P. Stokes, and D. Stökl Ben Ezra, "eScriptorium: An open source platform for historical document analysis," in *2nd International Workshop on Open Services and Tools for Document Analysis, OST@ICDAR 2019, Sydney, Australia, September 22–25, 2019*, 2019, pp. 19–24.

[4] B. Kiessling, D. Stökl Ben Ezra, and M. T. Miller, "BADAM: A Public Dataset for Baseline Detection in Arabic-Script Manuscripts," in *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing, HIP@ICDAR 2019, Sydney, NSW, Australia, September 20-21, 2019*, ACM, 2019, pp. 13–18.

[5] P. Stokes. "eScriptorium: Un outil pour la transcription automatique des documents." (2020), [Online]. Available: https://ephenum.hypotheses.org/1412.

[6] ——, "eScriptorium 1-6." (2020), [Online]. Available: https://vimeo.com/channels/1602497.

[7] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 369–376.

[8] M. Diem, F. Kleber, S. Fiel, T. Grüning, and B. Gatos, "cBAD: ICDAR2017 competition on baseline detection," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, IEEE, vol. 1, 2017, pp. 1355–1360.

[9] C. Wigington, C. Tensmeyer, B. L. Davis, *et al.*, "Start, Follow, Read: End-to-EndFull-Page Handwriting Recognition," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VI*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., ser. Lecture Notes in Computer Science, vol. 11210, Springer, 2018, pp. 372–388. DOI: 10.1007/978-3-030-01231-1_23.

[10] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *2011 International Conference on Computer Vision*, IEEE, 2011, pp. 1457–1464.

[11] L. Kang, P. Riba, Y. Wang, *et al.*, "GANwriting: Content-Conditioned Generation of Styled Handwritten Word Images," Lecture Notes in Computer Science, vol. 12368, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., pp. 273–289, 2020. DOI: 10.1007/978-3-030-58592-1_17.

[12] P. A. Stokes, "The digital dictionary," *Florilegium*, pp. 37–65, 2009.

[13] OCR-D. "The ground-truth guidelines." (2019), [Online]. Available: https://ocr-d.de/en/gt-guidelines/trans.

[14] A. Keinan-Schoonbaert. "Results of the RASM2019 competition on recognition of historical arabic scientific manuscripts." (2019), [Online]. Available: https://blogs.bl.uk/digital-scholarship/2019/09/rasm2019-results.html (visited on 09/13/2019).

[15] E. Pierazzo, "A rationale of digital documentary editions," *Literary and linguistic computing*, vol. 26, no. 4, pp. 463–477, 2011.

[16] ——, *Digital scholarly editing: Theories, models and methods*. Routledge, 2016.

[17] C. Huitfeldt and C. M. Sperberg-McQueen, "What is transcription?" *Literary and linguistic computing*, vol. 23, no. 3, pp. 295–310, 2008.

[18] M. Sperberg-McQueen and C. Huitfeldt, "Interpreting difference among transcripts," *Digital Humanities 2018: Book of Abstracts/Libro de resúmenes.*, 2018.

[19] P. Robinson and E. Solopova, "Guidelines for transcription of the manuscripts of the Wife of Bath's prologue," *The Canterbury Tales Project Occasional Papers*, vol. 1, pp. 19–52, 1993.

[20] D. Balogh and A. Griffiths, "DHARMA Transliteration Guide," Version 3 of a common transliteration system for Balinese, Cam, Javanese, Kannada, Khmer, Malay, Prakrit, Sanskrit, Sundanese, Tamil, Telugu., 2020. [Online]. Available: https://hal.archives-ouvertes.fr/halshs-02272407.

[21] P. A. Stokes and G. Noël, "Project Report," King's College London, United Kingdom, Tech. Rep., 2010. [Online]. Available: http://www.ascluster.org/techinfo/.

[22] T. Elliott, G. Bodard, and H. Cayless. "Epidoc: Epigraphic documents in tei xml." (2006–2020), [Online]. Available: http://epidoc.stoa.org/.

[23] P. A. Stokes, "Palaeography, Codicology and Stemmatology," in *Handbook of Stemmatology: History, Methodology, Digital Approaches*, P. Roelli, Ed., Walter de Gruyter GmbH & Co KG, 2020, pp. 46–56. DOI: 10.1515/9783110684384-002.

[24] A. Petrucci, *La descrizione del manoscritto. Storia, problemi, modelli. Seconda edizione corretta e aggiornata*. Roma, 2001.

[25] C. Sirat, L. Schramm, Ed., ser. Bibliologia elementa ad librorum studia pertinentia. Turnhout: Brepols, ISBN: 2-503-52116-9.

[26] J. Unsworth, "Scholarly primitives: What methods do humanities researchers have in common, and how might our tools reflect this," in *Symposium on Humanities Computing: Formal Methods, Experimental Practice. King's College, London*, vol. 13, 2000, pp. 5–00.

[27] A. Miles and S. Bechhofer, "SKOS simple knowledge organization system reference," *W3C recommendation*, 2009.

[28] J. Drucker, *Graphesis: Visual forms of knowledge production*. Harvard University Press, 2014.

Chapter 12

Conclusion and Perspectives

The focus of this thesis has been the retrodigitization of historical, handwritten and printed Arabic documents with an emphasis on its use in the Digital Humanities while being mindful of the need to reduce the bias of methods towards a particular script. More specifically, we have gained a deeper understanding of the limitations and capabilities of Arabic-script text transcription and layout analysis methods, adopted and extended the baseline paradigm for layout analysis, developed a simple system for multigraphic OCR, and integrated those methods in the open Kraken OCR software. The flexibility of this engine is attested through both its use for novel computer vision tasks like the character alignment presented in chapter 9, its straightforward integration into other workflows such as eScriptorium and OCR-D, and its use for a multitude of different languages, scripts, and document types. The eScriptorium VRE is a crucial puzzle piece in the retrodigitization ecosystem, both in its current state as a platform for accessible annotation and transcription and a foundation stone for the envisioned deep annotation incorporating more computational methods into the primitives of scholarly practice.

While the process is often rough around the edges and still requires some tinkering by the end user, understandably given the significant historical deficit of Arabic-script OCR in comparison to the treatment of other writing systems, we can now, in principle, digitize almost arbitrary documents with relatively modest training data requirements. We hope that the open nature of both eScriptorium and Kraken with its facilities to encourage sharing of training data and artifacts will be a part in this catch up process which will likely reduce the gap in accuracy and complexity between the digitization of "normal" and "rare" scripts in the comming years.

It is clear that substantial work remains. The most pressing requirements on an Arabic OCR system from our initial studies, layout analysis, segmentation-less transcription, and data annotation and curation, are largely solved but others remain in an unsatisfactory state.

The most immediate of those remaining tasks for Arabic manuscript OCR is reading order determination. As described in chapter 1, research on this topic is sparse, methodologically out-of-date, and largely focused on modern documents. Hand-crafted heuristics, incapable of accurately dealing with the complex structure of historical manuscripts, prevail in practical OCR systems. Datasets for evaluation and training of methods are non-existent or annotated implicitly in datasets for other tasks without communicated standards. Nevertheless, the capacity of machine learning algorithms for spatial reasoning with a large number of objects has grown in recent years and architectures like graph neural networks[1] and deep models for learning-to-rank such as [2], [3] offer the promise to advance the state of the art substantially in the near future.

While current OCR system are capable of impressive feats even on highly degraded, fragmentary, or atypical documents with requirements on training data in volume and complexity that is lower than ever before, training data acquisition remains the most labor- and time-intensive step of any retrodigitization project. Basic transfer learning functionality in Kraken reduces this workload to some

extent but its use is *ad hoc* and its practical efficacy depends heavily on the skills of the individual user.

Two non-exclusive avenues for more effective model adaptation are currently imaginable: a systematization of the data acquisition process, enriching datasets and models with metadata that aids in their manual or automatic selection for new material, and semi- and unsupervised model adaptation. Some digitization projects, such as OCR-D for Latin and OpenITI for Arabic, have started to enlist codicological support to put the selection and generation of training data on firmer scientific grounds [4], but this remains a rarity. Semi-supervised methods such as [5] and unsupervised training [6] on the other hand aims to reduce or remove human annotation in data generation completely. These approaches are considered one of the *holy grails* of DIA but even if methods were to reach the maturity for practical use important questions such as how to deal with different transcription standards effectively remain unanswered.

The development of eScriptorium will continue for the foreseeable future and will integrate advances in automatic computational methods to the extent that these aid in humanities research. Chiefly among these and not extensively considered in this thesis is deep annotation of textual and non-textual components. Applications such as image classification, document dating, automatic grouping of documents, and text reuse detection are imaginable. Additional means to share, publish, and distribute training data, models, and scholarly products in a way that is acceptable to both scholarly practice in the humanities and computer science are an important part of development in the near-term.

These means are extraordinarily important as even with the current availability of powerful and open computer vision tools, the dataset landscape remains fractured. While more and more researchers recognize the importance of the sharing not only to advance research in the humanities but also to direct computer vision research and practice in a way that is of utility to the humanities, the best current practice for public datasets remains a profane github repository with a non-descriptive README file. A combination of VREs conscious of the important of appropriate metadata, expanded input of archival and library practice in research activities, and the utilization of research data repositories already commonplace in other disciplines, has the potential to significantly improve this state of affairs in the coming years.

References

[1] H. Déjean, J.-L. Meunier, *et al.*, "Versatile layout understanding via conjugate graph," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 287–294.

[2] Z. Tan, X. Nie, Q. Qian, N. Li, and H. Li, "Learning to Rank Proposals for Object Detection," in *2019 IEEE/CVF International Conference on Computer*

Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, IEEE, 2019, pp. 8272–8280. DOI: 10.1109/ICCV.2019.00836.

[3] F. Çakir, K. He, X. Xia, B. Kulis, and S. Sclaroff, "Deep Metric Learning to Rank," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 1861–1870. DOI: 10.1109/CVPR.2019.00196.

[4] N. Weichselbaumer, M. Seuret, S. Limbach, *et al.*, "New approaches to ocr for early printed books," *DigItalia*, vol. 2, pp. 74–87, 2020.

[5] S. Keret, L. Wolf, N. Dershowitz, *et al.*, "Transductive learning for reading handwritten tibetan manuscripts," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 214–221.

[6] A. Gupta, A. Vedaldi, and A. Zisserman, "Learning to Read by Spelling: Towards Unsupervised Text Recognition," in *ICVGIP 2018: 11th Indian Conference on Computer Vision, Graphics and Image Processing, Hyderabad, India, 18-22 December, 2018*, ACM, 2018, 33:1–33:10. DOI: 10.1145/3293353.3293386.

Appendices

Appendix A

Résumé Long

A.1 Introduction

Cette thèse consiste en un certain nombre de publications qui étudient les difficultés de la rétrodigitalisation de l'écriture arabe historique et propose plusieurs méthodes pour faire progresser l'état de l'art en la matière. Bien que ces méthodes visent principalement à remédier les problèmes résultant des caractéristiques de l'écriture arabe, elles sont conçues pour être applicables à l'écriture et aux inscriptions dans une variété d'autres systèmes d'écriture.

A.2 L'Analyse d'Images de Documents et Reconnaissance Optique de Caractères

L'analyse d'images de documents (AID) est un sous-domaine de la vision par ordinateur (VO) qui cherche à comprendre le contenu des documents par le traitement de leurs images numériques. Par document, nous entendons les documents manuscrits et le texte imprimé sur papier, mais aussi l'écriture sur d'autres supports souples -tels que le papyrus ou les feuilles de palmier- ou même les inscriptions.

La différence par rapport à la vision par ordinateur en général ne se trouve pas dans les méthodes employées, mais dans la nature des images traitées. Ces images sont généralement obtenues par des caméras ou des scanners, souvent dans un contexte professionnel, ce qui permet d'obtenir un matériel source avec un minimum de bruit provenant d'éléments non pertinents (que l'on rencontre souvent dans les images de scènes naturelles traitées par d'autres branches du VO). Malgré des données d'entrée plus propres, les représentations structurées souhaitées en sortie ont tendance à être plus complexes et plus nombreuses avec l'AID qu'avec d'autres applications, car elle nécessite la détection, la classification et la mise en relation de dizaines d'éléments de documents tels que des lignes, des caractères, des illustrations et des tableaux.

Comme dans d'autres domaines de l'informatique, la recherche en AID peut être divisée en tâches spécifiques, dont une ou plusieurs sont résolues par une méthode particulière proposée. La tâche la plus importante de l'AID est la reconnaissance optique de caractères (ROC), mais il en existe d'autres (basées ou non sur la ROC) telles que la classification de documents, la datation ou le repérage par mot clé. La ROC est une conversion de textes imprimés, écrits ou inscrits, en texte codé par machine. C'est un processus établi depuis longtemps, aussi bien dans la recherche sur la vision par ordinateur que dans des applications comme l'analyse des adresses sur les courriers ou l'assistance aux personnes aveugles. Cette dernière est sans doute à l'origine de toute AID, avec un certain nombre de brevets remontant au début du XIXe siècle.

Ces premières approches, mises au point des décennies avant les premiers ordinateurs, ont maintenant évolué. Aujourd'hui, les technique d'AID sont utilisées quotidiennement pour la distribution du courrier, la vérification des chèques ou la rétro-numérisation des livres. En effet, on affirme aujourd'hui dans la profession que la ROC est fondamentalement résolue, au moins pour les documents modernes imprimés en anglais avec un niveau de bruit raisonnablement faible, où les logiciels commerciaux modernes de rétrodigitalisation atteignent régulièrement des taux de précision des caractères supérieurs à 99%. Toutefois, la ROC ne peut transcrire des écritures dont la typographie n'est pas conforme aux pratiques occidentales modernes, et ce même pour les écritures purement alphabétiques telles que le latin et le cyrillique. De plus, il existe près de 4000 langues écrites et plusieurs centaines de systèmes d'écriture, pour la grande majorité desquels il n'existe aucun système de ROC. Ainsi, il est clair qu'une part importante de la production littéraire humaine moderne n'est pas encore accessible à la rétronumérisation.

C'est encore plus clair en ce qui concerne le matériel historique. Force est de constater que si les projets de numérisation à grande échelle dans les pays riches ont permis de créer de vastes bibliothèques numériques, ces dernières sont de facto inutilisables. En effet, la reconnaissance par les logiciels adaptés aux documents modernes dégrade considérablement la qualité du texte en raison des variation orthographiques et typographiques, même pour des documents aussi récents que ceux de la fin du XIXe siècle.

Il s'agit très probablement d'un état temporaire pour les documents les plus nombreux dans les archives des pays développés où des projets tels que OCR-D[1] ouvrent la voie pour transférer les progrès de la recherche pure en AID à la pratique des bibliothèques. Dans le cadre de ces projets et d'efforts plus spécifiques tels que [1], un programme de recherche collective pour la numérisation du matériel historique et des systèmes d'écriture minoritaires a vu le jour, partagé par les chercheurs en humanités engagés dans les méthodes numériques et les experts de la vision par ordinateur. Néanmoins, ces communautés restent géographiquement isolées les une des autres, ainsi que par les barrières linguistiques et

[1] http://ocr-d.de

professionnelles.

D'autre part, le manque de financement combiné à la détérioration des collections dans les pays du Sud en raison de conflits et d'influences environnementales augmente le risque de perte permanente du patrimoine culturel, qui n'intéresse que des populations minoritaires et un petit nombre de chercheurs. Même des collections célèbres telles que les manuscrits de Tombouctou ont à peine échappé à la destruction au cours des conflits armés de ces dernières années et sont aujourd'hui menacés par l'humidité.

A.2.1 Tâches

En tant qu'application centrale de l'analyse d'images de documents, la ROC s'est divisée en un grand nombre de sous-tâches. Elles ne sont pas toutes strictement nécessaires à un système de ROC fonctionnel et, en fait, beaucoup d'entre elles ne peuvent être mises en œuvre que de manière spécifique en fonction des matériaux. Elles sont donc reléguées à des applications spécialisées.

Le pipeline typique de reconnaissance optique de caractères est construit autour de quatre grandes étapes :

Prétraitement Débruitage, redressement, binarisation.

Segmentation des pages Extraction des informations structurelles des images de pages de documents et enrichissement avec des informations sémantiques supplémentaires.

Transcription Extraction des informations textuelles de la totalité ou d'un sous-ensemble d'objets identifiés à l'étape précédente.

Si cette caractérisation est valable pour tous les pipelines de données, sauf les plus ésotériques, les blocs fonctionnels exacts dépendent fortement du type et de la structure du document à traiter. Chacune de ces étapes de traitement contient une ou plusieurs méthodes qui servent à résoudre une tâche particulière, comme par exemple :

Binarisation Classification des pixels d'une image en deux classes : le front, c'est-à-dire le texte, et le fond, c'est-à-dire tout le reste.

Débruitage Augmentation de la qualité de l'image de la page pour les tâches suivantes. Le débruitage comprend des processus tels que la normalisation de fond ou l'élimination des imperfections.

Redressement Correction à la fois de la distorsion de perspective inhérente à la capture par caméra, et des autres dégradations introduites pendant le scan telles que la rotation, la déformation le long de la reliure, etc.

Segmentation en regions Subdivision d'une image de page en éléments tels que du texte, de la décoration, des notes, des points, etc.

Segmentation en lignes Extraction des lignes de texte d'une image de page.

Segmentation en caractères Segmentation du texte sur une image de page jusqu'au niveau du glyphe ou même à un niveau inférieur. Bien qu'il s'agisse d'une opération courante dans les systèmes de ROC traditionnels, elle est le plus souvent superflue avec les méthodes de pointe.

Classification du système d'écriture et de la police de caractères Classification de la langue, du système d'écriture, du style ou de la police de caractères du texte. Cette classification peut être effectuée à différents niveaux, par exemple à l'échelle du document, ou pour tout ou partie d'une ligne.

Reconnaissance des tableaux Inférence de la structure logique des images d'un tableau.

A.3 Motivation et contributions scientifiques

L'écriture arabe représente l'une des plus grandes traditions littéraires de l'histoire de l'humanité, tant en termes de volume que de diffusion géographique. Les exemples vont des textes religieux, en particulier le Coran, le livre saint de l'Islam, à la poésie, en passant par les textes scientifiques et juridiques, sans oublier un vaste corpus de documents administratifs. Le nombre considérable de ces textes dans une multitude de domaines en fait une cible privilégiée pour les nouveaux paradigmes des humanités, qui utilisent des méthodes computationnelles telles que la lecture à distance et la paléographie quantitative. Ces méthodes nécessitent soit de grands corpus de textes, soit des méthodes précises d'AID basées sur une ou plusieurs des tâches susmentionnées d'un système de ROC. Comme la grande majorité des textes arabes n'a jamais existé sous forme numérique, la rétro-numérisation de haute qualité par la ROC constitue la base d'un nombre important de projets de recherche en humanités numériques arabes.

Lorsque j'ai commencé à travailler sur cette thèse, la ROC du texte arabe imprimé était largement rejetée par les chercheurs en humanités travaillant sur ces documents, même ceux qui étaient déjà très impliqués dans les humanités numériques, ainsi que par les organismes de financement, encore plus pour le matériel historique ou multigraphique. En effet, la reconnaissance précise de l'écriture à la main arabe semblait totalement inatteignable. Bien qu'il existe une longue tradition de publications sur la ROC d'ensembles de données arabes simples tels que KHATT[2] et qu'un certain nombre de logiciels de ROC libres et propriétaires tels que Tesseract, Abbyy FineReader et Sakhr offrent un soutien nominal pour la reconnaissance de textes en caractères arabes, ces solutions ne se sont jamais concrétisées dans la pratique réelle des bibliothèques ou à grande échelle. Les raisons en sont multiples : taux d'erreur élevé des classificateurs et des segmenteurs mal adaptés à la nature cursive du système d'écriture, manque de logiciels et de compétences techniques, et coûts et efforts substantiels nécessaires pour adapter les solutions existantes au matériel d'intérêt.

Il est rapidement apparu que les obstacles qui empêchent les chercheurs travaillant sur des textes imprimés et manuscrits en caractères arabes d'utiliser la ROC dans la pratique sont les mêmes que ceux rencontrés par de nombreux autres chercheurs engagés dans la rétro-numérisation de documents historiques et non-latins. Imitant l'opinion dominante sur la ROC arabe, [3] a revendiqué que les manuscrits médiévaux (en caractères latins) étaient pratiquement imperméables à la ROC contemporaine. On peut probablement trouver des évaluations similaires pour d'autres domaines.

En tant que telle, cette thèse doit être considérée à l'intersection entre les humanités et l'informatique. La recherche qui y est présentée n'est pas un ensemble de méthodes séparées permettant de résoudre des tâches dans l'AID arabe, mais fait partie d'un écosystème cohérent pour l'AID des humanités au sens large. Celui-ci est composé de deux éléments principaux : le système de ROC Kraken et l'environnement de recherche virtuel (ERV) eScriptorium.

Le Kraken est un système de ROC complet et modulaire. Il se distingue des autres logiciels pour plusieurs raisons : ses utilisateurs cibles sont des chercheurs en humanités (et non des informaticiens), et il permet une flexibilité maximale dans le type de documents qu'il est possible de traiter. Il a également été largement adapté en vue d'un meilleur traitement des documents historiques en caractères non latins, notamment en arabe.

Dans le cadre des travaux d'adaptation visant à faire de Kraken un outil plus performant pour le travail sur les textes arabes, deux études de faisabilité de rétro-numérisation ont été réalisées (la première à partir de livres imprimés en caractères arabes classiques, et la seconde sur une importante revue sur la langue arabe publiée par l'Université Américaine de Beyrouth). Ces études ont produit la première analyse détaillée sur les faiblesses et les capacités des méthodes de ROC de pointe sur les textes arabes imprimés.

Cette thèse contribue de deux systèmes de segmentation de lignes entraînables, un système élémentaire capable de détecter uniquement les lignes de base, et un système plus avancé permettant la segmentation des régions et des lignes en plus de la classification. Ce dernier est inclus dans le Kraken et permet la détection conjointe de régions et de lignes et l'inférence de l'orientation des lignes. Cette seconde méthode a également été optimisée pour l'utilisation de la mémoire, réduisant la consommation de mémoire d'environ cinquante pour cent par rapport aux réseaux de neurones à segmentation sémantique à convolution totale, ayant des performances similaires.

Un nouveau backend de réseau neuronal a été ajouté au Kraken. Basé sur la bibliothèque de réseaux de neurones pytorch, il permet la reconfiguration flexible des réseaux de neurones artificiels (ARN) utilisés à la fois pour la segmentation des pages et la transcription du texte, grâce à un langage de définition d'ARN léger capable d'exprimer de nombreuses caractéristiques d'architectures ARN courantes utilisées dans les tâches de vision par ordinateur. Cette couche d'abstraction permet l'ajout relativement simple de nouveaux types d'ARN, et donc un prototypage rapide et une optimisation efficace des hyperparamètres, comme l'a démontré [4].

Au cours des premières études de cas, plusieurs milliers de lignes de données d'entraînement pour la transcription de textes ont été annotées et mises à la disposition du public. J'ai participé à la conceptualisation technique et à la definition des standards de transcription pour ces ensembles de données. Un autre ensemble de données sous licence libre de quatre cents pages de manuscrits en écriture arabe, dans une variété de langues, de styles et de domaines, a été annoté avec des lignes de base et l'orientation des lignes pour permettre l'évaluation des méthodes d'analyse de la mise en page sur des documents historiques en écriture arabe. Cet ensemble de données très complexe reste le seul ensemble de données manuscrites non latines pour le paradigme de base de l'analyse de la mise en page.

La deuxième composante de cet écosystème de rétro-numérisation est l'ERV eScriptorium. Alors que le Kraken est conçu pour une flexibilité maximale, l'eScriptorium adopte une autre approche : il est conçu comme un environnement de recherche paléographique et de publication à part entière pour un usage scientifique. La fonctionnalité ROC n'est qu'une petite partie des caractéristiques prévues. Comme il n'est pas pratique d'exposer toutes les fonctionnalités du Kraken sur une telle plateforme, la conception d'eScriptorium permet une intervention manuelle et semi-automatique à chaque étape du processus, soit par une manipulation manuelle dans l'interface, soit par des interfaces d'échange de données graphiques et programmatiques.

Comme eScriptorium vise à offrir des fonctions scientifiques supplémentaires au-delà de la simple rétro-numérisation, la plateforme est également un cadre de test idéal pour la recherche en humanités assistée par la vision artificielle. Dans certains cas, ces fonctions avancées sont liées à la transcription de textes. Une méthode permettant de dériver les emplacements des graphèmes à partir de l'alignement implicite produit par une reconnaissance de texte par ligne ANN et son évaluation sur des fragments de matériel hébraïque est présentée.

A.4 L'Ecriture Arabe

Le système d'écriture arabe est l'un des systèmes d'écriture les plus répandu de l'histoire de l'humanité, géographiquement et chronologiquement. Il s'agit de l'écriture principale de la langue arabe, du farsi, de l'ourdou et de plusieurs autres langues du sous-continent indien. Historiquement, il a été utilisé pour produire des textes de l'espagnol au chinois.

Bien que ses origines exactes soient encore controversées, les historiens s'accordent à dire que l'écriture arabe a évolué à partir de l'écriture nabatéenne ou syriaque au Moyen-Orient au cours de plusieurs siècles, la maturation ayant eu lieu au cours du septième siècle de notre ère. Étroitement liés à la propagation de l'Islam, un certain nombre de variantes alphabétiques et de styles calligraphiques régionaux se sont développés au cours des siècles suivants. Néanmoins, elle est loin d'une écriture purement liturgique avec une abondance de documents administratifs, de traités philosophiques et scientifiques, de poésie, etc.

Il est donc erroné de parler d'une seule écriture arabe du point de vue de recherche de l'AID. Chaque style, avec ses particularités régionales et son contexte culturel, présente des défis particuliers.

A.4.1 Les Principes de l'Ecriture Arabe

L'écriture arabe est un *abjad*, un système d'écriture consonantique, ce qui signifie qu'elle ne nécessite que l'écriture des consonnes et des voyelles longues, le lecteur étant censé fournir lui-même les voyelles courtes appropriées selon le contexte. Les voyelles courtes et autres marques pour des caractéristiques telles que le doublage (gémination) et la nunation (ajout d'un n final), peuvent être ajoutées en option (*tashkil*) mais ne sont systématiquement utilisées que lors de la transcription du Coran ou de textes élémentaires pour les apprenants en langues. Comme le syriaque et l'hébreu, il s'écrit de droite à gauche, à l'exception des nombres qui s'écrivent de gauche à droite.

Contrairement à d'autres écritures très courantes comme le latin, l'arabe est écrit uniquement dans une forme cursive. Comme les variantes cursives d'autres écritures, les lettres individuelles changent ses formes en fonction de leur position dans un mot : des formes initiales, médianes, finales et indépendantes existent, en plus des règles calligraphiques spécifiques au style pour le placement des lettres. Une différence avec l'écriture cursive des autres écritures est que toutes les lettres ne peuvent pas être reliées entre elles. Comme les espaces n'indiquent pas nécessairement le début d'un nouveau mot, les calligraphes sont largement libres de faire varier l'espacement entre les mots, les syllabes et les traits comme ils le souhaitent. Cette variation de l'espacement peut perturber les systèmes de reconnaissance optique des caractères.

Une autre distinction est l'absence de majuscule et la ponctuation de type occidental, cette dernière n'ayant été introduite qu'au XXe siècle de notre ère. Au lieu de cela, des phrases et des mots particuliers sont utilisés pour introduire une nouvelle phrase ou une question, et les accents sont placés au-dessus des titres et des rubriques.

Il existe d'autres formes de lettres dans des variantes de l'écriture arabe adaptées à d'autres langues qui comportent des phonèmes qui n'existent pas en arabe. Elles sont généralement adaptées en ajoutant des points aux graphèmes représentant des sons similaires en arabe, comme le persan *pe* dérivé de *bāʾ* en ajoutant deux points supplémentaires en-dessous. Comme il y a beaucoup de langues qui sont ou ont été écrites en arabe, il existe un grand nombre de ces variantes.

Une difficulté particulière pour les systèmes de ROC est la façon particulière dont le texte arabe est justifié. La césure, c'est-à-dire la division des mots pour faciliter le retour à la ligne, est absente de tous les textes, sauf les plus anciens. Au lieu de la justification par espace blanc, courante dans les écritures alphabétiques, des alternatives plus agréables visuellement ont été conçues. L'allongement des liens entre les lettres, la courbure de la ligne de base et la superposition ou le déplacement de fragments du dernier mot au-dessus ou à côté de la ligne sont des

phénomènes courants. Ces deux dernières méthodes posent un défi particulier, même aux systèmes modernes de segmentation des pages.

A.4.2 Styles

Un grand nombre de styles calligraphiques ont été conçus au cours des siècles. Si certains sont reconnus dans l'ensemble du monde islamique, comme le *naskh*, d'autres sont spécifiques à certaines zones géographiques, par exemple le *maghribī* nord-africain ou le *nasta'līq* persan. Les premiers styles tels que le *ḥijāzī* et le coufique sont des familles de styles différents, car la standardisation était faible. Apres XIIIe siècle de notre ère, on compte six styles canoniques. Ils sont généralement appariés, une écriture d'affichage (majuscule) et une écriture de texte (minuscule) :

- *thuluth* et *naskh*
- *muḥaqqaq* et *rayḥān*
- *tawqīᶜ* et *riqāᶜ*

Les styles régionaux n'étaient pas seulement liés aux zones géographiques mais aussi à l'utilisation de la langue. Dans les parties du monde islamique influencées par le persan (Empire Ottoman, Iran, Inde), des styles suspendus tels que *nasta'līq* avec des mots individuels descendant sur une ligne de base commune étaient populaires car ils étaient plus adaptés aux différentes combinaisons de lettres en turc et en persan. D'autres styles étaient à la fois géographiquement limités et restreints à certains usages comme le *divanî* style de la chancellerie ottomane.

A.4.3 Critères pour les systèmes de ROC arabes

En résumé, la reconnaissance des textes arabes imprimés et manuscrits nécessite un certain nombre de caractéristiques que l'on ne trouve pas couramment dans les systèmes de ROC actuels. Les principales exigences sont, dans l'ordre de traitement à l'intérieur d'un pipeline typique :

Élimination de la binarisation La variété des supports, des encres et des décorations utilisés dans les textes arabes rend peu probable le développement d'une méthode générale de binarisation. Un système de ROC arabe devrait donc être sans binarisation.

Segmentation des lignes courbes et inclinées L'utilisation fréquente de lignes inclinées et courbes à des fins tant pratiques qu'esthétiques nécessite une méthode de segmentation des pages capable de les extraire et de les représenter efficacement.

Segmentation sémantique des pages L'utilisation extensive du paratexte nécessite un système de segmentation capable de séparer plusieurs textes sur la même page de document.

Détermination de l'ordre de lecture avancée En plus de classifier les différents éléments textuels d'une page, il est également nécessaire de les mettre dans le bon ordre pour une véritable compréhension du document.

Transcription sans segmentation Comme l'arabe s'écrit uniquement sous forme cursive et que les liaisons entre les lettres peuvent changer de manière significative d'un style à l'autre, une méthode de transcription arabe devrait fonctionner sur des lignes entières au lieu de les séparer en caractères.

Les outils de création et de conservation des données Même s'ils ne font pas directement partie d'un pipeline de ROC, les outils ergonomiques qui peuvent être utilisés pour annoter toute la gamme des caractéristiques du texte arabe sont essentiels pour les projets de numérisation concrets et pour créer des ensembles de données pour les recherches futures.

A.5 Études sur la ROC en écriture arabe

Deux études de précision ont été réalisées en 2017 et 2018 sur un certain nombre de textes classiques en écriture arabe et sur la principale revue scientifique de langue arabe, al-Abhath. Ces études ont utilisé le système de ROC Kraken (qui a depuis évolué) afin de déterminer les points forts et les limites des logiciels de ROC de pointe sur ces documents. En conséquence, nous avons déterminé deux recommandations clés pour améliorer substantiellement la ROC en écriture arabe : (1) une approche plus systématique de la production de données de l'entrainement, et (2) le développement de composants technologiques clés, en particulier des modèles multilingues et une meilleure méthode de segmentation des lignes et de la mise en page.

La première étude préliminaire a montré que malgré les faibles taux d'erreur de caractères (<3%) obtenus par les modèles spécifiques aux polices de caractères, les différences substantielles entre les polices de caractères se traduisent par des taux d'erreur nettement plus élevés desdits modèles sur des polices de caractères dissemblables. Bien que cette mauvaise généralisation puisse être attribuée dans une certaine mesure aux limites du réseau de transcription (LSTM peu profond entraîné sans régularisation) dans cette première version du Kraken, elle a montré la nécessité à la fois d'un backend de réseau neuronal plus puissant dans le système et d'une sélection minutieuse des données d'entraînement pour assurer la représentativité de l'ensemble des données d'entraînement pour le domaine cible souhaité.

La seconde étude a été construite dès le début autour d'une analyse beaucoup plus rigoureuse des polices de caractères présentes dans le matériel et d'une éva-

luation manuelle approfondie des résultats de la ROC par rapport à une évaluation purement computationnelle. Son sujet, la revue al-Abhath, comportait deux groupes de polices de caractères. Néanmoins, il y avait de légères variations intra-groupe, en particulier pour la première police de caractères qui a été utilisée pour la majorité du cycle de la revue. Pour se conformer à cette analyse, il a été décidé de produire 5000 lignes de données d'entraînement pour la première police de caractères et 2000 pour la seconde. Les lignes d'entraînement ont été tirées au hasard des deux groupes de caractères et transcrites manuellement avec le logiciel CorpusBuilder. Des modèles de transcription spécifiques aux polices de caractères ont été entraînés sur les deux ensembles de données, puis évalués par un prestataire extérieur et par OpenITI sur un ensemble de validation distinct (voir tableaux 4.1 et 4.2). À l'issue de l'examen, il est apparu clairement que Kraken surpasse significativement le logiciel ROC d'Abbyy pour la reconnaissance de textes arabes. Une évaluation manuelle détaillée de la transcription automatique a permis d'identifier plusieurs types d'erreurs courantes : mauvaise reconnaissance de ligatures peu courantes, de formes de lettres rares, de caractères d'allongement et de texte multigraphique, ainsi que la sortie de lettres doublées. Certains de ces problèmes ont pu être résolus grâce à des données d'entraînement de meilleure qualité. Une methode pour le traitement de texte multigraphique dans le système de ROC a été ajouté subséquemment (chapitre 7). L'évaluation a également identifié le besoin d'une meilleure segmentation des lignes des textes arabes, car le module optimisé pour le latin dans le Kraken avait tendance à tronquer et à diviser les lignes de texte.

A.6 Segmentation des pages

Pour le traitement des documents imprimés et manuscrits en caractères arabes, il est évident qu'une méthode d'analyse de la mise en page flexible, et de préférence entraînable, est nécessaire. En effet, l'impossibilité d'extraire des lignes de texte rend automatiquement impossible leur transcription correcte avec n'importe quelle méthode de reconnaissance de texte.

Alors qu'il existe un grand nombre d'ensembles de données d'entraînement suivant différentes représentations de lignes de texte pour les documents modernes et historiques en écriture latine, ce n'est pas le cas pour les manuscrits arabes. Pour aider au développement, nous avons préparé un ensemble de données de 400 pages de manuscrits arabes et persans annotés avec leurs lignes de texte.

Il est important de comprendre la nature exacte de la segmentation des lignes de texte dans un pipeline de ROC et son lien avec la méthode de transcription. Si l'objectif premier est d'identifier les lignes de texte, la tâche d'un segmenteur de lignes de texte est également d'aider à l'extraction de ces lignes de manière à optimiser les performances de la transcription. Ainsi, différentes représentations de lignes de texte peuvent être produites par un système de segmentation, par

exemple des boîtes englobantes, des polygones englobants, des nuages de pixels, des lignes de base, etc. Tous ces éléments ne sont pas adaptés aux conventions calligraphiques de l'écriture arabe. Par exemple, le tracé d'un rectangle englobant une ligne courbe ou inclinée inclura nécessairement des parties de lignes adjacentes à l'intérieur du rectangle. En outre, la précision de la transcription est améliorée lorsque les lignes sont centrées à l'intérieur de la bande d'image rectangulaire introduite dans le réseau de transcription.

Une représentation capable à la fois d'encoder une ligne sans bruit adjacent et de permettre la normalisation de la ligne sur une ligne droite est donc hautement souhaitable pour les documents qui contiennent des lignes non droites avec une certaine régularité. Pour l'ensemble de données, la représentation de la ligne de base a été choisie car elle est à la fois rapide à annoter, relativement facile à apprendre par les méthodes de vision par ordinateur, et suffisamment polyvalente pour permettre la normalisation de lignes de forme arbitraire.

Une ligne de base est une ligne virtuelle sur laquelle la plupart des caractères reposent. Alors qu'elles sont généralement droites dans les documents imprimés, elles peuvent être définies comme des polylignes, ce qui leur permet de suivre la courbure de la ligne de texte. En projetant les éléments courbes sur une ligne de base droite, nous pouvons transformer une ligne courbe en une ligne droite pouvant être traitée par le modèle de transcription. Si la ligne de base est associée à un polygone englobant, il est également possible de supprimer des éléments situés en dehors de la ligne de texte qui nous intéresse dans le processus.

Nous avons proposé une méthode de segmentation, basée sur un réseau neuronal de segmentation sémantique convolutive profonde (U-Net), suivant cette représentation de la ligne de base et l'avons évaluée à la fois sur notre ensemble de données et sur un ensemble de données latins cBAD. La méthode a atteint une précision similaire à celle d'autres méthodes de pointe (voir tableau 5.1). Les résultats obtenus sur l'ensemble de données arabes étaient légèrement inférieurs à ceux obtenus sur l'écriture latine. Neanmoins, notre ensemble de données est beaucoup plus diversifié, ce qui indique la pertinence générale de cette approche pour la segmentation des lignes de texte des manuscrits arabes.

Bien qu'efficace, il manque à cette méthode plusieurs caractéristiques nécessaires à un segmenteur pratique. Premièrement, elle ne comporte pas de moyen de calculer l'orientation des lignes, deuxièmement, elle ne dispose pas d'un algorithme pour calculer un polygone englobant pour la suppression du contenu non-ligne, et elle est incapable de reconnaître conjointement les lignes et les régions. En changeant la couche de sortie du réseau neuronal pour effectuer une classification de pixels multi-étiquettes, la nouvelle méthode est capable de détecter à la fois les lignes et les régions simultanément. Les nouvelles classes de marqueurs dans la carte de sortie indiquent où se situent le début et la fin d'une ligne de texte, ce qui permet de déterminer l'orientation de la ligne. En outre, une nouvelle méthode de post-traitement a été proposée pour extraire les lignes de base de la sortie de la carte de pixels bruts du réseau de segmentation sémantique, et pour calculer un polygone englobant. Enfin, le réseau de segmentation

sémantique a été changé en un réseau de type ReNet, plus efficace en mémoire, qui utilise des couches LSTM balayées verticalement et horizontalement au lieu de piles profondes de couches convolutionnelles pour obtenir de grands champs récepteurs.

Cette deuxième méthode a été évaluée à la fois sur l'ensemble de données arabes, une nouvelle version de l'ensemble de données cBAD et un certain nombre d'ensembles de données latines plus petites. Les résultats étaient comparables à la pointe de la technologie pour la segmentation des régions et des lignes de texte, avec une certaine amélioration par rapport à la méthode précédente dans les résultats sur l'ensemble de données arabes.

Enfin, nous avons proposé une méthode simple pour la détection du système d'écriture et de l'emphase dans les lignes de texte. Ce système est utile pour le traitement des textes et documents multilingues où l'emphase, c'est-à-dire le texte en italique, en gras, etc. est utilisée pour le balisage sémantique, tel qu'il se produit fréquemment dans les dictionnaires.

La méthode profite de l'alignement implicite fourni par le réseau de transcription de texte formé avec la fonction de coût CTC. Bien qu'il ne soit pas garanti que les activations pour un caractère particulier soient proches de son emplacement dans la ligne, les capacités limitées de modélisation à longue distance d'un réseau LSTM font qu'il le place presque toujours correctement. En entrainant un réseau de transcription de texte à produire une séquence de codes d'identification au lieu de caractères réels, nous pouvons diviser une ligne en bandes appartenant à un seul système d'écriture. Ces bandes peuvent ensuite être traitées par des modèles de reconnaissance spécifiques à l'écriture. Une propriété intéressante de notre approche est que le système peut être entraîné et que les données de son apprentissage peuvent être dérivées automatiquement des données d'apprentissage existantes pour les modèles de transcription.

A.7 La Transcription et l'Alignement

A.7.1 Le Logiciel ROC Kraken

Le Kraken est un logiciel de ROC modulaire et open source conçu pour être particulièrement utile pour la rétro-numérisation dans les humanités. Outre les méthodes de pointe pour la transcription et l'analyse de la mise en page, il comprend un certain nombre d'autres fonctionnalités qui le rendent intéressant pour les chercheurs en humanités.

Un grand soin a été apporté à son développement pour réduire les hypothèses implicites sur le fonctionnement du texte et pour rendre ses limitations explicites. Il a été étendu depuis ses origines en tant que bifurcation du système OCRopus avec un support Unicode complet de droite à gauche, bidirectionnel et vertical de l'écriture, la détection des scripts et la reconnaissance multigraphique. Une interface JSON simple permettant la configuration d'un mappage entre les sorties de modèles numériques et les séquences de points de code Unicode et vice versa. Ce

mécanisme est particulièrement utile pour les écritures logographiques de grande dimension telles que le système d'écriture chinois, car il permet la décomposition d'un point de code Unicode représentant un seul groupe de graphèmes en ses composants logiques dans la sortie du réseau neuronal.

Comme le Kraken est conçu pour être facilement intégré dans d'autres applications, il offre à la fois une API simple et un système de sérialisation flexible grâce à des templates. Des templates pour un certain nombre de formats tels que ALTO, hOCR, et TEI sont fournis par défaut. Les modules de traitement sont accessibles à la fois par l'API et par la ligne de commande qui permet la substitution flexible de blocs fonctionnels ou l'utilisation de sous-systèmes pour compléter ses propres méthodes.

Malgré notre efforts, les parametres choisi par défaut peuvent être désavantageux dans certains cas marginaux. Ils peuvent alors être désactivés ou adaptés. Les exemples vont du traitement textuel tel que la prise en charge de texte bidirectionnel[2] et de la normalisation du texte au changement des architectures et des paramètres d'entraînement des réseaux neuronaux artificiels, employés dans la segmentation et la transcription des pages.

Le module de transcription fonctionne comme un classificateur de séquences sans segmentation, utilisant un réseau neuronal artificiel pour mapper une image d'une seule ligne de texte en une séquence d'étiquettes qui sont ensuite mappées en points de code Unicode. L'ARN utilisé par défaut est un CNN-LSTM hybride entraîné avec la fonction de coût CTC. Un langage simple de spécification de réseau permet d'adapter le réseau à des tâches spécifiques. Les précisions des caractères pour un certain nombre de scripts différents utilisant ce classificateur sont indiquées dans le tableau 8.1.

La segmentation des pages est assurée par le système de segmentation des régions et des lignes décrit ci-dessus. Comme d'autres parties du logiciel, il est hautement configurable et permet la détection de régions et de lignes de texte arbitraires avec suffisamment de données d'entraînement. Les données d'entraînement peuvent être fournies dans un certain nombre de formats de fichiers standard tels que ALTO et PageXML ou via une simple API.

A.7.2 L'Alignement des Caractères

Une tâche d'un certain intérêt paléographique est l'alignement automatique de la transcription du texte avec les glyphes respectifs dans une image. Bien que cela puisse être fait naïvement avec une approche de segmentation des caractères similaire aux anciens logiciels de ROC, nous avons évalué une méthode qui utilise l'alignement implicite de la fonction de coût CTC pour localiser les graphèmes dans une image, à partir d'une transcription diplomatique, et nous l'avons comparée à un système SIFT-flow. La méthode est destinée à fonctionner sur les manuscrits de la mer Morte, des manuscrits très fragmentaires écrits principalement en hébreu.

[2] L'algorithme Unicode BiDi a des cas où un balisage explicite de la directionalité peut être requis.

Dans un premier stade, les manuscrits hébraïques fragmentaires sont segmentés à l'aide d'un modèle de segmentation des pages spécifiquement entraîné pour ce matériel. Les transcriptions diplomatiques par ligne de la base de données QWB sont ensuite mises en correspondance avec la sortie du segmenteur afin de créer des données d'entraînement pour un modèle de transcription de manière semi-automatique. Environ 2500 lignes provenant de 440 fragments ont ensuite été utilisées pour faire une apprentissage par transfer d'un nouveau modèle de transcription à partir d'un modèle de transcription de manuscrit hébraïque médiéval existant. Comme les données d'apprentissage varient énormément en termes de style, les caractères individuels sont souvent gravement dégradés, et le modèle est entraîné à surajuster sévèrement, le REC est assez élevé avec environ 30% sur l'ensemble de validation.

Les activations de ce modèle surajusté sont utilisées pour déterminer les positions des caractères sur le matériel dans l'ensemble d'entraînage créé semi-automatiquement. Lorsqu'il est évalué vis-à-vis des positions de glyphes annotées par l'homme, le système place le caractère le plus proche de la position réelle 90,3% du temps avec une IoU moyenne de 0,81, surpassant significativement la méthode SIFT-flow même lorsqu'il l'ancre avec les positions brutes des caractères identifiés par la ROC.

A.8 eScriptorium

eScriptorium est une plateforme libre d'analyse et d'annotation de documents. Elle cherche à combiner des techniques de calcul avec des outils numériques manuels pour la transcription et l'annotation approfondie de textes et d'images aux niveaux paléographique, philologique et linguistique. Il s'adresse aux chercheurs en humanités, mais aussi aux bibliothécaires et archivistes, aux étudiants, aux informaticiens et au grand public. Issu du projet Scripta, qui cherche à faciliter l'étude de l'écriture sous toutes ses formes au fil de l'histoire, ses principes de base sont la transparence, la flexibilité et l'indépendance de la langue et du système d'écriture.

Ce dernier point est particulièrement important car la gamme des langues et des systèmes d'écriture étudiés dans le cadre de Scripta est énorme, couvrant le Proche-Orient ancien, l'Iran et l'Asie centrale, l'Inde, l'Asie du Sud-Est et de l'Est, ainsi que l'Occident classique et médiéval. Par conséquent, comme pour le Kraken, un effort concerté a été fait pour réduire les hypothèses sur le fonctionnement du texte.

eScriptorium utilise le Kraken pour ses besoins en vision par ordinateur. Ainsi, la construction du pipeline de ROC est reflétée dans l'interface d'eScriptorium, avec une approche par étapes de l'importation des données, de la segmentation des pages (automatique ou manuelle), de la transcription (automatique ou manuelle), de l'annotation et de l'exportation.

La nécessité de s'adapter à une grande variété de systèmes d'écriture, en par-

ticulier la volonté de pouvoir traiter des écritures rares et historiques, impose à eScriptorium certaines restrictions de conception qui vont au-delà des mesures prises pour rendre les méthodes computationnelles de Kraken polyvalentes. Par définition, les langages rares manquent de grands ensembles de données préexistants qui peuvent être utilisés pour lancer le processus de ROC. Par conséquent, l'annotation et la vérification manuelles de la segmentation et de la transcription ne peuvent pas être une simple réflexion après coup, mais doivent être considérées comme une partie fondamentale de l'interface, à la fois pour permettre un travail pratique avec les plus petits ensembles de données qui ne peuvent pas encore être traités avec les méthodes automatiques mises en œuvre et pour aider au démarrage efficace du traitement automatique.

La variété des systèmes d'écriture empêche également l'utilisation de techniques courantes pour augmenter la généralisation et la charge de formation des méthodes automatiques telles que les modèles de langage statistique et les modèles généralisés pour des tâches comme la segmentation des pages. Les modèles linguistiques puissants pour les langues à faibles ressources telles que le vietnamien ancien sont tout aussi irréalistes qu'un segmenteur de pages capable d'extraire avec précision des lignes d'inscriptions chinoises, de manuscrits arabes, d'incunables et de journaux avec un seul modèle ARN. Par conséquent, la plateforme est conçue pour permettre un apprentissage et un réapprentissage fréquents grâce à des inventaires de modèles, des interfaces intermédiaires pour l'importation et l'exportation de données, et des rapports d'évaluation prospectifs détaillés comme ceux qui existent déjà dans le Kraken.

Un dernier aspect renforçant ces contraintes de conception dans la plateforme provient non pas du matériel source mais du type de travail effectué sur celui-ci. Les chercheurs en humanités effectuent un large éventail de recherches en utilisant un grand nombre de paradigmes différents sur le matériel textuel. Ce pluralisme méthodologique se traduit par des conventions de transcription différentes, même sur du matériel dans la même langue, en fonction des préférences particulières du chercheur et de son domaine. Il existe donc un besoin fondamental de s'adapter aux différentes normes et de les rendre visibles aux autres, en particulier dans le contexte des systèmes d'intelligence artificielle qui ne sont, après tout, que de puissants outils d'inférence statistique. Les systèmes ouverts peuvent aider à communiquer les normes et les hypothèses de ces procédures, mais il n'en reste pas moins que pour un simple utilisateur de humanités peu familier avec la terminologie de l'informatique, celles-ci sont cachées dans une boîte noire magique (et vice versa). Les ontologies peuvent principalement combler ce fossé, mais elles sont complexes à mettre en place et à entretenir et se heurtent souvent à la nature ad hoc de la recherche. Notre meilleure option reste de suivre les standards lorsqu'ils existent, d'offrir des interfaces pour prendre et apporter des données et des artefacts depuis et vers l'eScriptorium, et d'accepter qu'il est très peu probable qu'un seul outil soit à la fois pratique et universel.

A.9 Conclusions et perspectives

En conclusion, nous avons présenté dans cette thèse un travail qui représente un pas en avant vers la rétro-numérisation pratique des documents en écriture arabe et des documents historiques et non-latins en général. La segmentation des pages et la transcription sont maintenant en principe capables de numériser n'importe quel document en caractères arabes, mais surtout l'inclusion de ces méthodes dans un système de ROC de bas niveau et un ERV de haut niveau, qui sont tous deux totalement ouverts à l'adaptation, la réutilisation et le partage, rendant l'utilisation de ces outils dans les projets de humanités numériques, petits et grands, beaucoup plus attrayante.

Il est clair qu'un travail substantiel reste à faire. Nous avons étudié les exigences générales d'un système de ROC à usage général en écriture arabe et validé l'état de l'art au début de la thèse dans deux études. Bien que les questions les plus urgentes (à savoir l'analyse de la mise en page, des méthodes de transcription plus puissantes et de meilleurs outils pour la création, la conservation et la diffusion des données) aient été résolues dans une large mesure, toutes les tâches ne sont pas actuellement résolues de manière satisfaisante.

La tâche la plus urgente pour la ROC des manuscrits arabes est la détermination de l'ordre de lecture. Comme décrit ci-dessus, la recherche sur ce sujet est rare et les méthodes existantes sont des heuristiques artisanales incapables de traiter la structure souvent complexe des manuscrits historiques. Néanmoins, alors que les ensembles de données sont inexistants ou cachés dans des ensembles de données pour d'autres tâches, les capacités de l'intelligence artificielle pour le raisonnement spatial avec un grand nombre d'objets ont augmenté ces dernières années avec le développement des réseaux neuronaux de graphes.

Bien que les systèmes de ROC de pointe soient capables de réaliser des exploits impressionnants même sur des documents très dégradés et atypiques, avec des exigences en matière de données d'entraînement plus ergonomiques que jamais, le repérage des données d'entraînement est toujours la tâche la plus longue des techniques modernes de vision par ordinateur. Alors que nous pouvons maintenant utiliser efficacement l'apprentissage par transfert pour adapter les modèles existants à de nouveaux documents avec des quantités minimales de données, des méthodes d'adaptation de domaine plus avancées offrent de grandes promesses pour rendre plus de documents accessibles sans intervention humaine.

Le développement d'eScriptorium va sûrement se poursuivre et intégrer les progrès des méthodes automatiques, dans la mesure où il aide la recherche en humanités. Les pistes non explorées dans cette thèse comprennent les opérations non textuelles, telles que diverses tâches de classification d'images, la datation, le regroupement de documents similaires ou la détection de la réutilisation de textes.

Enfin, même avec la disponibilité d'outils de vision par ordinateur puissants et ouverts, le paysage des ensembles de données reste fracturé. Alors que les chercheurs reconnaissent plus que jamais l'importance du partage des données pour faire progresser non seulement les humanités mais aussi la recherche informa-

tique, le moyen préféré pour y parvenir reste le dépôt github profane avec un fichier README non descriptif. Une combinaison d'ERV conscients de l'importance de métadonnées appropriées, d'un apport élargi de la pratique archivistique dans la recherche scientifique, et de l'utilisation d'infrastructures de données de recherche ouvertes comme c'est déjà le cas dans d'autres disciplines scientifiques, a le potentiel d'améliorer considérablement cet état de fait dans les prochaines années.

Appendix B

Technical Overview of the Kraken Software

This appendix is a technical summary of the Kraken software in its current state. It is valid for the version deposited for this thesis in the Zenodo research data repository and assigned the DOI *10.5281/zenodo.4498925*.

This document is located in between the low-level descriptions of the methods and algorithms employed in chapters 5, 6, and 7 and the high-level conceptual overview of the software intended for humanists in chapter 8. The majority of the text is derived from the technical end-user documentation available on the Kraken web site.

B.1 Command Line Interface

The principal way to interact with Kraken for most users is through the command line interface (CLI). For practical purposes the CLI is split into two principal parts, the *kraken* command for all tasks related to inference, i.e. recognition, and the *ketos* command for tasks related to the training and evaluation of segmentation and transcription models.

B.1.1 Inference

The *kraken* command exposes each processing step of the OCR process as a separate *subcommand* which operates on a number of inputs to produce specific output files. In concordance with the linear workflow structure of OCR, these subcommands can be chained to perform multiple processing steps at the same time.

The general invocation of the command is thus:

```
$ kraken -i inp_1 outp_1 -i inp_2 outp_2 ... subcmd_1 subcmd_2 ... subcmd_n
```

Input files can be in different formats and defined in different ways. The above syntax is the most direct: each input file is directly mapped to an output file.

As this syntax is too verbose for more than a few files and does not allow the definition of multiple outputs for a single input file, as is desirable for multi-page TIFF or PDF files, batch input are handled in two different ways. The first allows the use of shell or glob patterns to match multiple input files and append a specific suffix to each output files:

```
$ kraken -I glob_pattern -o suffix ...
```

For example:

```
$ kraken -I '*.png'  -o '.xml' ...
```

The second enables the splitting of a single input file into multiple output files with dynamically created suffixes through a format string:

```
$ kraken -I glob_pattern -p format_string ...
```

For example:

```
$ kraken -I '*.pdf' -p '{src}_{idx:06d}.xml' ...
```

splitting each input file into output files starting with the original file name followed by a page index.

A variety of input file formats are supported, both for reasons of convience and because each processing steps expect different input data. As OCR is the conversion of image data into machine-encoded text, the expected default is unsurprisingly plain image files:

```
$ kraken -i image_1.png output_1 -i image_2.png output_2 ...
```

More complex data can be fed into kraken with files in the ALTO and PageXML formats. These are, for example, useful to only perform transcription on already pre-segmented images. Each subcommand will automatically retrieve the necessary information, i.e. executing the layout analysis subcommand on an ALTO file will cause Kraken to only load the image file defined therein. Input formats can be switched with the *-f* switch, e.g.:

```
$ kraken -i alto.xml output.xml -f alto subcommand_1 subcommand_2 ...
```

A special case are multi-page inputs. These can also be selected with the appropriate *-f* option, currently *-f pdf* for both PDF and multi-page TIFF files, but as they do not contain parseable structural and content information only image data is extracted. Valid values for the format option are currently *alto*, *page*, *pdf*, *image*, and *xml* (to automatically select the appropriate parser for each XML input file).

Binarization

Binarization is no longer mandatory with the new segmenter but the original OCRopus binarization algorithm is still available through the *binarize* subcommand.

```
$ kraken -i ... binarize
```

Layout Analysis

Layout analysis is accessed with the *segment* subcommand. Two segmenters are implemented in Kraken, the legacy non-trainable segmenter producing bounding box data and the new trainable segmenter that uses the baseline and bounding polygon paradigm. In addition to extracting text lines, the latter is also able to detect regions (both textual and non-textual) and assign classes to text lines if trained with the appropriate training data. We will only explain the use of the new segmenter here.

The segmenters can be selected with a subcommand option:

```
$ kraken -i ... segment -x # legacy segmenter
$ kraken -i ... segment -bl # baseline segmenter
```

When the baseline segmenter is selected a default model trained on modern Latin manuscripts will be used. This simple model only detects lines and a basic text region. Other segmentation models can be supplied with the *-i* option:

```
$ kraken -i ... segment -bl -i model_1.mlmodel
```

It is also possible to run multiple segmentation models at the same time over an image and obtain a joint segmentation:

```
$ kraken -i ... segment -bl -i line_seg.mlmodel -i region_seg.mlmodel
```

This functionality can for example be used to combine the output of a segmentation model that only produces regions with one that only detects text lines. It is important to note that no filtering is performed on the output, i.e. when combining multiple line-detecting segmentation models the output will contain ”duplicate”, largely overlapping lines. Apart from the convenience of merging multiple region and line segmentation automatically, performing joint segmentation in Kraken also allows the segmenter to use additional region information for bounding polygon calculation which generally improves polygon accuracy, especially on lines close to the boundary of the writing surface.

The segmenter not only finds lines and regions but also imparts a reading order on them using a basic heuristic. As Kraken does not know the principal text direction of the document it can be supplied through an option *--text-direction*:

```
$ kraken -i ... segment --text-direction horizontal-lr # horizontal lines
                                                       # left before right lines
$ kraken -i ... segment --text-direction horizontal-rl # horizontal lines
                                                       # right before left lines
$ kraken -i ... segment --text-direction vertical-lr # vertical lines
                                                     # left before right lines
$ kraken -i ... segment --text-direction vertical-rl # vertical lines
                                                     # right before left lines
```

This text direction is unrelated to the direction of the writing system in a line and only determines the inter-line and column order. Taking a parallel English and Arabic text as an example, it is possible that lines are read top-to-bottom, left column before right column (the page is typeset left-to-right, i.e. like a Latin-script document) or that lines are read top-to-bottom, right column before left column (the page is typeset right-to-left, i.e. like an Arabic document). The options for vertical lines behave correspondingly.

In some cases it is desirable to mask out parts of the input image which are known not to contain any lines or regions. Mask images have to be the same shape as the input image. Black pixels in the mask image will be ignored be the segmenter:

```
$ kraken -i ... segment -m mask.png ...
```

The output of the segmenter is a JSON file containing the verbatim data structure returned by the internal segmentation method of Kraken:

```
{"text_direction": "horizontal-lr",
 "type": "baselines",
 "lines": [{"script": "default", "baseline": [[877, 281], [1893, 318]],
                                 "boundary": [[877, 281], … [881, 325]]},
           {"script": "default", "baseline": [[1224, 552], [1351, 500]],
                                 "boundary": [[1224, 552], … , [1231, 555]]},
           …],
 "region": {"text": [[[500, 128], … [200, 325], […],
            "illustrations": …},
 "script_detection": true
}
```

Transcription

Transcription requires a color, grey-scale, or binarized image, a page segmentation for said image, and a model file containing a transcription model. The first two can either originate from earlier subcommands or directly from an XML file. Model files are defined through the *-m* option on the *ocr* subcommand:

```
$ kraken -i ... ocr -m trans.mlmodel
```

The *ocr* subcommand is multi-model capable, i.e. it is possible to selectively apply transcription models on parts of the provided text lines. Originally intended

for multigraphic transcription (see chapter 7), this selection can be made for arbitrary criteria, such as different hands, languages, or typefaces. The assignment of transcription models to text lines works through line types which are part of the segmentation parsed either from an XML file (see the examples in the Kraken git repository for exact attributes used) or the output of an appropriately trained layout analysis model (the value of the *script* field in the example segmentation above). It is therefore a simple mapping based on classifications performed beforehand. The general syntax for this mapping is:

```
$ kraken -i ... ocr -m type_1:m_1 -m type_2:m_2 -m type_3:m_3
```

Two special keywords exist for types and models. The *default* identifier is a catch-all and applies the specified model on every identifier that does not have a model assigned explicitly. The *ignore* model value causes Kraken to ignore text lines with this identifier and silently drop them from the output. If no *default* model is defined, unassigned types will cause Kraken to abort processing with an error message. An example for transcribing all lines except those assigned the *notes* type:

```
$ kraken -i ... ocr -m default:defmodel.mlmodel -m notes:ignore
```

The default output of the *ocr* subcommand is a plain text file with the text in a line corresponding to the respective line in the segmentation. As this output lacks metadata, such as line, word, and character locations, links to image files, and utilized transcription models, enriched XML output formats can be selected with options on the subcommand:

```
$ kraken -i ... ocr ... -t # text output
$ kraken -i ... ocr ... -h # hOCR output
$ kraken -i ... ocr ... -a # ALTO output
$ kraken -i ... ocr ... -y # abbyyXML output
$ kraken -i ... ocr ... -x # PageXML output
```

Output is serialized *de novo*, i.e. even if an input file was already an XML file in ALTO or PageXML output is not ”inserted” into the input but the segmentation and transcription are used to produce an entirely new file which can lack information contained in the input file.

It is also possible to use the transcription functionality without a segmentation through the *--no-segmentation* switch. In this case, each input image is treated as one whole line instead of a document page containing multiple text lines.

B.1.2 Training

Training functionality is provided through subcommands of the *ketos* command line tool. There are three principal commands: *train* and *test* for training and evaluating transcription models and *segtrain* for training layout analysis models.

While basic tooling for training data creation for transcription models was included in the past, these are only compatible with the legacy bounding box segmenter. For the annotation and transcription of baselines, regions, and text external tools like eScriptorium or Aletheia that can either export data in ALTO 4.2 and PageXML format or have tight Kraken integration are the preferred option.

Therefore, both transcription and layout analysis are trained primarily through datasets contained in ALTO or PageXML files. Legacy formats, line images and text files for transcription and JSON files containing line coordinate lists, are still supported but do not offer the full range of functionality.

B.1.3 Transcription Training and Evaluation

Training a transcription model from a collection of PageXML or ALTO files containing the necessary annotation (baselines, bounding polygons, and text) can be done in two ways, from scratch or based on an existing model. The latter is useful when a model for similar documents, such as a similar typeface or hand, already exists. In this case, transfer learning to the new data can reduce the training requirements substantially.

We will start with the simple case of training a model from scratch:

```
$ ketos train -f xml *.xml
[1.8139] alphabet mismatch: chars in training set only: ... (not included in
↪  accuracy test during training)
Initializing model
stage 1/   [####################] 1163/1163 Accuracy report (1) 0.1844 10092 8231
stage 1/   [####################] 1163/1163 Accuracy report (1) 0.1844 10092 8231
stage 2/   [####################] 1163/1163 Accuracy report (2) 0.2335 10092 7736
stage 3/   [####################] 1163/1163 Accuracy report (3) 0.3242 10092 6820
stage 4/   [####################] 1163/1163 Accuracy report (6) 0.4006 10092 6049
...
```

This command automatically parses the XML files in either of the supported formats, loads the images, splits off ten per cent of the training data as a validation set, and commences training. Training is divided into epochs, with an evaluation automatically performed on the validation set after each line in the training set has been seen at least once by the network.

The warning about an alphabet mismatch is the result of the training dataset containing characters that are not in the validation set. The network is not evaluated against these characters but still learns how to recognize them. This is usually the case with small datasets and rare characters. A corresponding warning if a character is in the validation set but not in the training set can also be printed.

Depending on the speed of the computer and the size of the data, training can take a substantial amount of time. Per default training stops automatically as soon as the character accuracy (the first number in the accuracy report in the output above) on the validation set does not improve above a certain threshold for

a number of epochs. This approach, called early stopping, uses default parameters that might not be appropriate for all datasets. For very small datasets of only a few dozen lines the default number of epochs before aborting (five) might be too low while very large datasets without much variation can cause the model to overfit between evaluation runs. To adjust these parameters a couple of options are available:

```
$ ketos train ... -F 0.5 # evaluates after half the training set
$ ketos train ... --lag 10 # waits 10 epochs for any improvement
$ ketos train ... --min-delta 0.001 # lowers improvement threshold to 0.1%
```

Instead of a random split into training and validation set that changes with each training run, it is also possible to force a fixed split to ensure reproducibility acros runs. The most explicit way is through manifest files that each contain the path to one XML file per line:

```
$ ketos train -f xml -t train.lst -e val.lst
```

Transfer learning an existing model works similarly to training from scratch but takes an existing model in addition to the training data:

```
$ ketos train -f xml -i model.mlmodel *.xml
[0.8616] alphabet mismatch {'~', '»', '8', '9', ' '}
Network codec not compatible with training set
[0.8620] Training data and model codec alphabets mismatch: {' A'}
```

If the characters in the training set differ from the existing possible outputs of the network, an error will be raised. As the transfer learning process initially changes the internal structure of the model in a way that makes it ”forget” some of the already learned information, this is a basic safety precaution. Two modes for adapting the model to the new alphabet: *add* and *both*. *add* resizes the model to be able to output all the characters in the training set without removing any existing characters. *both* will make the resulting model an exact match with the training set by removing both unused characters and adding new ones.

```
$ ketos train -f xml --resize add -i model.mlmodel *.xml
...
[0.8737] Resizing codec to include 1 new code points
[0.8874] Resizing last layer in network to 52 outputs
...
$ ketos train -f xml --resize both -i model.mlmodel *.xml
...
[0.7857] Resizing network or given codec to 49 code sequences
[0.8344] Deleting 2 output classes from network (46 retained)
...
```

In this example 1 character was added for a network that is able to recognize 52 different characters after sufficient additional training. It is important to remember that in *add* mode the model will first lose some accuracy for characters

it has already learned through the resizing process, a deterioration that is worse for large changes, but also unlearn already learnt characters that are not in the training set during training. This initial deterioration is also true in *both* mode but not the gradual unlearning as all possible output characters are contained in the targeted training data.

The command line interface for training also exposes various hyperparameters such as model architecture, learning rate, optimizers, weight decay, etc. The model architecture can be changed through VGSL (see section B.1.3), while various other parameters are set with options, such as *--optimizer*, *--lrate*, and *--weight-decay* (see the help message for valid values).

More command line options for various text normalizations, custom codecs, recalculation of bounding polygons, caching of training data, etc. exist. These are documented in the subcommand's help message and the full Kraken documentation.

Lastly, it is possible to substantially accelerate training with CUDA acceleration. This requires a properly configured graphics card (GPU) with sufficient memory to place the model to be trained. As transcription models are fairly small, all but the smallest GPUs are sufficient for this purpose. CUDA acceleration is activated by selecting a GPU with the *--device* option:

```
$ ketos segtrain --device cuda ...
```

After a model has been trained an in-depth analysis against a separate test dataset is often performed. More detailled than the simple character accuracy output during training and a better estimation of real world accuracy when multiple models have been trained on the same training data, these reports contain per-script accuracy rates and confusion matrices that can also pin-point weaknesses of the transcription model:

```
$ ketos test -m best.mlmodel -f xml *.xml
Evaluating $model
Evaluating  [####################] 100%
=== report best.mlmodel ===

7012 Characters
6022 Errors
14.12%       Accuracy

2    Insertions
5226 Deletions
794  Substitutions

Count Missed   %Right
1567  575    63.31%  Common
5230  5230   0.00%   Latin
215   215    0.00%   Inherited

Errors       Correct-Generated
```

```
773  { A } - {  }
536  { c } - {  }
328  { e } - {  }
274  { d } - {  }
...
```

The report start off with an overall accuracy, followed by the absolute number of errors and per-script[1] accuracy rates. The remainder of the report contains the confusion table sorted by frequency.

Layout Analysis Training

Training of layout analysis models is very similar to training of transcription models, just with a different subcommand:

```
$ ketos segtrain -f xml *.xml
Creating model ...
Training line types:
  $pac  3  6539
  $not  4  202
  $par  5  14803
Training region types:
  $tip^^I6  829
  text^^I7  8
stage 1/50  [####################] 46/46 Accuracy report (1) mean_iu: 0.0309
 ↪  freq_iu: 0.0975 mean_acc: 0.0309 accuracy: 0.0309
stage 2/50  [############-------] 16/46 00:05:11
```

Instead of stopping automatically after a period of accuracy stagnation, *segtrain* stops after fifty epochs per default. This is chiefly because the pixel accuracy rates are not directly linked to the actual baseline and region detection quality.

As can be seen in the above example, the model is trained per default on all the baseline types and regions in the training dataset (the object counts for each type are listed after the type). There are multiple options that control this behavior. It is possible to suppress either baselines or regions completely:

```
$ ketos segtrain --supress-baselines ...
$ ketos segtrain --suppress-regions ...
```

More fine-grained controls allow the merging and suppression of specific types with whitelists:

```
$ ketos segtrain --valid-regions reg_1 --valid-region reg_2 ...
$ ketos segtrain --valid_baselines type_1 --valid_baselines type2 ...
$ ketos segtrain --merge-baselines $par:$not
$ ketos segtrain --merge-regions $text:$tip
```

[1] Scripts are determined according to Unicode script property linked to ISO 15924 script codes which vary widely in granularity. Script identifiers are defined for variant forms of the Latin script such as Fraktur but only one identifier exist for Arabic and its derived scripts.

Both can be combined. The region/baseline whitelists are processed before merging, so it is necessary to whitelist even regions/baselines that are merged into others with the *--merge-** options.

Like transcription models, layout analysis models can be transfer learned to a new dataset. The same two modes *both* and *add* exist. In contrast to transcription models, adaptation does not directly affect the accuracy of other types, although transfer learning in *add* mode will still slowly unlearn types not in the new training data:

```
$ ketos segtrain --resize add -f xml *.xml
$ ketos segtrain --resize both -f xml *.xml
```

Likewise explicit splits between training and evaluation set can be provided:

```
$ ketos segtrain -f xml -t train.lst -e val.lst
```

In the same vein, hyperparameters and GPU acceleration can be set through identical options.

VGSL

Kraken implements a dialect of the Variable-size Graph Specification Language (VGSL), enabling the specification of different network architectures for image processing purposes using a short definition string.

A VGSL specification consists of an input block, one or more layers, and an output block. For example a grayscale line transcription network consisting of two convolutional layers (ReLU activation) with 32/64 3×3 filters, followed2 ×2 maxpooling after each layer, and a final bidirectional LSTM layer and 1D dropout regularization:

```
[1,48,0,1 Cr3,3,32 Mp2,2 Cr3,3,64 Mp2,2 S1(1x0)1,3 Lbx100 Do O1c103]
```

or a simple layout analysis model pixel labelling (4 classes) a 1200 pixel high RGB color image with two convolutional layers and one ReNet-like block:

```
[1,1200,0,3 Cr3,3,64 Gn32 Cr3,3,128 Lby32 Lbx32 O2l4]
```

The first block defines the input in order of (batch, heigh, width, channels) with zero-valued dimensions being variable. Integer valued height or width input specifications will result in the input images being automatically scaled in either dimension. Mixed variable and fixed input sizes, e.g. a height set to 400 and a width set to 0, will result in a proportional scaling of the image. The batch size is currently ignored in Kraken and can be set separately with command line options.

When channels are set to 1 grayscale or B/W inputs are expected, 3 expects RGB color images. Higher values in combination with a height of 1 result in the network being fed 1 pixel wide grayscale strips scaled to the size of the channel dimension, i.e. an internal transposition of the height and channel dimensions.

After the input, a number of processing layers are defined. Layers operate on the channel dimension; this is intuitive for convolutional layers but a recurrent layer performing sequence classification along the width axis on an image of a particular height requires the height dimension to be moved to the channel dimension, e.g.:

```
[1,48,0,1 S1(1x0)1,3 Lbx100 O1c103]
```

or using the aforementioned alternative formulation performing the transposition implicitly with the input definition:

```
[1,1,0,48 Lbx100 O1c103]
```

Finally an output definition *O...* is appended. When training transcription and segmentation models with the provided command line tools these are derived automatically from the training data based on the number of different code points or baseline and region types.

The two principal layer types available in VGSL are LSTM and GRU layers:

```
L(f|r|b)(x|y)[s]<n> LSTM cell with n outputs.
G(f|r|b)(x|y)[s]<n> GRU cell with n outputs.
  f runs the LSTM/GRU forward only.
  r runs the LSTM/GRU reversed only.
  b runs the LSTM/GRU bidirectionally.
  x runs the LSTM/GRU in the x-dimension.
  y runs the LSTM/GRU in the y-dimension.
  s (optional) summarizes the output in the requested dimension,
     outputting only the final step, collapsing the dimension to a
     single element.
```

and convolutional layers:

```
C(s|t|r|l|m)<y>,<x>,<d>[,<y_stride>,<x_stride>]
Convolves using a y,x window, with optional stride, valid padding, d outputs,
with configurable non-linearity.
(s|t|r|l|m) specifies the type of non-linearity:
s = sigmoid
t = tanh
r = relu
l = linear (i.e., None)
m = softmax
```

Multiple auxiliary layers exist. The *S* layer shuffles data between dimensions and is most frequently used to collape any remaining y-height before the recurrent layers in a transcription model:

```
S<d>(<a>x<b>)<e>,<f> Splits one dimension, moves one part to another dimension.
Takes dimension d, reshapes it into a (a,b)-shaped tensor, distributing a into
dimension e and b into dimension f. Setting a or b to 0 auto-fills to the
correct value.
```

Various regularization layers are implemented:

```
Do<p>[,(1|2)] Inserts a 1D or 2D dropout layer with probability <p>. Defaults
to 1D dropout.
Gn<g> Inserts a group normalization layer with <g> groups.
```

Maxpooling:

```
Mp<y>,<x>[,<y_stride>,<x_stride>] Adds a maxpooling layer with kernel size
^^I(y,x) and optional stride (y_stride,x_stride)
```

and finally output layers:

```
O(0|1|2)(l|s|c)<d> Adds an output layer for scalar, 1D, or 2D heatmap output
with d classes.
(l|s|c) select both non-linearity and loss function:
l = sigmoid (binary crossentropy)
s = softmax (crossentropy)
c = softmax (CTC loss)
```

As mentioned above output layers are added automatically by command line tools, so it is only necessary to create them when using the API.

B.1.4 Model repository

Kraken incorporates a simple model repository that stores layout analysis and transcription models with basic metadata in the Zenodo[2] research data repository. Models made available through the repository are public and can either be retrieved with Kraken's command line tools, through the Zenodo website, or its web API. Because of limitations of the Zenodo platform, publishing of models is currently not completely automated and requires manual approval of each submitted model by an administrator. Therefore publishing models is not instantaneous until the necessary changes to Zenodo's API are made to enable automatic approval.

Models in the repository are interacted with through DOI permanent identifiers. As these are globally unique, unalterable, and resolvable to the object in the repository; they can be used to reference a particular model precisely, e.g. in publications. To retrieve the list of models in the repository:

```
$ kraken list
Retrieving model list .
10.5281/zenodo.2577813 (pytorch) - A generalized model for English printed text
....
```

The get more details on the exact type of data, character accuracy, etc. of the model one can also retrieve the metadata record of a single model with its DOI:

[2] http://zenodo.org

```
$ kraken show 10.5281/zenodo.2577813
name: 10.5281/zenodo.2577813

A generalized model for English printed text

This model has been trained on a large corpus of modern printed English text
augmented with ~10000 lines of historical printed documents.
scripts: Latn
alphabet:
 ↪  !"#$%&'()+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]`abcdefghijklmno
          pqrstuvwxyz{} SPACE
accuracy: 99.95%
license: Apache-2.0
author(s): Kiessling, Benjamin
date: 2019-02-26
```

The record contains a natural language description of the models, describing usually the amount and type of training data used, and additional information like ISO 15924 script identifiers, code points in the model codec, character accuracy on the test set, and the authors's names.

To actually download a model, one simply executes:

```
$ kraken get 10.5281/zenodo.2577813
Retrieving model ...
Model name: en_best.mlmodel
```

Models are placed per default in the local user configuration directory which is often *.config/kraken* but can vary between operating systems. The Kraken subcommands search for models automatically in this directory so they can be used directly with:

```
$ kraken ... segment -i seg.mlmodel
$ kraken ... ocr -m en_best.mlmodel
```

no matter where those commands are executed.

Models are published with the *ketos publish* command. As it accesses the Zenodo API it requires an access token which can be generated in the web interface of the platform by any account holder. The *publish* subcommand asks for a number of values to fill the metadata record and uploads the record and model to Zenodo:

```
$ ketos publish arabic.mlmodel
Access token: $SUPER_SECRET
author: foobar
affiliation:
summary: this is a model for all arabic text ever written
accuracy on test set: 100.0
script: Arab
license: SISSL
Uploading .....
model PID: 10.5281/zenodo.2577814
```

The record is created immediately and the PID is valid but an administrator has to approve the record's accession to the OCR model group in Zenodo in order for it to be discoverable with Kraken's command line tools.

B.1.5 API

A simple API is available for both training and inference. The principal recognition tasks are encapsulated in single functions and a simple pipeline for OCR is only a few lines of Python code:

```
from PIL import Image

from kraken import blla, rpred
from kraken.lib import models, vgsl

seg_model = vgsl.TorchVGSLModel.load_model('la.mlmodel')
tr_model = models.load_any('tra.mlmodel')

im = Image.open('/path/to/image.png')
seg = blla.segment(im, model=tr_model)
for line in rpred.rpred(tr_model, im, seg):
    print(line.prediction)
```

These lines load first import the PIL library for image handling, the *blla* Kraken module for (baseline) segmentation, the *rpred* module for text transcription, and the *models* and *vgsl* modules for model loading. Afterwards, the respective layout analysis and transcription models, and an input image are loaded. The difference between the two model loading function is that transcription models are wrapped in a slim abstraction layer while segmentation models use the raw VGSL interface directly.

Next, we perform segmentation on the previously loaded image file and transcribe each found line with the transcription model. The transcription function *rpred* does not only return text but an object containing also character bounding polygons and confidences. *rpred* is a simplified transcription method for single model transcription; multi-model functionality capable of transcribing typed lines with multiple models is available through the more advanced *mm_rpred* function (see the full API documentation for further details).

Training is more complicated but a basic training run with the default parameters is just a few lines of code as well:

```
from kraken.lib import xml
from kraken.lib.train import KrakenTrainer

training_files = ['a.xml', 'b.xml', 'c.xml']
eval_files = ['d.xml', 'e.xml']

# callback called after each iteration
def step_callback(*args):
    return lambda: print('.')
```

```
# function to print the validation results after each epoch
def print_transcription_eval(epoch, accuracy, chars, error, **kwargs):
    print(f'Accuracy report {epoch} {accuracy} {chars} {error}')

# create a transcription model trainer
t_trainer = KrakenTrainer.recognition_train_gen(progress_callback=step_callback,
                                                output='model.mlmodel',
                                                training_data=training_files,
                                                evaluation_data=eval_files,
                                                format_type='xml')

# executing the transcription trainer.
t_trainer.run(print_transcription_eval)

# function to print the segmentation validation results after each epoch
def print_seg_eval(epoch, accuracy, mean_acc, mean_iu, freq_iu, **kwargs):
    print(f'Accuracy report ({epoch}) mean_iu: {mean_iu}')

# create a layout analysis model trainer
la_trainer =
↪  KrakenTrainer.segmentation_train_gen(progress_callback=step_callback,
                                                  output='seg.mlmodel',
                                                  training_data=training_files,
                                                  evaluation_data=eval_files,
                                                  format_type='xml')

# executing the transcription trainer.
la_trainer.run(print_seg_eval)

# retrieve of epoch of best validation error
print(f'best transcription model error: {t_trainer.stopper.best_epoch}')
print(f'best segmentation model error: {la_trainer.stopper.best_epoch}')
```

The basic principle is simple. A *KrakenTrainer* object is created through the constructors for transcription or layout analysis training. These constructors accept the arguments and options already known from the command line (see the API documentation for further details) and a callback that is executed each time after a sample has been ran through the neural network. Training runs are initiated by calling the *run* method on the object with another callback that is executed after the evaluation at the end of each epoch. *run* blocks and automatically returns once training is finished according to the stop parameters chosen, per default early stopping for transcription and a fixed number of epochs for layout analysis models.

Most options available on the command line are available on the respective API functions. A complete overview can be found on the Kraken website and in the Zenodo deposit mentioned in the introduction with example scripts showing low-level use contained in the contrib directory of the source code.

RÉSUMÉ

La transcription automatique de textes dans les documents historiques manuscrits et imprimés est devenue un processus établi dans les humanités numériques, son utilisation allant des archives ou des bibliothèques à grande échelle aux groupes de recherche et aux chercheurs individuels. Bien que des progrès considérables aient été réalisés ces dernières années pour comprendre les limites et faire progresser l'état de l'art, ces recherches restent largement limitées aux documents écrits dans les systèmes d'écriture européens, et plus particulièrement à l'écriture latine. L'une des cultures littéraires les plus vastes et les plus diverses, largement ignorée par les recherches actuelles sur l'analyse d'images de documents, est l'écriture arabe.

Cette thèse contient une étude compréhensive sur les caractéristiques des documents en écriture arabe et les défis qu'ils posent aux systèmes de reconnaissance optique de caractères de pointe, à travers une analyse théorique de l'écriture arabe et deux études de cas de rétro-numérisation sur des documents imprimés classiques et modernes. Les principales limites des méthodes courantes identifiées dans ces études ont ensuite été traitées. Deux méthodes entraînables de segmentation des pages suivant le paradigme de la ligne de base, permettant d'obtenir des résultats comparables à l'état de l'art et comprenant des caractéristiques supplémentaires nécessaires à la segmentation de pages de documents complexes, une méthode simple de traitement des lignes de texte multigraphiques et le logiciel ROC flexible Kraken intégrant ces méthodes sont présentés. On montre l'utilité de ce logiciel de ROC non seulement pour la reconnaissance de texte traditionnelle mais aussi pour une nouvelle tâche d'alignement des caractères. En outre, on présente l'environnement de recherche virtuel (ERV) eScriptorium pour l'annotation et la transcription. Cet ERV est spécifiquement conçu pour pouvoir traiter des textes non-latins, dont l'arabe, plus efficacement que les systèmes alternatifs existants. Au cours de ce travail, on a également préparé plusieurs ensembles de données d'entraînement et d'évaluation sous licence ouverte pour la transcription de textes arabes et la segmentation de pages.

MOTS CLÉS

segmentation des pages, reconnaissance de texte, environnement de recherche virtuel, écriture arabe, analyse d'images de documents

ABSTRACT

The automatic transcription of text in handwritten and machine-printed historical documents has become an established process in the Digital Humanities, its use ranging from large scale archival or library settings to research groups and individual scholars. While considerable progress on understanding limitations and advancing the state of the art has been made in recent years, this research remains largely limited to documents written in European writing systems, most importantly the Latin script. One of the largest and most diverse literary cultures largely ignored by current document image analysis research is the Arabic one.

This thesis contains a comprehensive study on the features of Arabic-script documents and their challenges posed to state of the art optical character systems through both a theoretical analysis of the Arabic script and two case studies of retrodigitization on printed classical and modern material. The principal limitations of common methods identified in these studies were subsequently addressed. Two trainable layout analysis methods following the baseline paradigm achieving comparable results to the state of the art while incorporating additional features necessary for the segmentation of complex document pages, a basic method for processing of multigraphic text lines, and the flexible Kraken OCR engine integrating these methods are presented. We show the usefulness of this OCR software not only for traditional text recognition but also a novel character alignment task. Further, we present the eScriptorium virtual research environment (VRE) for annotation and transcription. This VRE is specifically designed to be able to treat non-Latin, among them Arabic, script material more effectively than existing alternative systems. In the course of this work we also prepared multiple openly licensed training and evaluation datasets for Arabic text transcription and layout analysis.

KEYWORDS

layout analysis, text recognition, virtual research environments, arabic, document image analyis