Michael Schonhardt, Tim Geelhaar, Tobias Hodel, Jan Odstrčilík

Automated Text Recognition
Theory, Platforms, Best Practices
With a contribution by Achim Rabus

Digital Humanities Research

Digital Humanities is an evolving, cross-cutting field within the humanities employing computer-based methods. Research in this field, therefore, is an interdisciplinary endeavor that often involves researchers from the humanities as well as from computer science. This collaboration influences the methods applied as well as the theories underlying and informing research within those different fields. These implications need to be addressed according to the traditions of different humanities' disciplines. Therefore, the edition addresses all humanities disciplines in which digital methods are employed.

Digital Humanities Research furthers publications from all those disciplines addressing the methodological and theoretical implications of the application of digital research in the humanities.

The series is edited by Silke Schwandt, Anne Baillot, Andreas Fickers, Tobias Hodel, and Peter Stadler.

Prof. Dr. Michael Schonhardt is Professor of Digital Editing and Cultural History of the Middle Ages at the Technical University of Darmstadt and the Academy of Sciences and Literature | Mainz. His research focuses on the application of machine learning methods in digital scholarly editing and the automated datafication of historical sources.
ORCID: 0000-0002-2750-1900

Dr. Tim Geelhaar is a medievalist and digital historian. Since 2013, he has worked on various projects in digital history, including "Computational Historical Semantics", "Humanist Computer Interaction Revisited", and the "Latin Text Archive" at the Berlin-Brandenburg Academy of Sciences and Humanities. He has trained two complex handwritten text recognition models and works on digital annotation methods and text mining in historical corpora.
ORCID: 0000-0002-7653-5859

Prof. Dr. Tobias Hodel is Professor of Digital Humanities at the Walter Benjamin Kolleg at the University of Bern. His research focuses on machine learning methods in the humanities, with a particular emphasis on premodern sources. He is co-editor of the series "Digital Humanities Research".
ORCID: 0000-0002-2071-6407

Dr. Jan Odstrčilík is a Latin philologist and postdoctoral researcher at the Austrian Academy of Sciences, working in the research group Textual Networks and Cultures of Translation. His work centres on multilingualism in the Middle Ages, Latin sermons, and education. Beyond his involvement in ATR, he is interested in the connections between Digital Humanities tools and personal knowledge management.
ORCID: 0000-0001-9104-9827

Michael Schonhardt, Tim Geelhaar, Tobias Hodel, Jan Odstrčilík

Automated Text Recognition

Theory, Platforms, Best Practices

BIELEFELD UNIVERSITY PRESS

The publication of this work was generously supported by the University of Bern, the Publication Fund of Bielefeld University, and the Department of Digital Scholarly Editing and Medieval Cultural History at TU Darmstadt.

First published in 2026 by Bielefeld University Press, Bielefeld
www.bielefelduniversitypress.de

Cover Layout: Leon Pöhler, Bielefeld
Typeset: Nina Michler, Bielefeld
Illustrations: Marie Machatová, Prague
Printed by: mediaprint solutions GmbH, Paderborn

Print-ISBN: 978-3-69129-040-0
PDF-ISBN: 978-3-69129-041-7
https://doi.org/10.64136/nrlj5339

Table of Contents

Preface

Only a few years ago, the technological breakthrough in automated text recognition (ATR) brought together two historically distinct traditions of scholarly engagement with texts: the computational methods of computer and data sciences and the interpretive approaches of the humanities. This unexpected intersection marked the first opportunity for many scholars and scientists to collaborate across these independent disciplines. Although this collaboration proved immensely fruitful, it also highlighted the profound differences between these fields in vocabulary, concepts, research priorities, and methods. Soon, it became clear that the challenges of ATR cannot be addressed as purely technological problems, nor can they be framed solely as questions of traditional humanities. Instead, they force us to rethink traditional approaches regarding workflows, concepts, humanities research questions, and the development of artificial intelligence (AI) in general by bridging both perspectives.

It was within this context that we, the book's four authors, first met. Coming from diverse, yet complementary backgrounds – including digital humanities, digital editing, and scholarly engagement with pre-modern texts – we began sharing and discussing our experiences with applying ATR in our respective fields. These discussions quickly grew into a broader exchange of ideas about the implications of ATR and related technologies from the field of AI for our workflows, research practices, and interpretations. Now, as ATR is becoming increasingly mainstream in research and is adopted by many projects and scholars, we feel it is the right time to share and broaden our discussions more widely. We strongly believe the technological approaches available today are mature enough to be applied to a wide range of humanities questions. At the same time, the widespread adoption of the technology requires critical engagement with available methods, existing interfaces, and the underlying technological approaches, all while understanding their capabilities, limitations, assumptions, and epistemological implications.

Attempting to provide a guide to this dynamic and ever-changing landscape is no simple task. Advances in machine learning, new tools, and workflows are constantly reshaping the field on an almost weekly basis. Thus, we tried to focus not on particular tools but on a broader overview of machine learning approaches and core concepts that can be of

use beyond what is currently being used and developed. In doing so, we hope this book will help navigate the machine learning revolution that is currently sweeping through our daily lives and, increasingly, the humanities.

But first, we want to thank a number of persons and institutions who helped with this book: The Austrian Academy of Sciences allowed us to contract Marie Machatová, a designer who provided the visual layer to the book. Thanks to the support provided by the Universities of Bern, Frankfurt and the Burchards Dekret Digital project, we were able to hold meetings in person rather than only virtually. Achim Rabus kindly agreed to share his approach on ATR, which has significantly broadened the perspective of this volume. Dorothee Huff commented and advised us on an early draft of the first three chapters. Kerstin Tremble was responsible for the careful editing of the entire volume. Dominik Kilchmann managed the bibliography. The editors of the Digital Humanities Research series at Bielefeld University Press supported our endeavour and opened their series for us. Three reviewers gave constructive and valuable feedback that greatly improved the book. Our special thanks go to Vera Breitner, who made the whole process of writing and preparing for the press a pleasure. Finally, the Universities of Bielefeld, Bern, Darmstadt, and the Austrian Academy of Sciences supported the Open Access and HTML version of the book. We are grateful to all contributors.

Michael Schonhardt, Tim Geelhaar, Tobias Hodel, and Jan Odstrčilík

INTRODUCTION

1 Introduction

Chapter summary: This introductory chapter outlines the historical and scholarly significance of automated text recognition (ATR), emphasising its role in transforming analogue textual sources into machine-readable data. It explains how advances in machine learning have enabled the accurate transcription of handwritten documents and introduces key terms and definitions on which our arguments and assumptions are built.

How this fits into the broader argument: This introduction lays the foundation, situating ATR within the broader shift toward digital scholarship. It sets the stage for the book's dual focus on practical application and critical reflection, encouraging readers to engage not just with tools but with the underlying methods and implications of ATR.

1.1 From Books to Bytes

Since the invention of writing, the written word has been at the centre of people's engagement with their history. Only in writing can personal experience transcend the present and become part of a collective cultural memory (Assmann 1992) that is shared or researched by many generations, without reliance on memory and an unbroken chain of oral transmission. Although historiography can draw on many different sources – archaeological finds, oral histories, etc. – the written word remains the most important form of transmission, as it is the clearest and most durable record of human history.

Every piece of historical writing is bound to a physical medium – clay tablets, papyrus scrolls, parchment manuscripts, books printed on paper, or even magnetic storage devices – each with its own characteristics and firmly situated in the material world and its practices. Until recently, engaging with these artifacts and the text they convey meant traveling to archives or libraries, handling fragile documents, and painstakingly reading and transcribing texts page for page, line for line. This made research a labour-intensive, time-consuming, and costly endeavour, requiring scholars of all kinds to focus on few documents. These constraints were acceptable for a long time, as they were mirrored by similar limitations in scholarly practice, especially in the requirement to publish editions of texts as printed books.

Advances in digital technology in the late 20th and early 21st centuries have changed this situation fundamentally: Today, digital methods have become an integral aspect of the analysis, interpretation, and presentation of the past, which are, by definition, based on digital *data* instead of physical *artifacts* or their analogue copies (like film photography). At the same time, a significant increase in affordable computing power makes it possible for scholars to process vast amounts of such data, which in turn has boosted the popularity of methods such as distant reading or pattern recognition across a large body of sources. Consequently, as our engagement with historical material becomes increasingly digital, so too do our expectations of the accessibility, usability, searchability, and computational potential of this material.

In the past, the inherent fragility of historical documents was considered a natural, albeit inconvenient fact of life. Only a small fraction of historical documents could be photographed, and even these images were accessible only in a selected few libraries and archives. Recent developments have made this an unacceptable limitation and led libraries and archives to undertake enormous mass image digitisation efforts over the last two decades. However, if such images still require manual reading and interpretation, they offer limited advances over their physical counterparts beyond their remote accessibility. Without the ability to computationally search and analyse their content, these images remain part of a physical culture of engagement with the past. Only by transforming them into machine-readable textual data can the records of the past be effectively utilised in digital scholarship and the capabilities of digital tools and methodologies fully leveraged on a broader scale. Thus, this transformation presents us with one of the most significant and important challenges to modern historical research, particularly if these records were written by hand.

Luckily, a solution to this challenge has emerged in recent years: Automated text recognition (ATR), also known as handwritten text recognition (HTR). Particularly since the 2010s, machines have become able to analyse layout and transcribe text from images with remarkable (and growing) accuracy thanks to advances in the field of machine learning, like CNNs and LSTM cells. Since then, ATR has evolved from a technology that many scholars viewed sceptically to an indispensable tool in the humanities. As the development of user-friendly ATR platforms and tools, such as Transkribus and eScriptorium, has also made the

technology more accessible to non-specialists, scholars can now digitise printed and handwritten texts at scales and speeds previously unimaginable, allowing the conversion of large volumes of historical documents into machine-readable formats. Thus, by the time of writing, ATR had quickly made its way from an experimental technology to a tested and indispensable mainstream tool.

Despite its increasing use and usefulness, the field of ATR is far from easy to understand. On the contrary, it is complex and rapidly evolving, requiring a solid understanding of concepts of computer science, especially machine learning. Also, even experts may find it hard to keep up with ever-changing technologies, tools, methodologies, business models, and use cases. Understandably, this dynamic leaves many scholars feeling overwhelmed and uncertain how to approach their specific projects and make informed and sustainable decisions. At the same time, the growing availability of ready-to-use commercial platforms and tools makes it easy to take a back seat in this development. However, making *use* of these technologies without a deeper *understanding* of their technical, methodological, and even ethical consequences and challenges is risky: It creates dependency on the commercial interests that drive these developments, and even more importantly, it deprives the humanities of their critical voice at a time when machine learning and "artificial intelligence" are entering culture, society, and our daily lives on an unprecedented level and with yet unknown outcomes. Thus, understanding ATR and the machine learning technologies behind it is not only important in terms of making research more efficient but should be considered an important part of algorithmic literacy, i.e. the ability to understand and make informed decisions about the digital technologies that impact our lives and scholarship more every day (Ridley and Pawlick-Potts 2021).

1.2 Purpose and Outline of This Book

This book aims to help scholars navigate the dynamic realm of ATR and to reflect on its challenges and opportunities. Thus, it is intended as a guide, providing an overview of recent and future developments of ATR, yet primarily aiming to foster algorithmic literacy. It caters to two distinct audiences: It offers practical tips to users who want to implement ATR for their own research without having to write code and to work at the algorithmic level. To more experienced users who are able

to engage with machine learning methods at the programming level, it also offers deeper insight into what drives ATR. It shows how to tackle both small projects on individual sources as well as large projects handling big quantities.

Any book about a technology that is developing as rapidly as ATR risks quick obsolescence. For this reason, this book refrains from providing definitive answers and instead seeks to equip readers with the ability to ask the right questions and make informed decisions that outlast the rapid development cycles of technology and tools. To achieve this, the book is structured around four pillars, allowing for multiple perspectives on ATR and its associated technologies:

1. Understanding Automated Text Recognition: The second chapter focuses on understanding the technologies underlying ATR. It explains ATR as a fundamental computer science problem of creating strings (text) from image data (scans and photographs) by applying deep learning algorithms and neural networks, giving the reader a solid basic understanding of the technologies discussed throughout the book.
2. Tools: Chapter three explores the variety of ATR platforms and software available, assessing their features, advantages, and limitations to help readers choose the most suitable options for their projects.
3. Core Concepts: Chapter four is the heart of this book, addressing fundamental principles and concepts of ATR that need to be considered for its successful, efficient, and ethical implementation.
4. Case Studies: Finally, the technologies and concepts discussed in this book are applied in four case studies from current research. They demonstrate practical applications and real-world examples of ATR across different disciplines, demonstrating how ATR can be successfully implemented in your own research.

Let us finish with a couple of remarks on the structure of this book. It is intended to serve as a practical guide to ATR, yet during the process of writing it, we realised that the guide itself may require some guidance. In our effort to address a diverse audience – including scholars in the humanities, computer scientists, and practitioners from memory institutions – we kept adding more layers of information. Keeping a stringent narrative while also allowing readers to explore the book in sections turned out to be a challenge. As a result, early drafts were, frankly, a bit

chaotic. To bring clarity to this complexity, we structured the book very clearly to support both linear and modular reading. To provide both structure and navigability, we have implemented the following features:

References: While we strive to be a reliable and informative source, we deliberately limited the number of references to enhance accessibility. We included key literature where possible but covering the full breadth of this interdisciplinary field proved impractical. Instead, we trust our readers to follow up on the literature as needed. After all, this book is designed as a *guide*, not a work of original research.

Definitions and glossary: Given the number of technical and domain-specific terms, we defined key concepts upon their first introduction in the text. In addition, a glossary at the end of the book offers brief definitions and explanations of all terms that might need clarification. Terms included in the glossary are marked in the main text with the dotted underlining once every paragraph.

Cross-references and asynchronous reading: Recognising that many readers will use this book selectively or asynchronously, we embedded numerous cross-references throughout the chapters. The highlighting indicates that a relevant section appears elsewhere in the book. We intentionally avoided page numbers to keep the reading experience fluid and less visually cluttered. These cross-references are clearly labelled and easily found via the table of contents. In addition, we have tried to link to as many online resources as possible, either in text or in the bibliography. All of these links have been checked in April 2026.

Section overviews and reading aids: To support non-linear reading, each chapter begins with a brief overview box explaining the function and content of the section. When a chapter goes into more technical or abstract detail, we included boxes that simplify or encapsulate the core ideas or technical excursions. This allows readers to either engage more deeply or skim the material according to their interest and expertise.

Case studies: While the case studies in chapter 5 are stand-alone elements, they also reflect the methods and technologies introduced in earlier chapters. Since we gave the authors full freedom to present their application of ATR in their own voices, these contributions are not as tightly integrated into the narrative structure of the book. We recommend you

read them last, as you may see many of the decisions and concepts presented there in a new light.

With these aids in place, we trust that readers will be able to navigate this book as well as the complexities of ATR. To further enhance the usefulness of this book, we published our Zotero library, making all cited publications searchable (see: https://www.zotero.org/groups/4971878/).

1.3 Key Definitions

Before we begin our journey into the rich field of ATR, let us introduce some of the terms we will use throughout this book. The following definitions are by no means an exhaustive list of all the abbreviations and acronyms used; they point out key concepts and clarify our understanding of them. The first three are similar and sometimes used synonymously:

Automated text recognition (ATR), or sometimes automatic text recognition, is the general term for all forms of text recognition. Currently, neural networks are the primary technology used to achieve such recognition results, but ATR refers to the process regardless of the underlying technology.

Optical character recognition (OCR) is the oldest and probably best-known of three interrelated terms. Having its origins in the 1970s, it is a text recognition technique originally based on hard-coded individual character recognition. Today, only a very few applications still use this approach, which was highly sought after in the 1990s. Most current frameworks and applications now rely on machine learning approaches, but the term OCR is still often used to denote the recognition of printed and even handwritten characters.

Handwritten text recognition (HTR) is the most common term to refer to machine learning-based text recognition approaches, since the automated recognition of handwriting was considered the 'final frontier' for applications. Similarly to how the term OCR is sometimes used in relation to *handwritten* text, HTR occasionally refers to machine learning-based recognition of *printed* texts. We propose disregarding the writing mode (handwritten or printed) but rather focus on the process of recognition itself. For this reason, we will use ATR as our preferred term to refer to the recognition process.

Automated layout analysis or automated layout recognition is a closely related term for a necessary step preceding any ATR, the identification of written elements on a page. When this is based on computational methods, one or several algorithms identify text regions, baselines, and, in some cases, line polygons, making them explicit as coordinates on an image.

Finally, we decided to introduce one new term: An integrated transcription environment (ITE) is a platform that allows users to upload their own images, prepare ground truth, train models, use them for automated text recognition, and then export the results. All of this is done through a graphical user interface, either in an online or desktop application. ITEs are thus accessible to researchers without any knowledge of coding. ITEs differ from various programmatic ATR solutions on one hand, and from limited GUI solutions that do not allow training of custom models on the other. For other, more straightforwardly defined terms, please refer to the glossary.

As emphasised in this introductory section, there is no single, universally optimal path to building the perfect text recognition model. Instead, researchers and practitioners must choose from a wide range of tools and methods, each suited to different use cases, constraints, and goals. These options range from costly, highly optimised systems to flexible, lightweight solutions tailored for specific datasets or scripts.

In many fields, ATR is increasingly taking on the role of an auxiliary science, akin to disciplines such as sphragistics or, more prominently, palaeography. These auxiliary sciences are all fields of scholarly inquiry, shaped by internal debates, evolving paradigms, and shifts in methodological and technological practices. One of the aims of this handbook is to foreground ATR as such an auxiliary science: a domain not only shaped by technical progress but also by critical reflection on what is recognised, how recognition is achieved, and which epistemological, disciplinary, and practical implications emerge from these choices. While the tools and models may evolve, the underlying questions about goals, value, and impact of recognition remain central to digital approaches in the historical sciences.


UNDERSTANDING AUTOMATED
TEXT RECOGNITION

2 Understanding Automated Text Recognition

Chapter summary: This chapter introduces the technologies behind ATR, beginning with a conceptual overview of human reading and its digital counterpart. The historical evolution of ATR is traced from mechanical inventions to the breakthrough of deep learning, explaining how neural networks process image data to extract and recognise text. We discuss key technologies such as convolutional neural networks, recurrent neural networks, and transformers and their roles within the ATR pipeline. We also introduce fundamental concepts of machine learning – including supervised, unsupervised, and reinforcement learning – and highlight their relevance to modern ATR systems.

How this fits into the broader argument: This chapter lays the technical groundwork for the remainder of the book. It equips readers with a foundational understanding of how ATR works and why deep learning has become the dominant approach. By explaining key terms and processes, it informs the following chapters – particularly those on tools and core concepts – by providing a conceptual framework to discuss ATR systems and their practical application.

2.1 How to Recognise a Text

Many factors distinguish humans from other species, but writing is among our most distinctive capabilities. While spoken words can create complex social structures, writing allows us to transcend the present, giving enduring form to transient sounds or thoughts. The written word is one of the most significant and multi-faceted tools humans have devised. It can convey divine messages through the ages or inform a visitor of someone's expected return to the office. However, writing alone cannot facilitate communication; it requires its counterpart – the ability to read (Figure 1).

While writing involves advanced technology (such as a writing surface and writing tools) and refined motor skills, reading appears mundane and straightforward at first sight – just peruse a page for a while. But reading is a multi-faceted process, as well. It entails both physical and

cognitive efforts and multiple steps. Initially, we need to perceive the individual geometrical shapes of letters (glyphs, which can possibly be allographs), typically with our eyes, one after another, by moving the eyes, or the fingers when reading a tactile alphabet (Schotter and Rayner 2015; Rayner 1998).

Subsequently, we have to correlate these shapes with their corresponding functional abstraction (grapheme) in a writing system, such as an alphabet. This stage, known as text recognition, requires specific knowledge about the writing in question, whether it be a specific alphabet or a colleague's distinctive handwriting. To finalise the reading process, the recognised graphemes can be associated with specific sounds they represent to discern words and their respective meanings. This, again, demands another level of knowledge, specifically understanding language and the subject matter of the text (Treiman and Kessler 2015; Yap and Balota 2015). This stage is referred to as content recognition (Perfetti et al. 2005).

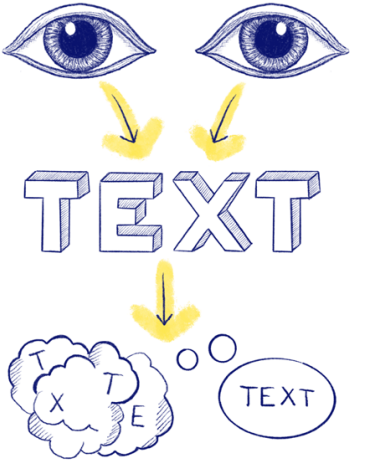


Figure 1: The process of text recognition, illustrated by Marie Machatová, CC BY NC 4.0

It is possible, albeit more challenging, to perform text recognition without any understanding of the content of a text. Conversely, content recognition, the final step in the reading process, cannot be performed without prior text recognition. While text recognition is an extraordinary

cognitive skill, it is nevertheless bound by three fundamental constraints (Perfetti and Hart 2002; Rayner et al. 2001) (Figure 2):

First, it requires sensory input, most commonly from the eye, as the initial gateway to the information presented. This sensory component is essential, because it facilitates the conversion of mostly visual stimuli into perceptual information. However, this also limits on the speed at which information can be processed, dependent on the musculature of the eye and its ability to move and refocus (Rayner 1998). This makes text recognition fundamentally linear, sequential, and thus time-bound, unfolding – at least theoretically – one letter at a time and restricting our ability to recognise and process large amounts of text swiftly and efficiently. Even taking into account speed-reading capabilities, the amount of text that can be processed will be starkly limited (Just and Carpenter 1980; Schotter and Rayner 2015).

Second, text recognition requires specific knowledge about the writing system. This domain knowledge is crucial for interpreting various symbols, characters, and structures inherent in different writing systems. Without adequate knowledge in this domain, text recognition is unattainable (Perfetti 2007).

Figure 2: Three constraints of text recognition, illustrated by Marie Machatová, CC BY NC 4.0

To overcome the inherent limitations of human text recognition, digital methods were developed to store, process, and provide access to text more efficiently. Attempts to develop automated text recognition began as early as the 1900s. Since then, technology has evolved through three significant phases. Initially, mechanical devices were developed to process written text, primarily to assist the visually impaired. In 1912, Edmund Fournier d'Albe invented the optophone, a device capable of scanning text and emitting tones to distinguish different letters. Two years later, in 1914, Hyman Eli Goldberg developed a process that involved printing characters in electrically conducting ink on a non-conducting surface, enabling the detection of the location and shape of printed characters (Buckland 2006: 162). In the 1920s, Emanuel Goldberg introduced a method to retrieve data from microfilms using optical scanning technology, a concept he subsequently patented in 1931 (Buxbaum-Conradi 2022).

While these innovations were marvels of human ingenuity, they failed to produce truly satisfying results until the advent of the second phase in the history of ATR, marked by the invention of the digital computer in the 1940s. This era witnessed "the birth of electronic data processing" (Schantz 1982: 6) and the advent of digital data creation, making the digitisation of written data a lucrative business model and channelling increased resources into text recognition. This processing evolution enabled innovations beyond the limitations of purely mechanical attempts.

The 1950s saw the development of devices utilising digital templates as exemplary images of single characters for character recognition, such as the OCR machine Gismo. The effectiveness of this OCR technology improved from the 1960s. The U.S. Postal Service adopted such processes for mail sorting, and corporations like IBM pioneered machines capable of extracting characters from typeset text (Schantz 1982). However, during these early stages of ATR, text recognition was confined to machines specifically designed for the task and therefore expensive.

A shift occurred in the late 1980s when ATR became accessible to personal computers thanks to stand-alone software solutions like Omnipage. These were able to recognise a diverse range of printed fonts (N. Y. Times 1988) relatively quickly and affordably.

These programmes operated by first segmenting individual characters from a text, then employed statistical methods to align the image of a character with existing templates (something like a silhouette), using a pixel grid as an overlay (Ningyang 1993). The cut-out of the image of a character was segmented into parts and compared to a library of templates. While the comparative process and the template database continually advanced and could attain up to 99% accuracy, this fundamental approach had its limitations: It functioned optimally only when the characters could be easily segmented, which meant they needed to be relatively uniform, and clearly legible. Therefore, these methods struggled with old prints and typewritten documents, whose fonts might be unfamiliar to the application, or which might be damaged or contain errors. Even more challenging was recognising handwritten texts whose characters often blend to create a more coherent appearance, don't represent any standardised font, and are often highly individualised. Therefore, although OCR generally yielded highly reliable results for modern printed text, the technology was not as effective in recognising historical and/or handwritten documents productively.

In fact, until recently, scholars deemed it near impossible to recognise such texts. However, the application of deep learning a decade ago permanently revolutionised the field of text recognition, resulting in the third and current phase of ATR. In 2009, Graves and Schmidhuber successfully utilised neural networks for recognising handwritten text (Graves and Schmidhuber 2009). This innovation in ATR was expanded in 2017 with the development of so-called transformer models (Vaswani et al. 2017), significantly enhancing the quality and efficiency of ATR.

We will delve into this in more detail 2.5 Deep Learning, but for now, it is sufficient to understand that neural networks enabled a highly adaptive approach to ATR, independent of any hard-coded templates for recognising characters. The network architecture allowed for the incorporation of language models in the form of lists of frequently occurring words and word combinations. Furthermore, the architecture was able to learn the domain knowledge essential for text recognition from given datasets, focusing on contextual sequences of text instead of isolated characters. This made neural network-based approaches exceedingly more flexible and versatile compared to traditional tools, incorporating a form of recognition which focuses on (a simple form of represented) content. This made these approaches so efficient that they have replaced

traditional OCR almost entirely, even for the recognition of printed text. The next section will provide you with a bird's-eye view on the process of neural network-based ATR.

2.2 A Bird's-Eye View on ATR

Every technical solution unfolds in three steps: an input, a processing mechanism, and a resulting output. In complex solutions, this sequence can be layered, with the output of one phase cascading into the next phase as input, eventually leading to the desired outcome. The principle of input-processing-output is also true for ATR, but necessitates at least two processes (Figure 3 and 4) to adeptly transform written text into a digital format: Initially, the machine must identify and segment all written elements on a page, a step known as layout recognition. Subsequently, the recognised sections must be extracted, and the text therein identified through text recognition.

Layout recognition: Any recognition process starts with a text requiring recognition. Ordinarily, this text is written on a physical object. For our purposes, let's consider a page in a book, although it might also be any other physical medium, such as scrolls, clay tablets, or a document. This object, serving as the input, cannot be utilised as-is in the recognition process, but must be transformed into a digital representation that is electronically processable (for difficult layout recognition processes see 5.1 Case Study 1 and 5.3 Case Study 3). This typically involves converting the object into a digital image, either through scanning or photographing, saving it as a raster image file consisting of pixels (Rehbein 2017).

This file format comprises a pixel matrix representing the colour or greyscale value of each pixel: for instance, a purely white pixel might have a value of 0, a purely black pixel a value of 1, and light grey a value of 0.5. Such an image file, typically saved in formats like jpg, png, or tiff, can then undergo further pre-processing before being fed into the layout recognition process.

Here, the pixel data are vectorised creating a so-called feature vector, which is essentially an ordered mathematical representation of the image's pixel values. A vector is nothing else than a numerical value that replaces the visual input. This vector is then introduced to a neural network (most often a CNN or a transformer-based network, as discussed

further below), producing an output vector (another numerical representation) that decides whether groups of pixels belong to a text region (e.g. a paragraph) or a line, resulting in coordinates of regions with a high probability of containing text. This output vector, usually transformed into a structured format such as PAGE XML (4.1 Data), can be employed to extract or visualise these regions on the original image and serve as input for the second step of the sequence, the text recognition process.

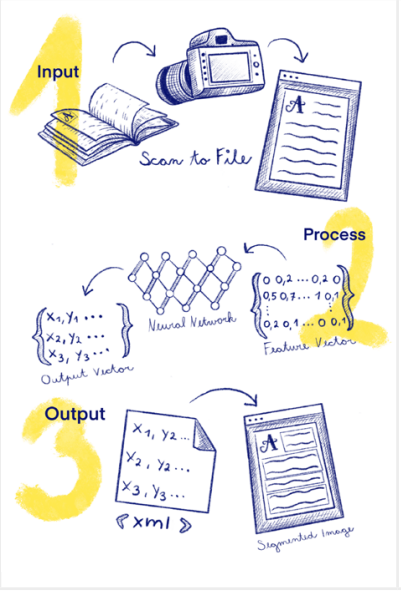


Figure 3: The process of ALA, illustrated by Marie Machatová, CC BY NC 4.0

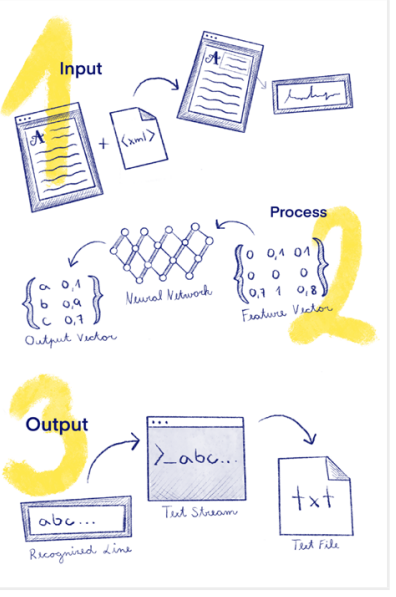


Figure 4: The process of ATR, illustrated by Marie Machatová, CC BY NC 4.0

Text Recognition: In this step, the data produced during layout recognition becomes the input for a dedicated text recognition process. Utilising the generated PAGE XML, this process extracts the individual text lines from the input image and re-converts them into a set of feature vectors – now a mathematical representation of the line's pixel values. Subsequently, these data are input into a neural network, specifically designed this time to map the vector to a sequence of a pre-defined set of characters (Jannidis 2017). In each of the steps, independent vectors are created, each representing different aspects of what can be found on a page.

During this process, the neural network tries to identify each written character based on previous training. It outputs a vector containing a list of possible Unicode candidates (4.1 Data), as well as the machine's confidence as a numeric value. For instance, the output vector for a written 'I' might look like: {'I': 0.65, 'l': 0.25, 'j': 0.14, 'm': 0.01}. Although this process encompasses more complex architectures, often including so-called language models, it essentially remains a simple mapping process, grounded in prior experience and stochastic probability.

Once the neural network has rendered its verdict on the various mappings, the characters with the highest probability are extracted from the output value and assembled into a sequence of characters, known as a string, which – hopefully – matches what a reader would read, on a given image input of the text line. This string can then be forwarded to other computational processes, or stored until all lines of a page are recognised, at which point they can be assembled to produce a textual representation of the written page in its entirety. Based on these data, numerous file formats can eventually be output, ranging from plain text to more complex formats, such as docx for Microsoft Word documents or TEI-XML, commonly for digital editions.

Without human intervention, the machine has successfully converted written text on a physical surface into a digital representation using a mechanism known as a neural network. In the next section, we will delve deeper into this technology and elucidate the recognition process in greater detail.

2.3 ATR as Machine Learning

As described above, the recognition of handwritten text by technological means seemed an unsolvable problem. This changed dramatically in the 2010s with the breakthrough of a new technology: deep learning based on neural networks. Deep learning can be considered part of the broader field of AI, a field of computer science that seeks to enable machines to solve problems in a way that rivals human capacity. As such, AI is not an intelligent entity or an artificial, self-aware being, but a variety of different technologies that are brought together to "do the right thing" regarding a problem from a human perspective (Russell et al. 2022: 19–20).

Traditionally, computers perform their tasks based on a set of rules, which are provided by the programmer in the form of an algorithm in a programming language. Think of the programmer as a cook who devises a recipe (programme) for the machine to execute in order to create a specific dish (output) from a list of ingredients (input), following processing instructions such as 'blending' or 'stirring'. If the recipe is meticulously followed, the dish will likely turn out as expected (and as planned by the author of the recipe, in our case the programmer), even if it is prepared with limited culinary skills. However, if certain specified ingredients or cooking utensils are missing, the dish might fail miserably, especially if there is no culinary expertise to improvise.

The same principle applies to traditional programming. Consider an ATR-related example, the task of expanding a large corpus of abbreviated Latin words. In this task, single abbreviations, e.g. 'sem℘' for Latin *semper* meaning 'always', need to be expanded to their full form, in this example, 'semper.' This task can be tackled efficiently by programming a simple script that replaces the brevigraph '℘' with the sequence of characters it represents, 'per'. By supplying the machine with this recipe, it can expand thousands of abbreviations in a matter of seconds, without having any general knowledge about the medieval abbreviation system – much like an unskilled chef can create elaborate dishes by strictly following a cookbook. However, if the machine is confronted with an abbreviation for which no rule has been provided (e.g. because it is using a yet unspecified special character), it will fail, because the machine lacks the intrinsic knowledge about the given task needed to improvise.

To prevent this, in traditional programming, the recipe must anticipate *all* likely occurrences to perform the task successfully, and provide strategies to prevent the programme from malfunctioning or introducing errors into the output in case of missing or erroneous input or unavailable tools. Thus, traditional computing requires a domain expert who knows which rules and exceptions will apply to a given task and dataset. In this example, it could be a palaeographer or another expert in medieval writing systems. But even then, ambiguous or context-dependent abbreviations might pose additional challenges. This can easily increase the complexity of the programme exponentially, making it hard or even impossible to create and maintain such an algorithm.

This example shows that the goal of AI to get a machine to *do the right thing* is quite difficult to achieve, as it is often not obvious what that might entail. Humans navigate this uncertainty by relying on their intelligence. In other words, we can *learn* to do the right thing on a given task, even if we are not explicitly told how, by identifying and inferring patterns or rules from past experiences or by inspecting existing data. This is why the process of cognition does not fail if confronted with the unexpected or unknown, but generally finds a way to handle or approach it.

Human intelligence

"[Human intelligence is a] very general mental capability that, among other things, involves the ability to reason, plan, solve problems, think abstractly, comprehend complex ideas, learn quickly and learn from experience. It is not merely book learning, a narrow academic skill, or test-taking smarts. Rather, it reflects a broader and deeper capability for comprehending our surroundings – catching on, making sense of things, or figuring out what to do."
(Gottfredson 1997: 13)

As this mental flexibility has proven very useful in problem-solving, it is unsurprising that computer science adopted the concept of deducing the next step from given data instead of fixed rules as early as the 1950s, establishing the field of machine learning. Although there are alternative categorisations (Kühl et al. 2022), machine learning can be understood as a subset of AI interested in the way a machine can figure out more or less independently *how* to do the right thing in regard to a given task.

Recently, this field has been elegantly defined as "the science (and art) of programming computers so they can learn from data" (Géron 2019: 2), but the box below offers a more formal definition. Instead of providing all possible rules and exceptions, in machine learning, a machine is given a set of data and statistical methods to 'recognise' these rules by itself, making it a much more flexible tool for real-world problem-solving. This is called deep learning, as the algorithm learns from the data, trying to see patterns humans could not. The metaphorical expressions 'learning' and 'recognising' have been introduced to help humans to understand the actual procedures through analogy, although these metaphors are not without problems, as we will see later.

Machine learning

"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience." (Mitchell 2013: 2)

The famous definition often attributed to Arthur Samuel ("[Machine learning is the field] of study that gives computers the ability to learn without being explicitly programmed"), as presented in many books on the subject, seems to be a misattribution, as it cannot be found in his works (Samuel 1959; Cattin 2018)

Today, machine learning technologies are used for many applications in everyday life as well as professional contexts, ranging from product recommendations in online shops, speech recognition by your favourite smart assistant, and tumour detection on brain scans to the recognition of handwritten documents. Despite the ethical problems it entails (Bender et al. 2021), it is safe to say that in the future, machine learning will be an even more integral part of our life, including the humanities and historical scholarship, starting (but not ending) with the introduction of ATR into the scholar's toolbox. Thus, understanding machine learning is not only important to grasp the opportunities and challenges of ATR as a particular application, but also the broader implications of this transformation on scholarship and society in general. So, let's see how machines "learn" to do the right thing.

2.4 How Machines "Learn"

The term machine learning subsumes different approaches of how algorithms can address a problem. While most algorithms follow a clear, "if this – then that" approach, machine learning algorithms are set up to imitate decisions based on previous examples or data. This was how the notion of "learning" came to be associated with this approach. But it is a problematic metaphor, as machines can't learn through experience as humans do, but through specific inputs put into context with an output. In order to get a basic understanding of machine learning approaches (for an overview see Géron 2019: 7–33) it is key to start with a classification. This allows us to introduce fundamental concepts before dealing with machine learning for the purposes of ATR. First, we approach the

issue from the perspective of the problems that need to be solved. They can be ordered into three categories:

Regression: Regression means that the problem at hand involves predicting an outcome based on input variables. Typically, this approach is used for predicting trends, such as economic or financial developments or weather forecasting. It relies on statistical methods such as linear regression, logistic regression, polynomial regression, or others. Usually, regression problems are addressed using supervised learning (see below). Regression is based on the concept of estimating or predicting a numerical value, in the sense of "regressing" to the mean or average value.

Classification: If the problem consists of sorting input into a particular category, we speak of a classification problem. In machine learning, such a problem is commonly addressed by supervised learning based on pre-labelled training data. After training, the machine should be able to classify new input based on the categories it has learned in that process. Classification problems are very common when dealing with images (is it an image of a dog or cat?) or text (is it spam or regular mail?), but it can also imply quite complex problems of NLP such as sentiment analysis or NER (Jurafsky and Martin 2009, ch. 17 and 22).

Clustering: In contrast to the previous approaches, clustering is based on unsupervised learning and tries to group unorganised and unlabelled data. In contrast to classification, the problem does not primarily lie in the application of pre-defined categories *to* data, but to derive such categories *from* data. This approach is used in topic modelling, for example.

Another way to categorise machine learning methods is by looking at the way machines are trained. Here, the most important distinction is along the line of supervised and unsupervised learning, or semi-supervised learning and reinforcement learning.

Supervised learning: In supervised learning, the machine is trained on data that include the desired outcome as a label attached by human intervention. An input is thus paired with an output through a label. For instance, a large set of images might include the label 'dog' or 'cat', depending on the image content, if that is the distinction the system is asked to learn. This method is not limited to classification but is also applied frequently to regression problems. Predicting the value of real

estate might be trained on a dataset including pre-defined variables such as postcode, proximity to the seaside or the city centre etc., but also the actual value. ATR, too, is a classic case of supervised learning, where a visual input in the form of an image pattern is matched with a corresponding character output based on a large dataset of images labelled with its corresponding text. Thus, in supervised learning, humans have a large influence on the machine's problem-solving capability: The data provided must have a meaningful relation to the problem (for instance, including factors that influence real estate prices) and also include sufficient and reliable labels. Simply put: A machine trained on 'dogs' and 'cats' will fail to identify 'birds', a system trained on 'togs' and 'Kats' or labels attached to the wrong image will lead to wrong classification, and images of text paired with incorrect transcriptions or a wrong set of characters will produce incorrect text recognition. Thus, in supervised learning, the training data needs to be prepared following clearly defined guidelines and annotated as carefully as possible.

Unsupervised learning: In contrast, unsupervised learning does not rely on pre-assigned labels. It is mostly used for problems that require the clustering of large amounts of unstructured and unlabelled data, but there are other areas of application as well, such as feature extraction, anomaly detection or association rule learning (to extract correlations between factors). In short, unsupervised learning is often applied to large and unlabelled datasets that need to be explored or put into some order.

Semi-supervised learning: There is also an approach combining supervised and unsupervised learning, called semi-supervised learning. This approach involves both labelled data, typically used for initial training, as well as unlabelled data. This approach usually applies a combination of supervised and unsupervised algorithms, mostly when only a small set of labelled data can be generated due to time and resource restraints.

Reinforcement learning: In the real world, many problems can neither be solved by pre-labelled data nor without any human intervention. This is particularly true when the machine's capabilities are not applied to static data, but to a changing and dynamic environment. Reinforcement learning is used if the goal of machine learning is to create an agent, for instance in a self-driving car, that can react to changes in its environment. In this method, the agent acts based on a perceived environment and gets a reward if it takes the desired action, or negative feedback for

an undesired action. The feedback strengthens or weakens the way a situation is processed, which improves the reaction to given environments. The machine thus learns patterns of desired responses to a given situation by trying to achieve as much reward as possible.

Classifying machine learning approaches along their degree of human intervention in the learning process is the most common approach to the issue. However, there are other classification systems that highlight other fundamental aspects of the learning process, as well. One is the fundamental division of approaches into online learning versus offline learning (which is the main method applied in current ATR systems). Despite the terminology, this has nothing to do with the world wide web but addresses if a system can learn continuously on a flow of input data (online) or if it needs to be trained on all available data at once (offline, also called batch learning). In offline learning, the processes of training and application are separated, making it easier to implement and less resource-intensive in an application. However, offline systems cannot adapt to changing input data on the fly but must be re-trained. Also, the process of training itself is usually more resource-intensive, as larger amounts of data need to be processed at once, and often several times.

In contrast, online learning is more adaptive to changing input data, as it can be re-trained based on single observations during production. That means that the system can adapt continuously to input data without the need to be re-trained fully. Thus, it can be less resource-hungry on the training side but is more difficult to implement and resource-intensive in production, when it needs to be able to process data in real-time. Also, the adaptiveness of online learning can be a disadvantage, as it might introduce unwanted changes to the system. All these methods are used to "teach" the machine how to do the desired thing in a given setting depending on the problem at hand. In principle, they can be applied using a variety of machine learning technologies, but for several years, a method called deep learning has proven to be the most effective one, particularly regarding ATR.

2.5 Deep Learning

The following section offers a detailed and somewhat technical introduction to deep learning, offering a better understanding of how ATR works technologically. If you're less interested in the inner workings and prefer a brief, non-technical summary of the aspects that pertain to practical application, feel free to skip ahead to the tl;dr (too long; didn't read) section at the end of the chapter.

Deep Learning From a Technical Point of View

Deep learning can best be described as a form of machine learning that relies on a specific form of digital architecture. All the above-mentioned forms of learning (supervised, unsupervised, reinforcement etc.) can thus be applied to the technology. Deep Learning is based on the application of neural networks and can be called 'deep' if these networks involve multiple layers that are only indirectly altered by applying stochastic algorithms. Stochastic means that some variables are influenced by randomness and uncertainty, but their behaviour follows predictable statistical patterns over time or across space. Some parts of the neural network are in a so-called 'hidden' state, since they are not accessed directly by human intervention but only through optimisation in training and application. The general idea of applying such neural networks for machine learning goes back to the middle of the 20th century. As early as 1943, McCulloch and Pitts (McCulloch and Pitts 1943) proposed a mathematical model of neural nets inspired by the function of its biological counterpart. This conceptual relation between artificial and biological neural networks still remains today, as neural networks "attempt at modelling the information processing capabilities of nervous systems" (Rojas 1996: 3), mainly in the way information is communicated within these networks, or more concretely, how the weights of single neurons are altered depending on the input.

In the human brain, information is stored and processed by a particular type of cell, the neuron. Each neuron has a cell body and is connected to other neurons through axons (for outgoing information) and dendrites (for incoming information). The contact points between two neurons are called a synapse. Here, information is transmitted from one neuron to another by changing the electrical potential of the other neuron through biochemistry once a certain threshold is reached (Rojas 1996: 8–23).

Neural networks try to abstract and simulate these biological mechanisms in a very simplified manner and apply them to computational tasks. For this, they use mathematical functions to activate a neuron, which is just a metaphor for a computing unit holding and processing numerical values.

The perceptron as a computational unit

The first artificial neurons were modelled as so-called perceptrons (Figure 5) as early as 1957 (Rosenblatt 1957). A perceptron is based on a binary step function for activation. That means it will take input values combined with a numerical weight and produce a single binary output (0 or 1). The perceptron is activated if the weighted sum of its inputs (and sometimes, a so-called bias) exceeds a particular threshold value.

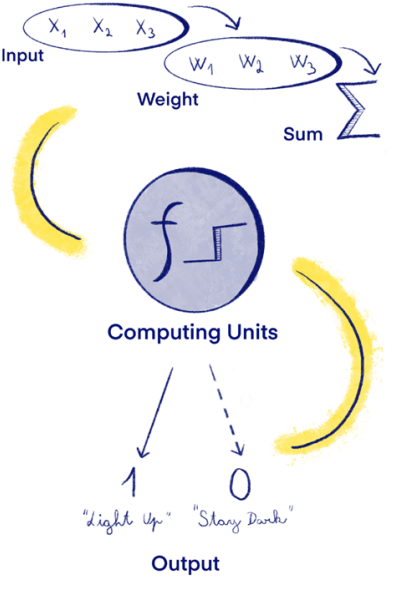


Figure 5: The perceptron as a computational unit, illustrated by Marie Machatová, CC BY NC 4.0

Early applications of this principle involved very simple structures consisting of only one neuron (perceptron). Although such a single perceptron can already solve very simple binary classification tasks, only

neural networks of multiple connected neurons are capable of complex tasks. In such networks, neurons are capable of exchanging signals with each other by taking numerical values from other neurons and passing them on after processing.

Sigmoid neuron

A sigmoid neuron is a particular type of artificial neuron utilised in neural networks. It is named sigmoid as it uses a logistic or sigmoid function for activation. While a perceptron can only handle binary input, this function is able to accept any real-valued number, such as 0.4 or 0.9, convert it, and normalise the output to a value between 0 and 1, depending on a defined threshold. This mathematical calculation is sometimes referred to as 'lighting up' or 'firing' a neuron, which simply means that the neuron calculates the value of 1 based on its activation function. The sigmoid neuron's ability to handle real values and produce non-linear output makes it capable of handling more complex operations and learn from error gradients. In this context, non-linear means that the output is not necessarily linear to a given input. A gradient represents a step to maximise or minimise an input towards a given output.

Although neural networks could be built from perceptrons yielding a binary output, in practice, they are built on neurons capable of taking and calculating more subtle values. These neurons are more capable and versatile than simple perceptrons, and they are particularly useful if they are part of a complex network of multiple neurons that can be linked together. To solve sophisticated tasks, networks of such neurons would typically be based on a particular architecture that organises its neurons in layers. In its simplest form, such an architecture would consist of an input layer representing some input data, a hidden layer processing the information, and an output layer representing the desired values. In practice, the three most important architectures can be categorised as follows:

Feedforward neural networks: Information is passed through such networks in one direction only. An important example of such an architecture are CNNs used for image classification and layout recognition.

Feedback neural networks: In these networks, information can be fed back from the output into the network using backpropagation. This makes such networks particularly suitable for problems involving sequences and context dependency but makes them highly complex and difficult to optimise. An essential implementation of this principle are RNNs, relying on LSTM cells that are capable of handling longer sequences of input, such as text (Leifert et al. 2016). Consequently, they are particularly suited for language processing and text recognition but require significantly more computational resources and data.

Transformer networks: The transformer architecture has been developed to overcome a particular weakness of CNNs and RNNs, namely their sequential treatment of input. By using so-called self-attention mechanisms (Vaswani et al. 2017), they can identify interdependencies between all elements of an input simultaneously, irrespective of their positions in a given sequence. This is particularly useful when dealing with complex forms of data such as language. This makes the technology very effective for complex tasks such as translation or text generation. However, this versatility comes at the cost of even higher computational power than RNNs. Large language models rely on transformers and wouldn't work as well without them.

Currently, CNNs (for image processing) coupled with RNNs (for the textual output) architectures are the most used applications for ATR. Most of the frameworks mentioned below (3 Tools) rely on such architectures that achieve the best results on datasets. However, transformer-based models, especially for large amounts of training materials, perform equally well or even better on many ATR tasks (Ströbel, et al. 2023b). Currently, two implementations of transformer-based models are used for text recognition, TrOCR on the one hand (Li et al. 2021), and Conformer on the other (Gulati et al. 2020). Increasingly, Transformer-based Large Language Models (LLMs) are being applied to ATR. While LLMs were traditionally restricted to post-correction, Multimodal LLMs (mLLMs) now integrate vision capabilities and can directly deal with images. First approaches show promising results for well represented writing styles (English and German after 1800), though their use for complex historical corpora remains to be studied.

Regardless of the architecture, all neural networks are based on the principle that their neurons (and layers) are connected and interact with

each other. Just as in the human brain, different groups and subgroups of neurons may be activated when confronted with different input, e.g. 'dog' or 'cat', making it possible to differentiate patterns in a dataset. To mimic this behaviour with computational units, each neuron carries a 'weight' as well as a 'bias', simple numerical values that influence the interaction of the neurons inside the network.

Weights and biases

In the context of deep learning, weights are fundamental components of a neural network, as they influence the relation between one neuron and another. Each connection between the neurons of different layers in a neural network has an associated weight, a numerical value that is initialised randomly. In the learning process, this value is dynamically adjusted in response to the errors the network makes during training. This process is known as backpropagation. The goal is to minimise an error function, often referred to as the loss function. Biases are additional parameters added to the weighted sum of inputs of a neuron before passing through the activation function. As such, they provide additional means to fit the mathematical model better to the patterns in the data, making the network more flexible.

Together, weights and biases essentially define the learned 'knowledge' of the neural network, which is thus nothing more than a mathematical representation. After 4.4 Training, a high weight between two neurons means that the neuron in the preceding layer has a significant influence on the neuron in the following layer.

When input is passed through the network, each neuron performs a mathematical calculation based on the input it received from previous layers combined with its distinct weight and bias, eventually activating a particular set of neurons in the output layer that represents the desired output, such as the label 'dog' or 'cat' if the goal is to categorise images of pets.

Let's explain this with the example of recognising one handwritten character. Usually, this example is based on the detection of handwritten digits (Michael 2015, chap. 1), but to adapt it to the matter at hand, imagine you want to recognise a fixed set of single handwritten

characters on images of equal size. In more technical terms, we try to convert a matrix of 256 input pixels with one of the 26 characters of the English alphabet. To achieve this, the neural network might receive an image as input and take each pixel as a neuron of the input layer. Let's assume these images are greyscaled with a resolution of 16x16, yielding a total of 256 input neurons that can hold numerical values between 0 (white) and 1 (black), depending on their greyscale value: A pixel from a thick black line might input the value 0.9 or even 1; a pixel from the white background a value closer to 0, and a grey pixel from the edge of a character approximately 0.5. These 256 values will form the neurons of our neural network's input layer.

The output layer of the network is a set of 26 neurons (using the English alphabet, and only considering lower- or upper-case) that represent its possible output (e.g. the letters a–z). Depending on the mathematical calculation of the neurons comprising the network, each of these output neurons will hold a number between 0 and 1, indicating how probable the network deems the given character to be the correct one. In our example, the neuron representing the letter 'l' has a value of 0.8, making it the most likely output, whereas the letter 'o' with a value of 0.01 is deemed to be the least probable.

As our example poses quite a complex problem, a network will have difficulty drawing such a conclusion directly from the input layer. Thus, neural networks have hidden layers of neurons between input and output, enabling more elaborate differentiation. In a hidden layer, each neuron after the input layer receives data from one or more neurons in the previous layer, passing the result on to the next layer and allowing a more complex analysis of the input data. After training (which will be explained further below), specific groups of neurons in the first hidden layer are activated by specific patterns in the data, such as horizontal lines or curves, based on the pixel values from the input layer. As this information is passed on to the next layer, neurons might be activated (meaning holding a numerical value close to 1) if the line is particularly long or almost vertical. Others might stay inactive (meaning holding a numerical value close to 0) and only fire if they recognise a circle. In doing so, hidden layers help generalise particular features present in the input data that can then be passed through the network and light up (again, holding a number close to 1) neurons of the output layer that are connected to the detected shapes. That is why our network was easily

able to differentiate between 'l' and 'o' but wasn't so sure about 'i'. However, it is important to note that we do not really know what features the network uses to come to its conclusion. We mentioned these geometrical shapes only for illustrative purposes.

In practice, ATR is much more complex than recognising a single isolated character, particularly when it comes to handwritten texts. Thus, real-world architectures of ATR applications focus not on the single character, but on entire lines of text, which they try to recognise using a so-called 'sliding windows' approach. This means the neural network is focused on one part of the image, which changes step by step (slides around). This makes it possible to handle the text as a sequence of characters that can be recognised based on the pre-defined set of characters used for training. Finally, the output of likely characters is further evaluated by a language model based on features of a given language, acquired during the training process. For instance, when recognising the Latin word 'natio', the character 't' might be chosen over the character 'i' if it appears in a sequence before the letters 'io', even if it is not written properly, as this is a common pattern in the Latin language, which the system has learned by ingesting the input data.

The non-technical introduction or tl;dr (too long; didn't read)

Deep learning is a branch of machine learning that uses a specific kind of digital architecture called neural networks. These networks are inspired by the way the human brain works: Information is passed between interconnected artificial neurons that adjust their behaviour based on experience. These networks are called "deep" when they contain multiple layers between input and output – layers that process information step by step and are not directly visible or changeable by humans. Instead, they adapt automatically during training through trial and error, using what we know as stochastic algorithms – processes that involve some level of randomness.

Originally inspired by biological neurons, artificial neurons are simplified computing units. The earliest form, the perceptron, was introduced in the 1950s and could only handle very basic decisions. More powerful types, like sigmoid neurons, can process a wider range of values and handle more subtle distinctions, making them suitable for complex tasks. In practice, many artificial neurons are connected in a network

that typically includes an input layer (e.g. pixels from an image), one or more hidden layers (where processing happens), and an output layer (e.g. predicted letters or words). These networks are used in several different architectures: some process input in one direction (feedforward networks, like CNNs for images), some handle sequences with feedback loops (RNNs with memory cells), and others (transformers) can process chunks of input at once using attention mechanisms.

These networks work by adjusting weights (which determine the strength of the connections) and biases (which help fine-tune decisions) during a process called training. This training involves showing the network many examples and using a loss function to measure how wrong it was, then improving step by step. For ATR, the system usually doesn't look at one letter at a time, but at entire lines of text. It processes these using a "sliding window" that moves across the line and suggests likely characters. A language model may help improve this output by checking if the result fits common patterns of a given language. While these systems can produce highly accurate results, they don't actually "understand" language or images but simply adjust values until the output matches the examples they were trained on.

2.6 How to Train Neural Networks

In the previous sections, we have discussed the basic components of neural networks and how information is processed inside them to come to a desired output. We have also introduced weights and biases as important factors for the activation of neurons, making it possible to generate highly differentiated outcomes. But how does the network know which weights and biases are appropriate in relation to a given problem and dataset? In deep learning, this is done by training a model.

Model
Weights and biases of a network are trained by adjusting them in relation to input data in an iterative training process. After a pre-defined number of runs, also called epochs, the parameters of a network are stored as a model, making it possible to reuse them for future tasks on similar data. The reliability is assessed on a validation set, checked after each epoch, and a test set that is applied after the training process.

This training or learning is achieved in a repetitive process that gradually changes the weights and biases of a network's neurons until it yields an optimal state. At the beginning of this process, weights and biases are distributed randomly to their neurons. Then the network is fed with data, for instance, images and corresponding lines of text, and trained by adjusting weights and biases using a mathematical algorithm called a loss function. This function will force the network to adapt its weights and biases until it yields optimal results based on the learning rate. The learning rate is a setting that controls how much the system adjusts itself after each round of training. You can think of it as the size of the steps the system takes as it tries to improve: Big steps mean faster changes, small steps mean slower and more careful adjustments. If the steps are too big, the system might miss the best solution; if they're too small, it might take too long to get there. Finding the right learning rate often requires trial and error because it depends on many other factors in the training set-up (4.4 Training).

Loss function

A loss function is a mathematical function used to measure how well a model's predictions align with the desired outcome values. It quantifies the disparity between the predicted and actual outcomes, providing a numerical value that represents the "loss" of the model's performance. The goal of the training process is to find weights and biases that minimise this loss function using optimisation algorithms like gradient descent.

As previously mentioned, models can be trained online or offline. In the online approach, the model weights are adjusted after each instance of learning data have been 'digested'. In offline learning, the weights are adjusted after processing a certain number of training instances, called a batch. When all the batches of a dataset have been processed once, an epoch is completed. The size of batches and the number of epochs are essential parameters that need to be adjusted depending on the dataset and problem. However, the quantity and quality of the data remain the most critical parameters for developing high-quality models. In the context of ATR, this training is based on a dataset comprising segmented images (one image per text line) with corresponding transcriptions. In each epoch, this dataset is shown to the learning algorithm once, with the total number of necessary epochs dependent on the type

of algorithm as well as the particularities of the dataset, such as regularity of a handwriting or characters used (4 Core Concepts).

While the training set is used to adjust the network's weights, the validation set is needed to evaluate the model's ability to adapt to unseen but similar data. This set is therefore used to monitor the model's performance during the training process and helps prevent what is called overfitting (to be addressed below). The test set, finally, evaluates the model's performance on unseen data based on a fully trained model. Ideally, the training set should comprise 80% of all the data, about 20% should be used for test and evaluation purposes. Once a predefined number of epochs has been completed or a certain quality been achieved, the training process stops, and the model can be saved and used on similar data. However, it does not necessarily mean that the model now understands why and how to do things; it merely imitates based on the training data. Having explored both the training and application of neural networks, it should be clear that the common conception of neural networks as an AI capable of learning and understanding meaning is somewhat misleading.

Machines do not really learn by understanding the problem they are trying to solve. They are trained purely by adjusting numerical values until the weights and biases align with a pre-determined output, much like shaping a key to fit a specific lock, where each small change brings it closer to the perfect match. However, this output holds no meaning for an algorithm beyond its statistical attributes. Therefore, while notions of cognition and intelligence can serve as useful metaphors, they should not lead to expectations of a learning process and understanding akin to ours. In fact, neural networks are best characterised by their famous description as 'stochastic parrots' (Bender et al. 2021), meaning that such systems can only reproduce features present in the applied dataset. Still, in practice, this 'parroting' can produce results that are indistinguishable from genuine understanding, which is impressive, and very helpful.

2.7 Traps and Common Misunderstandings

At the beginning of this section, we have shown that machine learning algorithms try to find fitting solutions to problems without explicitly being told how to behave. One way to achieve this is deep learning, a

process in which a network of artificial neurons represents the problem in mathematical terms based on a high volume of input data. The model's ability to transfer this representation to yet unseen data is called generalisation, and it is the key to solving a problem in the real world. However, this ability does not exist out of the box. On the contrary, there are many pitfalls that might prevent the model from generalising well or from doing the right thing:

Wrong understanding and framing of the problem: The first potential issue has nothing to do with the model itself but with our understanding of the problem. Often, humans solve problems based on experience and intuition without being able to formally express what is required to solve them. However, conceptualising the architecture of a neural network and curating sufficient data to feed to it requires a very good and formalisable understanding of the problem at hand and the expected outcome. One of the misconceptions of ATR is that it is a pure recognition problem that can yield objectively correct results. But it is much more than that, since reading is a form of navigating cultural knowledge. Thus, it should be understood as an interpretation problem: The machine tries to map what it sees to a set of characters a user has provided in the training process. However, this selection is by no means objective or self-evident, but depends on several assumptions, preferences, and choices on the part of the user: For instance, pre-nineteenth-century prints often include long-s and round-s, two distinct characters that might be preserved or expressed by a simple modern 's'. Thus, ATR is as much an epistemological problem as it is a computer vision task and users must carefully reflect and evaluate their assumptions and interpretations in order to produce a satisfying ATR result.

Another example of such a misconception might be over-emphasising text recognition, when in truth, the problem at hand is a layout recognition issue, for instance in highly annotated manuscripts or tables: Only if text regions, and more importantly, text lines are correctly identified and linked together to a logical sequence can the text recognition yield stable and high-quality results that actually make sense to a reader.

Wrong expectations: In the same way, it is crucial to take some time to reflect on realistic expectations toward a model. Careful planning should be devoted to its intended scope and the resources available to achieve it. Here, it is essential to realise that deep learning can be a very

powerful tool but requires much investment to perform with a certain degree of accuracy. If your resources are limited (as they usually are in an academic setting), you should not expect your model to rival the big-tech applications you might be used to. Instead, an accuracy above 90% should be regarded as solid, and anything above 95% a success, at least for text recognition of handwritings. Such rates show that you cannot expect models to operate without supervision and post-processing, which should be reflected in your planning.

Also, as the model needs to ingest well-prepared input first (at least in supervised learning), you should not expect the computer to solve tasks that are too difficult even for you. That means that you should at least be able to curate the data with which the model is trained or understand the decisions taken by others regarding these data. For instance, recognising names is especially challenging due to their particular patterns that might differ from the more general language patterns a model might have learned. Sometimes errors also cluster in highly problematic parts, such as damaged material, or due to faulty layout recognition. Thus, given the peculiarities of the text, even a very good model might produce a lot of errors in certain circumstances (or the other way around). So don't rely too much on quantifications of error rates and accuracy, and always test the model in real life.

Insufficient data: Data are the nutrition that fuels deep learning, and as a rule of thumb, the more, the better. Training a model that generalises well on new data might require thousands, sometimes millions of examples (transcribed lines in the case of text recognition), depending on the complexity of the problem a model is to address (Géron 2019: 23). As providing such data in a reliable quality is very cost-intensive, a solution is to share and reuse existing models and/or data as often as possible using FAIR principles (Wilkinson et al. 2016) and established platforms (Chagué and Clérice 2020) and standards (Romein et al. 2022). However, this can lead to serious issues if the existing data do not include needed features or is of low or different quality. In such cases, a deep learning approach might produce worse outcomes than traditional computing or a model might behave in unexpected ways. Thus, it is important to evaluate and document shared material properly.

Another aspect to consider is the quality of the dataset used for training. To be of high quality, a dataset must be a good representation of the data a model will be challenged with in its real-world application, which is particularly difficult if the dataset is very small, resulting in a so-called sampling bias (Géron 2019: 26). Such a bias might result from a misconception of the problem at hand but can also be the result of an unbalanced dataset. Strong bias not only makes a model less effective in problem-solving, as it will generalise more poorly, but can also lead to significant ethical shortcomings (Bender et al. 2021: 614–15), resulting in discrimination or marginalisation (4.8 Ethics). This is not only a challenge for applying artificial intelligence in everyday life but also a methodological problem in the humanities: Imagine, for example, a model trained on a large corpus of scientific texts of the 19th century (almost exclusively written by white men) being used to extract critical topics from a more diverse corpus. Naturally, the model would try to generalise based on concepts inherent to the training set and probably marginalise important views and topics – a big problem for discourse analysis. Lastly, the dataset could simply be poorly prepared and contain errors, introducing complications into the training process that may significantly impair the outcome.

For text recognition, a pre-trained model's inherent bias usually is unproblematic from an ethical perspective. Still, bias can exist in regard to special characters, rules of transcription, or languages included in a model (some examples can be found in the case studies below), and might become a serious problem, particularly if this bias only affects a subset of the documents, which are not represented in a test set. This is easily overlooked when checking the model's training set. Again, from an ethical perspective, it is very important to lay out the specifics of the dataset that is used to train a model in a transparent and well-documented way.

Overfitting and underfitting: A common and frustrating problem in deep learning is that models that perform exceptionally well on training data yield high error rates in handling unseen data, thus generalising poorly. This can happen if the model not only represents 'meaningful' patterns in the data but also data noise and random fluctuations. As a result, the model tends just to memorise the training data and performs poorly on new instances of data. As this is a very common problem in deep learning (particularly in small datasets), various techniques have been developed to prevent overfitting (Goodfellow et al. 2016: 107–17),

helping the model focus on learning the general patterns in the data rather than memorising specific details of the training examples. We will not go further into these methods but only emphasise a few: The number of epochs should not be too high, and, if overfitting occurs, it might be advisable to stop the training process. Early stopping of the training process ends the training process if the accuracy or the error rate on the validation set don't improve any further for a fixed number of epochs. In certain contexts, though, overfitting may even be beneficial, for instance, when local accuracy is more important than broader generalisation.

The batch size can also affect overfitting. A smaller batch size tends to reduce it, however, these variables depend on the quantity and quality of the training data and have to be evaluated individually. In contrast to overfitting, underfitting is understood as the model's inability to capture the underlying patterns in the training data, resulting in generally poor performance. Usually, this is due to a less complex architecture and can be addressed by choosing a more sophisticated network (Géron 2019: 29). Usually, overfitting is the more common problem in text recognition and addressed by a variety of steps: One approach is to improve the computer vision part (creating a vector based on a visual input) by creating multiple versions of the same image with slightly different colours (especially in the background) and different angles (Ströbel et al. 2023a).

Random initialisations: In addition to the issues already mentioned, it's important to know that training a neural network typically begins with random initialisation, unless you start from a pre-trained base model. As these random starting points can sometimes prevent the network from reaching its optimal state, it is recommended to train at least two models on the same data.

2.8 Conclusion

In this chapter, you've delved into the complex challenge of text recognition and the innovative technical solutions devised to automate this task. In particular, you've become acquainted with the technology of deep learning and neural networks, which are central to contemporary ATR solutions. While the specific architectures and methods are constantly evolving, you should now possess a robust understanding of the

underlying mechanics of ATR, as well as an appreciation of the technical, procedural, and ethical challenges associated with it (4.8 Ethics). As we progress through the book, we will revisit some of these concepts in greater depth. Especially in 5 Case Studies, you will find applications of the technology described here. The different use cases focus on specific needs and adapt their training and evaluation according to their goals. Now, let's have a look at some of the concrete applications in which these technologies are used for ATR.


THE BEST
TOOL

3 Tools

Chapter summary: This chapter surveys the current landscape of ATR tools, from simple drag-and-drop applications to highly customisable command-line interfaces and fully integrated transcription environments (ITEs). It guides readers through the selection process based on project scope, technical expertise, collaboration needs, and infrastructure. The chapter focuses on three major ITEs – Transkribus, eScriptorium, and OCR4All – comparing their features, usability, and sustainability, and highlighting how different tools serve different scholarly needs.

How this fits into the broader argument: Following the technical foundation we laid in the previous chapter, this chapter translates theory into practice by helping readers choose the right ATR tool for their specific use case. It supports the book's aim of promoting algorithmic literacy, not by prescribing any one solution, but by enabling informed decisions grounded in awareness of trade-offs, accessibility, and long-term sustainability. The chapter also prepares readers to understand and apply the core concepts of ATR discussed in the next chapter.

In the previous chapter, we discussed the technology driving ATR, which is currently based on deep learning approaches. Since it requires substantial expertise in machine learning and general programming to apply such processes from scratch, it is unsurprising to see an ever-growing number of newly developed tools to make text recognition and similar tasks accessible to a broader audience. Some tools are more accessible to users without a technical background while others require a certain level of technological knowledge. These tools range from local command-line interfaces, such as Tesseract or Kraken, to cloud-based platforms that include data storage and complex, pre-built ATR workflows, like Transkribus. Not only do these tools regularly register record numbers of users, but they have also become integral to many new research projects and applications. To meet this demand, the field is continually evolving, with new tools being developed and existing ones rapidly adapting to new technologies. Given this dynamic nature, it is quite challenging to stay up to date on all necessary aspects, even for informed users. Thus, in the following chapter, we aim to guide you through this challenging digital landscape and make you aware of the best solutions. Yet, rather than a 'one-size-fits-all' approach, the

reader will encounter a variety of diverging methodological focal points. We will first compare the current state of the most used platforms and then discuss important issues when choosing the right tool for your task. Unlike the previous chapter, we largely omit technical details and solutions requiring extensive programming experience. Instead, our primary focus will be applications that offer a graphical user interface, especially in what we will call integrated transcription environments (ITEs). These ITEs provide a holistic approach to ATR, encompassing everything from ground truth preparation and model training to layout recognition, ATR, and exporting results in multiple formats. Finally, we will not consider commercial solutions catering to large institutions, such as state libraries and archives, but only tools that are realistically available to a wider user base. Additionally, we will not discuss closed solutions and older programmes that are no longer maintained.

One important caveat: Given the continuous development of the programmes discussed, it is a challenge to convey enduring information in a static book. By the time you read this, some aspects and functionalities of these tools might have undergone significant changes in technology or business model, changed appearance, or even have been abandoned altogether. Consequently, any software-specific overview provided here can only be considered a snapshot. As such, this overview does not claim to present a definitive picture but aims to provide more general guidelines, enabling readers to navigate the diverse solutions.

3.1 Navigating the Thicket of the Forest

Before venturing into a dark forest, it is wise to reflect on your mission and understand how to move effectively through the wilderness. Without this preparation, it is easy to become lost in the forest's overwhelming vastness and fail your mission. The same principle applies to the multitude of ATR tools. While the consequences may not be as dire, the sheer volume of tools and functionalities is confusing and overwhelming. However, choosing the right tool is a critical task. Once a choice is made, each tool requires training and adaptation of your workflow (4.6 Workflow) and may involve substantial investments of time and, in some cases, financial resources, which might increase as the project progresses. Therefore, switching tools later in the process can be difficult,

time-consuming, and potentially expensive. This is why it is crucial to reflect on your goals before making your choice, and to find your bearings in the field of available tools for your purpose. To assist in this process, we suggest you explore the following nine areas, in addition to the core concepts discussed in 4 Core Concepts:

1. Project scope and intent: Are you merely curious about ATR or do you have a concrete project in mind? How many pages do you plan to process with ATR, just a few, or do you intend to handle large collections, perhaps even huge amounts of images?
2. Technical expertise: Can you dive deeper into ATR's technical foundations? Are you able to code and/or run your own server? Do you have access to technical support to handle these tasks for you?
3. User interface requirements: Is a GUI necessary for your work, or do you prefer working on the command line, anyway?
4. Collaboration needs: Do you require online collaboration features in your ATR tool?
5. Model training and ground truth preparation: Are you looking to train your own, specific ATR models or do you prefer to use existing ones out of the box?
6. Layout recognition: Is there a need to train a layout recognition model in addition to the ATR model to cope with specific or difficult layouts (5.1 Case Study 1)?
7. Further use: Do you need additional functionalities, such as annotating structural or textual information, NER, or similar features? Do you need built-in data export methods or do you prefer a flexible API that can be integrated into your processes?
8. License and expenditure: Do you need the tools to be open-source? Can you invest financial resources into the software, its application, or even proprietary developments? Are you dealing with sensitive information that may not be shared with anyone (e.g. for data protection reasons)?
9. Support requirements: How much software and community support do you need? Can you rely on a community or do you need professional customer service?

Ideally, your search for the right ATR tools should begin by reflecting on these key questions, as this helps filter the vast array of available software to identify those that best meet your specific requirements (Figure 6). For instance, if your goal is to occasionally transcribe a few

documents and save them as a pdf file, straightforward drag-and-drop tools like Transkribus.ai could be an ideal choice. They offer simple, free text recognition using publicly available models trained for various handwritings that might fit your needs just fine. While these tools might not support processing multiple images, custom model training, or advanced export functions, they might perfectly align with your current needs. If your goal is to frequently transcribe large amounts of text and incorporate ATR into broader workflows, drag-and-drop solutions will soon reach their limitations.

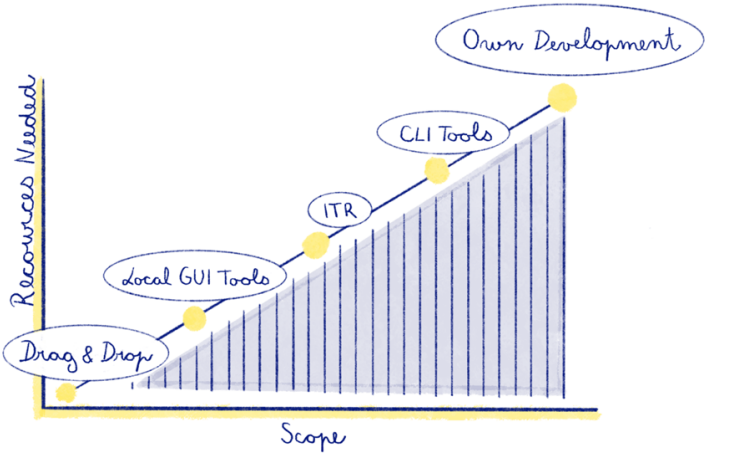


Figure 6: Increasing complexity of scope and applications, illustrated by Marie Machatová, CC BY NC 4.0

At this point, many scholars try one of the major commercial LLMs such as ChatGPT or Gemini on their material. Indeed, as is true for this technology more broadly, their ATR capabilities are rapidly evolving, making them a new standard tool in many workflows. For this reason, and for additional considerations discussed below, we will not discuss LLMs in depth here and only outline recent findings on the usefulness of LLMs for historical ATR, which might help you proceed down this path.

Recent studies suggest that LLMs such as Gemini 3 or Qwen 3.5 have the potential to outperform even fine-tuned traditional ATR systems on a variety of historical materials and under zero- and few-shot prompting conditions (Kim et al. 2025). Not only can they perform ATR on printed

material, but they are increasingly capable of recognising handwritten sources, as well. Humphries et al. (Humphries et al. 2024), for instance, demonstrate in their project *Transcription Pearl* that commercially available LLMs can transcribe 18th- and 19th-century handwritten English documents more accurately and rapidly than specialised ATR systems, particularly with an LLM-assisted post-correction step.

Another significant development is the advancement of multimodal LLMs (mLLMs), which integrate visual and textual modalities. Just recently, evidence has been provided that such models can outperform traditional ATR engines in both transcription and post-correction tasks for historical sources (Ghiriti et al. 2024, Greif et al. 2025, Humphries 2025), achieving an exceptional accuracy below 1% CER without the need for fine-tuning or pre-processing.

While these experiments yield very promising results, LLMs still have significant challenges to overcome, such as hallucinations and hyper-corrections. While these early experiments yield very promising results, mLLMs still face significant challenges, such as hallucinations and a tendency toward hyper-correction. Also, these systems currently do not provide explicit layout information, as their output is limited to unstructured text formats rather than explicit spatial metadata such as PAGE XML. These findings highlight that, while mLLMs hold considerable promise, they are not yet universally reliable for complex scholarly datafication at scale.

Another issue is that language coverage and domain bias still remain significant problems in modern LLMs. While Gemini, for instance, performs competitively with fine-tuned ATR models on English handwritten datasets, its performance declines noticeably on non-English materials (Li 2024). This suggests that training data biases still limit the broader applicability of LLMs and mLLMs for multi-lingual historical transcription, a limitation over which users have no control.

Still, given the rapid pace at which LLMs and mLLMs are evolving, it is plausible that they may outperform many of the traditional ATR engines discussed in this book. Nevertheless, we chose not to cover them in greater depth here, not because of existing shortcomings, but because their functionality remains fundamentally opaque: Researchers have no access to or control over crucial intermediate steps such

as layout analysis, text recognition, or sequence modelling (Greif et al. 2025). Thus, in practice, current LLM applications lack the configurability, modularity, and methodological transparency necessary to implement them in a responsible FAIR and CARE workflow. Thus, while multi-modal LLMs may introduce a paradigm shift in accessible historical text transcription, they are unlikely to fully replace specialised, modular ATR workflows (4.6 Workflow) tailored to the methodological requirements of historical research. Still, many of the foundational concepts discussed throughout this book will remain relevant as LLM-based approaches continue to develop and mature.

While drag-and-drop tools and mLLMs offer a low entry barrier, they are currently best suited for quick and simple ATR tasks. For research requiring greater methodological control and scalability, more advanced tools can be broadly categorised into four groups. They differ in the resources required for their successful application and the project scope they can handle. After drag-and-drop solutions, dedicated local GUI software comes into play for small to medium-sized projects. A notable recent example is Rescribe.xyz, a desktop tool for historic ATR. It offers an accessible way to utilise the OCR engine Tesseract for transcribing printed books, using pre-trained models for Latin, English, and French historical prints from approximately 1500–1800, and potentially for written manuscripts with a beta model for Carolingian minuscule. As a straightforward desktop application, Rescribe can be downloaded and used locally on a consumer computer like any regular programme. It allows for easy selection of source data from a folder, PDF, or even Google Books. Data processing with Rescribe is sufficient for many individual researchers and even smaller projects. However, while it is possible to adapt Rescribe to specific needs by altering the available source code, users without technical knowledge are limited to the performance of their machine and must manage their data locally, which complicates collaboration. Here, full ITEs such as Transkribus or eScriptorium are powerful tools. They offer comprehensive ATR solutions along with cloud-based image and data management and can handle small- and large-scale projects without necessitating programming skills. Current ITEs will be discussed in more detail below.

If your requirements are highly specialised, or you plan to integrate ATR solutions into existing workflows without a GUI interface, utilising ATR engines directly may be the best fit for your project. However, it

might demand significant resources to develop, adapt, and manage software independently. This approach can involve integrating existing CLI tools like Tesseract (utilised in Rescribe), Kraken (employed in eScriptorium and OCR4All), PyLaia (used in Transkribus), Loghi, or Yolo for various or all stages of the ATR process. Using CLI tools for ATR offers significant advantages to users who prioritise efficiency, customisation, and security. Although the transition to terminal-based operations might present a steep learning curve to unfamiliar users, once adapted to the terminal, their streamlined nature makes CLI tools easy to use, which eliminates complex (and sometimes poorly designed) GUIs.

One of the hidden benefits of CLI tools lies in their seamless integration into existing computational workflows (4.6 Workflow). If you employ ATR as part of a larger processing chain that might also comprise image processing or some sort of data manipulation, CLI tools can be incorporated into this chain without disrupting established processes. Additionally, CLI tools are often highly customisable, as they are not constrained by the limitations of a GUI. Thus, with little effort, CLI tools can be tailored exactly to your needs and data.

Another significant advantage of CLI tools is that they can be applied on your local machine, thus making you independent from external infrastructures. This may reduce costs but also guarantees stable workflows that are under your full control. Also, as your data are processed locally, it is unnecessary to upload it onto an external server, significantly enhancing data security and avoiding potential copyright or other legal issues. This advantage is particularly crucial for projects handling sensitive data, such as classified or otherwise protected documents.

While CLI tools offer significant advantages, they also present potential issues. First, as they are deployed on your machine, you are entirely responsible for providing and administering the necessary infrastructure. This involves having sufficient hardware, particularly if you want to train models from scratch but also keeping software packages up to date and handling any misconfigurations that might arise. Regarding hardware, mid-range laptops or older hardware may struggle to train or process larger amounts of data, which should be performed using a dedicated GPU with at least 12GB (v)RAM as well as sufficient System RAM.

Since they run locally, CLI tools are usually designed for individual use. Thus, using them in a collaborative setting may be challenging and require external services such as GitHub or Collab. Also, your individual set-up may restrict the scalability of your ATR project. What might have worked for a couple of documents may become impracticable when applied to several thousand items. In this case, CLI-based workflows can be transferred to a server. While this may solve collaboration issues, such a set-up requires significant amounts of time and effort to maintain, and in most cases, some backend engineering, as well. Finally, take into account that not all CLI tools cover the entire ATR workflow. This means that users may need to chain together multiple tools to achieve their objectives. However, assembling a workflow from several different tools can be quite complex and require significant amounts of tweaking and configuration.

One tool that strikes a balance between accessibility and functionality is Kraken OCR. This software package, written in Python, offers utilities for text recognition and layout analysis as well as training and evaluation in one package. As such, it is used in GUI software like eScriptorium as primary engine but also offers CLI usage via its 'ketos utility' command. What sets Kraken apart is its ability to manage the full ATR cycle with a few straightforward commands: Users can input data, train models for specific recognition tasks, and output and publish their results, all within Kraken. At the same time, it is highly customisable, allowing for exceptionally advanced options and specialisations, as needed.

Alternatively, you might even opt to develop your own ATR solution using common machine learning frameworks such as PyTorch or TensorFlow. However, this should only be your last resort, as text recognition architectures are highly elaborate and challenging. Still, by constructing your workflow with these tools, you can manage complex and large-scale projects, leveraging the software's full potential without the limitations of pre-built GUIs, but at the cost of significant investment and resource allocation.

For most users, the best solution probably lies somewhere in-between employing a CLI tool such as Kraken locally and exploring an existing cloud-based ITE, which provides a comprehensive set of ATR-related tools while relying on an established digital infrastructure. Thus, in

the next section, we will first show how to train and use ATR models using Kraken and then go into the platforms that are currently the most prominent in this domain.

3.2 Integrated Transcription Environments

As established in the previous section, most users will opt for one of the major ITE developments of recent years. From our perspective, to be classified as an ITE, a platform must meet the following requirements: It must have at least a GUI to manage the entered data. Furthermore, the ITE must be capable of layout analysis and text recognition, including options for manual corrections. Furthermore, it should support training custom models and export into various formats. Currently, only a handful of platforms meet these criteria (in alphabetic order): eScriptorium, OCR4All, and Transkribus. Thus, although the landscape is rapidly evolving, we will limit our survey to these ITEs. We will start with a quick overview of their context and development before going into more detail.

In the following, we will compare the three platforms in more detail regarding the following features, based on the nine questions formulated in the last section:

1. GUI: Assesses the user-friendliness and intuitiveness of the interface.
2. Custom model training: Checks the users' ability to train custom models for specific text types or historical periods.
3. Availability of public (general) models: Focuses on available and importable models for recognition.
4. Layout analysis: Looks at the effectiveness of layout recognition and the ability to handle complex document structures. Layout analysis also takes into account the order and direction of reading, which is crucial for proper recognition.
5. Data management: Evaluates how the ITE manages large datasets and the ease of data access and organisation.
6. Annotation and collaboration features: Assesses the tools available for team-based projects and multi-user collaboration.
7. Export options: Reviews the variety of formats available for exporting processed data, such as PAGE XML or ALTO XML.

8. Cost, licensing, and business model: This section considers the pricing structure and licensing options, which are important for users with budget constraints.
9. Community and support: Assesses the level of support and resources available, including community forums, tutorials, and customer service.
10. Scalability and sustainability: Assesses how well the ITE can handle increasing volumes of data or expanding project scopes.

eScriptorium

eScriptorium was developed as part of the SCRIPTA-PSL project by the École Pratique des Hautes Études (EPHE, PSL University, Paris). The platform is relatively new compared to others like Transkribus and is built upon the Kraken OCR engine. Its development emphasises ease of use and accessibility and is community-driven and open-source. However, the developers do not provide a server infrastructure for the wider public. Although several institutions started setting up eScriptorium servers, making them available to the public (e.g. FoNDUE at the University of Geneva or the University Library of Mannheim), deploying eScriptorium on a larger scale will probably require you to maintain your own server.

	eScriptorium
Graphical user interface	Web interface based on the Django framework. Clean set-up with limited features that are relatively easy to use. More complex tasks (e.g. adapting user rights and model-sharing) are done in a Django-Environment.
Custom model training	Text recognition is based on Kraken (Kiessling et al. 2019); text recognition, as well as layout models, can be trained.
Availability of public (general) models	Many models are available through HTR-united and other publications in repositories (e.g. Zenodo). All models trained with Kraken/eScriptorium can be exported and imported.
Layout analysis	Option to train custom models. Supports connections to ontologies (such as Segmonto or custom ontologies). Good results for standard layouts. So far, does not support table recognition.

Data management	Straightforward upload options, including IIIF import. Data are managed at the level of “projects”, which consist of “documents” that include images.
Annotation and collaboration features	Projects can be shared across users. Annotation is supported on the level of layout (or parts thereof). Texts can be annotated based on a defined or imported ontology. At the moment, the annotations are not exportable.
Export options	PAGE XML (version 2019-07-15, not fully compatible with PAGE XML produced in Transkribus due to the later use of an older standard) and ALTO XML for ground truth or for recognised pages are available. All trained models (for both layout and text) can be downloaded.
Cost, licensing, and business model	Open-source software, which needs to run on a dedicated (virtual) server. Recognition and especially training processes are quite resource-intensive. Accordingly, GPU or access to a GPU cluster are recommended. The standard implementation tries to access dedicated GPUs, the FoNDUE implementation accesses HPC infrastructures based on SLURM job management.
Community and support	A fast-growing community of open-source enthusiasts, mainly in France, the US, Germany, and Switzerland. Support through GitLab issues and a communication channel.
Scalability and sustainability	Relies on an open-source community. The server needs to be maintained on its own. Larger processes, especially training, will require large storage and computing power (ideally with consumer GPU).

OCR4All

OCR4All is an open-source project developed within the framework of OCR-D, a DFG-funded joint project to improve the accessibility of early prints. The software is primarily maintained by the University of Würzburg. The system was mainly developed to process historical prints, but it can also be used for handwritten sources. OCR4All offers a high degree of workflow customisation and combines several tools (especially LAREX), allowing users to adjust the OCR process to their specific project needs. Text recognition is based on the Calamari OCR engine; Kraken and Tesseract models can also be used for the workflow but need some tweaking.

	OCR4All
Graphical user interface	The GUI centres on workflows for processing data collections. From this angle, different processing parts can be accessed. It uses the LAREX interface to correct layout and/or text; the interface is comparable to commonly available ground truth and correction interfaces.
Custom model training	Text recognition is based on the open-source OCR Calamari (Wick et al. 2020).
Availability of public (general) models	Some models can be found online. Compared to the other ITEs, fewer models are generally available. Export and import of models is possible.
Layout analysis	Based on Kraken, no information is available about trainability.
Data management	The tool is process-oriented and aims at providing customised workflows. Images are uploaded via folder systems outside of the web GUI.
Annotation and collaboration features	Offers no user management and therefore, no document sharing. However, several users can access the same instance, enabling collaboration. Annotation of text is not supported; the layout can be annotated (following a limited but expandable custom data model, which was created with early prints in mind).
Export options	Export of results as TXT and PAGE XML.
Cost, licensing, and business model	Open-source software, with no financial business model and no open-running instance. The instance has to be run on self-hosted servers.
Community and support	Widely used in Germany. Offers no running instance but can be downloaded and run on a consumer computer.
Scalability and sustainability	Relies on an open-source community. The server needs to be maintained on its own. Larger processes, especially training, will require large storage and computing power, ideally in the form of GPUs.

Transkribus

Transkribus is a product of the projects Transcriptorium and READ (Recognition and Enrichment of Archival Documents), both initially funded by the European Union. Since its beginnings in 2013, Transkribus has evolved into a robust platform used by numerous archives, libraries, and research institutions. Transkribus first used to deploy Hidden Markov Models (not based on neural networks) and was later based on "HTR" and "HTR+" engines. Today, it is based on PyLaia and TrOCR (called TrHTR within Transkribus). It is particularly noted for its large user base and its organisation as a cooperative, known as READ COOP. Besides recognition applications, Transkribus offers significant markup capabilities. It works out of the box thanks to its large server infrastructure that can be used in a freemium model by registered users.

	Transkribus
Graphical user interface	Its initial JAVA-based GUI called "eXpert client" is currently being replaced by a web front-end. Some options are currently only available in one or the other application. A merge of major functionalities towards the web version is underway.
Custom model training	Users can train custom PyLaia models within the platform. Training of TrOCR/TrHTr models (transformer-based models) is only possible on demand and carried out on-premises.
Availability of public (general) models	Transkribus offers the most exhaustive collection of models for a wide variety of writing styles and scripts. Import and export of models is not possible.
Layout analysis	Based presumably on Detectron2 (Wu et al. 2019), layout analysis is trainable and can also be used to train semantic layout annotations. as well as table recognition models. For standard layout, the layout recognition works well out of the box.
Data management	Different upload options (as JPEG, PDF, or TIFF) via folder systems, FTP servers, IIIF imports, or via URL (as METS, only in eXpert client).

Annotation and collaboration features	Transkribus supports textual and structural tags. Comments are done using a special type of a textual tag. Transkribus offers the possibility to annotate transcripts according to categories of your choice and to search large amounts of text according to these categories. Projects and models can be shared across users.
Export options	Transcription results can be exported as PDF, TEI-XML, DOCX, TXT, ALTO XML, and PAGE XML (version 2013-07-15, which can cause compatibility issues with other ATR platforms). The image files can be downloaded as JPEG. Two Excel options (XLSX) are available for exporting text tags (which can also be exported as PDF and DOCX files) and tables.
Cost, licensing, and business model	Transkribus requires credits to perform text and layout recognition. Each user receives 50 free monthly credits to an account. Beyond the free credits, including using transformer-based recognition, credit packages or subscriptions must be purchased. Discounts are available for project groups and educational use. The platform is provided and run by the READ COOP. It is closed-source, while mostly open-source tools are implemented.
Community and support	A growing community of users (members) in over 30 countries. Transkribus provides support through an online help centre with instructions for individual application areas and a newsletter (online).
Scalability and sustainability	The platform is quite stable, with minor hiccups when processing heavy loads. Overall, there is high uptime and access to a stable infrastructure for training. However, processing is significantly slower under the free plan due to low priority.

3.3 Conclusion

The presented ITEs each come with different advantages and challenges. While Transkribus offers the opportunity to quickly start and train specific models, eScriptorium and OCR4All are more adaptable to particular needs. The main difference lies in the way the ITE is served to end users: While Transkribus is run as a fee-based platform, eScriptorium

and OCR4All are open-source projects and can thus be installed and run locally or on larger server infrastructures.

Although at first glance, ORC4All and eScriptorium seem to be the cheaper solutions, since there are no charges for relying on dedicated infrastructure, server maintenance and administration might become more cost-intensive in the long run. Still, open-source software allows for adaptations to achieve the desired development, which is more complicated with closed solutions. In terms of capabilities, all three ITEs are more than capable of handling most jobs efficiently and recognising handwritten material from various script systems. In the end, the decision will depend mostly on financial considerations, access to infrastructure, and a general approach to open- or closed-source solutions.

In this chapter, we have tried to provide an extensive overview of various ATR tools for different use cases. As we have shown, there are many solutions to many problems. Although most users will be perfectly happy with the more streamlined approaches of one or all ITEs presented, it might be worth considering other approaches, as well. In any case, it is essential to understand the fundamental concepts of ATR in relation to your particular needs before engaging with the wide array of available ATR implementations. While the previous chapters have examined both the underlying technologies and the practical tools that enable ATR, the rapid pace of innovation means that many of these tools and systems may soon evolve or be replaced. The next chapter therefore shifts the focus from highly dynamic implementations to enduring fundamental principles.


CORE
CONCEPTS

4 Core Concepts

Chapter summary: This chapter outlines eight key concepts – data, layout, transcription, training and models, evaluation, workflow, use and reuse, and ethics – that form the conceptual backbone of ATR. It explains how each concept shapes the outcome of text recognition, stressing the importance of methodological awareness, practical decision-making, and critical reflection beyond specific tools or technologies.

How this fits into the broader argument: As the book's conceptual core, this chapter deepens the reader's understanding of ATR by connecting its technical foundations with real-world practice. It prepares readers to evaluate and adapt tools critically, laying the groundwork for applying ATR in diverse scholarly contexts and for engaging with the ethical and epistemological implications of ATR.

The previous chapters were dedicated to technologies and tools that are currently used for ATR. However, due to the rapid pace of technical progress in these areas, our assessment is likely to be obsolete by the time you read it. In this chapter, instead of discussing highly dynamic technical implementations of ATR, we want to focus on its underlying conceptual principles. Therefore, we have selected seven core concepts (data, layout, transcription, training and models, evaluation, workflow, use and reuse, and ethics) which are central for a deeper understanding of ATR. By addressing the methodological implications of working with ATR along these seven concepts, the reader should be able to make informed decisions about its application, regardless of the technology used.

Since working on digital representations of historical sources requires a fundamental understanding of the material itself, we start with basic concepts, like sources and their understanding as data, moving along the processing chain to layout and transcription, and ending with training and evaluation, workflow, and use and reuse, and ethics. Independently of this suggested reading order, each part can be read separately for reference. You will therefore come across some repetitions, which are intentional. As many of the discussed concepts are still under research, be it on a methodological (layout recognition), an applied (tuning of machine learning parameters), or epistemological (data) level, the cited literature is not intended as an exhaustive reading list, but meant

to offer the reader an entry point to further research. Each of these concepts represents a research field in its own right, so rather than covering every detail, our aim is to provide a foundational understanding.

4.1 Data

Data comes in many shapes and forms. In the context of ATR, you will primarily work with three types of data: images, plain text, and structured text formats like XML. Images are the starting point of the ATR process, capturing the original artifact – such as a printed or handwritten document – for further computational processing. Therefore, we begin by examining the formats and methods used to acquire and prepare these images.

Image data

When it comes to datatypes, what we refer to as an 'image' can actually be encoded in various ways. Commonly, these encodings fall into two categories: vector and raster data, which represent visual information differently. Vector images essentially store image data as an XML-encoded text file containing coordinates and colours for the geometrical shapes within the image. In contrast, raster images represent visual information as a grid of pixels. To scan and process images for ATR, raster files are mainly used. They come in multiple formats, most commonly as TIFF, PNG, and JPEG. While these formats differ in how they compress pixel data, they are generally compatible with ATR processes. There are many image viewers and editors available to convert between these formats if necessary, such as the open-source tool Converseen (Mondello 2024).

For ATR purposes, it is advisable to use files of good quality and high resolution, with attention to dpi (dots per inch) and general resolution (optimal are around 300 dpi and 5 megapixel). However, there is no need to use extremely large file sizes, as are often seen in TIFF images, since they can slow down processing, introduce further errors, and complicate file management, especially when there is limited cloud storage space. Also, deep learning (2 Understanding Automated Text Recognition) technologies today are much better capable of handling lower quality data as older OCR methods.

Some ATR solutions support the direct upload of PDF documents containing images of scanned sources. In these scenarios, the software typically performs an automatic extraction and conversion of images into a raster format for subsequent processing. This conversion step can also be manually accomplished or scripted (4.6 Workflow) using various available tools if needed.

As both scanners and everyday devices like cameras or mobile phones can provide the necessary raster files, successful ATR does not depend on professional scanning equipment. In fact, READ-COOP, the organisation behind Transkribus, even offers DIY solutions, including a product called ScanTent (https://readcoop.eu/de/scantent/), a simple portable device enabling comfortable scanning with a mobile phone, which is accepted by more and more libraries. However, professional digitisation offers a significant advantage in its ability to handle difficult-to-scan documents, such as manuscripts that may be difficult to lay flat. Professional equipment helps avoid issues like highly skewed images or loss of text due to tight bindings or other codicological features. However, it's worth noting that even such images can be improved for effective ATR by any established image editing or processing software, or specific solutions for handling book scans, such as YASW (Chéramy 2012). This step is crucial for maintaining consistent dimensions in hand-captured images and preventing artifacts from adjacent pages in the ATR output.

Even the highest image quality is futile if it's based on flawed input, such as a mediocre microfilm or microfiche. These intermediaries are frequently blurred, scattered with dots from earlier processes and often black and white, limiting the ATR's ability to accurately capture the nuances of a script, which considerably lowers the recognition rate. Therefore, whenever possible, it's advisable to source images directly from the original documents for optimal ATR performance. As most modern computer vision tasks (especially those based on convolutional neural networks) no longer require binarisation (see YOLO), there is no need to process the original images before applying ATR, although it can be beneficial in some contexts, for example, in case of poor scan quality (Jacsont and Leblanc 2023).

Fortunately, today, many libraries and archives offer digital versions of their sources in open access and high quality. Increasingly, these institutions are adopting IIIF, which facilitates easy and legal access to images.

IIIF uses a specific data type known as a manifest file, containing links to all digital images associated with a particular unit (like a manuscript) along with relevant metadata, facilitating ATR purposes, as many ATR platforms now support direct IIIF import of sources.

Text data

Based on the image data, any ATR process will then extract the text written on that image, generating plain textual data. Although plain text is one of the most fundamental and basic data types in computing, presenting its content strictly sequentially without any formatting and layout, it is simple only at first glance. This seeming simplicity masks the complex technical nature of digitally encoding the diverse alphabets and symbols from around the world – past and present. From a strictly technical perspective, encoding involves converting written characters into a standardised digital format that computers can understand and process. To do so, each character is assigned a unique binary codepoint that can then be mapped to a particular font shape, making it possible to display the same text data differently by switching fonts (Korpela 2006; Haralambous and Horne 2007).

Historically, the range of possible codepoints was limited to the English language in the ASCII standard, and thus insufficient for global and historical languages using special characters such as ø, ä, ß, or even Ω, ъ, and か.To overcome these limitations, the Unicode standard was developed to provide a significantly enhanced and flexible text encoding standard using 8, 16, or even 32 bits per character. Today, most ATR systems use this specific standard. Each Unicode character is assigned a unique hexadecimal identifier, known as a codepoint, which ranges from 0 to 0x10FFFF, encompassing over a million potential characters from various scripts, historical writing systems, symbols, emojis, and control characters that are grouped into 328 blocks, such as Basic Latin, Greek and Coptic, but also emoticons and the so-called Private Use Area, a range of codepoints set aside for highly specialised requirements (Korpela 2006; Haralambous and Horne 2007). Thus, today, we use a great variety of different characters, ranging from medieval brevigraphs to hieroglyphs, to express many of the characters or symbols we find in historical documents, although that freedom also entails several issues (4.3 Transcription). Problems often arise with right-to-left scripts,

in particular, as Unicode characters include information about whether they are read left to right or right to left.

It is key to understand that although text data might appear familiar when presented on your screen in a font, under the hood, it is nothing more than a specified codepoint differentiating one character entity from another, regardless of the form and shape in which the character might manifest in a particular font. At the same time, it is important to understand that a ATR solution will not consider the visualisation of these characters in a font, even if that font was used for inputting the data in a GUI. Therefore, it will not be more likely to confuse 'l' and 'I' if those characters are almost identically shaped in a particular font but only output plausible codepoints it has been trained on. Thus, it is sometimes hard to visually differentiate the characters used for encoding: ä and a + ◌̈ may look the same in most fonts, but they are, in fact, two entirely different codepoints as specified in Unicode. Therefore, it is essential to carefully document all characters used, as their appearance may vary across contexts – and more critically, in the context of ATR, undocumented variations can silently but significantly diminish the quality and usefulness of the ground truth.

XML data

Despite its seeming simplicity, plain text data can encode complex information about characters used in a text. Extensible Markup Language (XML), a third data type commonly encountered in ATR, introduces yet another layer of complexity. Unlike programming languages such as Java or Python, XML is specifically designed to structure and mark up text, facilitating the retrieval and processing of information by both humans and computers. In contrast to databases, XML data are stored in a sequential plain text format, allowing it to be opened and read in any standard text editor. What sets XML apart from the plain text format is its capability to encode and structure additional information through inline annotations. This is achieved using tags that encapsulate the text to be annotated. These tags, demarcated by angle brackets (< and >), have distinct start and end points, such as <tag>annotated text</tag>. An illustrative example of minimal XML code is:

```
<book>
  <title>Vademecum</title>
</book>
```

Tags can also have additional information that is stored in a structured way as attribute values inside tags:

```
<book type="monograph">
  <title>Vademecum</title>
</book>
```

This way, information can be stored in a structured and explicit way, allowing machines to retrieve and process the data at hand.

By design, XML is very extensible and customisable to meet specific needs. In our example, we could utilise Latin instead of English to "encode" the same information:

```
<liber genus="monographia">
  <titulus>Vademecum</titulus>
</liber>
```

While XML offers great flexibility, it also poses a significant challenge: Creating new tags for every use case can make it difficult to share and retrieve information across individual instances. Generally, XML is only interoperable as a format but not as an information carrier. In our example, the information is encoded using Latin text, meaning a search for "books" in English would yield no results. To mitigate this issue, several standards have been developed for specific purposes. In ATR, we encounter three main XML formats: ALTO XML, PAGE XML, and TEI-XML. Each of these formats is tailored to suit specific needs within the field, ensuring a more standardised approach to data encoding, and facilitating easier data sharing and analysis.

ALTO XML (Analyzed Layout and Text Object) and PAGE XML (Page Analysis and Ground Truth Elements) are specialised XML formats designed for describing the layout of pages, including any recognised text, in a structured and machine-readable manner. Both formats are capable of storing information about various regions on a page, individual text lines, or even single words. These elements are annotated with

corresponding pixel coordinates and supplemented with additional information, such as image links or the layout function of the elements. These details are typically captured in individual files, with one file representing one physical page.

For instance, in the transcription of the medieval manuscript St. Gaul, SB, Cod. Sang. 614 (https://www.e-codices.unifr.ch/en/list/one/csg/0614), transcribed by Tim Geelhaar for his model on Carolingian scripts (5.2 Case Study 2), the encoding of a layout element is exemplified as follows:

```
<TextRegion orientation="0.0" id="r1" custom="readingOrder
 {index:0;}">
 <Coords points="201,342 201,1956 1387,1956 1387,342"/>
 <TextLine id="r111" custom="readingOrder {index:0;}">
   <Coords points="364,403 1357,397 1348,331 363,342"/>
   <Baseline points="99,691 207,690 194,504 […] 1294,391"/>
   <TextEquiv>
     <Unicode>POSTQUAM BEATISSIMI UIRI · SANCTUS UIDE¬</Unicode>
   </TextEquiv>
 </TextLine>
[…]
</TextRegion>
```

In ALTO XML, the same page looks slightly different:

```
<TextBlock ID="r1" HEIGHT="1614" WIDTH="1186" VPOS="342"
HPOS="201">
 <Shape>
   <Polygon POINTS="201,342 201,1956 1387,1956 1387,342"/>
 </Shape>
 <TextLine ID="r111" BASELINE="99,691 […] 294,391" HEIGHT="72"
  WIDTH="994" VPOS="331" HPOS="363">
   <String ID="string_r111" HEIGHT="72" WIDTH="994" VPOS="331"
    HPOS="363" CONTENT="POSTQUAM BEATISSIMI UIRI · SANCTUS
    UIDE¬"/>
 </TextLine>
[…]
</TextBlock>
```

While most ATR solutions are capable of processing and producing both ALTO XML and PAGE XML formats, many users would not find any difference between the two. PAGE XML is the more commonly used format. Still, there are situations where the particular encoding of your data is important and can pose significant challenges. Firstly, converting between PAGE XML and ALTO XML formats is more complex than converting image files, as it requires knowledge of XML transformation or dedicated software such as OCR-conversion (Neudecker et al. 2015). Secondly, both formats have various versions employing slightly different encoding methods, which complicates the task of conversion or integration into an ATR process when the specific version is not supported. Lastly, some ATR solutions introduce proprietary XML elements into these files that are not part of the official specification and sometimes poorly documented. Furthermore, each software might implement these formats slightly differently, potentially requiring additional adjustments (Chagué 2021).

Despite these potential issues, ALTO XML and PAGE XML remain fundamental data formats in ATR. They serve to encode both textual and layout information, allowing for seamless data exchange between various ATR platforms or other software solutions, preservation for future use, and application in further model training. Additionally, these formats serve well in other contexts and processing pipelines (4.6 Workflow). Therefore, even if you may not directly engage with these formats based on your specific use case, understanding their role and structure is crucial in the field of ATR.

Both PAGE XML and ALTO XML formats are primarily concerned with encoding the specific layout of a document. In contrast, TEI-XML, the third XML format significant for ATR, focuses more on the perceived content of the document, which can be encoded on different levels (philological, historical, linguistic, etc.). This fundamental difference is evident in the way these formats are saved: While PAGE XML and ALTO XML are typically stored in one file per original page, TEI is saved as a single file for the entire document or its relevant parts. Currently, TEI is in its P5 version, which has been in use since 2007. In TEI, textual content is emphasised, and the physical position of text on the page is optionally represented using <zone> elements. These elements are typically placed at the beginning of the XML document. For example,

the same passage discussed earlier would be encoded in TEI somewhat differently, with an optional focus on the line positions:

```
<TEI xmlns='http://www.tei-c.org/ns/1.0'>
 [...]
 <facsimile xml:id='facs_2'>
   <surface ulx='0' uly='0' lrx='1664' lry='2496'>
     <graphic url='pathToFile.jpg' width='1664px'
     height='2496px'/>
     <zone points='201,342 […] 1387,342' rendition='TextRegion'
       xml:id='facs_2_r1'>
       <zone points='364,403 […] 363,342' rendition='Line'
        xml:id='facs_2_r1l1'/>
       [...]
     </zone>
     [...]
   </surface>
 </facsimile>
 [...]
 <text>
   <body>
   [...]
     <p facs='#facs_2_r1'>
       <lb facs='#facs_2_r1l1' n='N001'/>POSTQUAM BEATISSIMI UIRI
       · SANCTUS UIDE¬
       [...]
     </p>
     [...]
   </body>
 </text>
</TEI>
```

Although TEI-XML is the primary format for textual encoding in the humanities, particularly in digital editing, its layout encoding approach is not ideally suited for use in ATR processes, especially since TEI allows for combinations of layout information with transcribed text. However, TEI often plays an important role at either the start or the end of many ATR endeavours. It is a foundational format for the preparation or final presentation of textual data. Consequently, some ATR platforms, like Transkribus, provide out-of-the-box TEI export capabilities. This

exported TEI file is essentially a transformed version of the PAGE XML data generated during the ATR process. While annotations added with platforms such as Transkribus may also be converted, the transformation can sometimes be suboptimal. Therefore, if the aim is to create complex TEI documents, further transformation and refinement post-ATR might be necessary (4.6 Workflow).

4.2 Layout

If you have ever worked with sources, regardless of the historical period, you know that layout is a fundamental dimension of every document analysis. From handwritten medieval charters to printed newspapers of the modern age, a significant part of the meaning they convey is encoded not only in letters but also in the way these letters are arranged as paragraphs, headings, or marginalia. Without a basic understanding of a document's layout, its content cannot be fully appreciated and consequently understood. Thus, before working with the content of a document, it is essential to analyse its structure. A human reader who is used to the writing system of a document can perform such an analysis on the fly and directly extract the meaning of its text. Machines, however, face a more complex challenge. Unlike humans, they do not inherently understand the nuances of spatial arrangement or the contextual significance of different text elements that are presented visually on the page. Automated systems deciphering a document layout require a sophisticated process comprising image processing and pattern recognition techniques (2 Understanding Automated Text Recognition).

The primary challenges for ATR systems and the first goal of this process are to distinguish text from non-text elements such as images, borders, decorative markings, or even unintentional artifacts like stains or shadows produced by scanning, as well as elements added by holding institutions such as their name, logo, or colour wedges. Although this might seem straightforward to humans, it is in fact a complex challenge for machines. The process typically begins with pre-processing an image file created from the document in question. This includes steps like adjusting contrast and binarising the image to black and white or greyscale to enhance text visibility and simplify the image for analysis. Following pre-processing, the image is transformed into a feature vector. This vector represents the image as a comprehensive set of numerical values, which can then be fed into layout analysis algorithms.

For layout analysis, there are several technical approaches available. One common method employs flexible deep learning techniques, such as CNNs, which have shown effectiveness in recognising complex patterns (Borges Oliveira and Viana 2017; Ren et al. 2016; Umer et al. 2021; Deng et al. 2023; Clérice 2023). Alternatively, classical computer vision techniques, which rely on explicit algorithmic rules and mathematical models, can also be utilised (as detailed in Nixon and Aguado 2012). Regardless of the chosen technical solution, the objective remains to identify patterns within the feature vector or in comparison to learned vectors that correspond to geometrical shapes in the image. These identified patterns help establish the document's structure, enabling the ATR system to accurately differentiate and process text (broader regions and individual lines) from other elements present in the document. These elements are then encoded in hierarchical order with their exact pixel location on the image, such as in PAGE XML (4.1 Data) or similar formats.

Although these technologies can yield impressive results, layout analysis still poses a great challenge to the ATR process, especially when confronted with complex document layouts, which often has a significant impact on the quality of text recognition. One very common issue is the segmentation of closely packed elements, such as very narrow gaps between multiple columns, where the ATR system might fail to distinguish between separate text columns, resulting in messy or out-of-sequence text recognition. Additionally, inconsistencies in text alignment or irregular spacing within and between lines complicate the text extraction process. The presence of mixed content types, such as glosses or chapter numbers adjacent to the main text, adds another layer of complexity. This can pose challenges during both the pre-processing and feature extraction stages. Pre-processing adjustments, crucial for clarifying text, might inadvertently merge closely spaced text lines or columns if not adapted to the material at hand.

Similarly, the choice of algorithm for layout analysis also plays a critical role: While deep learning models offer adaptability, they may not always effectively handle unconventional or complex layouts out of the box without extensive, layout-specific fine-tuning. Traditional computer vision methods, though efficient in handling standard layouts, often struggle with atypical or cramped text arrangements that are common in historical documents. These layout-specific challenges can

significantly affect the accuracy of text recognition, leading to errors in content interpretation and data extraction in ATR systems.

This leads to another error potential in layout analysis, centred around the somewhat philosophical question of what actually constitutes a 'text'. While *geometric layout analysis* (Quirós 2018: 1) is primarily geared towards the *visual* structure of a document, deciphering the meaning of pixel values in relation to the overall image, from a human perspective, layout is more than the position of text on a page, but instead, plays a crucial role in *logically* structuring text within the confines of a printed or handwritten page.

Thus, meaningful engagement with a text goes beyond the mere recognition of where it appears on a page. The identified text must be arranged in a logical order (in modern research, this is typically linear and sequential, but there are exceptions, such as facsimile editions that maintain the original visual structure). Also, it must be determined which textual layout elements are related and how various layers of a written page should be understood and prioritised for processing. For instance, let's consider a letter with multiple layers: The sender's original writing, corrections made by a second person, a postal service stamp, annotations by the recipient, and perhaps even a copyright statement and ID added by the digitising institution.

To a human, the importance of these layers varies significantly, but to a machine, they are all the same unless they are addressed explicitly. This can be done by applying *logical layout analysis* with neural networks that are capable of labelling detected shapes based on annotated data (Quirós 2018) or by manually annotating the structural elements of the document. Both approaches must be based on a thorough understanding of the document and ensure that the methods used for layout analysis, such as a particular model, align with your understanding of the document. Ideally, it is also based on a shared annotation standard, such as SegmOnto (Gabay et al. 2021) or very well documented if it is based on a more individual solution.

To sum up, layout and its interpretation remains one of the key dimensions of information encoding in analogue documents, and layout recognition one of the most important but challenging steps in the ATR

process, as it lays the foundation of subsequent text recognition. To avoid disappointing results, at least three aspects must be considered:

1. Don't simply rely on your ATR software's default settings. Before starting geometrical layout analysis, make sure you have a solid understanding of the document structure and potential pitfalls, such as narrow columns or additions. Choose layout recognition models best suited to your needs and, if necessary, re-train or fine-tune accordingly. Also, anticipate whether manual intervention will likely be necessary and reserve the appropriate time to check the results before proceeding to text recognition.
2. Don't take your own understanding of a text's logical structure for granted. Although most humans might share some implicit assumptions on the logical function and importance of layout elements (although that is far from a given), machines do not, unless they are explicitly instructed. You need to make your assumptions and particular interests explicit and convey them to the model or algorithm at hand, for instance by annotating a particular dataset used for fine-tuning. Alternatively, you can post-process the results afterwards (5.3 Case Study 3).
3. Don't create data silos: As layout analysis generates research data with significant value, it's important to rely on established standards to ensure sustainability and machine readability and to make the generated data as accessible as possible.

4.3 Transcription

Since the beginning of historical and philological research, the practice of 'transcribing' has been at the heart of it. Scholars worldwide have spent thousands of hours sitting in libraries and archives, pen in hand, deciphering strange and exotic writing or typefaces and capturing their content in a more familiar script. Given the enormous effort involved with transcription, the desire to let a machine take over this tedious process by applying ATR seems natural. However, although ATR can be used to produce something resembling a traditional transcription, essentially, both the process and the product of transcribing change fundamentally when taken to a digital dimension.

But before we go into this matter more deeply, let's define what a transcription is. Although it is an often-used term in the humanities, it is more often casually used than precisely defined and can mean various things in different contexts and disciplines. In general, the term 'transcription' can be applied on three different levels and denote a product (the transcription), an act (transcribing) and a relation (between original source and transcription) (Huitfeldt and Sperberg-McQueen 2008). In our context, we will ignore the latter and concentrate on the *act* and *product* of transcribing, focusing only on the historical and philological research perspective on transcribing in relation to written documents.

On a higher level, transcription may be defined as the "effort to report – insofar as typography allows – precisely what the textual inscription of a manuscript consists of" (Tanselle 1978: 201). More formally, one could say that a "document (the transcription, T) is [...] a transcription of another document (the exemplar, E), if T was copied out from E with the intent, successfully achieved, of providing a faithful representation of a text as witnessed in E. Often the purpose is to make some representation which is easier to use than E is. For example, T may be easier for modern eyes to read or easier to duplicate. Or T may be able to travel while E cannot" (Huitfeldt and Sperberg-McQueen 2008: 296).

However, exemplars can be very complex entities that transport meaningful information about the transmitted text on many layers, including materiality, damage, later changes, and of course, most importantly, a text visually encoded in a writing system that is foreign to the modern reader. At the same time, the process of re-encoding this information by means of transcribing is limited to the expressional potential of the transcriber's writing system and medium, as well as their particular interests and resources. Thus, the process of transcribing isn't as clear-cut as it might seem and involves a plethora of decisions regarding the scope and detail of a transcription.

While the approach to transcription can vary significantly based on individual preferences, transcribers generally adhere to a set of guidelines established by their own academic community. Given that these guidelines can differ widely across different disciplines, we will not go into specifics, but rather show the broader range of transcription practices, which Driscoll calls 'levels of transcription' (Driscoll 2007).

A good framework for understanding the diverse approaches to transcription underlying these levels is W. W. Gregg's renowned categorisation of 'substantives' and 'accidentals' (Greg 1950). 'Substantives' refer to the essential content and meaning of a document, encompassing the words and ideas conveyed by the text. Transcribers focusing on 'substantives' aim to preserve the original message and intent of the document, potentially allowing for silent modernisation or standardisation of language to enhance clarity and accessibility. On the other hand, 'accidentals' pertain to the physical and graphical characteristics of the original document, including spelling variations, punctuation, layout, and even the specific allographs of letters used. Transcribers prioritising 'accidentals' strive to create a faithful representation of the document's original form and appearance, preserving its historical authenticity and idiosyncrasies. Ultimately, the key question in transcription is whether the aim is to convey the meaning of the original text, potentially at the expense of graphical fidelity, or to remain as true as possible to the original's graphical presentation, even if this means preserving historical peculiarities.

In practice, the main area where both approaches differ is graphical faithfulness. Transcribers may choose to preserve the distinct allographs of letters, or they might opt for a more standardised representation. Similarly, the extent to which historical writing practices, like abbreviations, are maintained can vary from silent expansion to transparent expansion, which is marked by brackets or similar symbols, to a faithful representation of the abbreviations, including brevigraphs. Practices also differ in the way they include textual layers. When the original text includes corrections or amendments, a decision must be made about whether and how to represent all these layers in the transcription, or to only choose a particular one.

Naturally, these and other practices span a spectrum ranging from facsimile transcriptions, which aim for a high degree of graphical fidelity to the original, to normalised transcriptions, which adapt the text to contemporary language standards. Diplomatic and semi-diplomatic transcriptions strike a balance between the two, with the former leaning towards preserving the original's form and the latter allowing for some modifications, though we must stress that the exact meaning of these terms can vary across literature.

Each of these concepts and practices originates in the analogue, paper-based world where the writing system and additional features of a historical document were translated into a contemporary format by way of graphical encoding. For this process, a set of visually appropriate letters must be chosen to represent the distinct style of the original scribe while also relying on graphical symbols such as specific brackets '<>' or '[]' to encapsulate the essence of the original writing practices, depending on the intended detail of a transcription. All information necessary for a proper understanding of the exemplar must be stored either inline or through footnotes. However, they are inherently constrained by the limitations of the paper's visual surface, which sets boundaries on what elements of the original document can be accurately and fully translated, thereby shaping and defining the scope of the transcription.

Consequently, the various decisions, practices, and concepts related to transcription that we have explored are, in many ways, a direct result of these analogue constraints. While these practices retain their relevance and applicability and can be replicated in the digital domain, particularly by ATR using appropriately trained models, it is crucial to acknowledge that a proper *digital* transcription involves a technical and conceptual paradigm shift and a significant departure from traditional transcription methods, since it transcends the limitations of the physical page. At the same time, it poses new challenges and raises new issues that require decisions to be made (Sahle 2013a; 2013b; 2013c).

The first difference between a digital transcription and a 'printed' one stems from the different technologies that are used. It has important conceptual implications: In the digital realm, content, while often displayed on a flat screen, is encoded digitally and thus not only visually. This distinction becomes particularly important in the context of ATR, where the transcription is eventually manifested through graphical symbols defined by codepoints, made accessible through fonts (4.1 Data).

These codepoints serve as unique identifiers, which can *then* be mapped to various graphical representations depending on the font utilised for rendering. As such, the characters identified by an ATR system are not directly tied to specific graphical symbols. Rather, they exist as abstract entities that can take on multiple visual forms. This decoupling of the electronic encoding from the visual representation has profound

implications for transcription practices, especially when preserving or distinguishing special characters, such as brevigraphs or unique allographs, becomes imperative (5.3 Case Study 3).

In such cases, the dependency on fonts to accurately render these special characters means that only certain fonts, specifically designed to accommodate these unique codepoints, can successfully display the intended graphical features of the exemplar. Consequently, this may lead to scenarios where the fidelity of the transcription's visual representation to the original is contingent on the availability and use of specialised fonts, underscoring the complexity and nuanced nature of digital transcription practices.

This particular challenge is considered a reason to prioritise the display capabilities of characters when choosing adequate encoding for a digital transcription (Guéville and Wrisley 2024: 10). While this is a significant consideration, it is important to emphasise that the variance in encoding options that creates these issues can also be an advantage of digital transcribing, as the features of the exemplar can be differentiated and statistically captured in great detail, even when an adequate visual representation may not be available. This is becoming particularly relevant as the digital age increasingly integrates machine agents that are indifferent to the visual equivalence between the exemplar and the transcription. Consequently, we would argue that digital transcription should be evaluated not just based on the visual outcome but primarily in terms of the analytical potential it enables.

In light of these differences between the potential of analogue practices to digital ways of transcribing, Robinson and Solopova (Robinson and Solopova 1993) have proposed a slightly different system of characterising the level of transcription that is more suited for the digital sphere. They propose a division into four categories: 'graphic' ("every mark in the manuscript, every space, is represented in the transcription, even to the point of decomposition of letter forms into discrete marks"), 'graphetic' ("every distinct letter-type is distinguished"), 'graphemic' ("every manuscript spelling is preserved […] without distinction of separate letter forms as in a graphetic transcription") and 'regularised' ("all manuscript spellings are regularised to a particular norm, perhaps the spelling of a manuscript considered authoritative").

Regardless of what level is chosen, it is crucial to carefully consider the set of characters and symbols used, as it can affect the compatibility of one's data with other datasets, especially in the context of training models for ATR (Guéville and Wrisley 2024; Romein et al. 2022; Clérice et al. 2024). While the analytical potential should play a significant role in the selection of characters, it is indispensable to consult existing standards and datasets (e.g. through initiatives like MUFI (https://mufi.info). If possible, choosing standard Unicode characters might be the best choice. It is equally important to thoroughly document the used characters and symbols, including the specification of codepoints, to ensure that the transcriptions created are appropriately usable and reusable by others.

As we have said earlier, the transcription process is not solely concerned with the textual content; it is equally involved with the faithful replication of writing practices observed in the exemplar. This task extends to the encoding of graphical symbols, as such symbols have been used to represent various scribal practices in line. For example, abbreviations may be conveyed by inserting the expanded form in brackets, and specific symbols often indicate deletions. Driven by neural networks, ATR applications are not only capable of recognising such practices in the exemplar but can also simulate the traditional in-line methods of graphically encoding them (5.1 Case Study 1) – provided that an adequate dataset of sufficient size was used for 4.4 Training.

Thus, thanks to the sophisticated language models they utilise, these systems can replicate the nuances of a diplomatic transcription as effectively as they do a silently normalised one. They are even capable of correcting scribal errors with impressive success, even though they sometimes yield hyper-correct forms. This efficiency of ATR systems is seductive – they promise to streamline the labour-intensive tasks of human transcribers such as deciphering, grammatically correct expansion, proofreading, and normalising, all within a seamless operation (5.4 Case Study 4).

Yet, digital transcribing is more than outperforming humans in terms of speed; it is also about creating a representative data model of the original document, which might include more than capturing the textual content, but also renders the complexity of its writing practices and 'accidentals'. By utilising digital text encoding and markup standards like TEI-XML, it's possible – and current best practice – to produce

multi-faceted representations that, for instance, preserve both the original and expanded forms of abbreviations, as well as the specific allographs alongside a standardised set of characters.

However, current ATR systems are not capable of supporting this type of multi-layered encoding. Although they can often export their results into data formats such as TEI-XML, what is exported is typically a one-layered sequence of characters. Thus, if the goal is to create a truly digital, multi-layered representation of an exemplar, the transcription process cannot be fully automated using ATR alone. Instead, it needs to be part of an extensive workflow that might begin with an ATR-generated diplomatic transcription, which is then post-processed by additional applications to expand abbreviations and encode the text in TEI.

Just as with varying levels of transcription in the analogue sphere, the digital world, too, offers a wide spectrum of transcriptional approaches to choose from. And again, this choice is not a matter of right or wrong; but about understanding that the decisions made in the analogue world need careful reconsideration and reflection in a digital context.

4.4 Training

All forms of supervised machine learning rely on training and model creation. Accordingly, one of the core concepts in AI is the process of training. The following part gives you an understanding of such processes regarding text recognition. It is no technical introduction but focuses on specificities of training text recognition models.

Training in deep learning for text recognition can be framed as a sports metaphor: Just like athletes continually refine their skills and techniques to improve their performance, deep learning models for text recognition require iterative training to enhance their accuracy and reliability. Also like in sports, the training must be aligned with a specific goal. Whether you want to gain stamina or speed, your objective differentiates how you pursue your training. Finally, it's also a matter of professionalisation. An amateur will not be able to devote as much time as a professional athlete and will not rely on the same professional infrastructure. This metaphor underscores the importance of scope, goals, determination, and ongoing effort in the pursuit of refinement in text recognition.

One key aspect is identifying the optimal amount of training material. Creating training material, or more generally ground truth, is tedious work, requires expert knowledge, and is thus expensive. Unfortunately, there is no general rule on how much data are required, but it depends on several factors. An important one is the questions to be addressed and the tasks you want to carry out with the recognised text. Other factors are the regularity of the script (also involving the scribe's level of training) and if the goal is to train a specific or a general model. The range of different characters that a model should be able to recognise is another factor that impacts the amount of necessary data. In short, balancing the amount of training material is a dynamic process and, of course, also impacted by data quality. Independent of all the above-mentioned factors, the general rule is: the more training material, the better the results (for an example of a general model see 5.2 Case Study 2).

Defining clear objectives is a fundamental step in developing and training text recognition models. Iterative methodologies are often productive, as exploratory – at times even playful – engagements can illuminate both the potential and the limitations of current technological implementations. While perfection in text recognition remains an aspirational benchmark, it is rarely achievable in practice and typically requires extensive manual intervention. Once you acknowledge that absolute accuracy is likely outside of your reach, your main focus will be on training models with clearly defined, task-specific goals rather than pursuing unattainable perfection. Defining achievable and meaningful goals helps guide the training process. Consequently, this means researchers have to know what parts of a text are most important to them and what tasks should be feasible once text has been recognised. If the goal is to search within texts for specific words and leveraging wildcards, a higher CER can be acceptable compared to corpus analytical inquiries. And for approaches that rely on language models (large and small-scale), we are currently learning that perfect texts are not a necessity at all, or that higher CERs do not hamper meaningful use of the derived transcriptions (Hodel et al. 2023). Moreover, not all research questions require an analysis of every component of a textual representation. In certain cases, marginalia may prove particularly valuable for capturing the gist or intent of a document. But sometimes, they only mirror information that is irrelevant for specific questions. Depending on the perspective, specific text regions can vary in relevance, and the quality of recognition in these regions can be adjusted accordingly in a specific context.

Typical Hyper-Parameters in Machine Learning

In the context of deep learning, numerous factors can influence the output of a trained model. These factors are referred to as hyper-parameters in machine learning terminology, as they govern the learning process that determines the model's internal parameters. To provide a general sense of how such hyper-parameters shape the training process and its outcomes, the following section offers a brief overview of key hyper-parameters and their roles:

Epochs govern the number of iterations that a neural network is trained with the same data. In each epoch, all available data are being fed to the network and the optimisation process is guided. After each epoch, the validation set tests the capability of the newly generated model. Generally, in deep learning, overfitting (2.5 Deep Learning) is a major problem since neural networks start memorising certain inputs. In the case of text recognition, this means that a line is recognised while specific traits of a character are not "learned". Thus, it is helpful to check the output of the model development for signs of overfitting (see below). In contrast to other machine learning applications, having too many epochs is not as problematic as having too few, as experience indicates that more epochs can lead to more stable recognition.

Batch size: We split the data into so-called batches to enable the algorithm to handle the amount of data used for training processes. A batch is the amount of data that is run by an algorithm before the optimisation process is initialised. The size of a batch is thus essential for the model. A larger batch size means the computer learns from more examples at once, making the learning process faster, but also requiring more memory and computing power. A smaller batch size might, conversely, provide insufficient context and thus result in suboptimal learning within the model. Depending on the utilised framework or engine, the batch size varies greatly: In its standard Transkribus implementation, for instance, PyLaia operates with a batch size of 24, but can technically operate on other sizes as well, and in Kraken, the batch size for training can be set via the "--batch-size" parameter. However, larger batch sizes also lead to larger demand on computing power and the GPUs (v)ram.

Learning rate: The learning rate is a critical hyper-parameter that significantly influences the effectiveness of the training process. It determines the size of the updates made by the optimisation algorithm to the model's parameters during training. If the learning rate is set too high, the algorithm may take overly large steps, potentially overshooting the optimal solution and failing to converge. Conversely, a learning rate that is too low results in very minor updates, causing slow convergence and increasing the risk of the model becoming trapped in a local minimum that appears optimal. While there are various techniques to estimate an appropriate learning rate, none of them can guarantee the identification of an optimal value. Consequently, selecting a suitable learning rate typically requires empirical tuning and iterative experimentation.

To understand the concept of the learning rate in model training, it is helpful to imagine hiking down into a valley in complete darkness, where each step must be of a fixed size. The learning rate determines the size of these metaphorical steps as the model seeks to minimise its loss function. If the learning rate is set too low, as depicted in the top-left plot of Figure 7, each step is very small. Although this cautious approach ensures that progress is steady and safe, it also means that it will take an excessive amount of time to reach the valley floor – the point of minimal loss. Conversely, when the learning rate is tuned appropriately, as shown in the top-right plot, each step is large enough to make meaningful progress without overshooting. In this scenario, the descent is both fast and stable, allowing the model to converge efficiently to the minimum.

If the learning rate is set slightly too high, as illustrated in the middle-left plot, the model begins to overshoot the minimum: Each step moves the model past the lowest point and back again, causing oscillations. Although under certain circumstances convergence may still occur, it is less efficient and introduces instability into the training process. When the learning rate is far too high, as shown in the middle-right plot, the steps are so large that the model consistently overshoots the minimum by a wide margin, failing to settle near the optimal point. Instead, the model 'zigzags' across the loss surface, never properly descending into the valley and thus failing to converge.

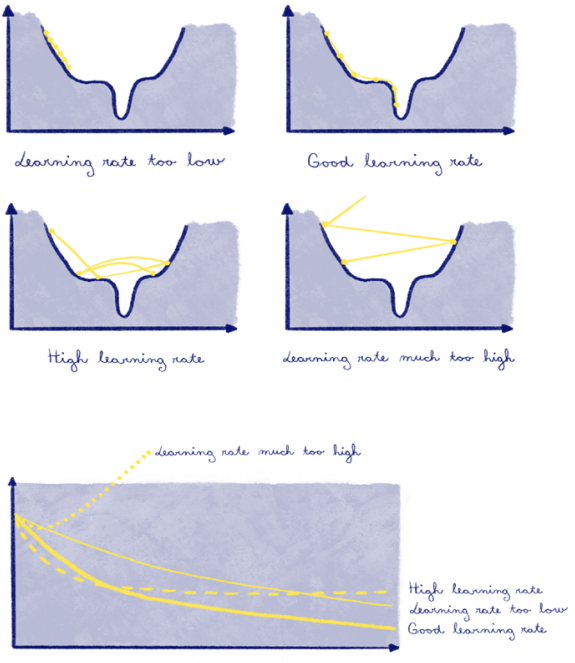


Figure 7: Different Learning Rates and their influence on Text Recognition, illustrated by Marie Machatová, CC BY NC 4.0

This is further illustrated by the bottom plot of Figure 7, which presents the model's training loss as a function of time. A well-chosen learning rate (solid thin line) yields a smooth and rapid decline in loss, indicating efficient learning. A learning rate that is too low (dashed line) results in steady but sluggish improvement. A high learning rate (medium-thick line) produces a rapid initial decrease, followed by oscillations and a slower convergence. A learning rate that is much too high (dotted line), however, causes the training loss to increase over time, signalling that the model is diverging rather than learning.

In sum, the learning rate acts as the critical parameter controlling the size of each adjustment made during training. If it is too small, learning is excessively slow; if it is too large, the model may fail to converge altogether. A properly tuned learning rate ensures that the model progresses quickly and steadily towards the minimum of the loss function. As Figure 7

illustrates, the training process can be conceptualised as a traversal of a "loss surface", a landscape whose lowest points represent optimal model states. The size of each step across this surface, governed by the learning rate, is thus central to the success and efficiency of the learning process. In short, parameterisation is quite specific and requires in-depth knowledge of deep learning approaches and insights into the material and what results can be achieved regarding the used frameworks.

Different Approaches to Utilising Models

To cope with the challenges of text recognition, as outlined in 2 Understanding Automated Text Recognition, it is key to understand the different results of training recognition models. Broadly speaking, the recognition process can result in two different models: specific, *specialised models* capable of recognising one specific hand or a well-defined set of hands, potentially coupled with prints; or *generic models* that can recognise entire styles of handwriting.

Of course, the two approaches require different input forms and entirely different amounts of training material. While specialised models need the greatest possible amount of training material from the specific hands or prints (including a representative set of the writing material used), general models require a balanced set of hands in order not to specialise towards highly represented hands or writing styles (Hodel et al. 2021). There's also an immense difference in the amount of required material. Depending on the regularity of the writing style or typeface, the first specialised models can be trained with an input of merely a few thousands of words. General models need large amount of training materials, at least some thousands of pages of input or about 100,000 word tokens.

Due to the required input, it's difficult to produce general models that are capable of following very specific transcription guidelines or dealing with abbreviations (4.3 Transcription). This mirrors the rule that models consistently reproduce their input, especially in large, general models. Based on the most frequent input treatment, the output will be modelled accordingly. At the same time, this means that specialised models can produce sophisticated as well as custom-made outputs, for instance, regarding visual or semantic interpretation: Italicised texts can be marked, or even place names can be identified with specific characters, although for the latter, NER approaches still yield better accuracy.

Going forward, the most effective and advantageous approaches will be a combination of general models that are used for fine-tuning to derive specialised models. Because a robust general model already provides the baseline, users only need to supply a modest amount of additional data – about 20 to 50 pages – to fine-tune it into a specialised model. With such a *base model* approach, text recognition will be much more efficient. However, this strategy pre-supposes the prior availability of broad, high-capacity base models encompassing every language, script, and writing style.

Any training process of text recognition models must necessarily rely on capable infrastructures. Most of the used engines require large amounts of computational power. It is advised to train on a dedicated server (Chagué and Clérice 2023) instead of personal computers. Generally speaking, a GPU on a more powerful computer allows for the training of PyLaia or Kraken models. Transformer-based infrastructures require access to more advanced GPUs with a suitable amount of vRAM.

As this chapter shows, training processes are complex procedures that require some tinkering independent of the used frameworks. Even if you do not plan to tune your model or trial your parameters any further, at least two or three training sessions of the same material are indicated from a technical point of view. However, as this comes with significant environmental costs, a reasonable balance must be found (4.8 Ethics).

4.5 Evaluation

Evaluation is a critical aspect of the development and implementation of text recognition models based on deep learning algorithms. Just as training must be tailored to a defined research goal, evaluation should also target the specific task at hand; hence, the aim is not to prescribe a universal method, but to differentiate between the various ATR use cases.

Therefore, the evaluation must be rooted in a theoretical framework that defines what is evaluated and for what purpose. This includes establishing criteria for assessing the effectiveness of a model in recognising text. Unlike in scholarly editions, a recognition process can be interpreted as temporary and dedicated to a single purpose. Examples of such goals can be the searchability of text for single words or whole sentences, or further processing using NLP tools like NER, sentiment

analysis, or other information extraction processes (for an introduction to NLP see Jurafsky 2009).

Consequently, the CER is the most frequently used, yet not necessarily the only metric to consider. The measure CER indicates only the percentage of correctly identified characters. However, as already discussed above (4.3 Transcription, 4.4 Training), not all characters are equally important for all goals of text recognition. This is why it make sense to use tools such as CERberus (Haverals 2023) that allow for a more nuanced interpretation of test material.

Splitting the data

Prior to evaluation, make sure not to evaluate data that has been used for training or validation within a model's training processes. Both training and validation influence the process of model creation and are not considered unbiased parts of the model's inception. Thus, it is necessary to strictly rely only on test split that has never been used for the training process, not even for validation purposes. Only after the training process can the test set be used to evaluate the newly trained model. From this perspective, the reporting of models within the Transkribus platform, which only consists of training and validation evaluation, needs to be treated with some caution, as it does not include a real test set.

A proper split should be made automatically and ideally be based on the line level instead of entire pages. Certain pages may be exceptional, containing only tables or a specific hand that only occurs on this page. Also, the layout analysis may fail on a given page due to scanning problems or other issues, resulting in very poor results that don't reflect the capabilities of a model. Designing test sets for general models is a particular challenge. They must comprise samples of handwriting that the model has never encountered during training. Otherwise, it's impossible to prove the recognition capabilities of a specific model for an entire writing style (Hodel et al. 2021).

Interpreting learning curves

An indication of a model's quality is the loss and CER curves of its training processes versus evaluation. Models that are overfitting (Figure 8) are easily identifiable by their very low CER on the training data but

a significantly higher CER on the validation data (Hodel 2020). At the end of the training process, the CER curve of the training and the validation set should ideally be parallel and as close to each other as possible. If the validation or the training curve still veers down, more epochs could result in more optimised models.

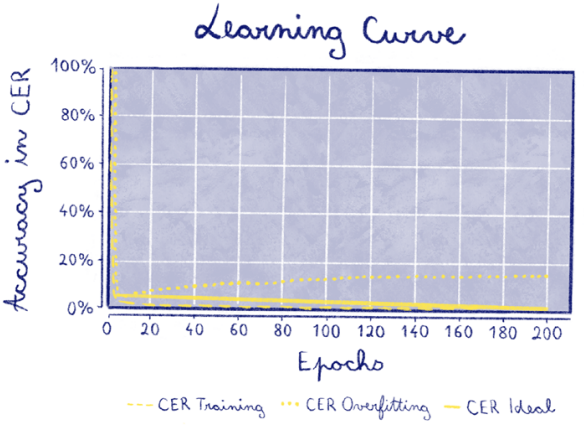


Figure 8: Example of overfitting, illustrated by Marie Machatová, CC BY NC 4.0

Character error rate

Since CER is still *the* most reported evaluation metric, we will briefly outline what it points to.

CER is a mathematical representation and quantification of the output calculated against a ground truth produced by humans that has been set aside in the form of a test set. CER is calculated at the character level, which means that even misrecognised punctuation marks or white spaces count as errors. Also, a CER can be above 100% since insertions, deletions, and displacements are counted. But let's keep in mind that this metric is only a very limited indication of a model's usefulness and that its "real-world" performance may vary.

The CER on the test set indicates at which error rates an ATR model will recognise similar writings. This is, of course, helpful to decide if a model is suitable for a given piece of writing. At the same time, it's only a limited indicator of a model's issues with the writing, for example, it's

unclear which characters are frequently not recognised or whether specific patterns are problematic (for a comparison of CER and real-world usage see 5.1 Case Study 1).

Quality benchmarks

As we just argued, CER is a means to measure the quality of a recognition process, even though it only hints at the potential of recognised text. Since CER is a standard measure, we will now give some pointers on how to work with this quantitative evaluation method. The following statements are valid only with regard to handwriting or mixed models (handwriting and print) and not for pure print recognition.

Generally, CERs above 15% are *hardly readable* for humans. Parts of words might make sense, but skimming texts is impossible and it takes an in-depth understanding of the content as well as palaeographic skills to read the handwriting and analyse it. *Any value above 15% should be considered unusable.*

CERs of 10–15% are generally *usable* but only to a very limited degree. Frequent and common words will be correctly identified. Unfortunately, those are most likely the words that could have easily been deciphered and transcribed manually, but it would still be faster to correct text identified at a 10–15% CER than transcribe it from scratch. Interestingly, given the large amount of human corrections necessary, most people still prefer to transcribe these texts from scratch.

At a CER of 8–10%, models achieve *decent* results. The text is readable, with errors in the parts containing irregular writing. Generally, the text is not reliable.

A CER of 5–8% is considered a *good* outcome. This can be achieved for regular scripts using general models. For most applications that do not require perfectly recognised proper names (a challenge for many models), this is a good target rate and can be achieved with training sets of less than one hundred pages most of the time.

Models based on large datasets (whether specialised or general) can realistically achieve a CER of 3–5%, which is considered a *very good* outcome. For specialised models, this usually (and depending on the

regularity of the writing) requires a hundred pages or more of training material or a very suitable base model. To reach this level, several iterations of training are required.

CERs below 3% can be achieved only in ideal cases, especially with a regular script. At this point, further training iterations will not lead to improvements, or only with enormous (and mostly unrealistic) amounts of additional training material. Most of the errors will concern uncertainties that are identified in the training material, such as uncertainties by human transcribers, alongside punctuation marks and problems with previous processing steps (such as layout recognition). When language models are used at this stage, even hyper-corrections must be considered. This means that the error is due to the language model, which normalises the spelling of a word.

As has been mentioned before, depending on the use case, a high or a low CER (within the range of 0–15%) is not the only measurement that should be taken into account. NER (as an example) can already yield usable results with a CER of 8% or higher on the produced transcription, as many taggers are capable of dealing with such outputs. Close reading, on the other hand, or classical corpus analysis will require a lower CER. We therefore advise not to take CER as the sole indicator of quality. Furthermore, qualitative comparisons are necessary to assess if the desired information can be extracted or identified.

Working with model outputs

When training models, we always need to analyse the output to gain an understanding of the model's capabilities and shortcomings. To determine precisely what characters a model does not identify correctly, we recommend CERberus (Haverals 2023), an open-source evaluation tool. The output of CERberus highlights which (perhaps crucial) characters are frequently misrecognised. Oftentimes, ciphers and uncommon characters like upper cases are not as important.

Regardless of the first run's performance, you should train at least two or three additional models with exactly the same training, validation, and test splits. Since models with no base models rely on random initialisation, the derived models may sometimes vary greatly, which can also be seen in 5.4 Case Study 4. But it's also helpful to train several base

models since the optimisation steps can lead to dubious developments within the net that are not recoverable. This is the likely cause when the CER curve shows spikes in the training process, and the resulting model is sub-optimal.

Tools such as CERBerus are employed for detailed error analysis beyond measuring pure CERs. Although CERBerus focuses on this metric, it also provides insights into the types of errors a model frequently commits, which is vital for the continuous improvement of text recognition models. Focusing on characters or specific digits, can yield improved insights.

When might it be useful to process recognised text with a CER above 15%? For fuzzy and wildcard searches (where characters like "*" or "?" can be used as place holders for one ("?") or many ("*") undefined characters), even a high CER can yield successful results; even more so if one can use approaches like smart search (provided by Transkribus for PyLaia outputs to search several different readings at once) or keyword spotting (as formerly implemented by HTR+). With both approaches, even with a subpar or rather bad CER, a high recall can be reached when searching for specific strings. Especially the smart search implementation is straightforward, as the model outputs not only the most likely recognition results but every recognition above a specified threshold. For instance, all recognition results with a likelihood above 40% are stored in PAGE XML and accordingly indexed and searchable.

How should you proceed when you are interested in proper names or numbers? Numbers and names are among the most difficult entities to recognize accurately. This is primarily because language follows relatively predictable linguistic patterns, but numerical sequences do not. Similarly, proper names are less bound by standard patterns, as naming practices are often creative, geographically mobile, and subject to individual taste. Fortunately, these challenges of recognizing suchg entities are mitigated by NER methods that are remarkably robust to mild ATR noise, allowing for the successful extraction of names and entities even from transcriptions that are not perfectly recognised.

4.6 Workflow

One of the most common applications of ATR is to use one of the available ATR platforms (3.2 Integrated Transcription Environments), which offer comprehensive functionality for seamless extraction of digital text from images right out of the box. However, although such platforms offer a very comfortable user experience, they make it easy to forget that ATR is not a monolithic tool but rather a complex and multi-layered process powered by a sophisticated internal workflow (Mühlberger et al. 2014; Stokes et al. 2021). Typically, this encompasses server-based microservices, which are small pieces of software specialised in one of the various tasks necessary to perform ATR. This may include importing documents, pre-processing, layout recognition, text recognition, and subsequent export into multiple formats, as well as training the necessary models. Behind the curtains of a GUI or the browser, these services are chained through RESTful APIs or comparable interfaces, which collectively execute complex operations, such as converting an uploaded historical document into an editable Word document.

Thanks to the substantial efforts of developers to streamline this complex process, users can take full advantage of ATR-related technology without the need to programme themselves or even understand the underlying architecture and methods of ATR. Instead, they can drag and drop an image into the browser and download a final text. Although this seamless experience creates accessibility to ATR technology for a broad user base, irrespective of their technical expertise, it also comes with some caveats: As the workflows used in these platforms are designed with mainstream use cases in mind, there is a considerable chance that the objectives embedded within the software are misaligned with your needs. This can be attributed to two reasons:

First, every project in the humanities is unique in its particular source material, guiding questions, and methodological approach. Even within the confines of scholarly editions – one of the more standardised productions in the humanities – each edition is somewhat distinct. This diversity is further enhanced within the digital sphere, where the many tools, techniques, and data formats available can add multiple layers of individuality to a project.

Second, ATR usually represents just one component of an extensive workflow for historical documents. This might involve important steps like (automatic) metadata curation, digital archiving strategies, and integrating interdisciplinary research methodologies such as advanced XML encoding or additional post-processing and analysis (see, for instance, Müller 2021a, or 5.3 Case Study 3). Producing text via ATR is rarely the final step but an intermediary process that contributes to a larger goal (Andrews 2023), which the standard outputs of ATR platforms may not fulfill, or even complicate.

To make the best use of ATR, it is therefore important to first reflect on a project's individual goals and needs. This must, in particular, include a solid understanding of the project's broader workflow, as ATR tools and the data they generate will have to be integrated and fit into the different steps. To make this clearer, let's explore four practical approaches:

1. Direct utilisation of built-in functionality: Many research projects, particularly in historical research, are primarily focused on the content of their sources. 4.3 Transcriptions facilitating this goal tend to concentrate on the 'substantives' of a document, as its 'accidentals' might only complicate the analysis (Greg 1950). This can be achieved easily with a model trained to extract normalised text that is output into a common file format suitable for direct and manual engagement, such as txt or docx, that can be post-processed and exported by all major ATR platforms. Such a direct application of built-in functionality meets the project's requirements without the need for further adjustments or complex workflows. This scenario usually applies to research projects that rely mostly on traditional methods but seek to benefit selectively from the efficiency of some digital tools such as ATR platforms.
2. Tailoring built-in functionality: If the intended output of a project is more digital, such as a digital edition of historical documents, producing txt and docx files may not be enough. Typically, such a project might seek to produce more complex and annotated data, such as TEI-XML files compliant with the project's proprietary annotation rules. Some platforms do offer such functionality beyond the limited scope of ATR, making it possible to annotate a document's transcriptions inside a GUI or browser window and export the final transcription into TEI. To do so, platforms leverage the transformability of XML documents through XSL scripting. In such a transformation,

the PAGE XML (4.1 Data) produced by ATR is converted into a coherent TEI file based on pre-defined rules in a XSLT file. In this process, annotations are also transformed from a so-called standoff format (Burghardt and Wolff 2009) where they are stored separately from the text with pointers or offsets indicating the specific parts of the text to which they refer, into XML elements according to the specified rules. To offer the user a built-in export functionality, platforms usually deploy a standard transformation file that covers the rules that they expect to be useful to most users. However, as many projects rely on specific TEI schematics that are more closely aligned with scholarly standards of their discipline, these rules might not apply to them. In this case, it can be helpful to contact developers, who might be open to deploy custom XSLT scripts that modify the ATR's output accordingly, particularly if it is potentially useful to a wider audience or funding is available. Alternatively, you can download a document's PAGE XML and transform the data yourself into any format you require, which leads us to the third scenario.

3. Advanced integration using APIs: As described earlier, platforms are based on microservices that communicate via APIs, which registered users can often also use directly. Thus, when a platform's pre-defined workflow does not suffice and simple adaptations are inadequate, these APIs can be leveraged directly to craft a workflow more suitable to your needs. Typically, such APIs are so-called RESTful APIs that rely on a set of protocols and standards for exchanging data between systems via HTTP requests to access, create, or manipulate resources (like data or services) on a server. Responses are typically returned in formats like JSON or XML for easy integration and processing. Often, there are means of integrating these APIs into a programmatical workflow through dedicated libraries (typically written in Python) such as Scriptorium Connector (Brown-deVost and Zandbank 2021) or TranskribusPyClient (Déjean et al. 2016). If you have sufficient programming skills, you can also quickly create your own functionality (Müller 2021b). This way, you can take the various steps of the ATR process out of the pre-defined workflow and insert your own processes. For instance, you might ask a platform to perform layout analysis but process the returned PAGE XML on your machine in a specific way before sending it back again for ATR. Similarly, you might recognise text in a more diplomatic way and perform normalisation using different technical means, later. Though this is more challenging and requires at least basic programming

skills, this level of workflow customisation can give you the power of established platforms without compromising too much on your specific needs, enabling you to perform complex pre- and post-processing on your data. However, be sure to verify the cost that might be associated with using a platform's API. Your workflows may also become vulnerable to unexpected changes made by the platform's developers.

4. Constructing a custom workflow: If your needs transcend the scope of existing ATR platform APIs, designing a custom workflow from the ground up may be the most viable solution. As these platforms themselves rely on microservices that have typically been developed in an open-source context, such as Pylaia (Puigcerver and Mocholí 2018) or Kraken (Kiessling 2015), it is possible to chain together the various parts of a process by yourself, or even programme your own machine learning applications using one of the established frameworks such as TensorFlow or PyTorch. This way, not only can you tailor your ATR environment exactly to your needs but also ensure maximal flexibility and independence. The scope of such a DIY approach can vary greatly in complexity, ranging from a simple CLI pipe, chaining together installed software into a functioning workflow, to programming your own elaborate GUI. However, depending on said complexity, this approach may require not only considerable personnel resources, but also dedicated hardware and server infrastructure, which entails additional administrative and maintenance costs. So unless you are a large institution with considerable resources or a skilled programmer yourself, this approach is most probably not practical.

As the optimal solution varies with each project's unique requirements, it is impossible to cast a final verdict on the best approach. Therefore, before embarking on the ATR journey, carefully reflect on your actual goals as well as the workflow that goes along with them. This is even more important for workflows of GLAM institutions. Such institutions usually depend on individually assembled or purchased workflow tools supporting ATR and other steps. Accordingly, there is no "best" solution for GLAM. Instead, we encourage you to look for best practices (Nockels and others 2022). Many distribution platforms (like Visual Library) might also allow the use of certain tools/platforms via API. For scholarly approaches, the key takeaways are: Ensure proper documentation and reproducibility (or at least know which pathways were

used). Ideally, elaborate your reasons and motivations for taking the approach you chose.

4.7 Use and Reuse

As the previous chapters have shown, creating training and evaluation material for text recognition, as well as for other deep learning-based applications, is tedious, time-consuming, requires expert knowledge, and is thus, in many regards, very expensive. Therefore, it makes sense to collaborate as much as possible on the creation of datasets that can be used and reused by a larger community for multiple purposes such as training or fine-tuning models for specific contexts. Such collaborative practices are very common in computer science, particularly in the creation of large, annotated datasets in the context of NLP, and is often referred to as "shared tasks" (Mühlberger et al. 2014), since they require the input of many volunteers, experts, and paid contributors.

In the humanities, the concept of shared tasks to provide datasets is less common. Here, projects often work only with their specific use case in mind, producing isolated solutions and datasets that cannot be used and improved easily by others, for instance due to restrictive licenses or particular design choices. However, since more and more deep learning applications are used on a daily basis in the humanities, requiring ever larger amounts of data, it is obvious that shared tasks will gain more traction. Thus, this subchapter briefly introduces the concept and the pitfalls of data reuse in deep learning. It outlines a low-resource approach to sharing data (transcriptions and images) and ends with deliberations about shared tasks with regard to text recognition.

Deep learning requires massive amounts of carefully annotated data to train capable models (2.2 A Bird's-Eye View on ATR). In the context of ATR, these data typically are composed of either annotated layout data (4.2 Layout) or prepared transcriptions (4.3 Transcription) that can be used for training or fine-tuning models. As the quality of such models typically improves when trained on large datasets, and ATR use cases often are very similar, it makes sense to reuse existing models or data as much as possible to reduce costs. There are two main options: You can either fine-tune an existing model using a smaller amount of data you have assembled or employ other datasets to train a new model.

By fine-tuning base models, you can quickly train (4.4 Training) new models for a specific writing style or other use cases. Because the weights of the base model have already been trained, the training process for fine-tuning is much faster, since you only have to adapt parts of the network rather than train a new model from scratch. Another advantage of fine-tuning is that it is much more energy-efficient than the full training of a neural network. However, in order to be effective, fine-tuning should be based on a model that somehow fits the data it is supposed to predict: Fine-tuning a model to a specific hand by the same scribe should work very well, whereas fine-tuning a model trained on modern handwriting will probably not work well on early medieval handwriting. But when possible, fine-tuning is the recommended approach for reasons of efficacy and to energy efficiency (4.8 Ethics).

Instead of fine-tuning an existing model, another approach is training a new model based on existing data. This can be advisable if your use case necessitates some changes in the way data are processed, for instance, if you are only using a specific set of characters. In this case, manipulating data – for instance by replacing or normalising characters – is much easier than trying to impose new rules to existing models. Reusing data can be as simple as recycling existing transcriptions in Word files that were initially intended for print editions buried deep down in your file system. But if you apply them on a larger scale, reuse can quickly raise complex issues, such as challenges in data compatibility, technical interoperability, copyright restrictions and accessibility. In fact, a lot can go wrong and prevent an effective use of the data.

Thus, we strongly recommend to always prepare even small datasets with reuse in mind, and to adhere to FAIR (Wilkinson et al. 2016) and CARE (Carroll et al. 2020) principles with a special focus on retrievability and accessibility. From a technical point of view, prepared datasets should always adhere to established ATR data formats, such as ALTO XML or PAGE XML, and include (or at least link to) images they are based on, in order to facilitate fast and easy import or aggregation into larger datasets. As for reusability, ensure legal compliance by choosing permissive licenses such as Creative Commons licenses (https://creativecommons.org/), especially CC0 or CC-BY. The latter raises awareness of the labour that went into creating the ground truth. If you use such data, make sure to give adequate credit, as you would when you cite a traditional publication.

Another critical factor in facilitating the reuse of models or datasets is comprehensive documentation of the decisions made during data preparation and model training. This includes, for example, transcription guidelines, annotation protocols, and the standards employed throughout the process. Ideally, such guidelines are based on established community standards. Only if these guidelines are made transparent can others decide if your data can be easily integrated into larger datasets, require normalisations using tools such as ChocoMuffin (Chagué and Clérice 2021), or cannot be used at all in a given context. And of course, the documented transcription guidelines need to be followed very strictly in order to not introduce errors into the model.

One last aspect is particularly relevant to the reuse of models whose training data cannot be easily inspected, but also essential for regular datasets: Data should always be accompanied by sufficient metadata such as period, language, scripts, used sources, as well as how balanced the dataset is.

Overall, making models and data truly reusable can be quite a significant, time consuming, and expensive effort. However, if properly prepared in the spirit of open science, such data – small or large sets alike – can have a huge impact on the development of future models and applications, as they enable the community to quickly and efficiently build and rebuild large and robust datasets that lead to high quality applications. In this light, the preparation of datasets should be treated as results of scholarly productivity and akin to publications. Similar to traditional articles, they require proper attribution of the contributors (see e.g. McGillivray and others 2022 and generally the concept of collections as data, Padilla 2017).

Prepared models and datasets can be shared in many ways, but a recommended approach to cluster available training material is provided by HTR-United (Chagué and Clérice 2020). Based on GitHub, HTR-United tries to collect datasets and metadata that can be reused for training or testing ATR models. Its unique framework utilises a versioned, low-resource methodology, emphasising the accessibility of diverse datasets. This approach is particularly notable for its inclusivity, as it does not prioritise specific transcription guidelines but rather integrates a wide array of handwriting resources. HTR-United provides comprehensive guidelines for licensing for various usage scenarios, including commercial reuse.

In any case, and independent of the development of the technology and frameworks such as HTR-United, if your data are prepared with reuse in mind, it will remain highly valuable or rather indispensable to train, evaluate, and test the future algorithms. The labour-intensive elaboration of the material is, therefore, also an investment in the future to support research relying on text recognition models. It is thus a shared task for scholars interested in recognising textual materials.

4.8 Ethics

It is highly misleading to think of ATR as just a technical solution to turn images into textual data: Models and workflows are not neutral, but strongly shaped by specific choices, assumptions, and power structures. Thus, far from being ethically inert, ATR entails a range of ethical concerns, including environmental costs, bias against marginalised languages and communities, and uneven access to the expensive infrastructures required for training and using ATR models. In this section, we explore these three areas further to help scholars and institutions reflect on ethical dimensions of ATR and develop responsible and sustainable practices.

As most ATR systems are based on deep neural networks, training and deploying them requires massive computing power and thus a lot of electrical power. The exact power requirements are difficult to determine and depend on several factors, including the type and size of the model as well as the technical infrastructure on which it has been trained. Training a transformer model for NLP, for instance, can use up to 650,000 kWh and emit as much CO2 as five cars in their lifetime, as Strubell et al. 2019 have calculated. That being said, not every deep learning model requires these massive amounts of power and ATR engines such as Pylaia and Kraken can be trained with fairly moderate resources. However, as platforms like Transkribus make training easy and quick, many users apply an iterative training approach (that we also advise from a technical point of view in 4.4. Training) and create several overlapping models rather than sharing pre-trained ones. Thus, although ATR models may not be as energy-intensive as LLMs, the cumulative energy use from training, retraining, and inference across digitisation projects can be substantial.

High energy consumption is a major driver of the climate crisis due to the high amount of CO_2 that is emitted by providing reliable sources of energy. The exact emissions depend on a number of factors and can vary significantly. Thus, carefully choosing providers of computing power can make a significant difference, as different companies rely on a different energy mix (Cook et al. 2017). As most users cannot influence the infrastructure of platforms or institutions, training your models locally might be an ecologically sound option if you have access to renewable energy.

Of course, the environmental impact of digital tools and methods is not exclusive to ATR, and by no means limited to CO_2 emissions (Baillot 2023). Thus, creating and adhering to sustainable practices in digital humanities requires a broad discussion across the community and funding institutions. Such a dialogue can be facilitated by initiatives such as the Digital Humanities Climate Coalition (https://sas-dhrh.github.io/dhcc-toolkit/), which also provides many resources. Regarding ATR specifically, however, it is important to reflect on the potential environmental footprint when choosing set-ups and workflows (4.6 Workflow). This can include considering sustainable practices such as sharing and reusing pre-trained models rather than re-training from scratch, using more energy-efficient algorithms and hardware, or running training jobs during periods when renewable energy is available (4.4 Training). Also, it can be good practice to disclose your consumption of fossil fuels and power if that information is available (Luccioni et al. 2025).

For historians, the archive is a place where they can listen to the sources of the past. However, it can also be a place of profound silence, particularly regarding minorities of race, gender or economic class (Olson 2024). This is not only true for the archival records themselves, which are predominantly created by writing elites, but also due to archival practices surrounding their preservation and retrieval. In this regard, ATR plays an ambiguous role: On the one hand, the technology promises to democratise access to historical texts or issues, even if they are not adequately catalogued, making it possible to break archival silence about minorities. On the other hand, such models often replicate and intensify cultural or archival biases that can lead to even more potent silencing (Nockels et al. 2024). As ATR models tend to converge around the dominant language and script in a dataset, they will most likely

perform best on traits of elite documents and poorly on low-resource languages and documents from marginalised communities.

Imagine, for instance, archival documents from a colonial context that are written in the colonialist's language but may include indigenous names or phrases: If the training set is not carefully balanced, ATR models will most probably misrecognise non-Western or low resource scribes or languages (Jaillant et al. 2025; Arnold 2022; Agarwal and Anastasopoulos 2024). This leads to an ambiguous role of ATR regarding the archival silence of marginalised groups: If the model performs well on their language and scribe, it can massively enhance the retrievability of hitherto untraceable documents; however, if it is biased against them, those documents remain effectively unreadable in the digital corpus, perpetuating or even worsening epistemic injustice where knowledge from minority groups is systematically excluded from the discourse due to structural and technological limitations (Melanson 2020).

These considerations highlight that ATR is not an ethically neutral tool but needs to be critically examined regarding its impact on minorities and marginalised groups and communities. This includes reflection on whether training data include potentially harmful biases, whether those can be reduced, and how they can be made explicit and transparent. For GLAM institutions, these questions extend even further: They need to ask what impact ATR models have on the accessibility and interpretability of their holdings, especially for traditionally marginalised voices. When they contribute training data to the creation of ATR models, they have to make sure that data have been ethically sourced and not perpetuate the very exclusions ATR seeks to address (Jaillant et al. 2025). Obviously, there are no easy answers to these questions. Yet it is important to recognise that seemingly technical decisions in ATR, such as data selection or training workflows, can have profound ethical implications that must be considered.

Finally, the infrastructural dimension of ATR must be considered from an ethical perspective, too. Training and deploying ATR models can require significant technical expertise, computational resources, and financial investment, which is unevenly distributed across institutions or regions, potentially creating a gap in research output. Although services (3 Tools) like Transkribus or LLMs are accessible to non-technical users, they require subscriptions that may not be affordable for users

from certain economic contexts. 'Free' services such as eScriptorium can turn out to be even more expensive, as they need to be run on a technical infrastructure (Chagué and Clérice 2023) that is often unavailable in many university departments, particularly in institutions or regions facing economic challenges. Moreover, using these systems effectively demands a high degree of digital literacy and technical knowledge. Institutions with in-house digital humanities labs, IT support, and funding can meet these challenges while others cannot. Thus, infrastructures should be developed as inclusively as possible, and funding bodies need to take this into account in their decision-making. Also, as the productivity of ATR continues to increase, the community should consider access inequalities when formulating their expectations of research practices and prospective projects.

To sum up, ATR is not an ethically neutral technology but an ambiguous tool with the potential to reshape the conditions under which scholarship is produced, for better or for worse. It can have a substantial environmental footprint, may embed strong bias that can marginalise certain languages and communities, and relies on infrastructures that are not available to everybody. As such, its responsible application requires ethical decision-making on many levels that needs to be carefully reflected regarding inclusion and sustainability (Luccioni et al. 2025).


CASE-STUDIES
1 0 0
0 1 0 1 1
1 1

5 Case Studies

Chapter summary: This chapter presents four case studies that demonstrate how ATR can be adapted to the challenges of complex historical materials. Each case engages with real-world problems such as mixed-language manuscripts, diverse scripts, complex layouts, and varying editorial practices. Together, the studies offer practical insights into how methodological decisions – ranging from annotation strategies to model design – directly shape recognition outcomes and scholarly reuse in research practice.

How this fits into the broader argument: Building on the conceptual groundwork laid in the previous chapter, these case studies illustrate how ATR is not a one-size-fits-all solution but requires tailored approaches depending on the research context. They show how methodological awareness, flexible workflows, and critical adaptation are essential when moving from theoretical understanding to practical application. In doing so, they reinforce the book's broader argument: that success in ATR depends as much on informed decision-making as on technical capability.

Following the conceptual introduction to ATR in the previous chapter, this chapter shifts focus from principles to practice. Rather than presenting universal solutions, the four case studies illustrate how ATR must be adapted to specific research aims, material conditions, and scholarly traditions.

Achim Rabus demonstrates how linguistic intelligence can be integrated into ATR models to enhance the readability and usability of recognition outputs. By adding word boundaries or enabling transliteration from Ottoman Arabic to Latin scripts, his approach shows how models can be tuned not only for character accuracy but also for the needs of human readers and further processing.

Tim Geelhaar proposes building a broad and robust model for the frequently encountered Carolingian script. His case study emphasises generalisability, illustrating how training on a wide and diverse dataset can yield a model that is resilient across manuscripts, scribal hands, and layout variations.

Michael Schonhardt takes a workflow-oriented approach, developing different models tailored to distinct stages of a scholarly edition. By doing so, he enables varied reuse scenarios, from strict, character-faithful transcriptions to editorially expanded versions of abbreviated texts. His strategy highlights the importance of planning for reuse and future-proofing ATR outputs beyond the immediate project needs.

Jan Odstrčilík addresses the critical role of transcription conventions and editorial traditions. His study shows how choices made during ground truth preparation – such as the handling of abbreviation marks or layout features – can either support or hinder ATR model performance. He demonstrates that seemingly small transcription decisions have major consequences for training outcomes.

Across these examples, common themes emerge. Careful annotation, thoughtful sampling of pages across a manuscript, and strategic decisions about layout and linguistic features are all shown to be essential for achieving reliable ATR results. Moreover, the case studies demonstrate that unconventional methods – such as selectively capitalising words in one language to aid recognition of code-switched passages – can sometimes yield better results than strictly traditional practices. Importantly, the case studies also expose typical pitfalls: how unbalanced bilingual data or inconsistencies in layout analysis can reduce recognition accuracy, and why an iterative, feedback-driven approach to modelling is crucial for success. They collectively show that ATR, especially for complex historical sources, requires flexibility, methodological reflection, and a willingness to experiment.

5.1 Achim Rabus: Pushing the Limits of ATR – 'Smart' Models with Extended Functionality

While the most dominant use of ATR models is a faithful diplomatic transcription (4.3 Transcription), they can be more sophisticated and more powerful, acquire a certain kind of linguistic intelligence, and be capable of working on different 'levels of transcription' according to Driscoll (4.3 Transcription). In the following, with a main focus on the platform Transkribus, we will discuss selected models with such 'smart' capabilities, that is, models able to conduct word separation in manuscripts written in *scriptura continua*, resolve abbreviations, insert

hyphens at line breaks to indicate word separation, transliterate to another script, reverse the reading directionality or modernise the orthography of a certain manuscript. Most of the use cases in this section are concerned with non-Western, i.e. non-Latin script systems that can pose specific challenges to ATR, such as the character set or the reading direction. This case study is intended to show the reader the current capabilities and limits of ATR models and their added value beyond diplomatic transcription.

Models That Resolve *Scriptura Continua*

In many medieval writing traditions, no blank spaces between individual words are inserted. Instead, the text is written in so-called *scriptura continua*. In a typical manual editorial workflow (4.6 Workflow), human editors would insert blank spaces to indicate the appropriate word separations and to make the text more legible. This very basic normalisation and modernisation step is applied even when no other normalisation steps regarding abbreviations, spelling variations, ligatures or superscript characters are taken. This is why it makes sense even for otherwise philologically faithful ATR models to incorporate the smart feature of inserting blank spaces. An example for such a model is the generic model for Church Slavonic (pre-modern Slavic) available both for the platform Transkribus and for Kraken/eScriptorium (Rabus 2019; Rabus and Thompson 2023; Rabus et al. 2023). The model is capable of transcribing both Cyrillic uncial and semi-uncial script from the 11th to the 16th century. On the Transkribus platform, the model has a CER of 3.71% and was trained on 393,079 word tokens (Rabus 2020). An example of the capabilities of this model can be seen in Figure 9.

As in many other languages, there are homonymous prepositions and prefixes in pre-modern Slavic, often causing both human and AI transcribers to miss or place an excessive amount of blank spaces. The real-world application of such models shows that, indeed, such errors do occur (Rabus 2024b), and it would make sense to eliminate them in a post-processing step. However, interestingly, training a model completely without blank spaces on the same training data do not result in a lower CER in this particular case. One has to keep in mind, though, that the CER is not always a good indicator for a model's real-world performance (4.5 Evaluation). Oftentimes, a specialised model with a low CER on paper and a low amount of training data tokens performs

worse in real-world scenarios as compared to larger, generic models with a CER that seems unattractively high at first sight. This discrepancy arises because large ATR models have acquired a significant amount of linguistic intelligence during training, which they leverage during the transcription process.

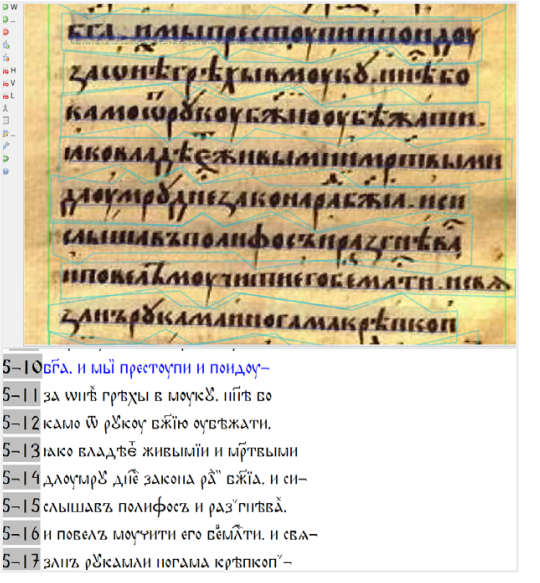


Figure 9: Public model for pre-modern Slavic. Smart capability: resolution of *scriptura continua*, image by the author, CC BY 4.0

Models That Can be Used for Formatting Purposes

As discussed above, ATR tools are usually part of a workflow that includes post-processing the transcribed text. Some of this post-processing entails (semantic or structural/visual) markup. There are many exciting possibilities to train tags for information extraction (such as NER) that are discussed elsewhere. Another possibility is to train smart models that facilitate post-processing in other digital formats such as TEI-XML (4.1 Data). For example, when retro-digitising a printed publication with ATR for reuse in a digital edition, it is possible to train smart models in a creative way, using specific Unicode values to mark certain tokens in the text as cursive or bold, for example. After transcription with an ATR model, this dummy markup can easily be converted – using

scripts or simple search-replace operations – with the appropriate XML tags. Bastianello and Baumgartner 2023 report on such a creative use of smart ATR models to efficiently convert a printed book to a digital edition, retaining its original formatting.

Models for Targeting Different User Groups

Apart from adding blank spaces, the Church Slavonic model presented above is a faithful, philological model. It conducts no additional normalisation or simplification steps, potentially restricting the target audience to expert philologists. In order to introduce pre-modern Slavic sources to a wider audience, we trained two different models using the same training data, but different transcription principles. (Rabus and Meindl 2025) One of the models is philologically faithful, while the other one has been trained to conduct several normalisation and simplification steps. In particular, this smart model replaces certain historical letters with their modern counterpart, e.g. Cyrillic <ѧ> with <я>, or <ѡ> with <о>. Moreover, as can be seen in Figure 10, first transcription line, it expands some of the most common abbreviations (e.g. **мєсяца** 'month' instead of **м̑ ͡ца**), replaces Slavic numbers with Arabic ones, and normalises punctuation, whereas the second transcription line shows the result of a philologically faithful model.

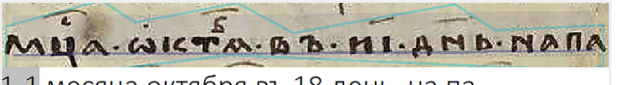
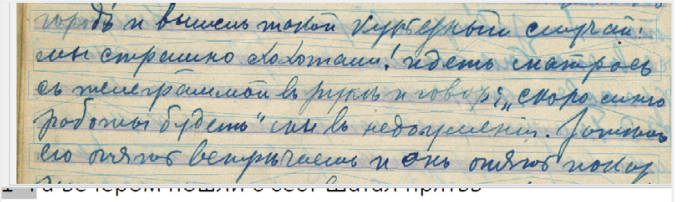


Figure 10: Example of two models for different target audiences: Smart capability of the first model (line 1): Modernise letters, expand abbreviations, change the Slavic numbers to Arabic ones. Second model (line 2): philologically faithful, image by the author, CC BY 4.0

A similar approach has been chosen for our public generic model for Russian handwriting. (Rabus 2022c; Tikhonov et al. 2022) This model automatically replaces several letters that are obsolete in modern Russian orthography but still present in 19th- and early 20th-century handwriting, with their modern counterparts. Moreover, it omits the redundant letter <ъ> at the end of certain words.



1-5 город и вышел такой курьезный случай
1-6 мыстрашно хохотали; идет матрос
1-7 с телеграммой в руке и говоря скоросиню
1-8 работы будет ми в недомении. Затем
1-9 его опять встречаем и он опять показ

Figure 11: Public model for Russian. Smart capability: Replacement and omission of obsolete characters, image by the author, CC BY 4.0

In the example shown in Figure 11, we see <ъ> frequently omitted. Models that have acquired the smart feature of orthographic modernisation greatly facilitate access to the output for a non-expert audience that untrained in reading pre-revolution Russian orthography.

Transliterating Models

As we've discussed extensively (2 Understanding Automated Text Recognition), ATR models learn correspondences between optical cues in the digital manuscript image and Unicode codepoints corresponding to computer-readable letters in the digital transcription. It is important to note that there is no need for the visual form of the letters in the digital manuscript and the corresponding digital transcription to be similar in shape. As a matter of fact, they can belong to different script systems, opening up the possibility of training and applying models with the capability of transliterating from one script to another. Such models may be used in several scenarios. The most obvious use case is in situations where the manual editorial tradition is one of transliteration, for example when the original handwriting in the manuscript is no longer commonly used. Other use cases apply in situations where the expert tradition is to retain the original script system of the manuscript in the editorial process, but reading the transcription requires expert knowledge, and a large portion of the potentially interested academic community lacks the necessary expert knowledge to read it. In such

cases, transliterating models can help wider audiences gain access to the sources, thus contributing to the democratisation of knowledge (Rabus 2024a). One public model belonging to the first category – the editorial tradition of transliteration – is the public generic Glagolitic handwriting model trained on 171,082 word tokens (Rabus 2022a; 2022b). Glagolitic is a Slavic script used, among others, in medieval Croatia. Its angular form features many ligatures. In Croatian philology, Glagolitic sources are routinely transliterated to Latin script, ligatures are resolved, and abbreviations are expanded and marked with parentheses. During the training (4.4 Training) of the model, we followed the recycling approach, i.e. we reused transcriptions originally intended for preparing printed editions that are available as Word files, created by different specialists in Glagolitic handwriting. These transcriptions followed the principles outlined (Latin script, resolution of ligatures, expansion of abbreviations), which is why the model acquired several smart features, above all, transliteration to the Latin script. An example of the model performance can be seen in Figure 12.

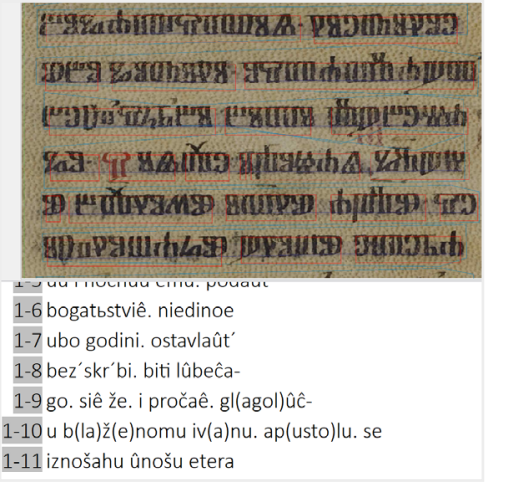


Figure 12: Sample of a transliterating model for handwritten Glagolitic. Smart capability: transliteration to the Latin script, resolution of ligatures, expansion of abbreviations, image by the author, CC BY 4.0

While the original manuscript is written in the Glagolitic script, the ATR model produces output in the Latin script, thus making the source more accessible to non-expert speakers of Slavic languages. Besides, as

can be seen, the model is capable of expanding ligatures and marking them with parentheses, even if numerous letters need to be added (line 1–9) or there are several abbreviations within one word (line 1–10).

While being remarkably smart – and, by the way, not restricted to the Transkribus platform (Rabus and Thompson 2023) – the ATR model for Glagolitic does not feature an extremely low error rate: The validation set CER of the public model for handwritten Glagolitic, a model with the capability to expand abbreviations, is 5.6%. Re-training the model without the smart capability to expand abbreviations yielded a considerably lower error rate of 3.8%, i.e. by removing the smart capability of expanding abbreviations, the CER decreased by one third. It is apparent that smart capabilities often come with the trade-off of a higher CER. Obviously, the smart features are more error-prone than simple character-for-character transliteration – a result that can also be observed with human editors.

Reversing Directionality

While Western writing systems such as the Latin, Cyrillic, or Greek scripts, are written from left to right (LTR), other writing systems such as Arabic or Hebrew are written from right to left (RTL). When transliterating text written in such writing systems to another, e.g. the Latin script, the directionality of the transliteration is different than the directionality of the original writing system in the manuscript. With respect to ATR, this means that a directionality feature has to be implemented into the model/the transcription workflow. On the Transkribus platform, the reverse directionality feature is activated by checking the appropriate box (Figure 13). Crucially, it is possible to leave Arabic numbers in left-to-right direction – at least in the Transkribus Expert Client. This feature is helpful for numerous real-world cases where an RTL language has to be converted to an LTR script while retaining the original LTR directionality of the numbers.

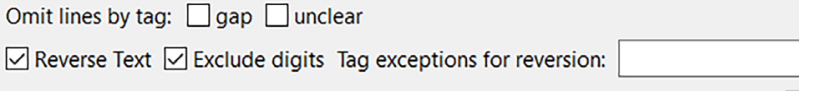


Figure 13: 'Reverse Text' checkbox in the Transkribus Expert Client, image by the author, CC BY 4.0

Use cases where the directionality feature comes in handy are models for transliterating Ottoman Turkish. Ottoman Turkish, the linguistic predecessor of modern Turkish, is written in the Perso-Arabic RTL script. In Ottoman philology, it is common to transliterate the transcription to the Latin alphabet using different transcription standards, some more philological, some more akin to modern Turkish.

In addition to the directionality issue, there is one more crucial feature that requires smart transliterating ATR models for Ottoman manuscripts, namely the way vowels are written: Sometimes, they are marked with diacritics, but oftentimes, they are not marked at all. The model needs to learn how to infer them during training. It goes without saying that the smart capability of guessing the right vowels to add to the transcription, in combination with different handwriting styles and transcription standards, leads to a comparatively high CER. An example of such a model (Matić-Chalkitis 2023) be seen in Figure 14.

Figure 14: Example of a model for Ottoman Turkish. Smart capabilities: Transliteration, reversion of writing directionality, addition of vowels (depending on sources), image by the author, CC BY 4.0

The model used here has been trained on 201,096 word tokens and has a comparatively high error rate of 11.6%.

Another use case where the directionality feature can be beneficial is Yiddish. While written in the Hebrew script, its linguistic core has many similarities with Germanic languages. Persons interested in Jewish history and culture, proficient in Germanic languages but unfamiliar with the Hebrew alphabet, will be able to access document contents if they are transcribed into the Latin script. This involves reversing the reading direction from RTL to LTR. This use case is a prime example of using AI in general and ATR in particular to reduce access barriers and to contribute to the democratisation of knowledge. An example of such a model for Yiddish can be seen in Figure 15.

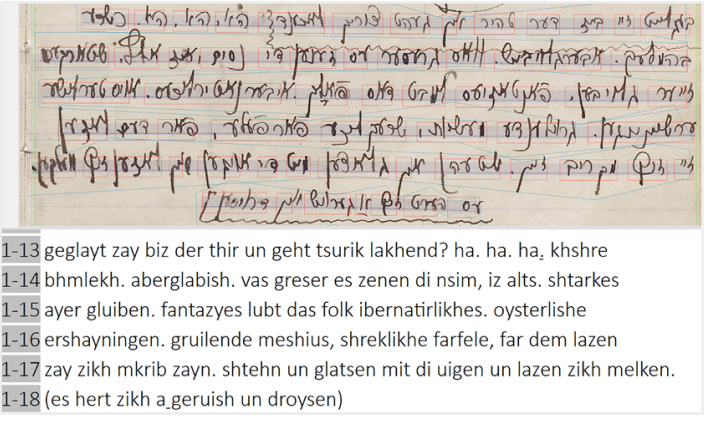


Figure 15: Transliterating model for Yiddish. Smart capabilities: Transliteration, reversion of writing directionality, image by the author, CC BY 4.0

This model is predominantly based on the public Dybbuk model (DYBBUK project 2022); the transcriptions of this model were converted to the Latin script. It was trained on 135,982 word tokens and has a CER of 5.0%.

Shorthand

The effort to develop smart models for writing systems that only a small minority can read today culminates in creating models for shorthand or stenography systems. Since shorthand does not have a one-to-one relation of one handwritten character to another one in the transcription and is heavily abbreviated, it is much harder for machines – and humans – to learn its respective features and to achieve a satisfying error rate. As

a rule, such models require considerably more training data. Since, in such complex cases, there is a clear misrelation between the available (or easily producible) and the necessary amount of ground truth, we created synthetic ground truth for Deutsche Einheitskurzschrift (DEK), using, among others, the so-called Stenogenerator by Jens Christian Wawrczeck (https://www.jens-wawrczeck.de/stenogenerator/index.htm). We then combined the synthetic ground truth with available handwritten data and trained a model for DEK with 144,709 word tokens and a CER of 9.50% (Rabus et al. 2022).

As can be seen in Figure 16, while the transcription results are far from flawless, the model enables speakers of German without any knowledge of Deutsche Einheitskurzschrift to gain access to documents written in this type of German shorthand.

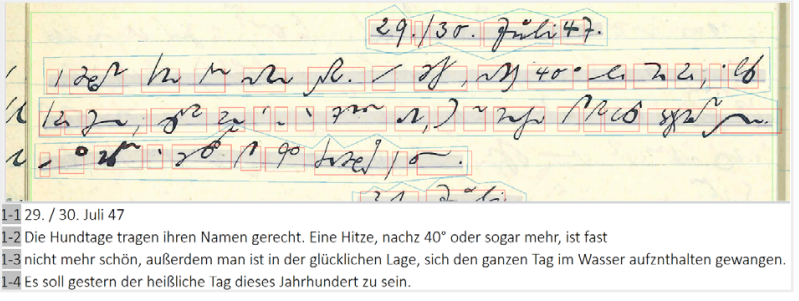


Figure 16: Example of a model for shorthand (Deutsche Einheitskurzschrift). Smart features: dealing with heavily compressed writing and many abbreviations, image by the author, CC BY 4.0

Models for other types of shorthand, such as Gabelsberger, are also being trained. At the time of writing, their CER is around 15%, making parts of the output hard to comprehend. Recycling already existing Gabelsberger transcriptions for training purposes might eventually result in a lower CER.

Conclusion

This overview has shown that ATR models can acquire a wide range of smart capabilities beyond basic character-to-character transcription. By virtue of their diverse capabilities such as modernising obsolete letters, expanding abbreviations, transliterating scripts into another

alphabet, and deciphering writing systems known only to a small group of experts, smart ATR models can help make sources more accessible to a wide audience of users, contribute to reducing hegemonic power, and spread the democratisation of knowledge. These smart functions usually come with the trade-off of a higher CER as compared to philologically faithful models.

Smart models have acquired additional functionalities to conduct tasks that are normally done by other, non-ATR tools in a digital pipeline, such as resolving abbreviations or formatting. By rendering such tools obsolete, smart ATR models simplify the overall digitisation workflow, albeit sometimes compromising control or error rates. They follow a trend towards multi-purpose tools in recent AI development. Large language models (LLMs) such as OpenAI's ChatGPT, arguably the most famous and impactful multi-purpose AI tools, continue to have a significant impact on humanities research in the digital age, steadily replacing certain legacy NLP tools. For instance, they already achieve remarkable accuracy for tasks such as NER (Rastinger 2024). In view of this, it will be interesting to observe how ATR and LLMs develop and interact in the future. For instance, LLMs can take some load from smart ATR models regarding tagging or the expansion of abbreviations. Multi-modal LLMs are even capable of transcribing – mostly modern and Western – handwriting. For historical and low-resource handwriting, however, particularly with respect to non-Latin writing systems and non-standardised orthographies, it is fair to assume that more specialised ATR tools that provide both smart and philologically faithful models will remain relevant for the foreseeable future.

5.2 Tim Geelhaar: A General ATR Model for Multiple Purposes – The *Caroline Minuscule Model*

As demonstrated in the preceding case study, ATR models are highly versatile. This chapter will provide a more detailed examination of a particular use case, demonstrating how a model can be adapted to align with a specific objective, and the potential benefits and challenges involved. The model under consideration is the Caroline Minuscule Model (CMM) (Geelhaar 2023), which has been developed to automatically transcribe texts written in the Caroline Minuscule (Figure 17).

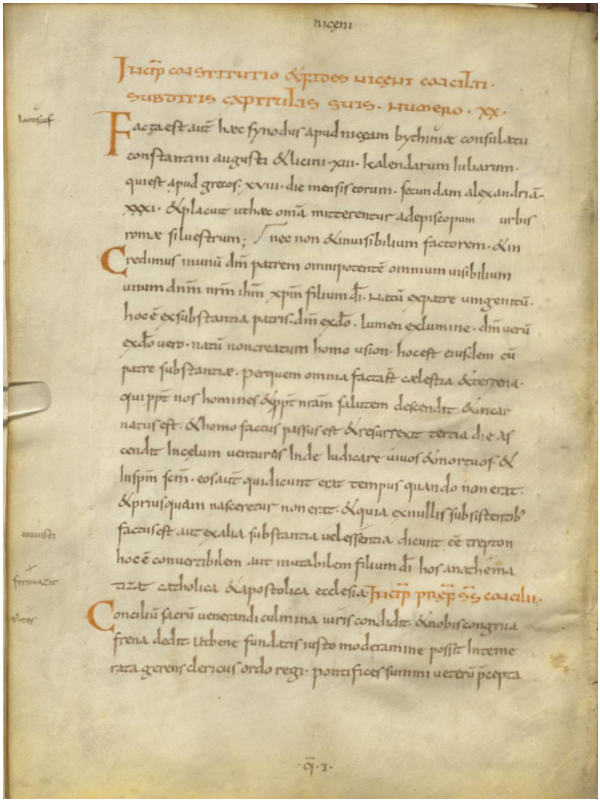


Figure 17: Frankfurt UB Ms. Barth. 64 7v, Collectio Dionysio-Hadriana, PDM 1.0

This script emerged in the second half of the 8th century during the reign of the Carolingians. It differed from its predecessors in terms of clarity, uniformity, and regularity. It was written in a predominantly lower-case style, which subsequently led to its designation. Capital letters, or majuscules, continued to be written in semi-uncial, an earlier writing system. The script was used long and widely after the end of the Carolingian era because of its simplicity in both writing and reading. It therefore dominated Latin European scripts until the introduction of the Gothic scripts during the High Middle Ages. In subsequent centuries, it served as the foundation for the Renaissance script known as Antiqua and later, for the Times New Roman typeface (Bischoff 1986; Derolez 2006; Ganz 2020).

The central goal of the CMM is to pave a way from edition-based collections towards corpora of manuscripts for digital historical semantics. This paradigmatic shift is necessary to fully grasp the mutual dependencies between a society and the language it uses (Geelhaar et al. 2023; Jussen and Rohmann 2015). By keeping the lexical diversity of these manuscripts, semantic observations will be more significant and allow a differentiated view of language and semantic representations used in the sources. Differences in spelling can be highly instructive, for instance, for legal historians and Latin philologists. Transitions from Latin to Romance languages have already been analysed in the context of digital historical semantics (Frank-Job and Henrichfreise 2015). However, this research has only been possible at the level of edited sources. Many linguistic variations could not be considered because they had already fallen prey to emendation. This makes it even more important to turn to the handwritten word inventory.

The CMM is only one piece of the puzzle. While there are strong models for the scripts of later centuries, such as the Gothic model discussed below, there is a gap for the early Middle Ages. The apparent lack of interest in a stable and reliable model may be because individual manuscripts are easy to read once you understand typical features such as the system of abbreviations and the peculiarities of interpunctuation. Moreover, many texts written or copied in the early Middle Ages are accessible through editions such as those of the Monumenta Germaniae Historica (MGH), so there is no apparent need to revisit the manuscripts. However, this also has some major drawbacks. We rely on the editorial work of earlier generations, with all their biases and shortcomings, and on the canon of edited works, while other manuscripts remain in their shadow.

This is where ATR comes into play, allowing the transcription of manuscripts at scale. But the models so far have rather been proofs of concept than actual productive tools. Tobias Hodel trained a model on Transkribus in 2019 using only St. Gall SB Cod. 914, pages 1–60 (5,531 words) and the algorithm CITLab HTR+ with a CER of 8,4%. Nick White and others created another model trained with Ocropus for Rescribe (https://rescribe.xyz/) in 2022 using snippets from 70 manuscripts from the Munich State Library (17,155 words). To overcome these shortcomings and to meet its goal, a newly trained model must be capable of transcribing manuscripts from many centuries and many different scribal

hands precisely and consistently. A general and robust ATR model is required to generate automated transcriptions that can be used in text mining processes with minimal post-correction.

The CMM is the first version of such a general and robust model. It meets several expectations. First, it performs well on a variety of layout configurations (one, two, or more columns) and even on text in the margins. Second, the model produces reliable transcripts regardless of the state of preservation. Third, the model can handle a high degree of diversity in writing itself, including personal as well as regional differences. Fourth, it resolves abbreviations to create an immediately readable and processable text. In this respect, the CMM differs from the next case study in this section, the Burchard Decretum Digital model (5.3 Case Study 3), that keeps the abbreviations in the first reading to resolve them later in the editorial process. The overall accuracy of the produced results attains 95%, which is the benchmark for robust ATR models (Pinche 2023).

For training, we used the integrated transcription environment *Transkribus*. It offers all necessary features without any need for programming skills. The CMM was trained from autumn 2021 to April 2023. It started as a personal project with a limited number of credits (the currency used to pay for *Transkribus* services). The REAC COOP SCE that developed *Transkribus* kindly supported the further training with additional credits when it became part of a university course in summer 2022 at the University of Bielefeld and part of a winter school on ATR for historical transcription at the Academy of Sciences in Vienna in 2022. The first versions had been trained with the text recognition algorithm "CITLab HTR+" that was no longer available as of November 2022. Subsequent versions had to be trained with the algorithm Pylaia (Puigcerver and Mocholí 2018) for text and layout recognition with default training parameters that will be explained below. The latest version of the CMM, the so-called "Carolingian Minuscule Model CMM 9th-11th c." (Geelhaar 2023) is published on the *Transkribus* Platform. It achieves a CER of 5,2% which can be regarded as a sufficient initial result. To understand the quality of transcription that the model currently offers and to find ways how to improve the model, it is necessary to describe the underlying principles and the training process.

The following chapters therefore address the selection of manuscripts, the handling of the layout, the transcription guidelines and the training process, including testing. The conclusion presents the results and further considerations.

Manuscript Selection

The creation of a robust, general ATR model requires a selection of manuscripts as a material basis, covering many different manuscripts from various backgrounds. A variety of scribal hands, schools, regions, time periods must be considered but also genres that have an impact on the quality of writing. Religious texts were written far more diligently and accurately than any other genre. The selected manuscripts cover the bible, sermons, exegetical and penitentiary works, canon and secular law as well as capitularies, hagiography and history, mathematical and philosophical works. This choice was also motivated by a desire to cover a wide range of vocabulary and consequently include abbreviations in the training.

All in all, samples of 10 to 20 pages from 46 manuscripts with 23,380 lines and 179,520 words were used for the training. The manuscripts are listed in the accompanying publication on Zenodo (https://doi.org/10.5281/zenodo.16613534). They appear there in the same order as in the corresponding *Transkribus* collection. During several training iterations, the list grew and its order changed. The description of each manuscript is deliberately kept very brief, as complete descriptions can be found in the manuscript catalogues of the holding institutions. The list contains the holding institution, the shelf mark, the title, the selected page range, the place and date of composition, the genre, and remarks on layout and script.

The length of each sample depends on the layout of each manuscript. Some feature more than 40 narrowly spaced lines per page, others contain only 18 lines in a very wide script. The idea behind working with samples was to avoid too much input from one or only a few sources that could bias the predictability of the ATR model. Since the vocabulary and consequently the abbreviations become increasingly repetitive after a while, it seemed sufficient to keep the samples to the chosen length. This is also why model training is different from transcribing manuscripts for editorial purposes.

While at first glance, 46 manuscripts appear rather few compared to the wealth of manuscript preservation, the sample is of usual size compared to other ATR models in *Transkribus*. An important benefit of artificial intelligence is its ability to achieve high-quality recognition with minimal training data, as demonstrated by initial test runs. However, it is necessary to capture different writing styles to enable the model to recognise untrained writing styles. Beyond a certain point, additional transcribed manuscripts and thus training material yielded only a slow increase in quality. At a CER beyond 5%, more training material only yields minor improvements. The excellent model "Gothic Book Scripts XIII–XV M4" by Tobias Hodel (Hodel 2022b) with its 4,1% CER only performs 1,1% better than the CMM, although the training corpus contains five times more material (163,157 lines or 902,344 words versus 23,380 lines and 179,520 lines). But these are only observations based on automated evaluation. The significance of these statistical analyses for assessing model quality will be discussed later.

The selected manuscripts were written by many different scribal hands from Bavaria to Western France, from Italy to England from around 800 until 1100 CE. It is important to cover a variety of locations to build a general model. This also means that the training was not meant to be specific for one scriptorium, such as St. Gall. More fine-grained models for specific production locations are possible based on the CMM, which may also help identify the origin of certain manuscripts. But again, this is not the aim of this training. For the same reason, the identity of the scribe or the prominence of the manuscript were also irrelevant. On the contrary, such choices can unnecessarily bias the training process.

Another criterion for selecting these manuscripts was the reusability of existing editions and transcriptions from other projects. This was particularly helpful to assure the highest possible confidence for the transcriptions. Some of these external transcriptions are available under a Creative Common Licence, as in the case for the Capitularia project (https://capitularia.uni-koeln.de/). Other existing transcriptions were reused, like those for Flavius Josephus's Antiquities by Richard M. Pollard or those in the project Burchards Dekret Digital (https://www.burchards-dekret-digital.de) and the project on Gregory of Tours' historiae by Helmut Reimitz (https://www.oeaw.ac.at/projects/histories-in-transition/home/histories-of-a-history-gregory). These transcriptions, however, had to be aligned with the transcription guidelines for the model

training. The transcriptions of the Capitularia project influenced the guidelines regarding interpunctuation. Other transcriptions, like those from the Gregory of Tours project needed to be simplified as they contain philological annotations that were not to be trained in the CMM.

An equally significant aspect in the creation of the training corpus was the availability of images. Most libraries like the Munich State Library, Heidelberg University Library, or the Bamberg State Library offer their digitisations under a Creative Common License if the material is not in the public domain. The same holds true for the e-codices.ch and the Bibliothèque nationale de France that makes its digitisations available under their own copyright. Other libraries like the episcopal libraries of Trier and Cologne kindly granted the author use of pages from their manuscripts for training purposes. The images used for model training follow the FAIR principles and are also available for reuse on Zenodo (https://doi.org/10.5281/zenodo.16613534).

Transcription Guidelines

Transcription guidelines are essential for the scientific analysis of spoken or written text, and this is particularly the case when training ATR models (4.4 Training). On the one hand, fixed rules must guarantee consistency and transparency; on the other hand, they facilitate the collaborative creation of ground truth. Transcription guidelines are based on the defined purpose of the transcription, i.e. to create transcriptions that can be easily post-processed for text mining purposes. The automated transcriptions for the CMM should be accurate enough to only require very few refinements before the outcome can be analysed with text mining methods. This can be best achieved by a model training on an intermediary transcription level. In contrast to a graphetic transcription, not every feature of the font's appearance needs to be encoded, as in hyper-diplomatic editions that consider every single stroke, deformed character, and other peculiarities. It is also not necessary to reproduce all non-standard characters as a corresponding digital representation, as the School of Salamanca Edition Project (https://www.salamanca.school/) does. The text must be transcribed true to the character so as to keep the mistakes. This helps train the algorithm not to subsequently interpret the scriptural evidence. This can be considered a philological approach, rather than a normalising approach where mistakes are emendated. However, the intermediary and the normalised level share

one important feature: Both resolve abbreviations. Luckily, the usage of abbreviations in the Early Middle Ages is consistent, which could be confirmed by manual evaluation of automatically transcribed pages. Spelling normalisation is also not needed as current pre-processing tools are able to lemmatise orthographical mistakes thanks to specific Latin lexica like the Frankfurt Latin Lexicon (Mehler et al. 2020) and taggers like LatinCy by Patrick Burns (Burns 2023). Moreover, normalisation could be detrimental since mistakes are also important indicators for textual understanding.

There are some best practices for transcription guidelines that can be used for orientation, such as the Capitularia Project or the guidelines from the School of Salamanca. But every transcription project and every model training need to adapt existing practices to the needs and demands of the source material. The process of setting up transcription guidelines in the early stages of a project can be divided into three phases. First, the basic principles must be established in accordance with the model's purpose. In the second phase, these principles must be concretised and adapted to the material. Not everything can or should be done as originally planned. The third phase is about fixing the guidelines and to avoid any further changes, if possible, as every change would require revising the entire ground truth. This is also how the guidelines for the CMM evolved. They concern three different aspects that are particularly important for the model training: the layout, the characters, and special cases.

Since the full transcription guidelines are published on Zenodo (https://doi.org/10.5281/zenodo.16613534), it should suffice to outline the rules here. Contrary to the BDD (5.3 Case Study 3), the text structure is excluded from the training and thus from the transcription guidelines. Furthermore, only the main text region should be transcribed. Additional text regions are to be considered only if they are written in Caroline Minuscule, as well. Initials should be excluded if their height exceeds three lines. Including initials beyond this size complicates their alignment to the correct baseline. Unclear sections should be omitted and indicated accordingly. The manuscript must match the character exactly, without correcting mistakes in Latin orthography. This prevents confusion in graphical representation and character matching. For the same reason, proper names are to be transcribed faithfully to the manuscript. Characters should remain in lower case when the

manuscript indicates a minuscule. Abbreviations must be expanded and interpunctuation preserved to the greatest extent possible. Interpunctuation presents a significant challenge due to its lack of standardisation, making it difficult to train accurately. The limited number of special characters is due to the intermediate transcription level. Training corrections in the manuscript is challenging due to their infrequency and the various practices for marking errors. Consequently, corrections have been included in the training material only when they are clearly recognisable. Otherwise, the model will be at a greater risk not to distinguish between normal characters and errors, potentially affecting the accuracy of the model.

Training

The model training was conducted with the help of the Transkribus Expert Client (Versions 1.12 to 1.26) with the standard parameters – a learning rate of 0.0003, a batch size of 24, a maximum of 200 and a minimum of 20 epochs – as there was no need for adjustment, like in the case of the late medieval English chancery writing, the so-called Anglicana. Here, it proved helpful to lower the learning rate from 0.0003 to 0.0001 to achieve better recognition results.

The first trained model was based on twelve manually transcribed manuscripts. The manuscript images were imported from digital libraries like gallica.fr or e-codices.ch through the IIIF-Import function or by uploading PDFs. Next, Transkribus recognised the layout of the uploaded manuscripts and prepared a setting for the manual transcription by numbering all lines in the manuscripts. At first, the library HTR+ was used but replaced with Pylaia later. Either way, the layout recognition was of good quality but far from perfect. One of the first lessons learned was that the layout recognition always needs to be checked and manually adjusted. This effort is necessary as the layout recognition contributes considerably to the later transcription quality. Luckily, most documents in Caroline minuscule are not too complicated so manual corrections do not take too much time. A typical case was the correct alignment of the ends of paragraphs. Sometimes initials had to be included and additional letters were put into the space behind the last word of a paragraph when there was not enough space for them in the next line.

Existing text editions could not be entered directly into Transkribus although there had been experiments to align computer-written text with the automated recognised text. Instead, editions and other transcriptions were used as a reading aid to ensure the correct reading of a text. Since most critical editions are based on the reading of several witnesses, the actual transcribed text will always differ from these edited versions. But even in cases where the text had been transcribed straight from the existing edition, it was useful to re-read the transcription later to correct errors and adjust in relation to the other transcriptions. After this check, the page in question was set as ground truth, which made it easier to collect all the ground truth pages for model training.

A particularly compelling feature of Transkribus is the option to use other models as base models for your own model trainings. This means that training can build on pre-existing algorithms from other models. Thus, machine learning becomes much easier and quicker as many more technical features can be reused. This allows for iterative and incremental training by using one model version as a base model for the next. This process leads to more and more fine-grained and more precise ATR models. For the CMM, nine models were trained this way, building on each other with additional manuscripts each time. The first model training was based on the rudimentary model by Tobias Hodel. Even the first result was already promising, and from the fourth version onwards, the model seemed to have a certain degree a stability and precision. This model was based on only 32,266 words resp. 4,123 lines from 150 pages. It had a CER on the validation of 2,6% but seemed to have overfitting from the 25th epoch (or iteration in the learning process) onwards. Although subsequent models did not achieve the low CERs of earlier versions, they no longer exhibited significant overfitting. But this is also related to the composition of the training and the validation set. It contained only seven pages for the CMM 4, which was not enough for effective validation. This leads to another aspect of how to assess a training model.

Evaluation

There are multiple methods for evaluating (4.5 Evaluation) the accuracy of a trained model. Transkribus employs an approach to model training whereby the material is initially divided into a training set and a validation set. The user has the option to either create the validation

set manually or to have the computer generate it with 2 to 10% of the training material. The training set for the published CMM contains 620 pages with 179,520 words and 23,380 lines. Another 62 pages or 10% of the complete material were used as validation set. The Transkribus Expert Client provides the results from the training in Figure 18. The y-axis shows the recognition error percentage, while the x-axis represents the number of iterations or epochs. Each iteration is a complete recognition process of the material. With each iteration, the number of correctly identified characters increases and errors decrease as the algorithm learns from the training material.

The CMM training stopped early after 96 iterations. Although it could run up to 200 epochs, the algorithm halts when results stop improving. The learning curve is far from perfect. The spike of the red validation line shortly after the 5th iteration hints at some technical issues in the neural network that can lead to a distorted and dysfunctional model. Since the hiccup is in the validation set, not the training set, it likely didn't affect the training.

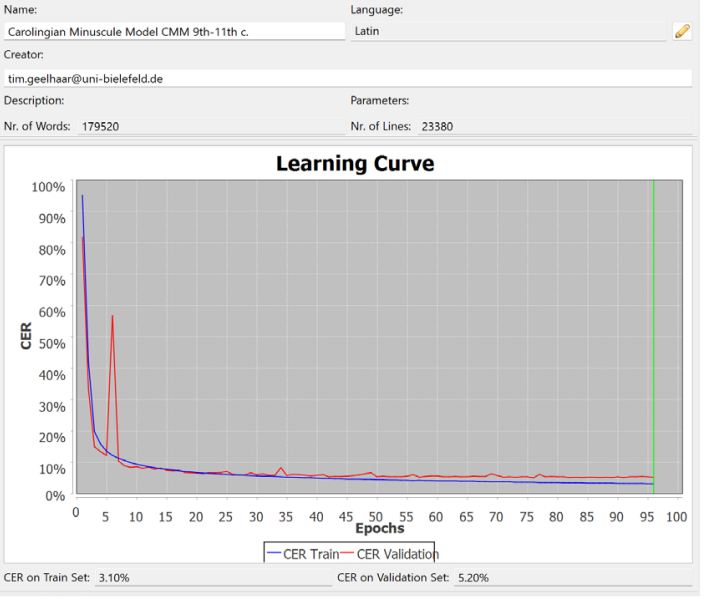


Figure 18: Spiked Learning Curve, image by the author, CC BY 4.0

To make sure that the model is not distorted, a tenth model training was started. The learning curve for this later model (CMM fine) was trained on 22 July 2023 with 175,215 words on 22,475 lines, essentially with the same set of ground truth but trained in one run. The curve looks smoother than for CMM and the best iteration seems to be at 50 before a slight overfitting kicks in. But the final CER for the training and validation set are close to the published model (Figure 19).

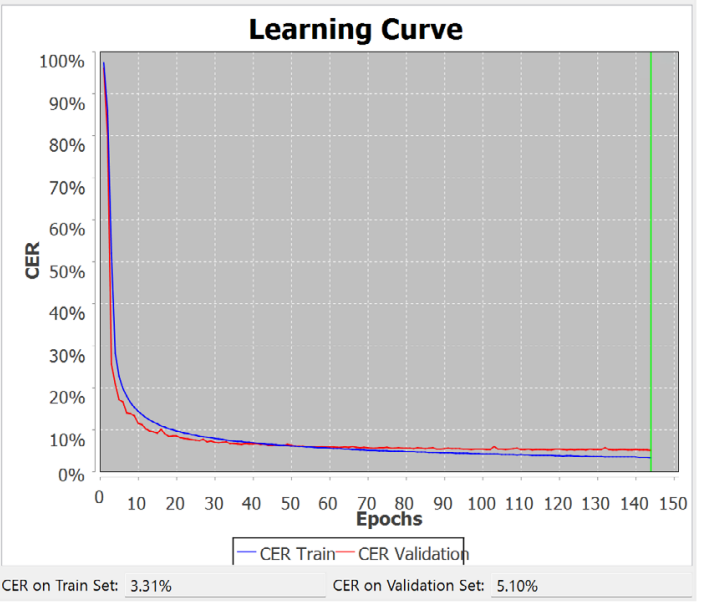


Figure 19: Smooth Learning Curve, image by the author, CC BY 4.0

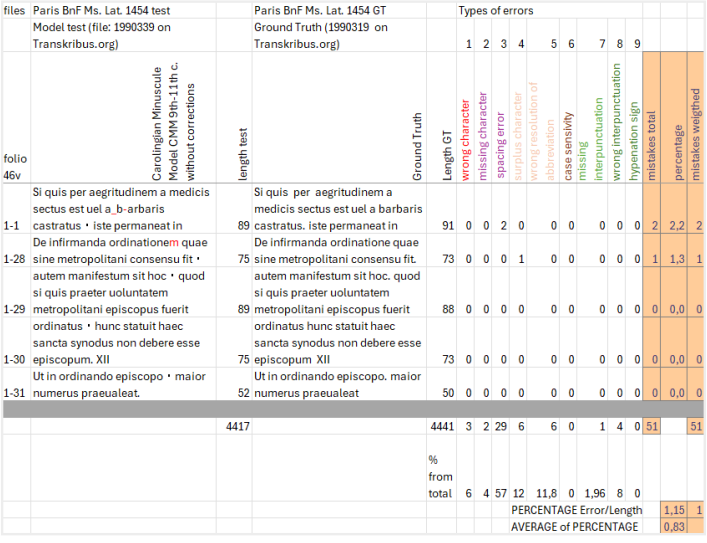
The real challenge for each model is to be tested against untrained material. Although there are also ways to compute accuracy on untrained material on the Transkribus Client, it seemed more promising to continue with manual testing to better understand the nature of mistakes and to assess the reliability of the automatically generated CER scores.

The manual test was done in three steps. First, the model was run on two pages from a completely unknown manuscript, Paris BnF Ms. Lat. 1454 (46v and 47r from the *Collectio canonum Quesnelliana*). Then, the result was duplicated as two separate files, while one of them was manually corrected. Both versions were finally copied to an Excel file for

comparison. Figure 20 represents only a snippet of the complete file that can also be found on Zenodo (https://doi.org/10.5281/zenodo.16613534). This comparison is based on the length of each line in number of characters (including spaces). Nine different typical error types have been assessed and percentages calculated (wrong character, missing character, spacing error, surplus character, wrong resolution of abbreviation, case sensitivity, missing interpunctuation, wrong interpunctuation, wrongly added hyphenation sign).

The most recurrent mistakes are segmentation errors/missing spaces with 57% (e.g. *ab arbaris* vs. *a barbaris* in line 1-1), while the lines with the most mistakes are those with majuscules (error rates attaining almost 9,7% on 47r, 1–22, not shown in the picture). Interestingly, it makes no difference if certain errors are weighed only 50% compared to others. The overall number of errors is 51 to 50,5 which translates to 1,15%. Regarding other kinds of errors, the manual comparison reveals that the model also struggles with surplus characters, i.e. the model adds non-existing characters (line 1–6) due to the resolution of abbreviations. But this error appears extremely seldom (0,14 %), like the wrong resolution of abbreviations and interpunctuation errors (5 in total).

The last observation is surprising since interpunctuation appears to remain one of the major challenges for the model. Interpunctuation in general is far less stable than the system of abbreviations. Not only were punctuation marks written differently but sometimes even served a different function, making encoding potentially difficult. In addition, later readers apparently tried to improve the punctuation to suit their reading habits and reading comprehension, usually with little success. Their adjustments and additions are extremely challenging to distinguish from the original interpunctuation for both human readers and the machine. A further improvement of the model has to take this challenge into account.



files	Paris BnF Ms. Lat. 1454 test		Paris BnF Ms. Lat. 1454 GT		Types of errors										
	Model test (file: 1990339 on Transkribus.org)		Ground Truth (1990319 on Transkribus.org)		1	2	3	4	5	6	7	8	9		
folio 46v	Carolingian Minuscule Model CMM 9th-11th c. without corrections	length test	Ground Truth	Length GT	wrong character	missing character	spacing error	surplus character	wrong resolution of abbreviation	case sensivity	missing interpunctuation	wrong interpunctuation	hypenation sign	mistakes total	percentage	mistakes weighted
1-1	Si quis per aegritudinem a medicis sectus est uel a_b-arbaris castratus · iste permaneat in	89	Si quis per aegritudinem a medicis sectus est uel a barbaris castratus. iste permaneat in	91	0	0	2	0	0	0	0	0	0	2	2,2	2
1-28	De infirmanda ordinationem quae sine metropolitani consensu fit ·	75	De infirmanda ordinatione quae sine metropolitani consensu fit.	73	0	0	0	1	0	0	0	0	0	1	1,3	1
1-29	autem manifestum sit hoc · quod si quis praeter uoluntatem metropolitani episcopus fuerit	89	autem manifestum sit hoc. quod si quis praeter uoluntatem metropolitani episcopus fuerit	88	0	0	0	0	0	0	0	0	0	0	0,0	0
1-30	ordinatus · hunc statuit haec sancta synodus non debere esse episcopum. XII	75	ordinatus hunc statuit haec sancta synodus non debere esse episcopum XII	73	0	0	0	0	0	0	0	0	0	0	0,0	0
1-31	Ut in ordinando episcopo · maior numerus praeualeat.	52	Ut in ordinando episcopo. maior numerus praeualeat	50	0	0	0	0	0	0	0	0	0	0	0,0	0
		4417		4441	3	2	29	6	6	0	1	4	0	51		51
				% from total	6	4	57	12	11,8	0	1,96	8	0			
														PERCENTAGE Error/Length		1,15	1
														AVERAGE of PERCENTAGE		0,83	

Figure 20: Evaluation metrics, image by the author, CC BY 4.0

All in all, the total error rate for this sample is about 1,15%, which is far better than the calculated error rate on the training set with 3,10 to 3,31%. It even attains error rates for modern printed texts like papers (like Transkribus Typewriter with 1,4%). Admittedly, the excellent result is also due to the very good readability of the unknown test material and will be higher for more challenging manuscripts with many comments and corrections, like in the case of Basel UB II 5 containing Sedulius Scottus's commentaries on the Pauline epistles. Nonetheless, the result shows first that models can perform even better than the automated calculated CER suggests; second, that special errors need to be addressed in further training – in this case the majuscules; and third, that even a model like the CMM can already be useful to transcribe large amounts of texts.

Conclusion

This chapter introduced the Caroline Minuscule Model (CMM) as a general, robust ATR model that can be used to efficiently transcribe large numbers of manuscripts for text mining purposes. It described how this model was built with this goal in mind. This included the selection of manuscripts for training, the transcription guidelines and their

underlying principles, and the way in which the training was carried out. The evaluation showed that the model performed even better than suggested by the character error rates automatically calculated by Transkribus. The resulting transcriptions could be passed directly to text mining tools, but it goes without saying that post-correction of the results would still improve the text mining result. This is particularly true for interpunctuation, which would need to be normalised in the post-correction process.

Furthermore, the results encourage us to present the CMM as an out-of-the-box solution for anyone who wants to work directly with manuscripts written in Caroline Minuscule. As a result, transcriptions can be produced much more efficiently, and the researcher will have to do less editing. In principle, however, researchers will still need to produce the best possible transcription, e.g. for editorial purposes. It can particularly support projects using highly repetitive material like miscellanies and canon law collections. As such, it is used already by the Carolingian Canon Law Project (https://ccl.rch.uky.edu/) that seeks to identify variations in textual history by establishing a conceptual corpus of all canons from the Early Middle Ages. It could also be beneficial for the Clavis Canonum (https://data.mgh.de/databases/clavis/db/) that gathers information on Latin canonical collections and their manuscripts from ca. 500 to 1234. Even projects as far advanced as the project Capitularia could still benefit from the model.

The model is available on Transkribus (Geelhaar 2023) and the ground truth can be found on Zenodo (https://doi.org/10.5281/zenodo.16613534). In this way, the material is also available to other researchers as a basis to train further models. This would be desirable because any further training can help eliminate existing weaknesses in the model, such as punctuation recognition. It can also serve as a basis for more fine-grained training of ATR models that focus on single scriptoria or even writers to identify the origin of manuscripts. The author aims to further develop the model in the direction of a general manuscript model for early medieval scripts, which can also automatically transcribe pre-Carolingian and other scripts of the early Middle Ages, including the pre-Gothic scripts of the 12th century.

5.3 Michael Schonhardt: ATR and Editorial Workflows – *Burchards Dekret Digital*

Introduction

The project *Burchards Dekret Digital* (BDD) focuses on creating a hybrid edition of one of the most influential texts in medieval canon law, the Decretum Burchardi or Burchard's Decree, henceforward referred to as BD. Compiled in the first quarter of the 11th century by Bishop Burchard of Worms, this collection of canon law quickly became a standard reference for scholars and practitioners. As a rich and extensive text that reflects the application of law and legal discourses, BD offers valuable insights for historians of canon law and contributes significantly to our understanding of cultural history in general (Austin 2009; 2019; 2022; Kéry 1999; 2017; Kynast 2020). Of particular interest are five manuscripts (listed in alphabetical order and referenced using the project's sigla) dating back to the decree's creation:

1. B: Bamberg, Staatsbibliothek, Msc. Can. 6 (312 leaves) (https://www.digitale-sammlungen.de/de/view/bsb00140701)
2. F: Frankfurt, Universitätsbibliothek, Ms. Barth. 50 (315 leaves) (https://sammlungen.ub.uni-frankfurt.de/msma/content/titleinfo/2035614)
3. K: Köln, Erzbischöfliche Diözesan- und Dombibliothek, Cod. 119 (204 leaves) (https://digital.dombibliothek-koeln.de/hs/content/titleinfo/284343)
4. V^2: Vatican, Biblioteca Apostolica Vaticana, Pal. lat. 585 (330 leaves) (https://digi.vatlib.it/mss/detail/Pal.lat.585)
5. V^3: Vatican, Biblioteca Apostolica Vaticana, Pal. lat. 586 (280 leaves) (https://digi.vatlib.it/mss/detail/Pal.lat.586)

Remarkably, not only do these manuscripts show numerous traces of textual development, but can also be located in the bishop's scriptorium at Worms, providing a rare glimpse into the processes and practices of compiling and reformulating legal matters during the High Middle Ages (Hoffmann and Pokorny 1991). Building upon these manuscripts, BDD aims to create an edition of the Decretum accessible in print and as an extended TEI-encoded digital edition online (www.burchards-dekret-digital.de). This edition will feature an established text but also a synoptic presentation of the various manuscripts' text and

images. Consequently, the editorial process focuses on: 1. A thorough philological and codicological evaluation of the manuscript sources; 2. Preserving the relationship between text and image; 3. Maintaining the text as presented in the manuscripts; and 4. Providing a normalised and collated version of BD's text.

BD covers all five textual witnesses and almost the whole manuscript, representing approximately 125,000 lines on 2,800 manuscript pages that need to be transcribed – a very challenging number. Fortunately, all five key textual witnesses were prepared during the same period and in the same scriptorium by scribes sharing notable handwriting features. In fact, most parts of the manuscripts mentioned above were written by the same scribes, whose hands are relatively easy to read and who employ a consistent and primarily unambiguous system of abbreviations (Hoffmann and Pokorny 1991).

Additionally, the layout and overall presentation are remarkably consistent across these manuscripts. At times, scribes even made significant efforts to replicate the layout as closely as possible from one another. All witnesses are based on a two-column layout with recurring modules specifically designed to accommodate their legal content: Each of the 20 books in which BD is structured is introduced by a table of contents listing the individual chapters dedicated to a particular legal matter. These tables of content consistently indicate the number of each chapter with Roman numerals in red ink in the margins and commence the subsequent title with a small red initial for improved readability.

Subsequently, these chapters are featured in the content section of each book, which begins with a title and a so-called *praefatiuncula* – a very brief introduction to the book. Again, all chapters consistently adhere to the same structure, with a marginal red Roman chapter number and a red title in the main column. A larger red initial marks the start of a chapter. Moreover, a so-called *inscriptio* is included below the chapter number in the margin, providing a reference to the chapter's source. Lastly, all pages are accompanied by a header that displays the title of the current book in the upper margin. This layout can be best examined using the links to the digital copies mentioned earlier, but it is also well represented by Figure 21.

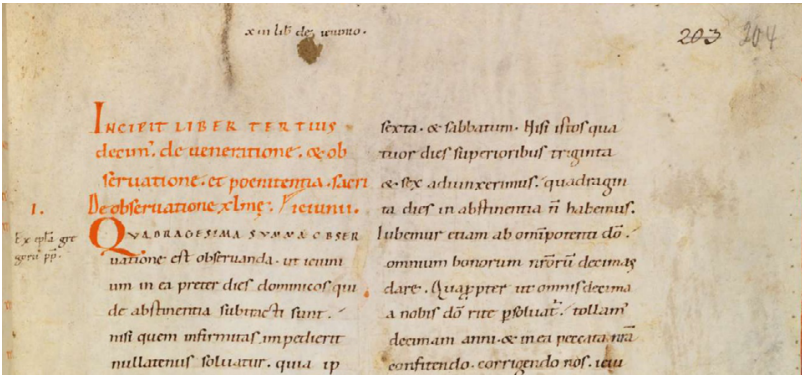


Figure 21: Bamberg, Staabsbibliothek, Can. 6, fol. 204r, Staatsbibliothek Bamberg, PDM 1.0

In the project, these witnesses serve not only as the philological foundation for a constituted text but also hold significant analytical value. As they reveal numerous indications of revisions and engagement with the text by its creator, they provide 'a glimpse onto the bishop's desk', which is the context in which they were created. Therefore, it is essential to create a detailed digital representation of each witness that preserves textual peculiarities, layout, and other codicological phenomena offering insights into this process. However, considering the limited time frame and funding, this representation must be achieved efficiently.

Considering the consistent features regarding scribes and layout mentioned earlier, it seemed plausible to rely on automated layout analysis and ATR. Although initial experiments with Transkribus produced impressive results in terms of CER, it quickly became evident that these methods can only be successfully applied in editorial practice if they are thoroughly considered, adapted, and integrated into the broader editing 4.6 Workflow. When executed effectively, though, ATR not only saves time by eliminating the need for manual transcription but also provides benefits beyond the mere transcription process.

In the following sections, I will outline the BDD's approach to model training and application, as well as provide an overview of the digital workflow into which ATR solutions typically are integrated. In doing so, I will emphasise the importance of a modularised workflow specifically tailored to the unique needs of digital editions rather than relying on

a generic, out-of-the-box software solution. Furthermore, I will discuss the necessity of employing a multi-layered transcription process based on graphemic models, as opposed to normalised transcription (4.3 Transcription), for digital editing.

Workflow

Although every historian interacts with textual sources in one way or another, the methods employed can vary significantly. Some research questions necessitate a detailed close reading of a single source, while others rely on distant reading of a vast corpus of texts, often employing quantitative methods. However, both ends of the spectrum typically approach sources from a primarily analytical perspective, focusing mainly on the content of the sources. In this context, the transcription process can be viewed as unearthing content from written records, which must then be refined through normalisation, standardisation, and correction to varying degrees. If the goal is to obtain a clean text that can be easily analysed and compared, selecting tools (3 Tools) and digital infrastructure is no particular challenge: Any modern ATR solution, such as Transkribus, is suitable, provided there are sufficient data to train a highly normalised model (2.3 ATR as Machine Learning, 4.3 Transcription, 4.4 Training). A clean text can then be exported and utilised as a text file via the standard export functions of the chosen software (3.2 Integrated Transcription Environment).

Digital editions, on the other hand, are a very different matter. While projects with an analytical focus tend to minimise the peculiarities of a textual witness, digital editions try to carefully and transparently create an edited text (or multiple texts) based on a thorough philological, codicological, and palaeographical analysis of its existing sources. As a result, scribal errors, unique readings, and other peculiarities of a manuscript must be preserved before being processed (e.g. corrected or normalised) in a traceable manner. In doing so, digital editions often generate multiple layers of textual representation and store them in TEI-XML (Burghart 2017), or similar formats (Figure 22).

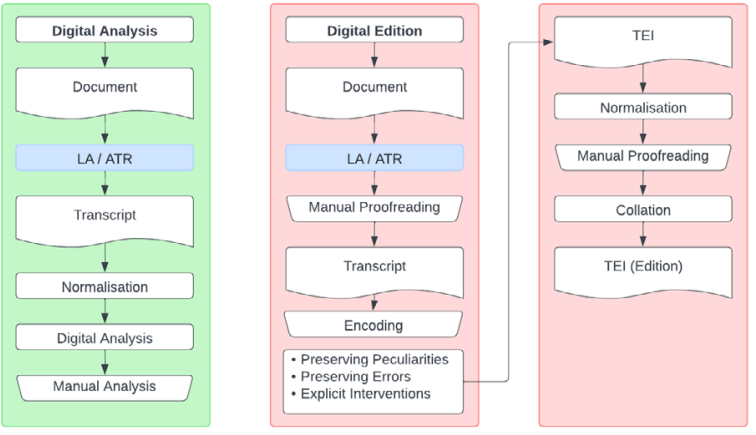


Figure 22: Analytical vs. editorial workflow, image by the author, CC BY 4.0

These requirements significantly impact the workflow and infrastructure needed to accommodate a digital edition. Generally, a digital edition's infrastructure is based on five different areas, which can overlap or be pursued in parallel:

1. Data creation
2. Data input
3. Data storage
4. Data transformation
5. Data output

Within this general framework, many editorial workflows can then be applied, which may differ significantly from project to project both in terms of complexity and technical implementation. The digital edition of BD not only takes textual phenomena into account but also layout and codicological phenomena that will be made visible using IIIF in the Mirador viewer. At the same time, it is trying to create several types of ground truth (4.1 Data, 4.4 Training) for later stages of the editorial pipeline (Schonhardt 2025a), making BDD's workflow quite complex and multi-layered (Figure 23).

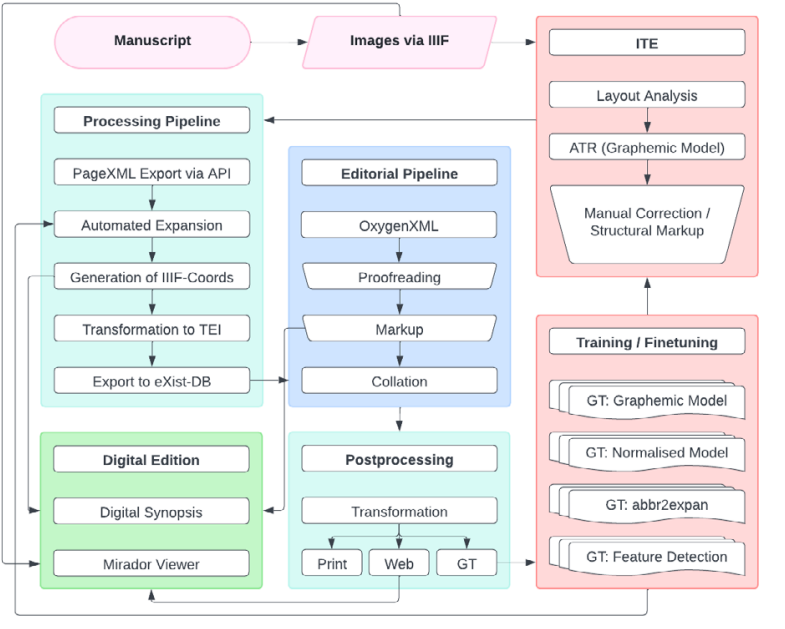


Figure 23: Flowchart of BDD, image by the author, CC BY 4.0

Technically, the editorial workflow is based on Transkribus, OxygenXML with a dedicated framework, eXist-db, and several custom processing pipelines written in Python. Manuscripts are input in the workflow via IIIF, processed in Transkribus, manually corrected, and structurally annotated. They are then exported as PAGE XML (4.1 Data) and transformed in several steps to TEI-XML suited to the project needs and eventually uploaded to eXist-db. From there, they are further annotated and proofread in OxygenXML and other programmes, taking the philological, codicological, and palaeographical aspects of each manuscript into account while making all peculiarities, such as scribal errors, traceable. To support this encoding (4.3 Transcription), frameworks have been developed that cater the various needs of the editors. Once encoding is finished, the data are post-processed in another Python-based pipeline and transformed into output formats for print (which then goes to an external typesetting pipeline) and for web (which is then relayed to a repository and displayed using web stack). This includes layout data generated by the automatic layout recognition and ATR process, which is needed for displaying text/image relations. Lastly, ground truth data

are generated from the final TEI-encoded transcription, which gets fed back into the training loop, making it possible to fine-tune existing models as well as create new ones (4.4 Training).

Without delving too deeply into technical details at this point, this description suffices to show that in this workflow, ATR and corresponding tools are only a small part of a larger editorial landscape. Although most aspects of the editing process, such as markup of specific phenomena, can technically be performed directly in some ITEs (3.2 Integrated Transcription Environments), there is dedicated software better suited for the task at hand, which is creating complex TEI-encoded and multi-layered representations of a manuscript witness. Thus, complex and modular workflows should be reflected in the complex and modular architecture of tools and protocols. Nevertheless, this does not diminish the value of ATR and powerful ITEs: Not only do these technologies rapidly increase efficiency by automating one of the most tedious aspects of editing – transcribing – but they also significantly enhance the encoding potential of manuscripts by generating layout information that can be used in other contexts, as well. Thus, layout analysis is the first process in BDD's workflow.

Automated Layout Analysis in Transkribus

The process of ATR is often conceptually simplified as the direct transformation of a binary image into machine-readable text (2.1 How to Recognize a Text). However, it is crucial to emphasise that accurate layout detection (4.2 Layout) is equally foundational for two reasons. First, the effectiveness of ATR is closely tied to the precision of layout recognition, which segments the image into meaningful zones, such as text blocks, lines, and regions. If text-bearing elements – such as decorated initials, marginal annotations, or punctuation at line ends – are not correctly identified and segmented during this stage, they may be excluded from the recognition process altogether. In this context, *correct* refers to the faithful identification and classification of relevant text zones in accordance with their actual function and placement within the document. Second, false positives in layout detection – i.e. the misclassification of non-textual elements such as smudges, watermarks, ornamentation, or parchment damage as text – can degrade recognition quality, as the ATR engine attempts to transcribe visual noise, thereby introducing

erroneous characters into the output. Thus, robust layout analysis is a prerequisite for achieving high-quality text recognition.

In medieval manuscripts, information encoded in the layout is often just as crucial as the textual content itself. This is especially true for manuscripts with a two-column or multi-column layout, which is quite common in high medieval manuscripts such as the witnesses of BD. Here, clean detection of columns is essential to produce a meaningful transcription, as horizontal lines need to be split correctly if they belong to separate columns. However, this can be challenging from a technical point of view, especially when the columns have a narrow gap or when/if other elements add further complexity. For these reasons, manuscripts of BD could not be segmented correctly by Transkribus's default segmentation.

Thus, manual correction had to be applied after layout recognition, particularly regarding lines and smaller text regions. In this very challenging and time-consuming process, separated recognition of text regions and lines proved to be beneficial. Not only was it easier to correct the text regions when line segmentation was initially omitted, but subsequent line recognition also improved when applied to regions that had been more accurately identified and delineated. At the time of writing (end of 2023), Transkribus also offers a very useful feature that can be selected in the tab 'Layout Recognition', which automatically cuts lines at region boundaries, solving the problem of lines overlapping narrow regions. This manual process helps improve text recognition but also lays the foundation of any layout-related feature of the digital edition, such as visualising certain phenomena on the manuscript image by providing coordinates.

This is particularly important, as layout not only serves as a carrier for text but also provides a way to encode information on the text presented through meaningful structure (as demonstrated in the manuscript image above). In the context of BD, for instance, it is crucial to identify small textual elements in the margin of the manuscript page, such as chapter labels belonging to other small elements beneath or beside them, and link them to a corresponding chapter in one of the two columns. Without this information, it would be challenging to make sense of the recognised text, since it would be forced into a strictly linear order that doesn't reflect the actual, non-linear layout of the manuscript.

Additionally, certain textual elements – such as inscriptions, chapter numbers, or marginal annotations – require distinct treatment during the editorial process, as they must be encoded differently from the main body of the text in BDD's TEI-XML schema. Therefore, the detection and representation of this structural information is a fundamental part of editing and also critical for any meaningful textual analysis of transcribed manuscripts.

Fortunately, layout recognition can also be leveraged to identify and add structural metadata to the various elements of the page. In BDD, the following tags are used to represent the structural elements:

1. header
2. footer
3. column_1 (custom tag)
4. column_2 (custom tag)
5. chapter_count (custom tag)
6. inscription (custom tag)

Most ITEs will display these tags in different colours on the image (see example of Transkribus below), but most importantly, they are stored as an @custom attribute in the text region's PAGE XML. Additionally, the coordinates of the text region are stored in a dedicated <Coords> element, as well, and refer to the pixel coordinates of the region in question on the image. The following example illustrates the text region corresponding to the inscription seen on the manuscript image (Figure 24) without lines and recognised text.

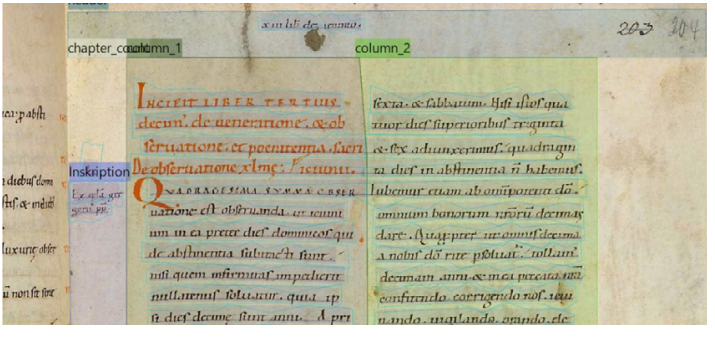


Figure 24: Bamberg, Staatsbibliothek, Can. 6, fol. 204r, coloured text regions in Transkribus, image by the author, CC BY 4.0

```
<TextRegion orientation="0.0" id="TextRegion_1654608976847_243"
custom="readingOrder {index:2;} structure {type:inscription;}">
  <Coords points="136,472 254,472 254,1605 148,1606"/>
</TextRegion>
```

This structural markup can be applied manually by assigning a specific tag to a text region or other structuring element (in Transkribus, this can be done in Metadata → Structural). In BDD, approximately 50 pages have been annotated this way in V². Then, this material was used to train a P2PaLa model for automated recognition of text regions (excluding lines) and their corresponding structural tag (Figure 25). Although the model occasionally identifies superfluous text regions – such as noise or non-textual artifacts – which must be manually removed, it consistently assigns appropriate semantic labels to the relevant textual elements it detects. Moreover, it significantly improved the correct detection of a two-column layout, achieving almost perfect results. By the time of writing, seven manuscripts have been fully annotated this way and published as a dataset (Schonhardt 2024b) as well as corresponding layout recognition-models in Kraken (Schonhardt 2024a; 2024c) to be used outside of Transkribus.

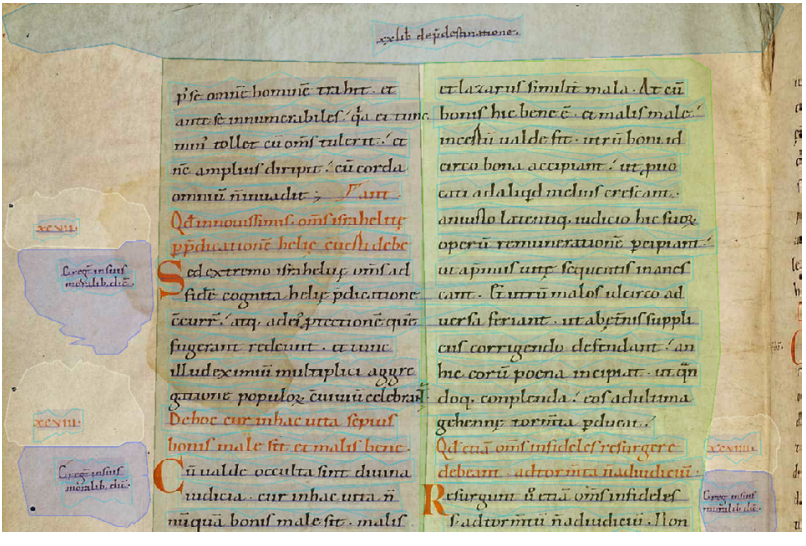


Figure 25: Bamberg, Staatsbibliothek, Can. 6, fol. 309v, regions recognised by P2PaLa without interventions, image by the author, CC BY 4.0

Once all text regions are detected and corrected, lines are added using the default layout detection algorithm provided by Transkribus. Since the text regions have already been identified, it is essential to deselect the feature to use the existing regions. Although the main text field is almost always accurately recognised as two separate columns, lines are sometimes incorrectly detected if the gap between the columns is too narrow. To avoid the tedious process of post-correction, line detection was performed twice on each page, with only one column selected at a time. This way, only the lines of the relevant column (as well as smaller elements on the margins opposite the column) are detected, and their coordinates are also stored in the PAGE XML. (Since at the time of writing, Transkribus added a feature for automatically cutting lines at region borders, this may no longer be necessary.)

```
<TextRegion orientation="0.0" id="TextRegion_1654608976847_243"
 custom="readingOrder {index:2;} structure {type:inscription;}">
 <Coords points="136,472 254,472 254,1605 148,1606"/>
 <TextLine id="TextRegion_1654608976847_2431l"
  custom="readingOrder {index:0;}">
   <Coords points="140,514 154,518 […] 139,473"/>
   <Baseline points="145,507 176,505 207,506 239,505"/>
   <TextEquiv>
     <Unicode></Unicode>
   </TextEquiv>
 </TextLine>
 <TextLine id="TextRegion_1654608976847_2431l2" custom="reading-
 Order {index:1;}">
   <Coords points="135,554 144,556 […] 132,517"/>
   <Baseline points="139,543 165,535 191,536 217,535"/>
     <TextEquiv>
       <Unicode></Unicode>
     </TextEquiv>
 </TextLine>
 <TextEquiv>
   <Unicode></Unicode>
 </TextEquiv>
</TextRegion>
```

Although layout detection is generally good enough to provide a solid foundation for ATR without manual intervention, the detected lines are checked and corrected manually because the underlying coordinates are essential in the later stages of the editorial process. Additionally, this step provides a crucial opportunity for close engagement with the sources, allowing editors to familiarise themselves with the material features, scribal practices, and structural conventions of the manuscripts – insights that are essential for making informed editorial decisions. Finally, the manuscripts are prepared for ATR. However, before automated transcription can begin, a model must not only be trained but also carefully conceptualised, ensuring that its design aligns with the specific characteristics of the script, layout, and editorial goals of the project.

ATR Model

When transcribing (4.3 Transcription) for a traditional context, the transcriber typically translates the medieval writing system directly into the framework of modern typography. The cognitive process of this translation involves expanding abbreviations and brevigraphs or matching different allographs with a standardised set of characters in a type font. This process is usually excluded from the printed output or preserved implicitly through layout and typesetting. For example, the abbreviation 'semp̨' in a manuscript might be transcribed as 'sem[per]' or even silently expanded as 'semper'.

In this regard, digital editing is fundamentally different from traditional editing (Sahle 2013a; 2013b; 2013c). In digital editing, a medieval source is not translated into typography but into a digital framework. This happens on two levels: First, the text is encoded at a lower level as a set of characters, usually in Unicode (Jannidis 2017), but secondly, at a higher level, certain aspects of this text are made explicit, such as abbreviations or special characters (4.1 Data). Encoding textual features and editorial interpretations in a structured, machine-readable format enables the transcription to serve as a genuinely digital representation. According to TEI guidelines, an abbreviation such as *semp̨* can be encoded using the <choice> element to capture both the abbreviated and expanded forms:

```
<choice>
  <abbr>sem<g ref="#per">p̄</g></abbr>
  <expan>semper</expan>
</choice>
```

This structure ensures that both the original manuscript form and the editorial expansion are preserved in a semantically meaningful way.

For this reason, a transcription of a medieval source in a word processor, although technically digital, still belongs to the print paradigm. It does not explicitly encode various aspects of the transcribed text but is done implicitly (Hodel 2022a). Although deep learning-applications like ATR are much more advanced than word processors, they also tend to follow a print paradigm. Paradoxically, this is due to one of the most significant advantages of deep learning, its ability to mimic the information processing of the human mind. As ATR models learn how to transcribe by deducing patterns implicit in training data, they also reproduce transcriptions that leave their editorial decisions implicit. If a model is trained on normalised data, it will produce transcriptions unsuitable for digital editing, where editorial decisions such as expanding abbreviations should be explicitly encoded. Thus, transcriptions made by ATR tend to be less explicit and formalised than those encoded manually.

However, this problem can be solved by training models to produce transcriptions that preserve much of the writing system present in the source. This 'raw' data can then be further processed into a more readable form in a separate step using various post-processing methods, including expanding abbreviations or normalisation. This way, editorial steps and decisions can be explicitly encoded into a digital representation of the source in the context of editing without losing the advantages of ATR. Additionally, these transcriptions yield better results in CER because they do not include characters that are absent from the source. Furthermore, they are much more versatile and sustainable, preserving much of the source's potential for future use (4.7 Use and Reuse).

However, there are many ways to preserve much of the writing system present in the source. In BDD, we opted for a pragmatic approach that preserves the function of the characters and symbols present in the source rather than their exact appearance. Thus, we prepared transcriptions

(https://gitlab.rlp.net/adwmainz/projekte/burchards-dekret-digital/data) that differentiate on the level of graphemes and brevigraphs rather than allographs and other forms of palaeographical phenotypes (4.3 Transcription). In our transcriptions:

- Ligatures are expanded into corresponding characters rather than preserved.
- Brevigraphs are transcribed using corresponding special characters.
- Abbreviations are not expanded.
- Interpunctuation is preserved using a set of dedicated symbols.
- Hyphens are only transcribed when present.
- The text is transcribed as it appears in the manuscript and not corrected.
- The goal is to preserve the function of a sign, not its appearance.

To represent brevigraphs or interpunctuation in a standardised manner, special characters strictly based on MUFI (https://mufi.info/) recommendations were chosen, in addition to standard Unicode characters. Using this set of characters, a certain amount of text was manually transcribed from manuscript V2. Once around 30,000 words were transcribed and corrected to ground truth, a model was trained using the now depreciated HTR+ engine over 70 epochs (with a CER of 1.20% on the training set and 2.20% on the validation set). This model was then expanded and material from the other manuscripts was included in an iterative process. The final model was trained using the PyLaia engine (Puigcerver and Mocholí 2018) on 111,800 words from all five manuscripts over 111 epochs (with a CER of 1.30% on the training set and 2.40% on the validation set, Figure 26) and is available as a public Transkribus model (Schonhardt 2023a) as well as a Kraken model (Schonhardt 2024e). Using this model, the manuscripts can now be transcribed automatically with only minor interventions. Preliminary tests also suggest that the model works sufficiently for other manuscripts from the 11th and 12th centuries.

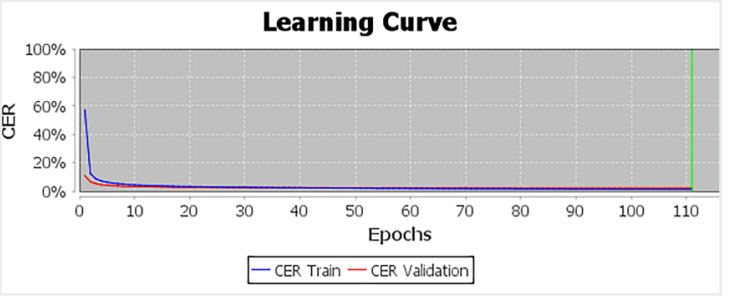


Figure 26: Learning curve of final PyLaia model, image by the author, CC BY 4.0

We call this model a graphemic model, because it preserves the writing system on the level of graphemes. Such a model is valuable for digital editing for several reasons. Firstly, it tends to produce better results in terms of CER compared to a 'smart model' that normalises the given text because it does not need to deal with characters that are not present on the page (5.1 Case Study 1, 5.2 Case Study 2). Secondly, it better preserves scribal errors and avoids hyper-correction through normalisation (although hyper-correction can still be an issue). Thirdly, it maintains abbreviations explicitly and enables various forms of expansion, normalisation, and corresponding encoding outside of ATR. Lastly, the generated material is very sustainable because it still retains the potential for other transcribing forms and can be used as ground truth for further training. In BDD, for example, these data are post-processed to ground truth suitable for training an ATR-model to expand the written text in Transkribus (Schonhardt 2023b) and Kraken (Schonhardt 2024d), but also to generate training data to train a text2text expansion model based on ByT5 (Schonhardt 2025b; 2025c; 2025d). If we had already applied normalisation in ATR, all peculiarities of the manuscript would have been lost.

Using this graphemic model, the witnesses were transcribed automatically and manually corrected. As the manuscripts were not hyphenated, line breaks within words were manually marked with '¬'. The text regions were linked together using a shorthand system to indicate their structural affiliation, such as chapter number, inscription, and title of chapter one indicated using '*1*'. Instead of Transkribus's internal method of layout structuring, a manual approach was used as it was quicker and easier for project staff to apply. Basic markup was provided

using a set of simplified TEI-XML tags and symbols, such as '#' for indicating a versal or <add></add> for indicating an addition. Although Transkribus provides the option to markup text via an internal tagging system, the manual approach was found to be the most efficient in terms of application and post-processing. This led to the final PAGE XML, which was exported and post-processed into TEI format suitable for the project's needs through a Python pipeline.

```
<TextRegion orientation="0.0" id="TextRegion_1654608976847_243"
 custom="readingOrder {index:2;} structure {type:inscription;}">
 <Coords points=”136,472 254,472 254,1605 148,1606”/>
 <TextLine id="TextRegion_1654608976847_2431l" custom="reading
 Order {index:0;}">
   <Coords points="140,514 154,518 […] 139,473"/>
   <Baseline points="145,507 176,505 207,506 239,505"/>
   <TextEquiv>
     <Unicode>⁎1⁎Ex epꝥa gre¬</Unicode>
   </TextEquiv>
 </TextLine>
 <TextLine id="TextRegion_1654608976847_2431l2" custom="reading
 Order {index:1;}">
   <Coords points="135,554 144,556 […] 132,517"/>
   <Baseline points="139,543 165,535 191,536 217,535"/>
   <TextEquiv>
     <Unicode>gorii pp¯·</Unicode>
   </TextEquiv>
 </TextLine>
 <TextEquiv>
   <Unicode>⁎1⁎Ex epꝥa gre¬
     gorii pp¯·
   </Unicode>
 </TextEquiv>
</TextRegion>
```

Export

Software tools like Transkribus usually offer a range of options for exporting textual data in various formats. In digital editing, TEI-XML is usually the preferred output format. While Transkribus does offer the option to export to TEI-XML, which can be customised using an XSLT

stylesheet, the project chose to export raw PAGE XML and process the data through a Python pipeline tailored to its specific needs and workflows. The goal was to automatically generate TEI-encoded representations of the witnesses that make explicit any editorial interventions while also being suitable for a synoptical view in a digital edition. This view should allow textual, codicological, or palaeographical phenomena, as well as editorial comments, to be visible in both textual and scanned image representations of the manuscript. The workflow developed for this project is as follows:

1. The PAGE XML is downloaded using Transkribus's REST interface.
2. Text regions are extracted and evaluated using the lxml library. (https://lxml.de/).
3. Lines in text regions are extracted using lxml library, and <lb> or <lb break="no"> elements are inserted based on the presence of '¬' at the end of a line.
4. Coordinates provided by Transkribus are extracted and converted to bounding boxes suitable for retrieving image data via the IIIF.
5. IDs are created to make enable matching of elements in PAGE XML.
6. The position of particular elements is determined via a Python function using these coordinates.
7. The text of each line is evaluated for brevigraphs or other signs indicating abbreviations. Abbreviations were initially expanded via a Python script according to a list and set of rules and transformed into a <choice> element. Currently, this rule-based method is replaced by a deep learning approach trained on existing data (Schonhardt 2025b; 2025c; 2025d).
8. The shorthand system is replaced with proper TEI markup via Python.
9. The post-processed PAGE XML is combined into a single TEI-XML file, and metadata such as the URL to the image or annotation files are added.

The TEI-XML resulting from this process (here in a simplified form) enables the handling of editorial decisions, such as the expansion of abbreviations, in an explicit but automated manner. It also efficiently transforms the complex structure of the manuscript into TEI, while preserving valuable information generated by Transkribus layout recognition.

```
<pb n="86r" facs="https://urlToFile/full/full/0/default.jpg"
 corresp="https://urlToFile/canvas/p0179"
 ana="/annotations/vatican-bav-pal-lat-586-annotation-p0179"/>
[...]
<div n="1" type="chapter" xml:id="vatican-bav-pal-586-13-con-001">
 <head type="chapter-title">[…]</head>
 <note type="inscription" place="margin left"
  facs="124,707,149,149">Ex
   <choice>
     <abbr>ep<g ref="#char-a749">ı̄</g>a</abbr>
     <expan>epistola</expan>
     </choice>Gregorii
   <choice>
     <abbr>pp<g ref="#char-0305">̄</g></abbr>
     <expan>papae</expan>
   </choice>
   <pc type="distinctio"><g ref="#char-f1f8">·</g></pc>
 </note>
 <lb n="7" facs="358,724,493,124"/>
 <p n="1">
 <hi rend="color:red initial">Q</hi>UADRAGESIMA
   […]
 </p>
</div>
```

In the final stages, this TEI-XML is uploaded to eXist-db, where it can be reviewed and further enriched by editors in OxygenXML or with similar tools. Eventually, it is transformed into HTML that is displayed using an http server. During this process, specific elements are automatically transformed into an Open Annotation Data Model and displayed in Mirador, utilising the coordinates extracted from Transkribus (Figure 27).

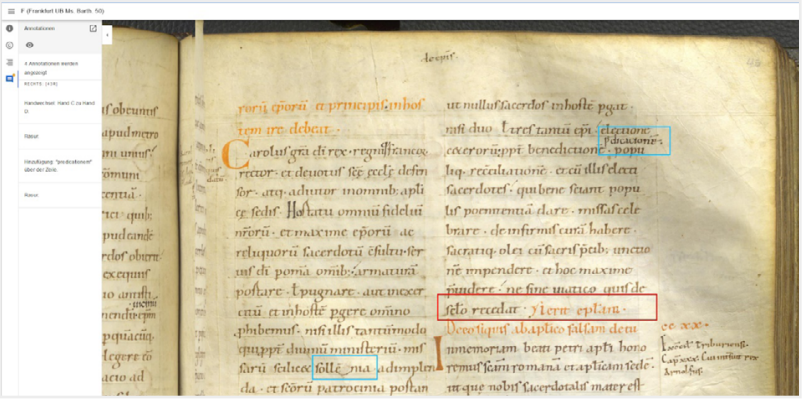


Figure 27: Display of phenomena in Mirador using Transkribus generated coordinates, image by the author, CC BY 4.0

Conclusion

In conclusion, ATR is an essential tool in digital editing of medieval texts if applied carefully and in the context of a broader workflow that includes various steps of data extraction, processing, and editorial evaluation. Digital editing requires a high degree of transparency and explicitness, including mistakes and other peculiarities found in the manuscripts, that cannot be disregarded by ATR applications. By using a graphemic model and a semi-automated workflow, BDD tries to achieve this goal while also taking advantage of the benefits of ATR technology. The graphemic model used in the project preserves the writing system on the level of graphemes, which not only produces better results in terms of CER, but also better preserves scribal errors and prevents hyper-correction by normalisation. It also helps keep abbreviations explicit and enables various forms of editorial intervention and corresponding encoding outside Transkribus. Furthermore, the generated material is very sustainable, as it preserves the potential for other transcribing forms and can also be used as ground truth for various models (Schonhardt 2025a). After all, ATR is only one step in a broader workflow, ranging from extracting data from the original manuscripts to further editorial processing. Thus, it is crucial to carefully conceptualise its integration into these workflows to achieve sufficient results not only in the recognition of text, but its editing, too.

5.4 Jan Odstrčilík: Multi-Lingual Automated Text Recognition of Medieval *Macaronic Sermons*

Multi-lingual Latin-vernacular sermons are a widespread yet underexplored phenomenon of the Middle Ages which can be found throughout Latin Europe. The mixture of languages can take various forms, ranging from single vernacular words glossing Latin text, to vernacular verses used for divisions in an otherwise Latin sermon, and even to sudden and seemingly random back and forth code-switches within single sentences. The latter type is traditionally referred to as macaronic in medieval studies (Wenzel 1994), although the term is not unproblematic (Demo 2014). A page from the National Library in Prague, XI F 3, fol. 1v, is an example of a Latin sermon with a vast number of Czech words (highlighted in Figure 28):

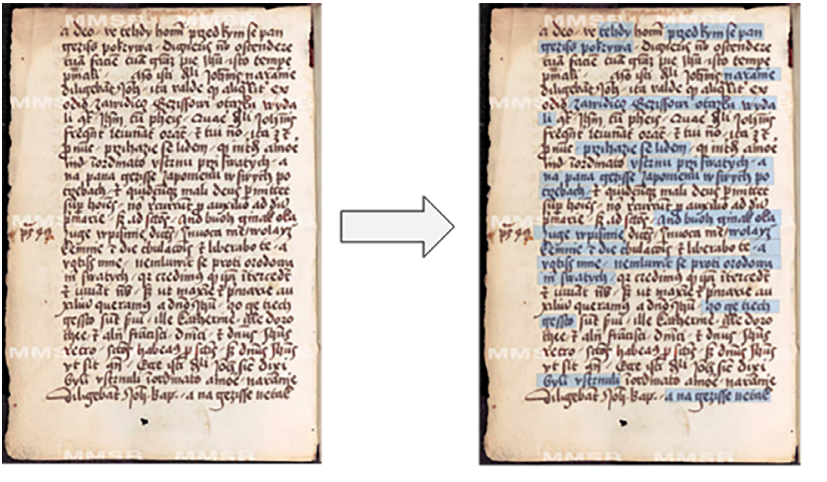


Figure 28: Prague, National Library, XI F 3, 1v with highlighted Czech words in blue, visualisation by the author, manuscript images by National Library in Prague, CC BY NC SA 4.0

Current research aims to analyse how these sermons originated and how they were used: Were they written by preachers themselves, or are they products of capturing speech in a church, a practice known as *reportatio?* Were these sermons delivered multi-lingually, or were they merely a written phenomenon? What triggers the switch from one language

to another? (Odstrčilík 2020) These are challenging questions because relatively few sermon collections of this type have been published so far (e.g. Horner 2006; Varischi 1964; Florianová et al. 2006). Could automated text recognition help solve this issue and make more of these texts accessible and analysable?

My research centres on 15th-century sermons that interweave Latin and Czech. Although both languages share the Latin script, their intermingling poses distinctive challenges for ATR:

1. The Latin language of the late Middle Ages was filled with abbreviations. Traditionally, when dealing with literary texts, Latin abbreviations are silently expanded in editions and transcriptions. However, there is an ongoing debate on this issue, and some researchers prefer to expand abbreviations only in post-processing (5.3 Case Study 3).
2. The Czech language of the late Middle ages had relatively few abbreviations; however, its orthography was more ambiguous and unstable than that of Latin. In response to this, Czech philology has developed a rather complicated and sophisticated interpretative method for rendering late medieval and early modern Czech within the rules of the modern Czech orthography as they were supposed to sound (Berger 2012, 265). This system is called "transcription" (Daňhelka 1985) and stands in opposition of diplomatic or palaeographical transcription, which is referred to as "transliteration" in the context of Czech sources.
3. In multi-lingual sermons, one language typically dominates. In my case, it is usually Latin. This creates, by default, unbalanced ground truth where one language dominates over the other.
4. Finally, there are usually no visual signs of code-switches, and both languages are written by the same scribe in the same script. Only in some cases have words in the second language been underlined by the scribe, rubricator, or a reader. Even in these cases, however, underlining is used for other purposes as well, such as biblical quotations and headings in Latin. All of this means that ATR models cannot use any visual signs to differentiate between languages. The situation is different for the modern period, where Latin was printed using a different typeface than the parts in Czech language.

Since I am interested in medieval code-switching, my goal was to investigate two main questions: how well the ATR would perform with mixed languages and, additionally, whether ATR could be used to detect passages written in Czech within Latin sermons, which would allow further analysis. I decided to use Transkribus as my ITE because I am familiar with it and it contains an integrated system for training annotations directly into ATR models.

I decided to conduct two specific case-studies. The first one focuses on the Latin-Czech sermons of Michael Polonus (†1480), a Prague preacher of Polish origin. The second is an anonymous Latin translation of *Czech Sunday Postil* of Jan Hus (*Česká nedělní postila*), which in many places retains, and sometimes even adds Czech vocabulary. Both texts are from the 15th century, a turbulent and unique epoch in Czech history. Until now, there have been no editions of these texts, except for smaller excerpts in the case of Michael Polonus and a partial edition of the Latin *Czech Sunday Postil* (Odstrčilík 2015).

In both cases, I chose to expand abbreviations, which occurred almost exclusively in the Latin parts. Regarding transcription rules, in the first text, I used semi-diplomatic transcription for Latin, while in the second, I reused my previous transcription with regularised lower- and uppercase letters and punctuation. In both cases, I used transliteration (i.e. diplomatic transcription) for Czech medieval words.

Each of these decisions has advantages and presents certain issues. Expanding abbreviations may lead to a loss of precision in ATR, but it may also have other negative consequences. In the realm of multi-lingual medieval texts, some abbreviations can have a diamorphic character (Wright 2011; ter Horst and Stam 2017). This means that they are inherently ambiguous and can be read in multiple ways as belonging to different languages, e.g. *x̂ane* can be read as Latin *vocative* or Czech *křesťane*. Both words mean the same, i.e. Christian (Odstrčilík 2023). Fortunately, the number of these abbreviations in the corpus under study is limited.

Michael Polonus's *Sermons* about saints

Few periods in Czech history are as dynamic as the 15th century. During this time, Czech lands became the birthplace of the first successful heterodoxy within the Catholic Church in Latin Europe that authorities were unable to fully suppress. Following the execution of Jan Hus, a popular priest, university master, and critic of contemporary Church vices, at Constance in 1415, a new religious group emerged: the Hussites. After years of conflict, the moderate faction – known as the Utraquists – joined forces with the Catholic side, defeated the most radical Hussites, and achieved recognition by the Catholic Church at the Council of Basel (1433–1435). The Utraquists formally became part of the Church, but largely remained separate and later faced renewed persecution by Catholic rulers. Nevertheless, Bohemia became unique in Western Europe as a place that offered some, albeit limited freedom of choice between different forms of Christianity until the Thirty Years' War, though not everywhere and not equally (David 2003).

Michael Polonus first came to Prague in the 1430s for his studies. His subsequent journeys took him briefly home to Poland and then back to Bohemia. We do not know much about his activities in 1450s and 1460s, but it seems that after becoming a priest, he was active in the Bohemian countryside. In the 1470s, we find him again in Prague as one of the most prominent Utraquist preachers of the Bohemian capital. After he stood up against the oppression of Utraquists in 1480, he was imprisoned and tortured. He died on 2nd November 1480 and became a Utraquist martyr (Bartoš 1955).

There are three sermon collections attributed to Michael Polonus (Bartoš 1955: 72; Spunar 1978, 243–44):

1. *Sermones de sanctis* (*Sermons about saints*), preserved in Prague, National Library, XI F 3;
2. *Sermones de tempore* (*Temporal sermons*), preserved in Prague, National Museum, I F 14;
3. *Sermones de festis principialioribus* (*Sermons on the principal feasts*), preserved in Prague, Metropolitan Chapter Library, F 16.

All the sermons are highly bilingual, featuring frequent and seamless code-switching between the main Latin language and Czech language, as is readily apparent in the following excerpts, for example:

dicit 'venite post me'. Tehdy chcze Cristus wssudy mieti naprzed, non tantum in ecclesia	'he says "follow me". And so (= Tehdy) Christ wants to have a prerogative everywhere, not only in the Church' (*Sermons about saints*, Prague, National Library, XI F 3, fol. 11v).
Modo, mi cristiani, pomniemez, quod hoc dicit Paulus	Now, my Christians, let us remember, that Paulus says this (*Sermons about saints*, Prague, National Library, XI F 3, fol. 11v)
fuguiamus temerarie a drzie cogittare, loqui, aneb tazati sie de ista trinitate sancta	let us not inconsiderately and imprudently think, speak or question the holy trinity (*Sermons on the principal feasts*, Prague, Metropolitan Chapter Library, F 16, fol. 4r)

So far, only excerpts from these extensive sermon collections have been published (Šimek 1936; Hanuš 1861: 111–15; 1863: 31–39). I have chosen *Sermons about saints*, preserved in the manuscript Prague, National Library, XI F 3. It consists of 479 pages. To prepare the ground truth for the Latin language, I decided to silently expand all abbreviations, apply modern word separation, and use only "s" for both forms of the characters "ſ"/"s", but refrained from other normalisations, including capital letters, diacritics and normalising "u"/"v". For the Czech language, I took a very similar approach, using palaeographical transcriptions (transliterations), i.e. silently expanding the very limited number of abbreviations in the Czech language, substituting "ſ" with "s" and applying modern word separation. The resulting transcription rules for both languages are thus very similar.

I used a transcription loop as my strategy (Figure 29), training the first model after preparing 38 pages. This already resulted in a promising CER of 10.7% on a validation set.

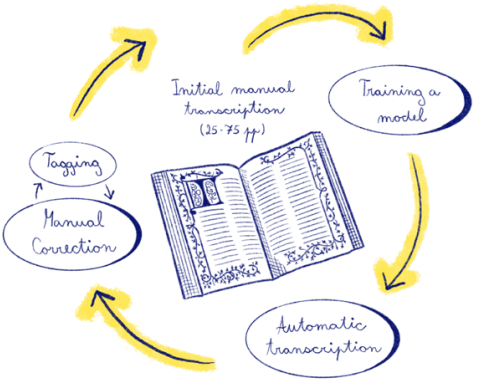


Figure 29: Transcription loop, illustrated by Marie Machatová, CC BY NC 4.0

I progressed gradually from the beginning of the manuscript, repeatedly training, correcting, and manually tagging Czech passages with the "foreign" tag to annotate a change in language.

When training models, we often don't want to waste ground truth pages on the validation sets. In the case of the transcription loop, this can result in a validation set that is not fully representative of the entire document. To address this and to provide better insight into model quality at different stages of transcription, I decided – after reaching the end of the document – to experiment with various ground truth sets and some very large validation sets (Figure 30).

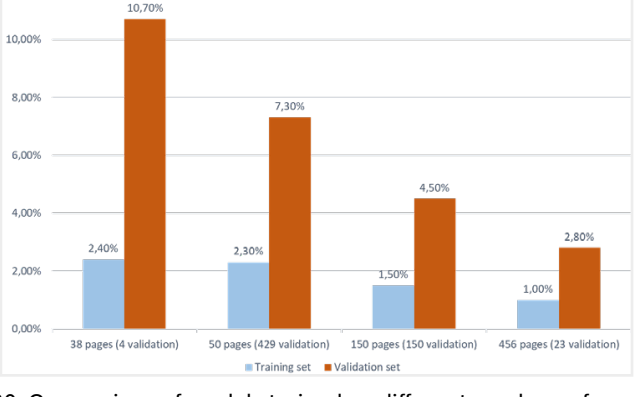
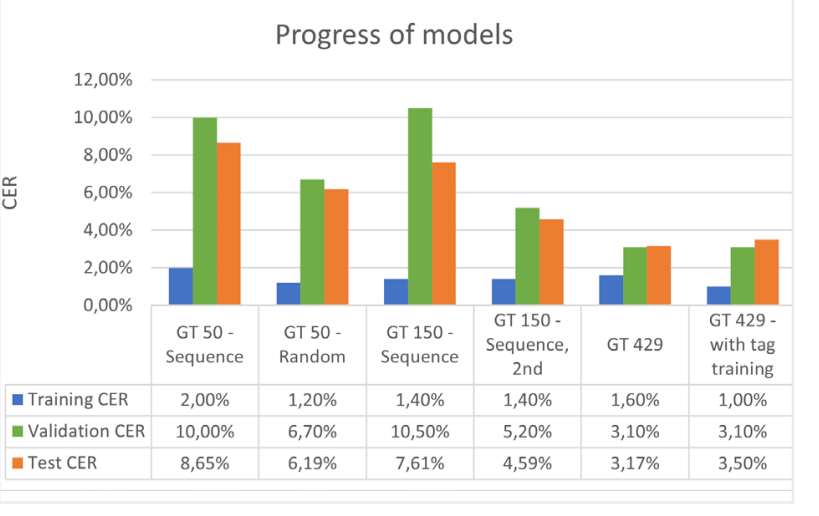


Figure 30: Comparison of models trained on different numbers of pages, image by the author, CC BY 4.0

Out of caution, and in line with machine learning best practices, I added a separate test set, reserved exclusively for post-training evaluation. From the 479 pages of the manuscript, I randomly selected 25 pages for validation and another 25 pages as a separate test set. This is something I have recently learned because the validation set in Transkribus does not necessarily provide the most relevant values. I also decided to test my theory that even within a manuscript written by one person, the training data should be selected from different parts of the manuscript. This is an experience we made at the Winter School of HTR of Medieval Documents 2023. I chose to train models with the ground truth of 50 pages, 150 pages and 429 pages, i.e. the whole remaining document (Figure 31).



	GT 50 - Sequence	GT 50 - Random	GT 150 - Sequence	GT 150 - Sequence, 2nd	GT 429	GT 429 - with tag training
Training CER	2,00%	1,20%	1,40%	1,40%	1,60%	1,00%
Validation CER	10,00%	6,70%	10,50%	5,20%	3,10%	3,10%
Test CER	8,65%	6,19%	7,61%	4,59%	3,17%	3,50%

Figure 31: Comparison of models trained on different numbers of pages with ground truth selected either sequentially or randomly, image by the author, CC BY 4.0

There are two important outliers among these models: GT 50 with random selection of ground truth performed significantly better than GT 50 with the sequence of pages taken from the beginning of the manuscript. This suggests that, at least for sermon collections, it is reasonable to take samples from different parts of the manuscript. This is likely because of the way PyLaia trains simple language models within Transkribus. In sermon collections, it is likely that different topics will be

addressed in different sermons, thus random sampling can improve the language model and consequently CER values.

GT 150 performed unexpectedly poorly, especially on the validation set. Re-training with the same values (GT 150, 2nd) improved the results substantially. It is unclear why the first model yielded such poor results, but it is likely that the random initialisation of the training underperformed due to accident or coincidence. As is pointed out in 4.4 Training, it is useful to repeat the training process for the same data to produce several models in order to avoid such effects.

One of the leading questions is if there are any differences in the quality of the transcription for Latin and Czech words. This can be computed only manually. I extracted all Czech words from the test set. There are 4,779 words, of which around 790 are Czech words, thus representing 16.53% of the whole corpus. Then I extracted all Czech words from the transcription of the test set provided by the best model (GT 429 with a validation CER of 3.10% and CER of 3.17%). Finally, I used CERberus (Haverals 2023) to compute CER values.

At default settings of CERberus, the CER is 4.24%, on the most permissive settings (i.e. ignoring punctuation, case, whitespace, numbers, and newlines) the CER is 2.82%. While the results are thus worse than the CER for both Latin and Czech, they are very acceptable, showing that bilingual texts of this sort do not pose substantial problems once enough GT can be provided as training data.

For the linguistic analysis, the necessary step in my work is tagging Czech passages within the Latin text. During my transcription loop in Transkribus, I manually added a "foreign" tag to all Czech words. This allowed me to use the XLSX export function of Transkribus to obtain an overview of all Czech words and their context. I also exported the document as TEI-XML, which I used together with AntConc and Voyant Tools for further analysis of code-switching.

Since annotation training is available for PyLaia models in Transkribus, I tested this functionality to reduce the laborious process of manually tagging Czech passages with the "foreign" tag. This is particularly relevant because lemmatisation tools for medieval Latin are difficult to

access, and are unavailable for medieval Czech altogether, making automated language identification in post-processing challenging.

Typically, PyLaia's automatic annotation is used to tag abbreviations and provide their expansions, but it can be applied to any type of textual tag. It is important to emphasise that there is no visual distinction between Latin and Czech words in the manuscript: both languages are written in the same script.

I trained a model based on 429 pages with tag training enabled for "foreign" and then used it to read and tag the test set. Out of 186 Czech passages with 790 words in total, only 12 have been identified. There were no false positives, i.e. Latin passages identified as Czech.

The main issue is that the model sometimes skips words in the middle of the lines (Figure 32).

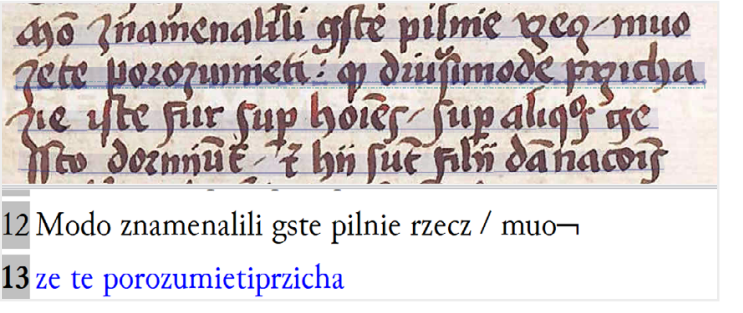


Figure 32: Skipped words in the automatic transcription of Prague, National Library, XI F 3, fol. 23v, manuscript image by National Library in Prague, CC BY NC SA 4.0

After inspection, it seems that the Latin passages are missing exactly between two Czech passages on the same line. Out of these, the model tags only the first one as "foreign". The inspection of the page XML source also confirms that the text is in fact missing and that it is not just rendering a bug of the Transkribus GUI.

This happens within Czech passages, as well. In this case, "slychate zdawna" is completely missing (Figure 33).

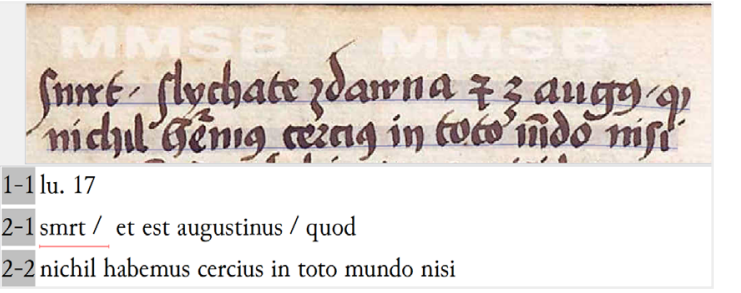


Figure 33: Skipped words in the automatic transcription of Prague, National Library, XI F 3, fol. 23v, manuscript image by National Library in Prague, CC BY NC SA 4.0

Nevertheless, even if the model did not delete passages, the results would be still too erroneous for further analysis. In summary, for the sermons of Michael Polonus, the automated reading proved to be successful and of great help while the automated tagging, at the time of writing this study, is unusable to identify parts in Czech.

In the Transkribus framework, the training of annotation is combined with text recognition. This means that visual elements are the main driver to distinguish the tags. Consequently, this works fine for elements which are italicised or for which other script systems are used (e.g. Greek characters). During text recognition training, certain language-specific patterns are picked up. This can be compared to a very simple language model, in the sense of a representation of how language patterns are applied. Accordingly, language switches are occasionally detected and annotated correctly. This mostly concerns words the recognition model has seen annotated in the process of training.

Latin Translation of Jan Hus's Czech *Sunday Postil*

Jan Hus is the most famous Czech preacher and, generally, historical figure in Czech history. He became known for his Czech-language preaching against the vices of the Church in the first decades of the 15th century in Prague. This, together with his treatises on various subjects (like simony, heretical books, the Church) and disobedience towards the Pope ultimately sealed his fate: He was condemned as a heretic at the Council of Constance and burnt at the stake in 1415.

Jan Hus was the author of many works, both in Latin and in Czech. Despite being a famous Czech preacher, he published the vast majority of his sermons in Latin, as was common in his time. *Česká nedělní postila* (The *Czech Sunday Postil*) is the only complete sermon collection he composed in the Czech language. One of its purposes was to communicate with his audience after a papal interdict forced him to leave Prague in 1412. It is thus a fine example of "preaching in the armchair", as Michel Zink put it (Zink 1976: 477–78). Its 59 sermons on biblical readings throughout the year are full of personal passages in which Hus defends himself against accusations of heresy and criticises the Church, even naming specific adversaries.

Probably more than 20 years later, a Latin translation of this work was created. Its author and other circumstances remain unknown, and it is preserved in only one manuscript: Mk 91, dated ca. 1448–1460, held at the Moravian Library in Brno (Dokoupil 1958: 153–55). The Latin translation is, on the one hand, extremely literal (almost word-for-word), but on the other, it makes precise cuts and small changes that radically modify the resulting work (Odstrčilík 2019).

I previously published an edition of the first 20 sermons of this text (Odstrčilík 2015) and currently have a provisional transcription of the entire manuscript. My goal is to publish the whole text in a physical form as a printed book. For this purpose, I follow traditional transcription guidelines for Latin medieval texts of Czech origin, as defined by Bohumil Ryba in the 1950s (Ryba 2024), with very few exceptions.

As with Michael Polonus, I silently expand abbreviations and normalise the separation of words. In the case of Mk 91, however, I also normalise u/v towards its linguistic use, add modern punctuation, capitalisation, and even quotation marks. I also decided to follow the example of editors of other Latin texts containing Czech words (like Florianová et al. 2006) and transcribed Czech words in capital letters, even adding a dash (–) between Latin and Czech words if they were used as glosses. All these decisions were made in 2013, before I even knew about Transkribus and ATR. The resulting text looks like following (Odstrčilík 2015: 107) (Figure 34).

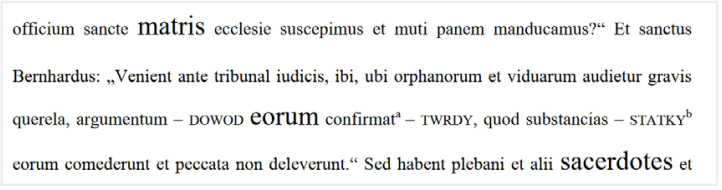

officium sancte matris ecclesie suscepimus et muti panem manducamus?“ Et sanctus Bernhardus: „Venient ante tribunal iudicis, ibi, ubi orphanorum et viduarum audietur gravis querela, argumentum – DOWOD eorum confirmat[a] – TWRDY, quod substancias – STATKY[b] eorum comederunt et peccata non deleverunt.“ Sed habent plebani et alii sacerdotes et

Figure 34: Example from the edition of the *Czech Sunday Postil*, image by the author, CC BY 4.0

What, then, could be the use of an ITE in this case when the text has been already manually transcribed? First and foremost, I found it useful for manual correction and improvement of my transcription thanks to the visual line-to-line alignment with the manuscript. Even more useful for me is the text search functionality with image overlay, which allows me to quickly check whether I read specific Latin abbreviations correctly and consistently (Figure 35).

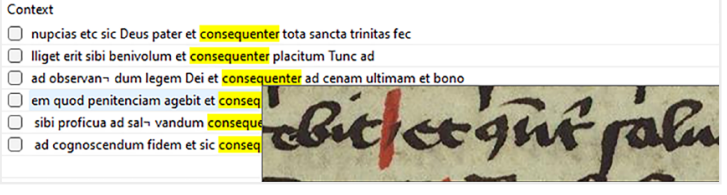


Figure 35: Search function in Transkribus displaying manuscript snippets, image by the author, CC BY 4.0

Could we use these transcriptions for training ATR, as well? One possible problem is my traditional transcription approach, which adds characters to the transcription (such as the dash “–” or punctuation) that are not present in the manuscript, and even completely changes the formatting of some passages (e.g. capitalisation of Czech words). This goes against all best-practice advice for ATR. Nevertheless, for the sake of experiment, there is no reason not to try such an approach.

Training on 333 pages achieved a CER of 7.41% on the validation set of 17 pages. This is better than expected. As said, I added many non-visible characters (punctuation, quotation marks etc.) and modified others (capitalisation), so it is also worth inspecting how these do affect the results. For this, I used CERberus to calculate CER values independently from the Transkribus CER calculator. Applying the most permissive

evaluation parameters, i.e. ignoring case, punctuation, whitespace, numbers and new lines, we achieve 4.65% on the same validation set (Figure 36).

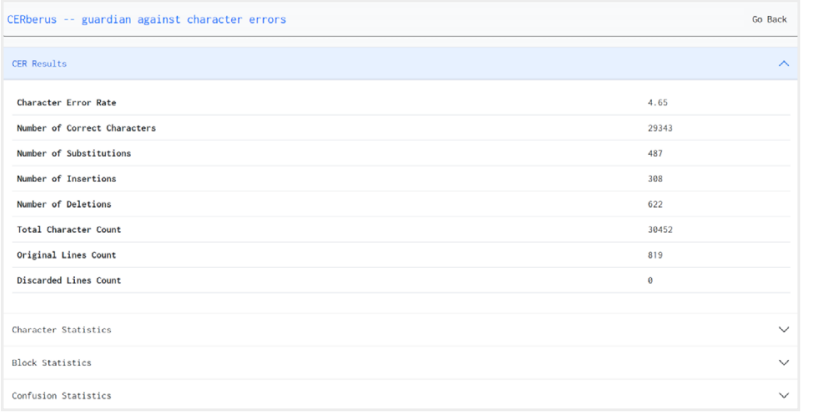


Figure 36: CER values as calculated in CERberus , image by the author, CC BY 4.0

There is one more aspect to this: When Transkribus uses a validation set, it reuses its layout. This is a correct approach because it tests ATR, not the layout recognition model. This means that in practice, without training any specific layout model, we will encounter additional issues. For instance, the manuscript contains Czech interlinear glosses in the Latin biblical readings at the beginning of each sermon (fol. 327r) (Figure 37).

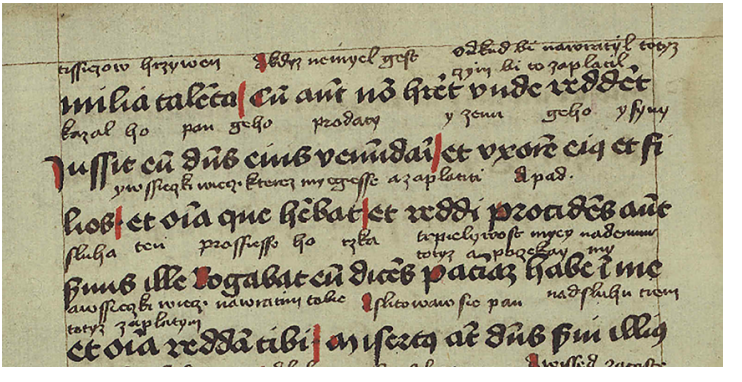


Figure 37: Brno, Moravian Library, Mk 91, fol. 326, manuscript image by Brno, Moravian Library CC BY NC SA 4.0

The default Transkribus Layout Analysis tool recognises those lines incorrectly (Figure 38).

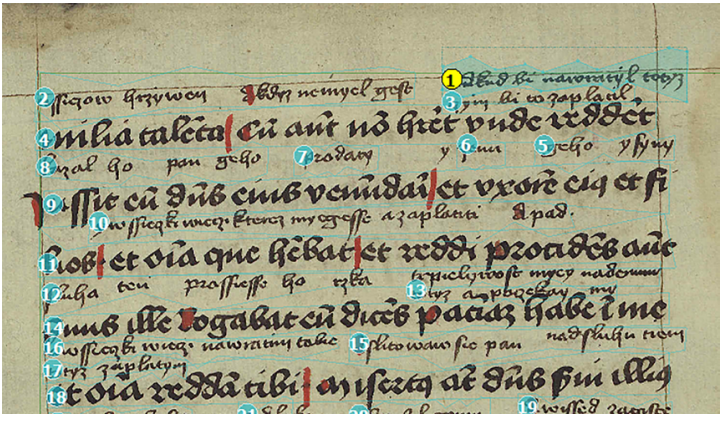


Figure 38: Regions and lines recognised in Brno, Moravian Library, Mk 91, fol. 326, manuscript image by Brno, Moravian Library CC BY NC SA 4.0

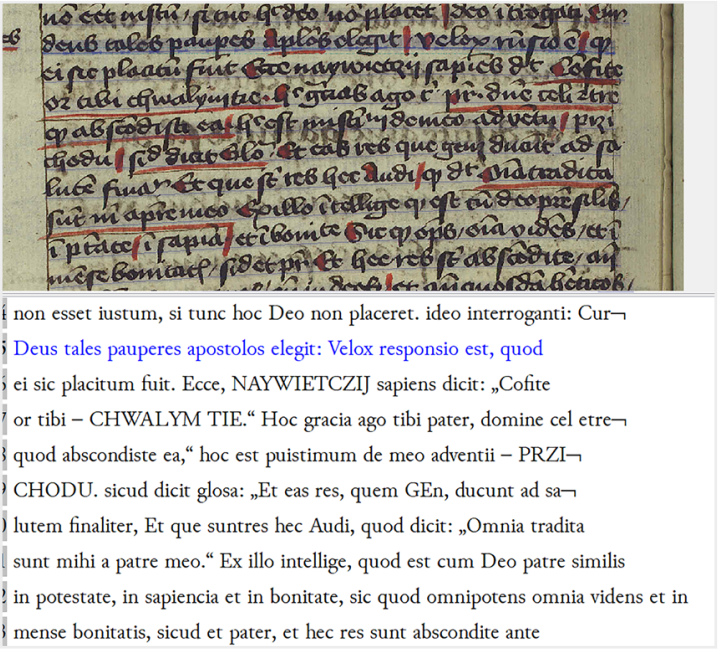
This causes an extremely high error rate of 37.14% (!), while on the preceding folio (fol. 326r), it is just CER 5.84%. Neither of these pages was used for training or validation. This shows how immensely the line order will affect the quantified results, while the text itself is still very legible.

If we now return to the validation set of 17 pages, delete all recognised regions, and then run the model on it, we achieve a CER of 8.79% in comparison to the initial 7.41% without considering the recognition of lines and text regions. With case-insensitivity, the CER is 8.12%. Part of the validation set are four pages with a complicated layout, including interlinear glosses (albeit one contains just one line). On them, the CER reaches a highly problematic CER of 16.42%, while on the remaining part, the CER is only 6.91%.

As in the case of Michael Polonus's work, I extracted Czech words, this time from the validation set, and used CERberus to manually compute CER values. There are 321 Czech passages on 17 pages with 562 words in total. In comparison to Michael Polonus's work, there are thus fewer Czech words and the Czech insertions are shorter. With default settings, I obtained a 9.7% CER on Czech words and with the most permissive

settings (i.e. ignoring punctuation, case, whitespace, number, and newlines), it is 6.19% CER. These are worse numbers than for Michael Polonus, but they are comparable with the general results for Mk 91.

The most intriguing part is how the model deals with Czech words. In the Mk 91 model, it learns to transcribe them with capital letters, and it does so at much higher precision than the model for Michael Polonus with trained annotations, using the foreign tag (Figure 39).

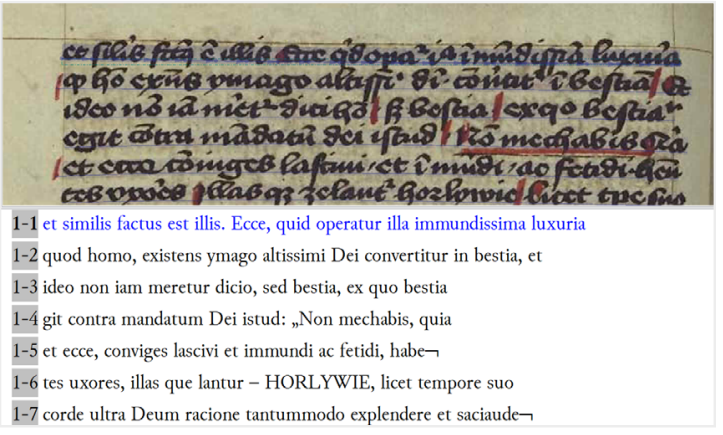


non esset iustum, si tunc hoc Deo non placeret. ideo interroganti: Cur¬
Deus tales pauperes apostolos elegit: Velox responsio est, quod
ei sic placitum fuit. Ecce, NAYWIETCZIJ sapiens dicit: „Cofite
or tibi – CHWALYM TIE.“ Hoc gracia ago tibi pater, domine cel etre¬
quod abscondiste ea,“ hoc est puistimum de meo adventii – PRZI¬
CHODU. sicud dicit glosa: „Et eas res, quem GEn, ducunt ad sa¬
lutem finaliter, Et que suntres hec Audi, quod dicit: „Omnia tradita
sunt mihi a patre meo.“ Ex illo intellige, quod est cum Deo patre similis
in potestate, in sapiencia et in bonitate, sic quod omnipotens omnia videns et in
mense bonitatis, sicud et pater, et hec res sunt abscondite ante

Figure 39: Brno, Moravian Library, Mk 91, fol. 175v, transcription by the author, manuscript image by Brno, Moravian Library, CC BY NC SA 4.0

Most of the Czech words were capitalised and there were just a few wrong capitalisations of Latin words. There are still false positives (i.e. falsely capitalised letters) but in general, this approach renders much more useful results than the previous one with training tags in PyLaia.

The manuscript itself is also large with 346 folios (i.e. 692 pages). *Czech Sunday Postil* covers only a little bit more than half of the manuscript (146r–345r). The other works are written in the same or a similar hand. The model thus performs relatively well on them as well, including recognition of Czech words (fol. 51v) (Figure 40).



1-1 et similis factus est illis. Ecce, quid operatur illa immundissima luxuria
1-2 quod homo, existens ymago altissimi Dei convertitur in bestia, et
1-3 ideo non iam meretur dicio, sed bestia, ex quo bestia
1-4 git contra mandatum Dei istud: „Non mechabis, quia
1-5 et ecce, conviges lascivi et immundi ac fetidi, habe¬
1-6 tes uxores, illas que lantur – HORLYWIE, licet tempore suo
1-7 corde ultra Deum racione tantummodo explendere et saciaude¬

Figure 40: Brno, Moravian Library, Mk 91, fol. 51v, transcription by the author, manuscript image by Brno, Moravian Library, CC BY NC SA 4.0

Conclusion

For both analysed cases, it has been shown that unbalanced bilingual models featuring frequent code-switching don't present a major challenge for training ATR models. For both languages, Latin and Czech with a differently dense representation, the results were usable. But, as was to be expected, in both cases, the general CER was better than the one just for Czech words:

Michael Polonus's *Sermons about saints*:

- ground truth: 429 pages
- validation (25 pages) CER: 3.10%
- test (25 pages) CER: 3.17%
- Czech words only test (25 pages) CER 4.24% (CERberus default)
- Czech words only test (25 pages) CER 2.82% (CERberus most permissive)

Mk 91 – *Czech Sunday Postil*

- ground truth: 333 pages
- validation (17 pages) CER 7.41% (Transkribus values)
- validation (17 pages) CER 4.65% (CERberus most permissive)
- Czech words only validation (17 pages) CER 9.7% (CERberus default)
- Czech words only validation (17 pages) CER 6.19% (CERberus most permissive)

Utilising Transkribus for automatic identification of Czech passages yielded unexpected results – both positive and negative. While training tags in PyLaia resulted in a complete failure, even skipping entire passages for no reason, using capital letters to indicate Czech words worked surprisingly well.

Finally, multiple models trained on Michael Polonus's *Sermons about saints* show the viability of using Transkribus for late medieval manuscripts using an iterative process for adding transcriptions. As little as 50 pages resulted in CER values well under 10%. The experiment suggests that it is recommended to sample from different parts of the document from the beginning rather than transcribe consecutively page by page.

CONCLUSION

6 Conclusions and What's Next in the Field of Text Recognition

As we approach the end of this book, we would like to draw a conclusion and consider what lies ahead for the dynamic field of automated text recognition. While we have presented ATR as a mature technology, capable of serving the daily needs of humanities scholars, we have also emphasised the importance of careful deliberation and well-considered approaches to its application. ATR is not about recognising "the text" in a singular, definitive manner; rather, it is about creating many possible digital representations of textual artifacts, each shaped by specific needs, contexts, and research interests. Thus, our goal has been to facilitate these deliberations by equipping you with the theoretical knowledge and practical experience needed to navigate the complexities of ATR in a landscape of ongoing technological advancement.

After introducing the topic in the first chapter, we delved into the details of text recognition and its foundations in machine learning in chapter two. Here, we provided context for the challenges of ATR by offering a brief history of the field and presenting a bird's-eye overview of the recognition process. We then explored the technical background of machine learning, explaining the fundamental concepts and technologies that underpin ATR.

In chapter three, we turned to a question frequently posed by interested users: "What is the best tool?" While there is no definitive or universal answer to this question, we proposed a framework to help readers evaluate tools based on their specific needs and requirements. To complement this framework, we also provided an overview of the most widely used implementations of ATR and highlighted the three leading integrated transcription environments (ITEs) currently available: Transkribus, eScriptorium, and OCR4All.

The fourth chapter shifted focus to the core concepts of ATR that we consider to be of lasting relevance. These concepts are designed to help users critically reflect on the process of creating digital representations of textual artifacts through machine learning approaches. They aim to empower users to select and use ATR tools in an informed manner

while remaining attuned to their unique needs, goals, and the broader implications of their work.

After these theoretical discussions, we wanted to show how ATR can be used across very different domains, disciplines, and materials: Achim Rabus showed how ATR-models can not only recognise pixels on the page but that linguistic patterns trained into a 'smart model' can be employed to conduct a whole editorial pipeline, greatly simplifying the task of text analysis. Tim Geelhaar detailed the work necessary to train general ATR models capable of recognising text from various manuscripts on the exemplar of his *Caroline Minuscule Model* and the challenges this poses. Michael Schonhardt discussed how ATR can and must be integrated in broader editorial workflows by introducing 'Burchards Dekret Digital' and the models used there. And Jan Odstrčilík showed both the challenges and potential of applying bilingual models to multi-lingual sources. Together, these case studies showed the vast landscape of possible applications of ATR in the humanities and that there cannot be a one-fit-for-all solution for all problems.

Next frontiers

Automated text recognition in its current stage represents a remarkable achievement, as it has largely solved the technical challenge of recognising text in historical documents. However, the more we use this technology, the clearer it becomes that text recognition alone is only part of the equation. While 'text' conveys an important part of the information embedded in a document, it is insufficient on its own to fully capture the source.

Historical documents are not mere vessels for text; they are complex data structures, arranged to encode culturally and individually embedded knowledge, which is often deeply entangled with the document's layout and visual structure. Consider, for example, medieval glossaries or early modern account books that cannot be fully understood without recognising and interpreting their spatial organisation. Even with the best ATR models, generating a coherent transcription of such sources is impossible without a meaningful recognition and annotation of its layout features. Titles, marginalia, page numbering, and other visual segmentations do not merely structure the document – they represent deliberate choices that preserve historical practices (Gabay et al. 2021).

Thus, in our opinion, ATR's next frontier lies in advancing the technological and conceptual dimension of document understanding and layout recognition. On a technological level, methods must be further enhanced, as current models have made progress but leave considerable room for improvement. On a conceptual level, the scholarly community must reflect on how we understand and engage with this dimension of the sources and develop shared vocabularies, standards, and datasets to support the training of models capable of document understanding. Only by shifting our focus from merely recognising text to comprehending the structure, context, and intent of historical documents can we unlock their full potential.

The role of large and small language models in these developments is currently debated. First examples hint at the usefulness for post-correction, especially for English (Kanerva 2025), while many point towards the pitfalls of the approach (Boros 2024, Kanerva 2025, Greif et al. 2025). Hyper-correction, wrongful adaptation, bias, and plain hallucination show that large language models, especially, are not reliable enough to be part of unsupervised pipelines and workflows.

Accordingly, we don't address the use of LLMs in ATR in this book. More experience, and even more importantly, better results are needed for LLMs to become part of automated processing, though we admit that transformer-based LLMs already play a small part in automated text recognition, as TrOCR approaches leverage BERT models for the decoding (Li et al. 2021).

While these perspectives are as exciting as the current state of ATR, it is important to reflect on the ethical and political implications that accompany this technology. Like all machine learning applications, text recognition is deeply rooted in data production – both in the creation of models and the selection of training data. These foundational processes inevitably shape the ways research in the humanities can and will be conducted in the future, which is especially significant as ATR systems are already advancing to the point where they can perform large-scale text recognition without human intervention and with good success. Especially for highly represented texts, a CER below 5% is attainable (4.5 Evaluation).

In this regard, ethical considerations must address biases embedded in training data, fairness in algorithmic outcomes, and the broader societal impacts of automating systems of knowledge. While biases in ATR may not entail the same immediate consequences as in fields like medicine or law, they can still have profound implications. For instance, consider the historical focus on male figures in archival preparation: Letters or writings of a prominent historical man are often better preserved, catalogued, and prepared for recognition than those of female or non-binary contemporaries. This disparity naturally extends to ATR models and datasets, which are likely based on a disproportionate number of male hands. Consequently, these models may become more adept at recognising male handwriting while struggling with female hands – a bias that mirrors and reinforces the historical underrepresentation of women in archival records.

This example can be extended to other forms of bias, including those related to race, language, ethnicity, and other minority groups. If ATR becomes a primary tool for archival work, such biases risk perpetuating a cycle where – compared to majorities – marginalised voices become more difficult to find, read, and to be included in the digital record of the past. To prevent this, it is important to reflect on potential biases and their consequences as part of ATR model design. This is the responsibility of the scholarly community and must become a critical component of research methodology discussions about the ethical use of technology. The CARE principles (Carroll et al. 2020) serve as a fitting indicator for the whole data life cycle, here also applicable to ATR processes.

Moreover, the ethical considerations surrounding data production extend beyond its creation to how it is reused, acknowledged, and credited to its providers – whether institutions or individuals. Ensuring transparency and fairness in these processes is essential to foster trust and collaboration within the research community. The reliance on the efforts of both paid contributors and volunteers further raises questions about equitable recognition and compensation. Thus, the more ATR becomes a fundamental part of humanities research, ethical frameworks must be established to value this often-invisible work and ensure that contributors are appropriately credited for their efforts.

In addition to these considerations, the ecological footprint of machine learning cannot be ignored. The energy demands of training ATR models and maintaining the computational infrastructures required for their use have real environmental consequences, as they require significant energy, which is not always generated renewably (Baillot 2023). To address these challenges, collaboration among institutions and researchers ensures resources are used responsibly in developing and applying ATR models. One key practice is reusing existing models wherever possible rather than training new models repeatedly for similar sources. Sharing trained models publicly can significantly reduce the need for duplicative training efforts, saving both computational resources and time (Chagué and Clérice 2022).

When creating new models, another good practice is to minimise unnecessary iterations during the training process. For instance, instead of repeatedly transcribing small increments of text (e.g. 2,000 words at a time) and re-training the model after each addition, a more efficient strategy is to plan training in larger, more focused stages until the desired quality is achieved. This approach conserves computational resources and helps limit the environmental impact of model training.

Political implications emerge as well, as the humanities depend on robust infrastructures to enable open and inclusive research. Open data are increasingly seen as a necessity, with cultural heritage institutions, such as galleries, libraries, archives, and museums (GLAM) playing a vital role in providing accessible resources, including image collections. In this context, the humanities serve as guardians of hermeneutical interpretation, ensuring that the rich complexity of source materials and its contextualisation are not lost in the drive for efficiency and unfounded assumptions. Interpretation remains a key part of working with historical and cultural materials, and embedding interpretive, academically sound knowledge in machine learning models is essential for preserving the depth and context of humanities research.

Generally, alongside the much more popular FAIR principles, CARE principles (Carol et al. 2020) also need to be addressed. Considering the impact of data and document use for the greater good is one of the guiding principles to explicitly consider communities that benefit but might also be harmed by the use of data and/or technology (4.8 Ethics).

In this sense, minority languages would be a great and worthwhile target for text recognition models, minimising colonial harm.

Finally, we strive for practitioners of ATR to make deliberate and informed choices when applying ATR to "their" documents. For this, a basic understanding of the technology is key, besides information about best practices and successful applications. We are certain that GLAM and scholarly practices will change based on the availability of ATR and the possibility to process large amounts of images to reveal texts of the past. At the same time, this requires an emphasis on scanning documents at scale, allowing straightforward access to the content (e.g. through well-documented APIs) and a focus on reusability with low legal hurdles.

We began this book with the premise that text recognition, while often seen as an auxiliary science, is in fact a fundamental practice of human life and the humanities. Although for many, ATR might appear to be a practical tool – a means to quickly unlock the content of historical records, it is much more a method in its own right that requires knowledge about its application. Besides identifying ATR as a new auxiliary science form, our aim has been to show that ATR is much more than a neutral, technical solution: It is a complex and transformative technology that not only assists in interpreting the past but also challenges us to rethink the very nature of the methods we represent, understand, and engage with historical artifacts and their epistemology in the digital age. And while these questions open up entirely new mazes to enter, we hope this book serves as both a guide and an invitation to continue your explorations.

7 Glossary

AI (Artificial intelligence): A broad field of computer science dedicated to creating systems capable of performing tasks that normally require human intelligence. This includes problem-solving, learning, perception, and language understanding. Machine learning is a prominent subfield of AI, and Automated Text Recognition (ATR) is a practical application of AI.

Algorithm: A finite sequence of well-defined rules or instructions that a computer follows to solve a specific problem or perform a task. In ATR, an algorithm might define the steps for segmenting a page, detecting text lines, and classifying characters within an image.

Algorithmic literacy: The ability to understand, critically evaluate, and make informed decisions about algorithms and their applications. It involves not only knowing how algorithms work but also recognizing their potential biases and societal impacts.

Allograph: A distinct, graphically different variant of a grapheme. For example, the historical long ‚s' (ſ) and the modern round ‚s' are allographs of the same grapheme ‚s'. A transcription that distinguishes between these variants is called graphetic or allographic.

ALTO (Analyzed layout and text object) XML: An XML format used to describe the physical layout of a page and its recognised text content. It standardizes the storage of ATR results by encoding coordinates for regions, text lines, and words, as well as the transcribed text itself. It's frequently used in digital library and archival contexts.

API (Application programming interface): A set of definitions, protocols, and tools that allows different software applications to communicate and exchange data with each other. For example, an ATR service might offer an API that lets developers send an image and receive the recognised text programmatically.

Batch-learning: A machine learning method where the model is trained on a batch of the dataset at once. The model's parameters are updated after processing all training examples of a batch. This is also known as offline-learning and is the standard approach for training ATR models with a fixed ground truth dataset.

BERT: A Transformer-based language model that learns deep word context.

Brevigraph: A symbol or a special form of a letter used in historical manuscripts and early prints to represent an abbreviation. For instance, a line over a vowel could signify a following ‚m' or ‚n'.

CARE principles: A framework for Indigenous data governance, ensuring data is used to empower Indigenous communities. The acronym stands for Collective Benefit, Authority to Control, Responsibility, and Ethics. These principles are crucial when working with culturally sensitive heritage data in GLAM institutions.

CER (Character error rate): A primary metric for measuring the accuracy of ATR results. It calculates the percentage of incorrectly recognised characters by comparing the model's output to the ground truth.

CLI (Command-line interface): A text-based user interface used to interact with a computer's operating system or software. Users execute commands by typing text strings instead of clicking of graphical elements.

Codepoint: A unique numerical value that represents a specific character in a digital character set. In the Unicode standard, for example, the codepoint U+0041 represents the character ‚A'.

CNN (Convolutional neural network): A class of deep learning models particularly well-suited for processing grid-like data, such as images. CNNs use special layers called convolutional layers that apply filters (kernels) across the input image to automatically learn a hierarchy of features, from simple edges to complex shapes. They form the backbone of many modern ATR systems for feature extraction.

Deep learning: A subfield of machine learning based on neural networks with multiple hidden layers (i.e., "deep" architectures). These models can learn complex patterns and representations from vast amounts of data, leading to significant breakthroughs in fields like computer vision and ATR.

FAIR principles: A set of guiding principles to make research data more valuable and reusable. The acronym stands for Findable, Accessible, Interoperable, and Reusable. Applying FAIR principles to ATR means making models, ground truth data, and results discoverable and usable by others.

Feature vector: A numerical representation of an object's characteristics (features) used for processing in a machine learning model. In ATR, a feature vector could represent a vertical slice of a text line image, encoding properties like pixel distributions and gradients that the model uses to identify a character.

Fine-tuning: The process of adapting a pre-trained model to a new, more specific task. A model trained on a massive dataset (e.g., millions of pages of modern print) is used as a starting point and then retrained with a smaller, specialised dataset (e.g., 18th-century letters). This transfer learning technique significantly reduces the amount of ground truth needed for a new task.

GLAM (Galleries, libraries, archives, museums): An acronym for cultural heritage institutions that preserve and provide access to historical artifacts, documents, and data. These institutions are the primary source of material for ATR projects in the humanities.

Glyph: The concrete visual representation or specific shape of a character. While a grapheme is the abstract concept of a character and an allograph is a recognisable variant, a glyph is the actual rendered mark on a page.

GPU (Graphics processing unit): A specialised processor designed for parallel computation, making it highly efficient at handling the matrix and vector operations that are fundamental to training deep learning models. Using GPUs drastically accelerates the training time of ATR models compared to using traditional CPUs.

Grapheme: The smallest abstract unit of a writing system that distinguishes one word from another. For example, the letters ‘a’, ‘b’, and ‘c’ in English are graphemes. It‘s the conceptual unit, distinct from its visual representations (allographs and glyphs).

Graphetic: A transcription approach that focuses on capturing the specific visual forms (allographs) of characters rather than normalizing them. For instance, a graphetic transcription would distinguish between different forms of the letter ‚r‘. It is also known as an allographic transcription.

Graphemic: A transcription approach that normalizes different character forms (allographs) into their standard abstract units (graphemes), thereby preserving the original spelling of the text but not the specific shapes of the letters. It is also known as a graphematic transcription.

Ground truth: The high-quality, verified reference data used for training and evaluating a supervised machine learning model. In ATR, the ground truth consists of pairs of images and their accurate, manually created transcriptions. The quality of the ground truth is a critical factor for model performance.

GUI (Graphical user interface): A type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators, as opposed to a text-based CLI. Tools like eScriptorium or Transkribus provide a GUI for the ATR workflow.

Hyper-parameter: A configuration variable that is set before the training process begins and controls how the model learns. Examples include the learning rate, batch size, and the number of layers in a neural network. The selection of hyper-parameters significantly influences training speed and model performance.

IIIF (International image interoperability framework): A set of open standards for delivering high-quality, attributed digital images online. IIIF allows users to view, compare, manipulate, and annotate images from different institutions in a standardised way, facilitating work with digital surrogates in ATR and DH projects.

ITE (Integrated transcription environment): A software platform that combines tools for the entire ATR workflow, often including image management, layout analysis, automated transcription, manual correction, and data export.

Layout analysis/recognition: The process of identifying and segmenting the structural elements on a page image. Key sub-tasks include page segmentation (identifying text blocks, images, tables), text line detection (locating individual lines), and baseline detection (finding the imaginary line on which characters sit). It's a crucial pre-processing step for many ATR systems. Also known as Document Layout Analysis.

Learning rate: A hyperparameter that controls the step size a model takes when adjusting its internal parameters during training. A learning rate that's too high can cause the training process to be unstable and overshoot the optimal solution, while a rate that is too low can make training extremely slow or get stuck in a suboptimal solution.

LLM (Large language model): A transformer-based neural network typically for text generation, based on large amounts of training data.

Loss function: A function that quantifies the error between a model's prediction and the ground truth. The goal of training is to adjust the model's parameters to minimize the value of the loss function.

LSTM (Long short-term memory): A special type of Recurrent Neural Network (RNN) designed to overcome the limitations of traditional RNNs in learning long-range dependencies in sequential data. LSTMs have internal mechanisms (gates) that regulate the flow of information, allowing them to remember relevant context over long sequences, which is ideal for transcribing lines of text.

Machine learning: A subfield of artificial intelligence that focuses on developing algorithms that allow computers to learn patterns and make predictions from data without being explicitly programmed for the task. Supervised learning, unsupervised learning, and reinforcement learning are major branches of machine learning.

Model: The output of a machine learning algorithm after it has been trained on a dataset. In ATR, the model is a complex mathematical function (usually a neural network) containing all the learned parameters (weights and biases) that enable it to map image data of text to a sequence of characters.

NLP (Natural language processing): A field of research for information extraction, text generation and, manipulation with natural language (≠programming languages). NER is a typical example of NLP.

NER (Named entity recognition): A common Natural Language Processing (NLP) task that seeks to locate and classify named entities in text into pre-defined categories such as persons, organisations, locations, dates, etc. NER is often applied to the output of ATR to extract structured information from historical documents.

Neural networks: A computational model inspired by the structure of biological brains, consisting of interconnected nodes (neurons) organised in layers. Each connection has a weight that is adjusted during training. By processing data through these layers, the network learns to identify complex patterns.

OCR (Optical character recognition): A technology for automatically recognizing machine-printed text in images and converting it into editable, machine-readable text. OCR is a subfield of the more general ATR.

Offline-learning: Another term for batch-learning, where a model learns from a static, pre-collected dataset. The model is trained to completion on this data before being deployed. This contrasts with online learning, where the model updates incrementally as new data arrives. Most ATR applications use an offline learning approach.

Overfitting: A common problem in machine learning where a model learns the training data too well, including its noise and random fluctuations. An overfitted model performs exceptionally well on the data it was trained on but fails to generalize to new, unseen data. Techniques like regularisation and early stopping are used to prevent it.

PAGE XML: An XML schema designed for encoding information about page images, including detailed layout structures (text regions, lines, baselines, glyphs) and transcription results. It is widely used in the Digital Humanities for creating and storing ground truth for ATR.

Raster image: A digital image composed of a grid of pixels (picture elements), where each pixel has a specific colour value. Common formats include JPEG, PNG, and TIFF. Raster images are the standard input for ATR systems. They are distinct from vector images.

RNN (Recurrent neural network): A type of neural network designed for processing sequential data. RNNs have connections that form a directed cycle, allowing them to maintain an internal state or „memory" of previous inputs in the sequence. This makes them suitable for tasks like transcribing a line of text, where context from previous characters is important. LSTMs are an advanced type of RNN.

Sentiment analysis: An NLP task that involves identifying and categorizing opinions or emotions expressed in a piece of text to determine whether the writer's attitude is positive, negative, or neutral.

String: A data type representing a sequence of characters. The final output of an ATR process is typically a string.

TEI (Text encoding initiative): An XML-based standard for representing texts in digital form, widely used in the Digital Humanities. While formats like ALTO and PAGE XML describe the physical layout and recognised characters, TEI is used for rich semantic encoding of a text's content and structure (e.g., marking people, places, paragraphs, and editorial changes).

Transformer: A state-of-the-art neural network architecture that relies on a self-attention mechanism. Unlike RNNs, which process data sequentially, transformers can process all elements in an input sequence simultaneously, allowing them to model complex, long-range dependencies between all parts of the data.

TrOCR (TransformerOCR): An end-to-end ATR model that applies the transformer architecture to the task. It treats text recognition as a pure image-to-sequence problem, using an image encoder and a text decoder without requiring complex pre-processing steps common in CNN/RNN-based systems.

Unicode: An international computing industry standard for the consistent encoding, representation, and handling of text in most of the world's writing systems. It assigns a unique number, or codepoint, to every character, ensuring that text can be exchanged and displayed correctly across different platforms and languages.

Vector image: A digital image created from geometric shapes (points, lines, curves) defined by mathematical equations. Vector images can be scaled to any size without losing quality. Common formats include SVG and PDF. They are typically converted to a raster image format before being used as input for an ATR model.

8 Bibliography

Agarwal, Milind, and Antonios Anastasopoulos. 2024. "A Concise Survey of OCR for Low-Resource Languages." In *Proceedings of the 4th Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP 2024)*, 88–102. Mexico City: Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.americasnlp-1.10.

Andrews, Tara. 2023. "Where Are the Tools? The Landscape of Semi-Automated Text Edition." In *Digitale Edition in Österreich*, edited by Roman Bleier and Helmut Klug, 3–17. Schriften des Instituts für Dokumentologie und Editorik 16, 1–36. Norderstedt: BoD. https://kups.ub.uni-koeln.de/70445/.

Arnold, Matthias. 2022. "Multilingual Research Projects: Non-Latin Script Challenges for Making Use of Standards, Authority Files, and Character Recognition." *Digital Studies / Le Champ Numérique* 12 (1), 1-36. https://doi.org/10.16995/dscn.8110.

Austin, Greta. 2009. *Shaping Church Law around the Year 1000: The Decretum of Burchard of Worms.* Church, Faith and Culture in the Medieval West, Farnham: Ashgate.

——. 2019. "Burchard of Worms." In *Great Christian Jurists and Legal Collections in the First Millennium*, edited by Philip Lyndon Reynolds, 458–70. Cambridge Studies in Law and Christianity. Cambridge: Cambridge University Press.

——. 2022. "Canon Law in the Long Tenth Century, 900–1050." In *The Cambridge History of Medieval Canon Law*, edited by Anders Winroth and John C. Wei, 46–61. Cambridge: Cambridge University Press. https://doi.org/10.1017/9781139177221.004.

Baillot, Anne. 2023. *From Handwriting to Footprinting: Text and Heritage in the Age of Climate Crisis*. Cambridge: Open Book Publishers. https://doi.org/10.11647/OBP.0355.

Bartoš, František Michálek. 1955. *Dvě studie o husitských postilách*. Rozpravy Československé akademie věd. Řada SV 65 (4). Prague: ČSAV.

Bastianello, Elisa, and Reto Baumgartner. 2023. "L'applicazione del riconoscimento testi neurale per la realizzazione di ristampe digitali." In *La memoria digitale: forme del testo e organizzazione della conoscenza. Atti del XII Convegno Annuale AIUCD*, edited by Emmanuela Carbé, Gabriele Lo Piccolo, Alessia Valenti, and Francesco Stella, 15–23. Quaderni di Umanistica Digitale. Siena: AIUCD. https://doi.org/10.6092/unibo/amsacta/7721.

Bender, Emily M., Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜." In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 610–23. New York: Association for Computing Machinery. https://doi.org/10.1145/3442188.3445922.

Berger, Tilman. 2012. "Religion and Diacritics: The Case of Czech Orthography." In *Orthographies in Early Modern Europe*, edited by Susan Baddeley and Anja Voeste, 255–68. Berlin, Boston: De Gruyter Mouton. https://doi.org/10.1515/9783110288179.255.

Bischoff, Bernhard. 1986. *Paläographie des Römischen Altertums und des Abendländischen Mittelalters,* 2nd rev. ed. Grundlagen der Germanistik 24. Berlin: Schmidt.

Borges, Oliveira, Dario Augusto, and Matheus Palhares Viana. 2017. "Fast CNN-Based Document Layout Analysis." In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 1173–80. Venice: IEEE. https://doi.org/10.1109/ICCVW.2017.142.

Boros, Emanuela, et al. 2024. "Post-Correction of Historical Text Transcripts with Large Language Models: An Exploratory Study." In *Proceedings of the 8th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature (LaTeCH-CLfL 2024)*, 133–59. St. Julians, Malta: Association for Computational Linguistics. https://infoscience.epfl.ch/handle/20.500.14299/203985.

Brown-deVost, Bronson, and Itay Zandbank. 2021. *Escriptorium Connector*. Software. GitLab. https://gitlab.com/sofer_mahir/escriptorium_python_connector.

Buckland, Michael K. 2006. *Emanuel Goldberg and His Knowledge Machine: Information, Invention, and Political Forces*. New Directions in Information Management. Westport: Libraries Unlimited. http://dx.doi.org/10.5040/9798400644856.

Burghardt, Manuel, and Christian Wolff. 2009. "Stand off-Annotation für Textdokumente: Vom Konzept zur Implementierung (zur Standardisierung?)." In *Von der Form zur Bedeutung: Texte automatisch verarbeiten*, edited by Christian Chiarcos, R. Eckart de Castillo, and M. Stede, 53–59. Tübingen: Narr. https://epub.uni-regensburg.de/14223/.

Burghart, Marjorie. 2017. "Transcription or Diplomatic Edition." In *Digital Editing of Medieval Texts: A Textbook*, edited by Marjorie Burghart. DEMM. https://projects.history.qmul.ac.uk/wp-content/uploads/sites/6/2017/09/02-Transcription-MB.pdf.

Burns, Patrick. 2023. "Synthetic Trained Pipelines for Latin NLP". Preprint, submitted May 7, 2023. arXiv:2305.04365v1. https://arxiv.org/abs/2305.04365v1.

Buxbaum-Conradi, Björn. 2022. "Hyman Eli Goldberg and Emanuel Goldberg – Two Pioneers of Optical Character and Code Recognition – or Why One of Them Was Not Properly Credited." In *The Orcus of Oblivion (blog)*, November 2022. https://bbc-blog.net/2022/11/hyman-eli-goldberg-and-emanuel-goldberg.html.

Carroll, Stephanie Russo, Ibrahim Garba, Oscar L. Figueroa-Rodríguez, et al. 2020. "The CARE Principles for Indigenous Data Governance". Data Science Journal 19 (1): 43. https://doi.org/10.5334/dsj-2020-043.

Cattin, Pierre. 2018. "Source of Arthur Samuel's definition of machine learning." Data Science Stack Exchange, August 2018. https://datascience.stackexchange.com/questions/37078/source-of-arthur-samuels-definition-of-machine-learning.

Chagué, Alix. 2021. “De Transkribus à eScriptorium.” *Léctaurep* (blog). https://lectaurep.hypotheses.org/documentation/de-transkribus-a-escriptorium.

——, and Thibault Clérice. 2020. *HTR-United*. GitHub. https://github.com/HTR-United/htr-united/.

——, and Thibault Clérice. 2021. *Choco-Mufin*. GitHub. https://github.com/PonteIneptique/choco-mufin.

——, and Thibault Clérice. 2023. “Deploying eScriptorium Online: Notes on CREMMA’s Server Specifications.” *A Research (b)Log (blog)*, December 22, 2023. https://alix-tz.github.io/phd/posts/017/.

Chéramy, Robert. 2012. *YASW (Yet Another Scan Wizard)*. Software. GitHub. https://github.com/tibob/yasw.

Clérice, Thibault. 2023. “You Actually Look Twice At It (YALTAi): Using an Object Detection Approach Instead of Region Segmentation within the Kraken Engine”, *Journal of Data Mining & Digital Humanities*, Historical Documents and Handwritten Text Recognition: 9806, https://doi.org/10.46298/jdmdh.9806.

——, Ariane Pinche, Malamatenia Vlachou-Efstathiou, et al. 2024. “CATMuS Medieval: A Multilingual Large-Scale Cross-Century Dataset in Latin Script for Handwritten Text Recognition and Beyond.” Preprint. HAL: hal-04453952. https://univ-paris8.hal.science/hal-04453952v1.

Cook, Gary, Jude Lee, Tamina Tsai, et al. 2017. *Clicking Green: Who Is Winning the Race to Build a Green Internet?* Washington: Greenpeace. https://www.greenpeace.de/publikationen/20170110_greenpeace_clicking_clean.pdf.

Daňhelka, Jiří. 1985. “Směrnice pro vydávání starších českých textů.” *Husitský Tábor* 8: 285–301.

David, Zdeněk V. 2003. *Finding the Middle Way: The Utraquists’ Liberal Challenge to Rome and Luther*. Baltimore, London: The Johns Hopkins University Press.

Déjean, Hervé, Jean-Luc Meunier, and Eva Maria Lang. 2016. *TranskribusPyClient*. Software. GitHub. https://github.com/Transkribus/TranskribusPyClient.

Demo, Šime. 2014. "Towards a Unified Definition of Macaronics." *Humanistica Lovaniensia* 63: 83–106. http://www.jstor.org/stable/24868899.

Deng, Qilin, Mayire Ibrayim, Askar Hamdulla, and Chunhu Zhang. "The YOLO Model That Still Excels in Document Layout Analysis." *Signal, Image and Video Processing* 18: 1539–1548. https://doi.org/10.1007/s11760-023-02838-y.

Derolez, Albert. 2006. *The Palaeography of Gothic Manuscript Books: From the Twelfth to the Early Sixteenth Century*. Cambridge Studies in Palaeography and Codicology 9. Cambridge : Cambridge University Press.

Dokoupil, Vladislav. 1958. *Soupis rukopisů mikulovské dietrichsteinské knihovny*. Soupisy rukopisných fondů Universitní knihovny v Brně 2. Prague: Státní pedagogické nakladatelství.

Driscoll, J. M. 2007. "Electronic Textual Editing: Levels of Transcription." Web page. https://tei-c.org/Vault/ETE/Preview/driscoll.html.

DYBBUK project. 2022. "The Dybbuk for Yiddish Handwriting." Transkribus Model. https://readcoop.eu/model/the-dybbuk-for-yiddish-handwriting/.

Florianová, Hana, Dana Martínková, Zuzana Silagiová, and Hana Šedinová. 2006. *Quadragesimale Admontense = Quadragesimale admontské*. Fontes Latini Bohemorum 6. Prague: OIKOYMENH.

Frank-Job, Barbara, and Bianca Henrichfreise. 2015. "Diskurstraditionelles im Sprachwandel: Korpuslinguistische Untersuchungen zum Spätlatein." In *Diskurstraditionelles und Einzelsprachliches im Sprachwandel*, edited by Esme Winter-Froemel et al., 159–81. ScriptOralia 141. Tübingen: Narr.

Gabay, Simon, Jean-Baptiste Camps, Ariane Pinche, and Claire Jahan. 2021. "SegmOnto: Common Vocabulary and Practices for Analysing the Layout of Manuscripts (and More)." Paper presented at the 1st International Workshop on Computational Paleography (IWCP@ICDAR 2021). Lausanne. https://hal.archives-ouvertes.fr/hal-03336528.

Ganz, David. 2020. "Early Caroline: France and Germany." In *The Oxford Handbook of Latin Palaeography*, edited by Frank T. Coulson and Robert G. Babcock, 235–61. Oxford: Oxford University Press. https://doi.org/10.1093/oxfordhb/9780195336948.013.54.

Geelhaar, Tim. 2023. "Carolingian Minuscule Model CMM 9th-11th c." Transkribus Model. https://app.transkribus.org/models/public/text/carolingian-minuscule-model-cmm-9th-11th-c.

——, Ludolf Kuchenbuch, Nicolas Perreaux, Juliane Schiel, and Isabelle Schürch. "Historical Semantics – A Vade Mecum." *Österreichische Zeitschrift für Geschichtswissenschaften, 34* (2): 18–47. https://doi.org/10.25365/OEZG-2023-34-2-2.

Géron, Aurélien. 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd ed. Sebastopol: O'Reilly.

Ghiriti, Alex, Wolfgang Göderle, and Roman Kern. 2024. "Exploring the Capabilities of GPT4-Vision as OCR Engine." In *Linking Theory and Practice of Digital Libraries*, edited by Apostolos Antonacopoulos et al., 3–12. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-72440-4_1.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. Adaptive Computation and Machine Learning. Cambridge, MA: The MIT Press.

Gottfredson, Linda S. 1997. "Mainstream Science on Intelligence: An Editorial with 52 Signatories, History, and Bibliography." *Intelligence* 24 (1): 13–23. https://doi.org/10.1016/S0160-2896(97)90011-8.

Graves, Alex, and Jürgen Schmidhuber. 2009. “Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks.” In *Advances in Neural Information Processing Systems 21*, edited by Daphne Koller, D. Schuurmans, Y. Bengio, and L. Bottou 1–8. Curran. https://papers.nips.cc/paper/2008/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

Greg, Walter. 1950. “The Rationale of Copy-Text.” *Studies in Bibliography* 3: 19–36. http://www.jstor.org/stable/40381874.

Greif, Gavin, Niclas Griesshaber, and Robin Greif. 2025. “Multimodal LLMs for OCR, OCR Post-Correction, and Named Entity Recognition in Historical Documents.” Preprint, submitted April 1, 2025. arXiv:2504.00414. http://arxiv.org/abs/2504.00414.

Guéville, Estelle, and David Joseph Wrisley. 2024. “Transcribing Medieval Manuscripts for Machine Learning.” *Journal of Data Mining & Digital Humanities*. https://doi.org/10.46298/jdmdh.9805.

Hanuš, Ignác Jan. 1861. “Příspěvky k historii literatury české.” *Časopis Musea Království českého* 35: 107–22.

Hanuš, Ignác Jan. 1863. *Malý výbor ze staročeské literatury: podle rukopisův C.K. Knihovny vysokých škol prážských XIV.-XVII. století posud z větší části netištěných*. Prague: I. L. Kober.

Haralambous, Yannis, and P. Scott Horne. 2007. *Fonts & Encodings*. Sebastopol: O’Reilly Media. https://hal.science/hal-02112942v1/.

Haverals, Wouter. 2023. *CERberus: Guardian against Character Errors*. Software. GitHub. https://github.com/WHaverals/CERberus.

Hodel, Tobias. 2020. „Best-practices zur Erkennung alter Drucke und Handschriften. Die Nutzung von Transkribus large- und small-scale“. https://doi.org/10.5281/zenodo.4621728.

——, David Schoch, Christa Schneider, and Jake Purcell. 2021. “General Models for Handwritten Text Recognition: Feasibility and State-of-the Art. German Kurrent as an Example.” *Journal of Open Humanities Data* 7. https://doi.org/10.5334/johd.46.

——. 2022a. “Chapter 6: Supervised and Unsupervised: Approaches to Machine Learning for Textual Entities.” In *Archives, Access and Artificial Intelligence: Working with Born-Digital and Digitised Archival Collections*, edited by Lise Jaillant, 157–78. Digital Humanities Research 2. Bielefeld: transcript. https://doi.org/10.14361/9783839455845-007.

——. 2022b. „Gothic_Book_Scripts_XIII-XV_M4.” Transkribus Model. https://app.transkribus.org/models/public/text/gothic-book-scripts-xiii-xv-m4.

Hoffmann, Hartmut, and Rudolf Pokorny. 1991. *Das Dekret des Bischofs Burchard von Worms: Textstufen, frühe Verbreitung, Vorlagen*. MGH Hilfsmittel 12. München: Monumenta Germaniae Historica.

Horner, Patrick J. 2006. *A Macaronic Sermon Collection from Late Medieval England: Oxford, MS Bodley 649*. Toronto: Pontifical Institute of Mediaeval Studies.

Horst, Tom ter, and Nike Stam. 2017. “Visual Diamorphs: The Importance of Language Neutrality in Code-Switching from Medieval Ireland.” In *Multilingual Practices in Language History: English and Beyond*, edited by Päivi Pahta, Janne Skaffari, and Laura Wright, 223–42. Berlin, Boston: De Gruyter Mouton. https://doi.org/10.1515/9781501504945-011.

Huitfeldt, Claus, and Christopher M. Sperberg-McQueen. 2008. “What Is Transcription?” *Literary and Linguistic Computing* 23 (3): 295–310. https://doi.org/10.1093/llc/fqn013.

Humphries, Mark, Lianne C. Leddy, Quinn Downton, et al. 2024. “Unlocking the Archives: Using Large Language Models to Transcribe Handwritten Historical Documents.” Preprint, submitted November 1, 2024. arXiv:2411.03340. http://arxiv.org/abs/2411.03340.

--. 2025. "Gemini 3 Solves Handwriting Recognition and It’s a Bitter Lesson", Substack newsletter, Generative History, 25 November 2025. https://generativehistory.substack.com/p/gemini-3-solves-handwriting-recognition.

Jacsont, Pauline, and Elina Leblanc. 2023. “Impact of Image Enhancement Methods on Automatic Transcription Trainings with eScriptorium.” Preprint, submitted June 2023. HAL: hal-03831686. https://hal.science/hal-03831686.

Jaillant, Lise, Olivia Mitchell, Eric Ewoh-Opu, and Maribel Hidalgo Urbaneja. 2025. “How Can We Improve the Diversity of Archival Collections with AI? Opportunities, Risks, and Solutions.” *AI & Society* 40: 4447–4459. https://doi.org/10.1007/s00146-025-02222-z.

Jannidis, Fotis. 2017. “Zeichen und Zahlen.” In *Digital Humanities: Eine Einführung*, edited by Fotis Jannidis, Hubertus Kohle, and Malte Rehbein, 59–67. Stuttgart: Metzler.

Jurafsky, Dan, and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2nd ed. Hoboken: Prentice Hall.

Jussen, Bernhard, and Gregor Rohmann. 2015. “Historical Semantics in Medieval Studies: New Means and Approaches.” *Contributions to the History of Concepts* 10 (2): 1–6. https://doi.org/10.3167/choc.2015.100201.

Just, Marcel A., and Patricia A. Carpenter. 1980. “A Theory of Reading: From Eye Fixations to Comprehension.” *Psychological Review* 87 (4): 329–54. https://doi.org/10.1037/0033-295X.87.4.329.

Kéry, Lotte. 1999. *Canonical Collections of the Early Middle Ages (ca. 400-1140): A Bibliographical Guide to the Manuscripts and Literature*. History of Medieval Canon Law. Washington, D.C.: Catholic University of America Press.

Kéry, Lotte. 2017. “Burchard von Worms (1000-1025) und die Entwicklung des kirchlichen Strafrechts.” In *Theologia Iuris Canonici. Festschrift für Ludger Müller zur Vollendung des 65. Lebensjahres*, edited by Christoph Ohly, Wilhelm Rees, and Libero Gerosa, 571–98. Kanonistische Studien und Texte 67. Berlin: Duncker & Humblot.

Kiessling, Benjamin. 2015. *Kraken*. Software. GitHub. https://github.com/mittagessen/kraken.

Kim, Seorin, Julien Baudru, Wouter Ryckbosch, Hugues Bersini, and Vincent Giniset. 2025. "Early Evidence of How LLMs Outperform Traditional Systems on OCR/HTR Tasks for Historical Records." Preprint, submitted January 20, 2025. arXiv:2501.11623. http://arxiv.org/abs/2501.11623.

Korpela, Jukka. 2006. *Unicode Explained*. Internationalize Documents, Programs, and Web Sites. Sebastopol: O'Reilly.

Kühl, Niklas, Max Schemmer, Marc Goutier, and Gerhard Satzger. 2022. "Artificial Intelligence and Machine Learning." *Electronic Markets* 32 (4): 2235–44. https://doi.org/10.1007/s12525-022-00598-0.

Kynast, Birgit. 2020. *Tradition und Innovation im kirchlichen Recht. Das Bußbuch im Dekret des Bischofs Burchard von Worms*. Quellen und Forschungen zum Recht im Mittelalter 10. Ostfildern: Thorbecke.

Leifert, Gundram, Tobias Strauß, Tobias Grüning, Welf Wustlich, and Roger Labahn. 2016. "Cells in Multidimensional Recurrent Neural Networks." *Journal of Machine Learning Research* 17: 1–37.

Li, Lucian. 2024. "Handwriting Recognition in Historical Documents with Multimodal LLM." Preprint, submitted October 31, 2024. arXiv:2410.24034. http://arxiv.org/abs/2410.24034.

Luccioni, Alexandra Sasha, Giada Pistilli, Raesetje Sefala, and Nyalleng Moorosi. 2025. "Bridging the Gap: Integrating Ethics and Environmental Sustainability in AI Research and Practice." Preprint, submitted April 1, 2025. arXiv:2504.00797. https://doi.org/10.48550/arXiv.2504.00797.

Matić-Chalkitis, Milanka. 2023. "OttomanTurkish_generic." Transkribus Model. *https://app.transkribus.org/models/public/text/ottoman-turkish-generic*.

McCulloch, Warren S., and Walter Pitts. 1943. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *The Bulletin of Mathematical Biophysics* 5 (4): 115–33. https://doi.org/10.1007/BF02478259.

McGillivray, Barbara, Paola Marongiu, Nilo Pedrazzini, Marton Ribary, Mandy Wigdorowitz, and Eleonora Zordan. 2022. "Deep Impact: A Study on the Impact of Data Papers and Datasets in the Humanities and Social Sciences." *Publications* 10 (4): 39. https://doi.org/10.3390/publications10040039.

Mehler, Alexander, Bernhard Jussen, and Tim Geelhaar. 2020. "The Frankfurt Latin Lexicon: From Morphological Expansion and Word Embeddings to SemioGraphs." Preprint, submitted May 21, 2020. arXiv:2005.10790. http://arxiv.org/pdf/2005.10790v1.

Melanson, Joe. 2020. "Towards Epistemic Justice in the Archives." *Emerging Library & Information Perspectives* 3 (1): 89–112. https://doi.org/10.5206/elip.v3i1.8617.

Michael, Nielsen. 2015. *Neural Networks and Deep Learning*. Online book. http://neuralnetworksanddeeplearning.com.

Mitchell, Tom M. 2013. *Machine Learning*. Reprint. McGraw-Hill Series in Computer Science. Columbus: McGraw-Hill.

Mondello, Francesco. 2024. *Converseen*. Version 0.12.1.0. Software. GitHub. https://github.com/Faster3ck/Converseen.

MUFI: The Medieval Unicode Font Initiative. 2015. Version 4.0. https://mufi.info/q.php?p=mufi.

Mühlberger, Günter, Sebastian Colutto, and Philipp Kahle. 2014. "Handwritten Text Recognition (HTR) of Historical Documents as a Shared Task for Archivists, Computer Scientists and Humanities Scholars: The Model of a Transcription & Recognition Platform (TRP)." Report. Innsbruck: University of Innsbruck.

Müller, Markus. 2021a. "Uncovering Censorship in the 16th Century with Transkribus and Python. Episode I: OCR with Latin Prints." *Digital Humanities Lab* (blog), July 2, 2021. https://dhlab.hypotheses.org/2022.

Müller, Markus. 2021b. "Uncovering Censorship in the 16th Century with Transkribus and Python. Episode II: Let Python Speak to Transkribus." *Digital Humanities Lab* (blog), July 23, 2021. https://dhlab.hypotheses.org/2114.

Neudecker, Clemens, Mike Gerber, and Robert Sachunsky. 2015. *OCR Conversion*. Software. GitHub. https://github.com/cneud/ocr-conversion.

Ningyang, Li. 1993. *An Implementation of OCR System Based on Skeleton Matching*. Canterbury: University of Kent. https://kar.kent.ac.uk/21129.

Nockels, Joseph, Paul Gooding, and Melissa Terras. 2024. "The Implications of Handwritten Text Recognition for Accessing the Past at Scale." *Journal of Documentation* 80 (7): 148–67. https://doi.org/10.1108/JD-09-2023-0183.

"Now, PC's That Read a Page and Store It." 1988. *New York Times*, August 17, 1988, p. 6.

Odstrčilík, Jan. 2015. "Analýza dvou latinských překladů Husovy České nedělní postily v rkp. Brno, Moravská zemská knihovna, Mk 56 a Mk 91 a jejich částečná edice." Prague. http://hdl.handle.net/20.500.11956/79733.

——. 2019. "Translation and Transformation of Jan Hus's Czech Sunday Postil." In *Pursuing a New Order II: Late Medieval Vernacularization and the Bohemian Reformation*, edited by Pavlína Rychterová and Julian Ecker, 153–84. The Medieval Translator 17.2. Turnhout: Brepols. https://doi.org/10.1484/M.TMT-EB.5.116115.

——. 2020. "Multilingual Medieval Sermons: Sources, Theories and Methods." *Medieval Worlds*, no. 12: 140–47. https://doi.org/10.1553/medievalworlds_no12_2020s140.

——. 2023. “Unbearable Lightness of Multilingual Sermons? The So-Called Wilhering Adaptation of Three Czech Sermons of Jan Hus.” In *Medieval Translations and Their Readers*, edited by Pavlína Rychterová and Jan Odstrčilík, 223–50. The Medieval Translator 20. Turnhout: Brepols. https://doi.org/10.1484/M.TMT-EB.5.133070.

Olson, Kevin. 2024. “Archival Silence: How Do We Write the History of the Subaltern Who Cannot Speak?” *Political Science & Politics* 57 (1): 92–94. https://doi.org/10.1017/S1049096523000525.

Padilla, Thomas. 2017. *On a Collections as Data Imperative*. Report. eScholarship. https://escholarship.org/uc/item/9881c8sv.

Perfetti, Charles, and Lesley Hart. 2002. “The Lexical Quality Hypothesis.” In *Studies in Written Language and Literacy*, edited by Ludo Verhoeven, Carsten Elbro, and Pieter Reitsma, 189–213. Amsterdam, Philadelphia: John Benjamins Publishing Company. https://doi.org/10.1075/swll.11.14per.

——, Nicole Landi, and Jane Oakhill. 2005. “The Acquisition of Reading Comprehension Skill.” In *The Science of Reading: A Handbook*, edited by Margaret J. Snowling and Charles Hulme, 227–47. Malden, MA: Blackwell Publishing. https://doi.org/10.1002/9780470757642.ch13.

——. 2007. “Reading Ability: Lexical Quality to Comprehension.” *Scientific Studies of Reading* 11 (4): 357–83. https://doi.org/10.1080/10888430701530730.

Peter Robinson and Elizabeth Solopova. 1993. *Guidelines for Transcription of the Manuscripts of the Wife of Bath’s Prologue*. Online document. https://doi.org/10.5281/zenodo.4050359.

Pinche, Ariane. 2023. “Generic HTR Models for Medieval Manuscripts. The CREMMALab Project.” *Journal of Data Mining & Digital Humanities*. https://doi.org/10.46298/jdmdh.10252.

Puigcerver, Joan, and Carlos Mocholí. 2018. *PyLaia*. Software. GitHub. *https://github.com/jpuigcerver/PyLaia*.

Quirós, Lorenzo. 2018. "Multi-Task Handwritten Document Layout Analysis." Preprint, submitted December 12, 2018. arXiv:1806.08852. http://arxiv.org/abs/1806.08852.

Rabus, Achim. 2019. "Recognizing Handwritten Text in Slavic Manuscripts: A Neural-Network Approach Using Transkribus." *Scripta & E-Scripta* 19: 9–32.

——. 2020. "Combined_Full_VKS_2." Transkribus Model. https://app.transkribus.org/models/public/text/russian-church-slavonic-1.

——. 2022a. "Handwritten Glagolitic." Transkribus Model. https://app.transkribus.org/models/public/text/48703.

——. 2022b. "Handwritten Text Recognition for Croatian Glagolitic." *Slovo* 72: 181–92. https://hrcak.srce.hr/file/391285.

——. 2022c. "Russian Generic Handwriting 2." Transkribus Model. https://app.transkribus.org/models/public/text/russian-generic-handwriting-2.

——, Martin Meindl, and Milanka Matić-Chalkitis. 2022. "DEK_German_combined." Transkribus Model. https://app.transkribus.org/models/public/text/47882.

——, and Walker Thompson. 2023. "Performance of Generic HTR Models on Historical Cyrillic and Glagolitic: Comparison of Engines." *Scripta & E-Scripta* 23: 11–34.

——, Walker Riggs Thompson, and Daniel Stökl Ben Ezra. 2023. "Generic HTR model for Old Cyrillic uncial and semi-uncial script styles (11th-16th c.)." Kraken Model, March 21, 2023. https://doi.org/10.5281/ZENODO.7755483.

——. 2024a. "Möglichkeiten und Grenzen KI-gestützter Manuskript-Transkription für unterschiedliche Einsatzzwecke." In *Post aus Nürnberg. Interdisziplinäre Forschungen zu den Briefbüchern des 15. Jahrhunderts*, edited by Mechthild Habermann, Peter Fleischmann, and Klaus Herbers, 47–60. Nürnberger Forschungen 34. Neustadt an der Aisch: Schmidt.

——. 2024b. “Tolerating Imperfection: Uncorrected Transkribus Transcriptions in Church Slavonic Studies.” In *Učitelnoto evangelie na Konstantin Preslavski i južnoslavjanskite prevodi na chomiletični tekstove (IX-XIII v.)*, edited by Lora Taseva, Achim Rabus, and Ivan P. Petrov. Studia Balcanica 37. Sofia: Bălgarska akademija na naukite. https://doi.org/10.62761/491.SB37.19.

——, and Martin Meindl. Forthcoming. “Digitizing Cyrillic Manuscripts Using Handwritten Text Recognition Technologies: Recent Developments and Future Perspectives.” In *Cyrillic Manuscripts: From Medieval to Digital*, edited by Viacheslav V. Lytvynenko, Małgorzata Skowronek, Achim Rabus, Dimiter Peev, and Boban Petrovski, 249–266. South-East European History 16. New York and Berlin: Peter Lang.

Rastinger, Nina. 2024. “Kleine Texte, dichte Daten. Listen in historischen Zeitungen als Schatz für die Digital Humanities.” *Digital History Berlin* (blog), January 8, 2024. https://doi.org/10.58079/vjru.

Rayner, Keith. 1998. “Eye Movements in Reading and Information Processing: 20 Years of Research.” *Psychological Bulletin* 124 (3): 372–422. https://doi.org/10.1037/0033-2909.124.3.372.

Rayner, Keith, et al. 2001. “How Psychological Science Informs the Teaching of Reading.” *Psychological Science in the Public Interest* 2 (2): 31–74. https://doi.org/10.1111/1529-1006.00004.

Rehbein, Malte. 2017. “Digitalisierung.” In *Digital Humanities: Eine Einführung*, edited by Fotis Jannidis, Hubertus Kohle, and Malte Rehbein, 179–98. Stuttgart: Metzler.

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. 2016. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” Preprint, submitted January 6, 2016. arXiv:1506.01497. http://arxiv.org/abs/1506.01497.

Ridley, Michael, and Danica Pawlick-Potts. 2021. “Algorithmic Literacy and the Role for Libraries.” *Information Technology and Libraries* 40 (2). https://doi.org/10.6017/ital.v40i2.12963.

Rojas, Raúl. 1996. *Neural Networks: A Systematic Introduction*. Berlin, Heidelberg: Springer.

Romein, C. Annemieke, et al. 2022. *Exploring Data Provenance in Handwritten Text Recognition Infrastructure: Sharing and Reusing Ground Truth Data, Referencing Models, and Acknowledging Contributions. Starting the Conversation on How We Could Get it Done*. Report. Zenodo. https://doi.org/10.5281/zenodo.7267244.

Rosenblatt, Frank. 1957. *The Perceptron – a Perceiving and Recognizing Automaton*. Report. Ithaca: Cornell Aeronautical Laboratory.

Russell, Stuart J., Peter Norvig, and Ming-wei Chang. 2022. *Artificial Intelligence a Modern Approach*. 4th ed. London: Pearson.

Ryba, Bohumil. 2024. "Bohumil Ryba's Rules for Transcription of Latin Manuscript Texts (English Translation)." Translated by Jan Odstrčilík. Zenodo. April 24, 2024. https://zenodo.org/records/11060034.

Sahle, Patrick. 2013a. *Digitale Editionsformen. Zum Umgang mit der Überlieferung unter den Bedingungen des Medienwandels. Teil 1: Das typografische Erbe.* Vol. 7 of *Schriften des Instituts für Dokumentologie und Editorik*. Norderstedt: BoD. https://kups.ub.uni-koeln.de/5351/.

——. 2013b. *Digitale Editionsformen. Zum Umgang mit der Überlieferung unter den Bedingungen des Medienwandels. Teil 2: Befunde, Theorie und Methodik.* Vol. 8 of *Schriften des Instituts für Dokumentologie und Editorik*. Norderstedt: BoD. https://kups.ub.uni-koeln.de/5352/.

——. 2013c. *Digitale Editionsformen. Zum Umgang mit der Überlieferung unter den Bedingungen des Medienwandels. Teil 3: Textbegriffe und Recodierung.* Vol. 9 of *Schriften des Instituts für Dokumentologie und Editorik*. Norderstedt: BoD. https://kups.ub.uni-koeln.de/5353/.

Samuel, A. L. 1959. "Some Studies in Machine Learning Using the Game of Checkers." *IBM Journal of Research and Development* 44 (1.2): 206–26. https://doi.org/10.1147/rd.441.0206.

Schantz, Herbert F. 1982. *The History of OCR, Optical Character Recognition*. Manchester Center, Vermont: Recognition Technologies Users Association.

Schonhardt, Michael. 2023a. "Bdd-Wormser-Scriptorium-0.2." Transkribus Model. https://www.transkribus.org/model/bdd-wormser-scriptorium-0.2.

——. 2023b. "Bdd-Wormser-Scriptorium-Expanded-0.1." Transkribus Model. https://www.transkribus.org/model/bdd-wormser-scriptorium-expanded-0.1.

——. 2024a. "Michaelscho/Bdd-Segmentation-Baselines: V1.0." Version v1.0. Kraken Model. Zenodo, March 29, 2024. https://doi.org/10.5281/ZENODO.10894507.

——. 2024b. "Michaelscho/Bdd-Segmentation-Data: BDD Segmentation Data." Version v1.0. Dataset. Zenodo, March 26, 2024. https://doi.org/10.5281/ZENODO.10882816.

——. 2024c. "Michaelscho/Bdd-Segmentation-Regions: Bdd-Segmentation-Regions." Version v1.0. Kraken Model. Zenodo, March 28, 2024. https://doi.org/10.5281/ZENODO.10890966.

——. 2024d. "Model Trained on 11th Century Manuscripts to Produce Expanded Transcription (Latin)." Kraken Model. Zenodo, September 9, 2024. https://doi.org/10.5281/ZENODO.13736583.

——. 2024e. "Model Trained on 11th Century Manuscripts to Produce Graphematic Transcription (Latin)." Kraken Model. Zenodo, September 10, 2024. https://doi.org/10.5281/ZENODO.13741956.

——. 2025a. "Abbreviationes." Version 1.0.0. Software. Zenodo, July 31, 2025. https://doi.org/10.5281/ZENODO.16628613.

——. 2025b. "Bdd-Abbreviations-Augmented." Dataset. Zenodo, August 4, 2025. https://doi.org/10.5281/ZENODO.16735498.

——. 2025c. “Die digitale Edition als Schnittstelle.” In *Schnittstelle Mediävistik. Kollaborationen der Mittelalterforschung im digitalen Zeitalter*, edited by Luise Borek, Karoline Döring, Nora Ketschik and Katharina Zeppezauer-Wachauer, Das Mittelalter. Perspektiven mediävistischer Forschung, 30,1: 54–69. https://doi.org/10.17885/HEIUP.MIAL.2025.1.25120.

——. 2025d. “mschonhardt/Byt5-Base-Bdd-Expansion-Lora-v4-L40s.” ByT5 Model. Zenodo, August 4, 2025. https://doi.org/10.5281/ZENODO.16736385.

Schotter, Elizabeth R., and Keith Rayner. 2015. “The Work of the Eyes During Reading.” In *The Oxford Handbook of Reading*, edited by Alexander Pollatsek and Rebecca Treiman, 44–60. Oxford: Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199324576.013.2.

Šimek, František. 1936. “Neznámá sbírka kázání Rokycanových.” *Sborník filologický* 11 (1): 1–73.

Spunar, Pavel. 1978. “Literární činnost utrakvistů doby poděbradské a jagellonské.” In *Acta reformationem bohemicam illustrantia - I: Příspěvky k dějinám utrakvismu*, edited by Amedeo Molnár, 165–269. Prague: Kalich.

Stokes, Peter, Benjamin Kiessling, Daniel Stökl Ben Ezra, Robin Tissot, and El Hassane Gargem. 2021. “The eScriptorium VRE for Manuscript Cultures.” In “Ancient Manuscripts and Virtual Research Environments,” edited by Claire Clivaz and Garrick van Allen, special issue, *Classics@* 18, 1. https://classics-at.chs.harvard.edu/the-escriptorium-vre-for-manuscript-cultures.

Ströbel, Phillip, Tobias Hodel, Franz Fischer, et al. 2023a. *Bullingers Briefwechsel zugänglich machen: Stand der Handschriftenerkennung*. Report. Zenodo, March 10, 2023. https://doi.org/10.5281/ZENODO.7715357.

Ströbel, Phillip, Tobias Hodel, Walter Boente, and Martin Volk 2023b. “The Adaptability of a Transformer-Based OCR Model for Historical Documents.” In *Document Analysis and Recognition – ICDAR*

2023 Workshops, edited by Mickael Coustaty and Alicia Fornés, 34–48. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-41498-5_3.

Strubell, Emma, Ananya Ganesh, and Andrew McCallum. 2019. "Energy and Policy Considerations for Deep Learning in NLP." Preprint, submitted June 5, 2019. arXiv:1906.02243. https://doi.org/10.48550/arXiv.1906.02243.

Tanselle, G. Thomas. 1978. "The Editing of Historical Documents." *Studies in Bibliography* 31: 1–56.

Tikhonov, Aleksej, Lesley Loew, Milanka Matić-Chalkitis, Martin Meindl, and Achim Rabus. 2022. "Multilingual Handwritten Text Recognition (MultiHTR) or Reading Your Grandma's Old Letters in German, Russian, Serbian, and Ottoman Turkish with Artificial Intelligence." In *The Palgrave Handbook of Digital and Public Humanities*, 215–33. Basingstoke: Palgrave Macmillan Cham. https://doi.org/10.1007/978-3-031-11886-9_12.

Treiman, Rebecca, and Brett Kessler. 2015. "Writing Systems: Their Properties and Implications for Reading." In *The Oxford Handbook of Reading*, edited by Alexander Pollatsek and Rebecca Treiman, 10–25. New York: Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199324576.013.1.

Umer, Saiyed, Ranjan Mondal, Hari Mohan Pandey, and Ranjeet Kumar Rout. 2021. "Deep Features Based Convolutional Neural Network Model for Text and Non-Text Region Segmentation from Document Images." *Applied Soft Computing* 113: 107917. https://doi.org/10.1016/j.asoc.2021.107917.

Varischi, Carlo. 1964. *Sermoni del Beato Bernardino Tomitano da Feltre nella redazione di fra Bernardino Bulgarino da Brescia in tre tomi.* Vol. 3. Milan: Cassa di risparmio delle provincie lombarde e Banca del Monte.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, et al. 2017. "Attention Is All You Need." Preprint. arXiv:1706.03762. https://doi.org/10.48550/ARXIV.1706.03762.

Wenzel, Siegfried. 1994. *Macaronic Sermons: Bilingualism and Preaching in Late-Medieval England*. Ann Arbor: University of Michigan Press.

Wick, Christoph, Christian Reul, and Frank Puppe. 2020. "Calamari - A High-Performance Tensorflow-Based Deep Learning Package for Optical Character Recognition." *Digital Humanities Quarterly* 14 (2). https://dhq.digitalhumanities.org/vol/14/2/000451/000451.html.

Wilkinson, Mark D., Michel Dumontier, IJsbrand Jan Aalbersberg, et al. 2016. "The FAIR Guiding Principles for Scientific Data Management and Stewardship." *Scientific Data* 3 (1): 160018. https://doi.org/10.1038/sdata.2016.18.

Will, Larissa. 2022. "Modellübertragung von Transkribus nach eScriptorium." Web page. https://ub-mannheim.github.io/eScriptorium_Dokumentation/Modell%C3%BCbertragung_Transkribus_nach_eScriptorium.html.

Wright, Laura. 2011. "On Variation in Medieval Mixed-Language Business Writing." In *Code-Switching in Early English*, edited by Herbert Schendl and Laura Wright, 191–218. Berlin/Boston: De Gruyter Mouton.

Wu, Yuxin, et al. 2019. *Detectron2*. Software. GitHub. https://github.com/facebookresearch/detectron2.

Yap, Melvin J., and David A. Balota. 2015. "Visual Word Recognition." In *The Oxford Handbook of Reading*, edited by Alexander Pollatsek and Rebecca Treiman, 26–43. New York: Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199324576.013.4.

Zink, Michel. 1976. *La prédication en langue romane avant 1300*. Paris: Honoré Champion.