


Text Digitization

Daniel Stökl Ben Ezra

 <https://orcid.org/0000-0001-5668-493X>

Abstract Text digitization describes the conversion of digital image data of inscribed objects into machine-readable texts.*

Keywords HTR, OCR, Layout Analysis, Handwriting, Machine Learning, Neuron Networks

1. Introduction

Let us define text digitization as the conversion of digital images of inscribed objects of any kind (manuscripts, inscriptions, cuneiform tablets, prints, etc.) into machine-readable texts. Databases with full-text search options in retroactively digitized prints such as JSTOR (since 1994) or Google Books (since 2004) have fundamentally changed the way research is carried out in all academic disciplines. In the last ten years, progress in automatic document analysis, especially in machine learning, has revolutionized the possibilities for researchers to analyze not only difficult prints of the most important cultural texts, but even historical manuscripts, with computerized means.

Transcription of the letters is not the only level of analysis. Text goes beyond a sequence of letters. Gerard Genette (1982) has highlighted the importance of layout and non-main text sections for the (preliminary) understanding of texts. Layout contains critical information, such as the distinction between title and main text, main text and notes, speakers in dramas, verses in poetry, or text connections in translations or commentaries. The choice of and changes in writing style or typeface, width, register (e.g. normal, italic, slanted), weight, color, and alphabets are also essential information carriers that can significantly deepen the depth of analysis of an analyzed text beyond the simple letter sequence (Beinert 2021). In addition, machine paleography, layout analysis, and codicology are used to evaluate these subtle differences for network analysis, dating, and localization of individual objects. This *big data* provides the traditional auxiliary sciences a completely new meaning.

* This chapter, including quotations in foreign languages, was translated from German by Brandon Watson.

After a brief introduction about the encoding of image and text, neural networks, and existing programs, this chapter treats layout analysis and reading order, computer paleography and text recognition. The analysis is in constant engagement with the discipline known in computer science as “image processing” (Maier et al. 2020).

2. Image Encoding

Computers can only distinguish between the values 0 and 1. Both texts and images must initially be represented as sequences of 0s and 1s. Usually, the image is then transferred with a grid into a table/matrix with rows and columns, in which each cell contains a value for one pixel. High-resolution images have more pixels for the same object surface than low-resolution images. Black and white images are the simplest image format, only knowing 0 or 1 for each pixel, such as foreground (ink) or background (paper). This format leads to the familiar stair steps (Fig. 1), especially at low resolution or high magnification.

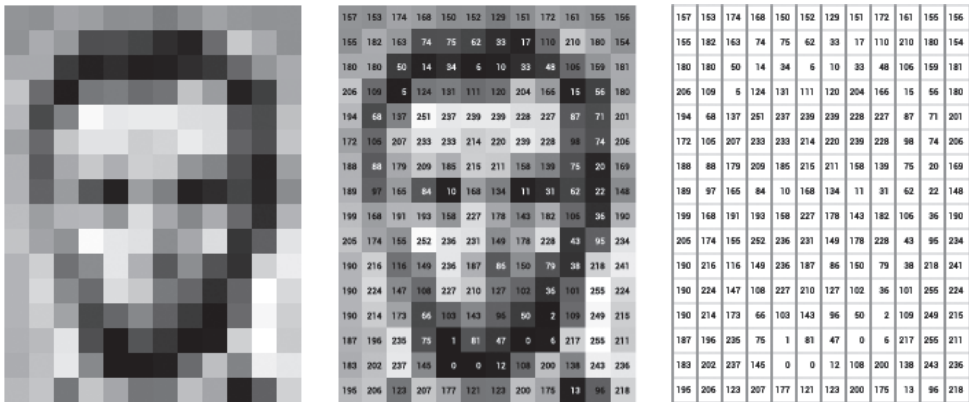


Fig. 1 Abraham Lincoln as a grayscale image with the values on the scale of 0–255

For more nuanced image, one can use grayscale images that allow intermediate levels. 0 still stands for black, but instead of 1, 255 is used for white, and all numbers in between denote gray values, depending on whether they are closer to white or black. These are so-called 8-bit images, i.e., a combination of eight memory cells (bits) for each pixel ($256 = 2^8$). For finer gradation, one can also use more extensive grayscale and then work with values from 0 to 65,535 ($16\text{-bit} = 2^{16}$) or more. The lowest value is used for black and the highest value for white, and the values in between represent the possible shades of gray.

Color images use a combination of three superimposed grayscale images with an 8-bit scale of 0–255. The color composition (e.g., red-green-blue = RGB) produces the mixed colors for eyes, e.g., violet, brown, orange, or pink. Instead of RGB, other color channels can also be used, e.g., cyan, magenta, and yellow, as in color printers. And, as with grayscale images, a much finer 16-bit scale can be selected instead of an 8-bit scale.

The greater the height and width, resolution or color scale, the more memory an image requires. Image data is often compressed to save space or increase processing speed. Each camera or scanner manufacturer has its own proprietary format (*Raw*). When exported, this format is converted into TIFF, PNG, or JPEG image files. Depending on the compression, Tiff and Png require considerably more storage space than JPEG image files (extension jpg), which are created using a compression algorithm that accepts more loss of information to take up less storage space. This process creates artifacts, which can be easily recognized at high magnification the image looking like bathroom tiles. As long as the image resolution is good (at least 30 pixels for an ‘a’ or ‘κ,’ much more for more complex scripts such as Chinese), jpps are sufficient for layout analysis and text recognition. Automatic paleography or writer identification achieves better results with tiff and png files.

3. Text Coding

In a computer, a text is also stored as sequences of 0s and 1s. In the past, when memory was more expensive, 8 bits were reserved for all variants together for each character, so that computers only knew 256 different values (code points), which could only represent a selection of either Latin, Greek, Cyrillic, or Hebrew, depending on the linguistic or geographical workplace, and was known as the Extended ASCII table. An additional code in the text file indicated which alphabets were meant by the raw numbers in the file. It was thereby possible to work with the local alphabets in Germany, Bulgaria, Israel, or Saudi Arabia, but never with all scripts at the same time. More complex scripts with a total number of characters greater than 256, such as Chinese, were impossible. Furthermore, if the encoding schema was unknown, the only solution was to try out all encodings until the text was displayed legibly. This limitation to 256 different characters was a major challenge for philological work in theology, where Syriac, Arabic, Armenian, Georgian, or Coptic – and sometimes also Akkadian, Egyptian, Ethiopian or Sanskrit – were often used in addition to Greek and Hebrew.

The introduction of the Unicode standard in 1991 has progressed towards solving this problem. Like grayscale images with finer nuances, the memory reserved for each character was initially doubled from 8 bits to 16 bits, allowing $2^{16} = 65,536$ different code points in the table. In 2022, a total of 161 fonts could be defined in a single table using this encoding, also known as UTF-8. Even though the introduction of the

Unicode was a major step forward, difficulties for the digital modelling of historical scripts remain, e.g., there is still no Unicode for the Babylonian vocalization of Hebrew texts, and Egyptian or Akkadian have only been partially standardized.

Humanities scholars should be familiarized with the intricacies of Unicode because the basic questions of script encoding that need to be solved are tricky. The present chapter will return to further questions, like reading order in bidirectional texts that mix Hebrew and Latin, or character combination coding, in the text recognition section.

4. Neural Networks

The rapid progress in automatic document analysis in recent years has five main interdependent causes: Hardware (memory, speed), software (neural networks), training data volumes, mass digitization and open-source policies. Processors have become much faster and can process much larger amounts of data simultaneously thanks to increased memory. Hard disk storage and internet data transmission (fiber optics) are also cheaper and of much better quality than 10 years ago. Mass digitization projects of culturally significant manuscript collections, archives, and libraries have led to a flood of image data. The interest of large corporations in processing large amounts of written and oral text (Google Books, YouTube, Netflix, Zoom) has not only improved existing algorithms but also developed new ones. Some of these algorithms are freely available to the public in open-source packages of the most important programming languages (e.g., *pytorch* from Facebook, *TensorFlow* from Google). There are research projects that have published their training data under open licenses, thus enabling others to use them to develop or optimize new algorithms.

Different forms of artificial neural networks are used for almost all stages of automatic document analysis. The basic principle has been known since Rosenblatt in 1958, but it was the above-mentioned constellation of simultaneous progress in hardware, software, data, and open source that led to their success (starting with Jürgen Schmidhuber and Yann LeCun's works in the early 1990s). The common principle is a very complex formula with thousands, millions, or even billions of parameters that are optimized by the computer in a learning process called *training*. The result of a trained network architecture is called a *model* because it mathematically models the problem (Fig. 2).

To a certain extent, artificial neural networks imitate the way brains work. The three most relevant network types at present are *Convolutional Neural Networks* (CNN), *Recurrent Neural Networks* (RNN) and Transformers. The most important common principle is the inclusion of context for each data point. CNNs are particularly interesting for images because a data point (i.e., a pixel) is considered in the context of a rectangle. For example, the computer can learn abstract concepts such as curves of

different curvature and lines at different angles and orientations and combinations. RNNs are interesting for sequences such as audio recordings or texts, as they are flexible in learning how much context to include for a certain phenomenon. For longer sequences, *Long-Short Term Memory Neural Networks* (LSTM), are often used. In deep learning, several layers of CNNs and/or RNNs are combined, resulting in complex architectures that require large amounts of data and often a lot of time to train. At the same time, these combinations also deliver excellent results, both for layout analysis and for transcription as well as many other tasks like classification.

Transformers have become commonplace through recent *Large Language Models* (LLMs), such as BERT (*Bidirectional Encoder Representations from Transformers*) and GPT (*Generative Pre-trained Transformer*). With the right configuration and an optimal learning process, they can learn to perform layout analysis simultaneously with transcription, recognize proper names (Named Entity Recognition), translate, or summarize the resulting texts or even answer questions.

When training, a distinction is made between supervised, self-supervised, and unsupervised training processes. In the supervised training process, the computer is shown *questions* with the corresponding *answers* and tries to optimize the parameters of the formula to arrive at the answer from the question step by step. Question/answer pairs depend on the learning objective and can be very different depending on the task: (1) the image of a line of text and the corresponding transcription; (2) the image of a manuscript page and the corresponding polygons of the layout; or (3) the image of a book page and the corresponding print type. Both question and answer are represented as a number (scalar), number sequence (vector), matrix or tensor etc., since computers know nothing else. At the beginning, all parameters are often initialized randomly. After each learning step (question/answer calculation), the distance of the calculated answer to the correct one is measured and the parameters are adjusted so that the next time the computer is confronted with the same or similar question, the calculated answer is closer to the correct one. If the distance gradually diminishes (and it is not always the case), then the model is said to converge. At regular intervals, the computed formula is subjected to a test in which it is presented with question/answer pairs without the computer learning from them, i.e., without changing the formula, but only determining the current precision of the current model. The specialist tries to formulate the network architecture so that that it can achieve the best models with as little training material as possible and in the fastest possible computing time. Training is usually terminated when the user determines from the test results that the computer is not improving any further. Finally, the model that delivered the best results in the comparison is saved.

The training process is not uniform. Consider the following illustration: a spaceship is supposed to fly as far as possible through a complex labyrinth of caves but is only allowed to make turns at a certain angle and must then fly a certain distance in this direction. If the selected distance is too short, the spaceship will crawl through the large opening space without finding one of the entrances to the labyrinth. If the

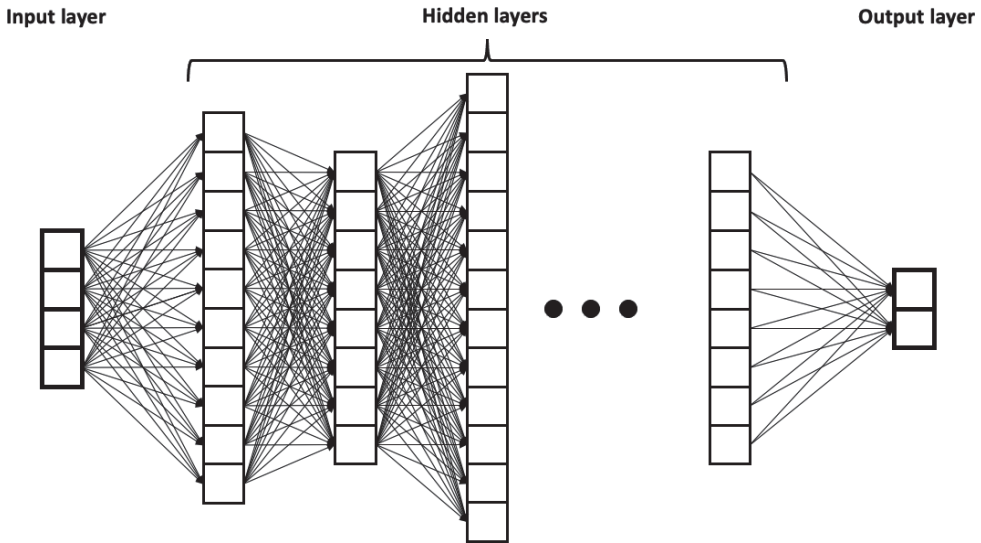


Fig. 2 Example diagram of a neural network with a one-dimensional input layer (with four variables) that leads through a complex network to an output layer with two variables. Each of the lines is a mathematical operation, the parameters of which must be optimized.

distance is too long, the spaceship will not be able to pass through tight turns. Angle and distance are among the *hyperparameters* of the training process, perhaps best compared to the *learning rate*.

In the *self-supervised* training process, the computer automatically calculates the answers from the questions. For example, the computer is shown many images of lines of handwriting, some of which are blacked out, and is asked to suggest a substitute image of what the blacked-out section of the line might have looked like. The model is optimized by analyzing the difference between the computer's suggestion and the original image. In this case, the computer gradually learns the principle of which pixel clusters are usually located between which other pixel clusters. Self-supervised training is sometimes used as pre-training before supervised training because it allows basic principles to be learned beforehand and reduces the amount of training data with manually generated responses.

One potential danger is overfitting. We can compare this to a student who memorizes the answers to the practice questions without understanding the underlying principle. The student can answer the practice questions almost perfectly but is unable to solve unseen questions. The same can happen, for example, if the training corpus is not adapted to the network architecture (too little training material for a question that is too complex) or if the learning rate at which the computer tries to adapt the parameters after each learning process is set too high or too low. Of course, we have only been able to describe a few hyperparameters here.

5. Existing Programs

Until recently, most companies and individual researchers interested in OCR worked with commercial programs like *ABBYY Finereader*, which was very successful with modern English or German texts but could not handle handwritten material or rarer printed texts such as the Syriac alphabets, for which the market seemed too small. Google's widely used open-source program *Tesseract* is only recommended for OCR but not for HTR.¹ Currently, there are several successful programs used for mass digitization of handwritten material.

Since 2016, *Transkribus* has enabled the automatic layout analysis and transcription of written objects, the manual correction of layout analysis and recognition, as well as the training of own transcription models based on the entered data with excellent results via a complex JAVA app or a simplified web app (Kahle et al. 2017, Mühlberger et al. 2019). The program, originally developed in several European research projects, was commercialized in 2019 in the form of a European cooperative.² Currently, users pay per page for automatic layout analysis and/or automatic text recognition. The platform and trained models are therefore closed-source. Other commercial programs include *Ocelus* and *Calfa* (Vidal Gorène 2021a).

In open-source, *OCRopus/ocropy*, developed by Thomas Breuel (2008), was a decisive step forward. Although, as the name suggests, it was only developed for OCR, our Paris team, with the help of Marcus Liwicki, has also been using it for handwriting recognition since 2015. However, programming knowledge was a prerequisite for use. There was only a very rudimentary way to enter transcriptions, only for even lines and only very simple segmentation. Since 2018, *eScriptorium* has been developed around Benjamin Kiessling's *Kraken* (Kiessling et al. 2019, Stokes et al. 2021). It is currently the only open-source program for handwriting analysis with an ergonomic user interface for layout and transcription correction as well as text alignment. It can be installed directly on Linux, Mac OS, and Windows computers using WSL (*Windows Subsystem for Linux*). If a team wants to collaborate on the same document(s), a server is required. A GPU with sufficient RAM is needed to train layout or transcription models.

1 FAQ: <https://tesseract-ocr.github.io/tessdoc/FAQ.html#can-i-use-tesseract-for-handwriting-recognition> (Accessed: 15 June 2024).

2 See <https://readcoop.eu/a-short-history-of-transkribus-with-gunter-muhlberger> (Accessed: 15 June 2024).

6. Layout Analysis

Computerized layout analysis has two objectives. First, computerized analysis is – thus far – a necessary step before text recognition. Second, the layout contains essential information for the hierarchy, reading order (see below), different text types, differentiation between image and text, etc. Layout analysis is crucial in text comprehension even after text recognition.

Previously, morphological operations were used for layout analysis to recognize different text blocks and lines. Currently, this is accomplished by neural network architectures (Fig. 3) that manage both the segmentation of regions and their division into types (column, header, marginalia, illustration, table, apparatus, etc.) as well as the recognition of lines and their division into types (main text line, interlinear line), and the writing direction (horizontal, vertical, upside down). A segmentation ontology determines which region and line types can be used for which phenomena. Incidentally, zones do not necessarily have to be text regions. Users can also use image segmentation to locate library stamps, coins, illustrations, etc.

There are currently two different approaches to region segmentation. One approach uses principles for object recognition such as traffic lights or signs in self-driving cars (Clérice 2022). This approach works very well for text objects with only rectangular regions that have been digitized in a precise fashion, e.g. without tilting, sheering or rotation. However, problems quickly arise with more complex layouts, e.g., L-shaped regions, or with small rotations.

The other approach is a pixel classifier (Kiessling 2020). All image pixels are assigned to one or more desired types of regions. The pixel cloud is then determined for each region type and one or more polygons are reconstructed. This approach better manages complex layouts or rotated digitized images but has difficulties assigning pixel groups of the same type that are very close to each other to the same polygon. The approach therefore tends to classify two closely spaced main text columns as a single zone.

Line segmentation occurs simultaneously to or after region segmentation (Grüning 2017). There are also two approaches in line segmentation. Either a neural network is trained first to detect the baseline and writing direction of each line and then calculate a polygon that surrounds this baseline so that all ink traces of the characters in this line, including any dots and dashes above or below them, are included. Or the neural network is trained to recognize the line polygon directly and then derive the writing direction.

If the training data is homogeneous and numerous enough, in *kraken/eScriptorium*, very complex segmentation models can be trained with 20 different region and line types (Stökl Ben Ezra 2022b). Training simple specific segmentation models is possible with just a few training pages.

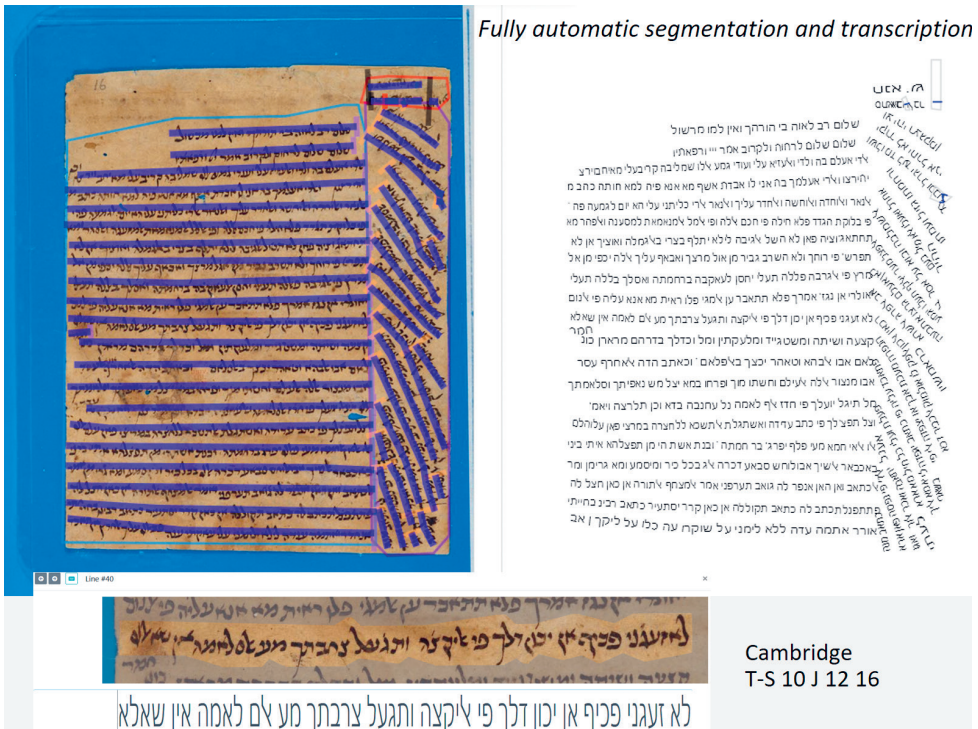


Fig. 3 Fully automated layout segmentation and transcription of the Genizah fragment Cambridge, T-S. 10 J 12 16 with *eScriptorium*, created in the HTR4PGP project.

7. Reading Order

Different text types diverge in their reading order. There are very different ways to read texts, such as the sections of one or more newspaper pages, a critical edition, a bilingual edition, a table, a manuscript with a basic text or with commentaries, a letter, or postcards. Poetic texts are often written stychographically looking like two columns of a prose text. Reading order analysis is closely related to layout analysis. In *kraken*, layout analysis can currently be trained in a version still under development, the integration of which into *eScriptorium* will take a few more months to design and implement the necessary ergonomic user environment (for the method, see Quirós 2022).

8. Computational Paleography

One area in which neural networks come very close to or even surpass humans is the classification of objects. Computational paleography can be used for a wide variety of purposes. Objects can be classified according to the types of printing or fonts used, schools, or scribes, or even dated or geographically located (Seuret et al. 2021, Droby et al. 2022, Popovic et al. 2021). Banks have been using this technology to verify signatures for a long time. Object classification can also be used to analyze whether a handwriting was written by one or more person. The Friedberg Genizah Project used these techniques early on, where users searching for similar fragments to fragment X, which may originate from the same codex, were suggested other candidates from the approximately 300,000 fragments by the system, which led to a large number of “joins” (Wolf et al. 2010).

9. Text Recognition

Text recognition previously required isolating each character. Modern neural networks, however, are based on entire lines of text and thereby achieve better results, as the context of each data point can be taken into consideration. Furthermore, when recognizing handwriting, isolating letters is difficult as such, and even the correct deciphering of whole words depends on the context of the preceding and following words and characters.

Current state of research does not permit to determine precisely how many lines are needed to train a model, or how often a character must occur in the training data set for the computer to learn it. Learning the lines and characters depends on factors like the complexity of the network, the number of different characters to be learned, legibility and uniformity of the script, contrast, uniformity of orthography and vocabulary, image quality, among others. When creating the training corpus, strict homogeneity of the data must be ensured, especially in the case of collective annotation or the affiliation of different projects. The same phenomenon should always be transcribed in the same way. More training material is required if the scholar wants to teach the computer to resolve abbreviations or to transcribe orthographic variants in a normalized way. Finally, in principle, the computer can also be trained to differentiate between different characters, e.g., between italics, bold and normal, or between different font types.

Usually, a preliminary analysis of the final purpose is worthwhile. If visually similar characters are encoded differently, for the computer they are just as apart as visually completely diverse characters are for humans. Do all different types of quotation marks (“”«») really have to be differentiated in exactly the same way, or can some or

all of them be standardized? Depending on the project, it may prove advantageous to go into the one or the other direction.

Here it might be helpful to return our attention to Unicode. Some letters appear multiple times in the Unicode table. E.g., there is ‘s’ and ‘ſ’. Using these and other subtle differences, an allographic or hyper-diplomatic transcription can be trained to differentiate between allographs. There are thus fundamental questions that every edition project (but also the Unicode Consortium) must answer: what is the same character, where are differences mandatory, and where should the project give users the freedom to differentiate for themselves?

A circle can be a Latin ‘O’ or ‘o,’ a Greek ‘O’ or ‘o,’ a Hebrew ם (a letter pronounced as ‘s’), or an Arabic ٥ (the sign for ‘5’). What a circle is in a historical document depends on the context. Sometimes the Unicode table distinguishes between different alphabets, but not always. For Latin, English, and German, the same encoding is used for ‘O’. ‘Ö’ is either an ‘O’ combined with a Trema “” or a single character ‘Ö’. They cannot be visually distinguished. But if one half of the training data has been created with the combination of ‘O’ + “” and the other half with ‘Ö’, then the network becomes confused, since for the computer these are completely different entities. With Greek accents or Hebrew diacritics and vowels, there is also the sequence, because sin, sin-dot, dagesh, and a kamatz can be written in 24 different permutations. Therefore, Unicode includes the possibility of a code point normalization, which (a) either decomposes all characters as much as possible and then puts them in the same order (NFD – decomposed), or (b) combines all characters in the same order (NFC – combined).

Some visually identical characters appear several times in the Unicode table. Latin capital ‘A’ and Greek ‘Α’ (“Alpha”) are visually identical but are represented in the computer by two different encoding points. For the graphically identical letters in Greek and Coptic, there was originally a single common code point for each in the Unicode table. It was not until version 4.1 in 2005 that separate code points were introduced for each.³ Most of the digits in Persian and Arabic are also identical in appearance, yet they are all encoded twice in the Unicode table, once for Persian, and once for Arabic.

When planning a project, it can pay off to start with simple texts, or even with existing digital texts than with more complex or interesting or non-transcribed texts. Often the best starting point is to train a base model for all documents in a script type capable of transcribing as many scribal hands as well as possible, then apply this base model to a few pages of a new manuscript, correct these pages and then retrain (*fine-tune*) this base model with this new data to optimize it for this particular manuscript. If transcriptions already exist elsewhere, then a large training corpus can be created even faster with text-to-text alignment.

Neural networks trained with purely visual information learn only a very primitive language model linked to the probabilities of a certain character appearing be-

3 See https://en.wikipedia.org/wiki/Greek_and_Coptic (Accessed: 15 June 2024).

tween two or more other characters. However, a more complex language model can also be added during training or as a separate post-OCR step. However, doing so involves the risk of hypercorrection in the case of historically variable orthographies, especially when using language models trained on modern texts.

There are now many transcription projects involving more and more languages and scripts. Perhaps the largest project relevant to theology to date is the ERC Synergy project MiDRASH, which has begun in October 2023 and aims to analyze and transcribe a large part of the approximately 100,000 digital Hebrew manuscripts collected in the KTIV project of the National Library of Israel. Many projects are now uploading their data to the *HTRUnited* catalog.⁴ In addition to the projects and publications already mentioned, there are also patristics projects on Greek texts in Berlin (von Stockhausen) and on Coptic texts in Berlin (Lincke 2019, 2021), Oklahoma (Schroeder), and Tokyo (Miyagawa 2018, 2019, 2021). Much work on Armenian and Georgian manuscripts and prints is being done on CALFA (Vidal Gorène 2021). For other data sets, see also Nikolaidou et al. 2022. Bullinger's correspondence is being analyzed in a project led by Tobias Hodel (Scius-Bertrand 2023, Ströbel 2023).

10. Text2Image Alignment

Text2image alignment is indispensable for paleographic studies as well as for digital editions. This type of alignment calculates the approximate regions for each letter and word in a transcription line. With certain neural networks, this approximation is part of the automatic transcription. However, this information is lost when the automatic transcription is corrected manually. With text-image alignment the approximate positions of the letters and words can be recalculated retrospectively. There are current projects underway on the Books of Hours by Dominique Stutzmann (Hazem 2020), on Qumran (Stökl Ben Ezra et al. 2020), on rabbinic texts (Stökl Ben Ezra, in press), and on the Hebrew Bible (Bambaci et al. 2023, Stökl Ben Ezra et al. 2021).

11. Text2Text Alignment

A practical method of using existing high-quality electronic texts for the creation of training data is *text2text* alignment. After correcting an automatic layout analysis, the best existing text recognition model is applied. The computer then calculates how best to align the electronic text with the faulty automatic transcription and swaps the latter with the former. If there are no line or page breaks, then the computer inserts

4 See <https://htr-united.github.io> (Accessed: 15 June 2024).

them.⁵ The *Text2Image* tool in *Transkribus* works in a similar way. The further connection with entire text corpora will allow the automatic creation of training material (Smith 2023).

12. From HTR Platform to Edition

The way from an automatic transcription to a digital edition is not (yet) clear; there are still stumbling blocks in the way. In editions, interlinear and marginal improvements are marked with brackets as additions but integrated into the running text. In transcription platforms, on the other hand, these are in separate lines. A continuous text is essential for the use of collation programs for critical editions of texts with several manuscripts. A pipeline for Hebrew texts promises initial approaches to solutions (Stökl Ben Ezra 2022a).

Literature Cited

- Bambaci, L., Stökl Ben Ezra, D. (exp. 2024). Enhancing HTR of Historical Texts through Scholarly Editions. A Case Study from an Ancient Collation of the Hebrew Bible. In *Computational Humanities Research Conference. CEUR Workshop Proceedings 2023* (pp. 554–576). Paris: Computational Humanities Research. URL: <https://ceur-ws.org/Vol-3558/paper6310.pdf> (Accessed: 15 June 2024).
- Beinert, W. (2021). Schriftstil. In Id. (Ed.), *Das Lexikon der Typografie*. URL: <https://www.typolexikon.de/schriftstil> (Accessed: 15 June 2024).
- Breuel, Th. (2008). The OCRopus open source OCR System. In B. A. Yanikoglu & K. Berkner (Eds.), *Document Recognition and Retrieval XV, part of the IST-SPIE Electronic Imaging Symposium*, San Jose, CA, USA, January 29–31. DOI: <https://doi.org/10.1117/12.783598> (Accessed: 15 June 2024).
- Droby, A., Irina, R., Vasyutinsky-Shapira, D., Kurar-Barakat, B., & El-Sana, J. (2022). Digital Hebrew Paleography. Script Types and Modes, *Journal of Imaging*, 8(5.143), 1–22. DOI: <https://doi.org/10.3390/jimaging8050143> (Accessed: 15 June 2024).
- Genette, G. (1982). *Palimpsestes. La littérature au second degré*. Paris: Éditions du Seuil.
- Grüning, T., Leifert, G., Strauß, T., Michael, J., & Labahn, R. (2019). A two-stage method for text line detection in historical documents, *International Journal of Document Analysis and Recognition*, 22(3), 285–302.

⁵ See <https://github.com/dasmiq/passim> (Accessed: 15 June 2024).

- Hazem, A., Daille, B., Kermorvant, Ch., Stutzmann, D., Bonhomme, M.-L., Maarand, M., & Boillet, M. (2020). Books of Hours. The First Liturgical Data Set for Text Segmentation. In *Proceedings of the Twelfth Language Resources and Evaluation Conference* (pp. 776–784). Marseille: European Language Resources Association. URL: <https://aclanthology.org/2020.lrec-1.97> (Accessed: 15 June 2024).
- Kahle, Ph., Colutto, S., Hackl, G., & Mühlberger, G. (2017). Transkribus. A Service Platform for Transcription, Recognition and Retrieval of Historical Documents. In *14th IAPR International Conference on Document Analysis and Recognition* (pp. 19–24). Kyoto: IEEE Xplore. DOI: <https://doi.org/10.1109/ICDAR.2017.307> (Accessed: 15 June 2024).
- Kiessling, B., Tissot, R., Stokes, P.A., & Stökl Ben Ezra, D. (2019). eScriptorium. An Open Source Platform for Historical Document Analysis. In *International Conference on Document Analysis and Recognition Workshops (ICDARW)* (pp. 19–24). Sydney: IEEE Xplore. DOI: <https://doi.org/10.1109/ICDARW.2019.10032> (Accessed: 15 June 2024).
- Kiessling, B. (2020). A Modular Region and Text Line Layout Analysis System. In *17th International Conference on Frontiers in Handwriting Recognition (ICFHR)* (pp. 313–318). Dortmund: IEEE Xplore. DOI: <https://doi.org/10.1109/ICFHR2020.2020.00064> (Accessed: 15 June 2024).
- Lincke, E.-S., Bulert, K., & Büchler, M. (2019). Optical Character Recognition for Coptic fonts. A multi-source approach for scholarly editions. In *DATeCH2019 – Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage* (pp. 87–91). New York: Association for Computing Machinery. DOI: <https://doi.org/10.1145/3322905.3322931> (Accessed: 15 June 2024).
- Lincke, E.-S. (2021). The State of the affairs in optical character recognition (ocr) for Coptic. In C. G. Zamacona & J. Ortiz-García (Eds.), *Handbook of Digital Egyptology. Texts* (pp. 139–164). Alcalá de Henares: Universidad de Alcalá.
- Maier, A., Christlein, V., Breininger, K., Würfl, T., Unberath, M., & Riess, Ch. (2021). Bildanalyse. In G. Görz, U. Schmid & T. Braun (Eds.), *Handbuch der Künstlichen Intelligenz* (pp. 673–712). Berlin/Boston: De Gruyter. DOI: <https://doi.org/10.1515/9783110659948-015> (Accessed: 15 June 2024).
- Miyagawa, S., Bulert, K., Büchler, M., & Behlmer, H. (2019). Optical character recognition of typeset Coptic text with neural networks, *Digital Scholarship in the Humanities*, 34(1), 35–41. DOI: <https://doi.org/10.1093/llc/fqz023> (Accessed: 15 June 2024).
- Miyagawa, S., Zeldes, A., Büchler, M., Behlmer, H., & Griffiths, T. (2018). Building Linguistically and Intertextually Tagged Coptic Corpora with Open Source Tools. In Suzuki, Ch. (Ed.), *Proceedings of the 8th Conference of Japanese Association for Digital Humanities. Leveraging Open Data*. (pp. 139–141). Tokyo: Center for Open Data in the Humanities.

- Miyagawa, S. (2020). Digitization of Coptic Manuscripts and Digital Humanities. Tools and Methods for Coptic Studies, *The International Journal of Levant Studies*, 2, 29–61.
- Mühlberger, G., Seaward, L., Terras, M., Ares Oliveira, S., Bosch, V., Bryan, M., Colutto, S., Déjean, H., Diem, M., Fiel, S., Gatos, B., Greinoecker, A., Grüning, T., Hackl, G., Haukkovaara, V., Heyer, G., Hirvonen, L., Hodel, T., Jokinen, M., Kahle, Ph., Kallio, M., Kaplan, F., Kleber, F., Labahn, R., Lang, E. M., Laube, S., Leifert, G., Louloudis, G., McNicholl, R., Jean-Meunier, L., Michael, J., Mühlbauer, E., Philipp, N., Pratikakis, I., Puigcerver Pérez, J., Putz, H., Retsinas, G., Romero, V., Sablatnig, R., Joan-Sánchez, A., Schofield, Ph., Sfikas, G., Sieber, Ch., Stamatopoulos, N., Strauß, T., Terbul, T., Toselli, A. H., Ulreich, B., Villegas, M., Vidal, E., Walcher, J., Weidemann, M., Wurster, H., & Zagoris, K. (2019). Transforming scholarship in the archives through handwritten text recognition, *Journal of Documentation*, 75(5), 954–976.
- Nikolaidou, K., Seuret, M., Mokayed, H., Liwicki, M. (2022). A survey of historical document image datasets, *International Journal of Document Analysis and Recognition*, 25(4), 305–338.
- Quirós, L., & Vidal, E. (2022). Reading order detection on handwritten documents, *Neural Computer Applications* 34(12), 9593–9611.
- Rosenblatt, F. (1958). The perceptron. A probabilistic model for information storage and organization in the brain, *Psychological Review*, 65(6), 386–408. DOI: <https://doi.org/10.1037/h0042519> (Accessed: 15 June 2024).
- Scius-Bertrand, A., Ströbel, Ph., Volk, M., Hodel, T., & Fischer, A. (2023). The Bullinger Dataset. A Writer Adaptation Challenge. In *ICDAR 2023. Document Analysis and Recognition. Conference Proceedings*, 1 (pp. 397–410). San José: Springer. DOI: https://doi.org/10.1007/978-3-031-41676-7_23 (Accessed: 15 June 2024).
- Seuret, M., Nicolau, A., Rodríguez, D.-S., Weichselbaumer, N., Stutzmann, D., Mayr, M., Maier, A., & Christlein, V. (2021). ICDAR 2021. Competition on Historical Document Classification. In Lladós, J., Lopresti, D., Seiichi, U. (Eds.), *Document Analysis and Recognition. ICDAR 2021* (pp. 618–634). Lausanne: Springer. DOI: https://doi.org/10.1007/978-3-030-86337-1_41 (Accessed: 15 June 2024).
- Smith, D., Murel, J., Parkes-Allen, J., & Miller, M. T. (2023). Automatic Collation for Diversifying Corpora. Commonly Copied Texts as Distant Supervision for Handwritten Text Recognition. In *Computational Humanities Research Conference. CEUR Workshop Proceedings 2023* (pp. 206–221). Paris: Computational Humanities Research. URL: <https://ceur-ws.org/Vol-3558/paper1708.pdf> (Accessed: 15 June 2024).
- Stokes, P., Kiessling, B., Tissot, R., Gargem E., & Stökl Ben Ezra, D. (2021). The eScriptorium VRE for Manuscript Cultures. In C. Clivaz, & G. V. Allen (Eds.), *Ancient Manuscripts and Virtual Research Environments* (no. pag) [= *Classics@Journal* 18(1)]. URL: <https://classics-at.chs.harvard.edu/classics18-stokes-kiessling-stokl-ben-ezra-tissot-gargem> (Accessed: 15 June 2024).

- Stökl Ben Ezra, D., Brown-DeVost, B., Dershowitz, N., Pechorin, A., & Kiessling, B. (2020). Transcription Alignment for Highly Fragmentary Historical Manuscripts. The Dead Sea Scrolls. In *International Conference on Frontiers in Handwriting Recognition* (pp. 361–366). Dortmund: IEEE Xplore. DOI: <https://doi.org/10.1109/ICFHR2020.2020.00072> (Accessed: 15 June 2024).
- Stökl Ben Ezra, D., & Lapin, H. (in print). From HTR to Digital Critical Scholarly Edition. Reflexions on the Use of Machine Learning. Computational and Digital Humanities in the Sofer Mahir Project. In U. Henny-Krahmer et al. (Eds.), *Machine Learning and Data Mining for Digital Scholarly Editions*. Norderstedt: Books on Demand [= *SIDE*].
- Stökl Ben Ezra, D., Lapin, H., Brown DeVost, B., & Jablonski, P. (2022a). HTR2CritEd. A Semi-Automatic Pipeline to Produce a Critical Digital Edition of Literary Texts with Multiple Witnesses out of Text Created through Handwritten Text Recognition. In *Digital Humanities 2022. Responding to Asian Diversity* (pp. 690–691). Tokyo: DH2022 Local Organizing Committee. URL: <https://dh2022.dhii.asia/dh2022bookofabsts.pdf> (Accessed: 15 June 2024).
- Stökl Ben Ezra, D., Rustow, M., & Witty, D. (2022b). Segmentation Mode for Archival Documents with Highly Complex Layout. In *Conference Documents anciens et reconnaissance automatique des écritures manuscrites*. École national des chartes, Paris: YouTube. URL: <https://www.youtube.com/watch?v=dE1XUXiuitU> (7:07–7:30). (Accessed: 15 June 2024).
- Ströbel, Ph., Hodel, T., Fischer, A., Scius, A., Wolf, B., Janka, A., Widmer, J., Scheurer, P., & Volk, M. (2023). Bullingers Briefwechsel zugänglich machen. Stand der Handschriftenerkennung. In A. Busch, & P. Trilcke (Eds.), *DHd 2023. Open Humanities, Open Culture* (pp. 98–102). Belval/Trier: Zenodo. DOI: <https://doi.org/10.5281/zenodo.7688631> (Accessed: 15 June 2024).
- Wolf, L., Littman, R., Mayer, N., German, T., Dershowitz, N., Shweka, R., & Choueka, Y. (2010). Identifying Join Candidates in the Cairo Genizah, *International Journal of Computer Vision*, 94(1), 118–135.

Literature Related

- Camps, J.-B., Vidal-Gorène, Ch., & Vernet, M. (2021). Handling Heavily Abbreviated Manuscripts. HTR Engines vs Text Normalisation Approaches. In E.H. Barney Smith & U. Pal (Eds.), *Document Analysis and Recognition – ICDAR 2021 Workshops. ICDAR 2021* (pp. 306–316). Cham: Springer [= *Lecture Notes in Computer Science*, 12917]. DOI: https://doi.org/10.1007/978-3-030-86159-9_21 (Accessed: 15 June 2024).
- Chagué, A., & Thibault, C. (2023). I'm here to fight for ground truth. HTR-United, a solution towards a common for HTR training data. In *Digital Humanities*

2023. *Collaboration as Opportunity*. Graz: Zenodo. DOI: <https://doi.org/10.5281/zenodo.8107449> (Accessed: 15 June 2024).
- Clérice, Th. (2022). You Actually Look Twice At it (YALTAi). Using an object detection approach instead of region segmentation within the Kraken engine, *Journal of Data Mining and Digital Humanities*, 1–13. DOI: <https://doi.org/10.48550/arXiv.2207.11230> (Accessed: 15 June 2024).
- Perdiki, E. (2023). List of manuscripts containing John Chrysostom’s Homilies and the relevant manual transcriptions, 1(2). *Zenodo*. DOI: <https://doi.org/10.5281/zenodo.8102662> (Accessed: 15 June 2024).
- Popović, M., Dhali, M. A., & Schomaker, L. (2023). Artificial intelligence based writer identification generates new evidence for the unknown scribes of the Dead Sea Scrolls exemplified by the Great Isaiah Scroll (1QIsaa), *PLoS ONE*, 16(4. e0249769), 1–28. DOI: <https://doi.org/10.1371/journal.pone.0249769> (Accessed: 15 June 2024).
- Vidal-Gorène, Ch., Dupin B., Decours-Perez A., & Riccioli T. (2021a). A Modular and Automated Annotation Platform for Handwritings. Evaluation on Under-Resourced Languages. In Lladós, J., Lopresti, D., Seiichi, U. (Eds.), *Document Analysis and Recognition. ICDAR 2021* (pp. 507–522). Lausanne: Springer. DOI: https://doi.org/10.1007/978-3-030-86334-0_33 (Accessed: 15 June 2024).
- Vidal-Gorène, Ch., & Decours-Perez, A. (2021b) A Computational Approach of Armenian Paleography. In E. H. Barney Smith & U. Pal (Eds.), *Document Analysis and Recognition – ICDAR 2021 Workshops. ICDAR 2021* (pp. 295–305). Cham: Springer [= *Lecture Notes in Computer Science*, 12917]. DOI: https://doi.org/10.1007/978-3-030-86159-9_20 (Accessed: 15 June 2024).
- Wick, Ch., Reul, Ch., & Puppe, F. (2018). Comparison of OCR Accuracy on Early Printed Books using the Open Source Engines Calamari and OCRopus, *Journal of Language Technology and Computational Linguistics*, 33(1), 79–96.

Figure Credits

Fig. 1: Abraham Lincoln as a grey scale image. CCo. The image was probably originally created by Leon Harmon in 1971, who wanted to find out how much visual information an image could do without in order to still be recognisable.

Fig. 2: Wikipedia © “BrunelloN” CC-BY-SA 4.0

Fig. 3: Screenshot Daniel Stökl Ben Ezra. Manuscript © CC-BY Cambridge University Library

Funded by the European Union (ERC, MiDRASH, Project No. 101071829). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.