# HDPA: Historical Document Processing and Analysis Framework

**Ladislav Lenc**[2] · **Jiří Martínek**[1] · **Pavel Král**[1,2] · **Anguelos Nicolao**[3] · **Vincent Christlein**[3]

Accepted for Evolving Systems journal

**Abstract** Nowadays, the accessibility of digitized historical documents is extremely important to facilitate fast and efficient retrieval of historical information and knowledge extraction from such data. To provide such functionality, it is necessary to convert document images into plain text using optical character recognition (OCR). Many OCR related methods and tools have been proposed, however, they are often too complicated for a standard user, some important parts are missing or they are not available in free versions.

Therefore, this paper describes a complex and flexible web framework for historical document manipulation and analysis with the main focus on OCR. The framework contains eight modules to facilitate three main tasks: image preprocessing and segmentation, creation of data for OCR model training and the OCR itself. This framework is freely available for non commercial purposes.

We have experimentally evaluated this framework on real data and we have shown that this system is efficient and can save human labour in the process of annotated data preparation. Moreover, we have reached state-of-the-art OCR results.

**Keywords** CNN · Document Analysis · Framework · Historical Documents · LSTM · Neural Networks · OCR

[1] Dept. of Computer Science & Engineering
Faculty of Applied Sciences
University of West Bohemia
Plzeň, Czech Republic

[2] NTIS - New Technologies for the Information Society
Faculty of Applied Sciences
University of West Bohemia
Plzeň, Czech Republic
{llenc,jimar,pkral}@kiv.zcu.cz

[3] Pattern Recognition Lab
Friedrich-Alexander-Universität Erlangen-Nürnberg
Erlangen, Germany
vincent.christlein@fau.de

## 1 Introduction

A significant number of historical documents are already digitized and stored in archival databases and portals. A crucial task consists in making such documents easily accessible to historians, archival researchers, and to the general public for information retrieval and knowledge extraction. First, the document images are converted into plain text using optical character recognition (OCR) eventually handwritten text recognition (HTR).

Several OCR methods and tools have been proposed, however, to the best of our knowledge, none of them provides a complex functionality for free and it is complicated to adjust and train by standard users.

Therefore, the main contribution of this work consists in the proposition of a novel web framework for historical document image manipulation and analysis which is freely available for non commercial purposes at `http://ocr-corpus.kiv.zcu.cz/`. It provides a set of sophisticated functions for processing document images with the main focus on OCR of printed materials. This framework is widely applicable for various types of users. For a programmer /researcher, it opens possibilities for improvement of the presented modules. The framework also allows to simplify the work of annotators by predicting almost perfect outputs. Last but not least, for historians and other archival workers, the framework offers a simple way of uploading a new set of images and, in the sequel, they can review the results. The main motivation though remains a fast and efficient information retrieval.

The web application is written in Django, which allows us to create a pipeline of individual python scripts (modules). The modules can run separately and Django represents the role of a sandbox that connects the user interface with the results of desired modules. Modules are stored in `app/libs` folder and Django per se is responsible for running a relevant set of scripts and displaying results using HTML.

The proposed HDPA framework contains functional units that facilitate three main tasks. The first unit handles image pre-processing and segmentation. The second one provides tools for creation of data (ground truth) for OCR model training. The last functional unit is the OCR engine. The main strength of the framework lies in its extensibility. Our goal is to facilitate an easy integration of new modules. This way, the users can easily customize the framework for their specific needs.

We focus on processing of printed historical German newspapers from the end of the nineteenth century printed in the *Fraktur* script. The newspaper pages have a variable layout and the scans usually suffer from several quality issues such as noise, skew, warped lines and even missed character parts. The above mentioned aspects make the document analysis including OCR of such documents very challenging.

The framework currently provides the user with two basic pre-processing modules, namely “Image Binarizer” and “Page Deskewer”. The goal of the binarization process is to remove noise and to increase the contrast between foreground and background pixels. It is also a prerequisite for some subsequent tasks. Page deskewing ensures that the page is upright and the text lines are horizontal. The user can easily integrate custom specific modules that can be chained with the existing ones.

The training data creator is useful when we desire to train a new OCR model. This functional unit contains utilities that allow the user to prepare a synthetic

dataset containing generated sentences from a requested domain and era. Such data allow the network to learn a specific language model that can improve the resulting accuracy. We also provide tools that serve for the creation of a set of single letter images cropped from real data. From such images, we can compose sentences that are very similar to the real ones. These hybrid data (we call it hybrid according to [23]) can be used for an initial training of the OCR model. Another way of creating synthetic data is the usage of a text image generating tool (generated synthetic data).

Similarly as in [23] we use a two-stage training of the model: 1) Training on large synthetic data which helps to learn basic glyph shapes and also the above mentioned language model; 2) Fine-tuning on a small amount of real text lines which ensures that the model can learn some specific aspects of the real data that cannot be mimicked in the synthetic lines.

To allow creation of ground truths for real data, we also provide a tool for annotating the line images produced by the second functional unit. It incorporates an OCR model that suggests the transcription and the user can correct the errors if necessary.

Commonly, neural networks that process whole text lines are utilized for OCR. Recurrent neural networks (RNN) are usually used for this task [5]. A great benefit of the RNN-based approaches is that a character segmentation is not necessary. Some approaches also combine RNNs and convolutional neural networks (CNN) [30, 26]. In such approaches, the CNN [18] serves as a feature extractor for the recurrent layers. It helps mainly in cases when the data are not well aligned and contain noise. Our system is also based on these principles and uses a CNN for feature extraction and a bidirectional LSTM recurrent neural network for recognition.

The main contribution of this paper is possible to summarize as follows:

- Proposition and development of a novel easily extendable framework for historical document analysis and processing;
- Providing a tool for annotation of the line images for OCR training;
- Integration of state-of-the-art methods into this framework;
- Provision of all source codes and developed corpus for non commercial purposes.

The rest of the paper is organized as follows. The next section summarizes the main tools and methods for image ground truth creation as well as modern OCR methods and systems. Section 3 presents the architecture of the proposed framework including three functional units composed of eight modules. Sections 4–6 detail the individual units, image pre-processing and segmentation, training data creator and OCR engine. Section 7 shows the experiments conducted using the data from the Porta fontium portal. The last section concludes the paper and proposes some further directions.

## 2 Related Work

Transkribus [19, 32] is a complex tool developed within the READ (Recognition and Enrichment of Archival Documents) project[1] at the University of Innsbruck

[1] `https://read.transkribus.eu/about/`

dealing with automated recognition, transcription, and searching of historical documents. It offers a number of tools for automated processing of historical documents, such as handwritten text recognition (HTR), layout analysis, document understanding, writer identification or optical character recognition (OCR). For the OCR, Transkribus utilizes ABBYY Finereader engine 11[2]. A drawback is the necessity to upload a document to the server to be able to process it. Furthermore, to the best of our knowledge, Transkribus has no support for creating any kind of synthetic data.

With the need to master the role of semantic segmentation, fully convolutional neural (FCN) networks have been evolved, characterized primarily by an image to image architecture. Such a network takes an image as an input and outputs another image. This kind of architecture can be considered as a function of pixels that transforms an input set of pixels to the output one. In a nutshell, every output pixel has a label with the information which part of the image it represents. By deploying this idea to the document analysis system, we get a relatively simple approach to classify layout (i.e. every pixel represents either background or foreground – text). The most popular representative of such an FCN is U-Net [27] and its derivatives that are focused on text line detection in documents – e.g. ARU-Net [14]. ARU-Net labels pixels belonging to one of several classes (e.g. baseline, separator etc.) Text line segmentation of historical Arabic documents with a classification of text blocks was presented by Zahour et al. [34]. Although their approach does not utilize a neural network, it achieves interesting results.

Methods and tools for image ground truth creation are absolutely necessary for the acquisition of training data to be able to use the FCN for segmentation. Van Beusekom et al. [33] proposed an interesting method for automated ground truth generation which detects a robust and accurate alignment of a scanned image and corresponding electronic document. It involves printing of an electronic document and scanning it again. Ground truths are created on pixel level and the algorithm is evaluated on the UW3 dataset. The estimated ground truths are compared with the real ones and the authors show that the resulting accuracy is less than one pixel difference.

A powerful tool that can be used for this task is Aletheia [8]. It can automatically detect objects on four levels: regions, text lines, words and glyphs. The outlines of the objects can be adjusted by users who also specify the segmentation of words to single glyphs. The ground truths are stored in the PAGE XML format [24]. Ground truth creation is not the only feature of Aletheia though. It offers a number of automated and semi-automated annotation tools. It is worth noting that the pro version with a complete document analysis system is not free. In our opinion it is also less flexible and there is no possibility to add a custom module.

Another tool which is used for semi-automatic document image analysis is GraphManuscribble [10]. A user-centered segmentation method is utilized in this tool. A sparse representation of the document structure is automatically captured by the graphs and it can be further edited and approved by the user.

TrueViz [16] is a tool for ground truth creation and visualization. This application is freely available for research purposes. It allows multi-lingual text processing and the output format is XML.

[2] `https://www.abbyy.com/`

Many other specific tools and systems have been proposed as for instance for Arabic synthetic data generation and OCR [22] or for Rusian artificial OCR dataset generation [6]. However their the functionality is limited to some particular task and therefore, general usage is not possible. The impact of synthetic data generation is also presented in several studies [9].

Nowadays OCR systems are trained with line images and corresponding ground truth text (labels). Such a label represents a text sequence that is depicted in the line image, but it does not express which part of the image exactly is mapped to the concrete letter of the ground truth label.

Labelling unsegmented sequences can be realized by using connectionist temporal classification (CTC) loss [11]. It gives a probability distribution over all possible label sequences conditioned on the input sequence. The network uses an objective function that maximizes the probabilities of correct labelling, which is differentiable and thus standard backpropagation can be used. This approach was first used for automatic speech recognition, however, it can be directly applied for text recognition as proposed in [13].

There are many examples of well-performing OCR systems that utilize a CNN and/or an RNN. Breuel et al. [5] propose an efficient OCR system based on the LSTM model which is a part of the open-source OCRopus system [4]. The method utilizes a bidirectional LSTM network in combination with a text line normalization, where a dictionary of connected component shapes associated with baseline and x-height information is computed on external annotated data. The baseline and x-height lines are mapped into two straight lines using spline interpolation. The system is applied on printed English and Fraktur texts, where it obtains 0.6 % character error rate (CER) on English data and varies from 0.16 % to 0.82 % on German Fraktur depending on the quality of the scans.

An approach combining CNN and RNN is proposed in [26]. The system utilizes a CNN for feature extraction and an LSTM is then used for sequence modelling. The model is evaluated on handwritten and printed data where such a model performs well for both data types. This work also presents a weighted finite state transducer (WFST) that supplies a language model to the decoding procedure.

## 3 Framework Overview

The framework has a modular architecture which is depicted in Figure 1. It is composed of eight modules (*M1* – *M8*) that can interact with each other. The modules are encapsulated in three functional units ($U1$ – $U3$). Every module can be used separately in order to create another image processing system. First, we briefly introduce each functional unit in this section and then we describe them in detail in the following sections. The framework and also our OCR dataset can be downloaded from `http://ocr-corpus.kiv.zcu.cz/`.

### 3.1 Functional Unit $U1$ – Image Pre-processing and Segmentation

The first functional unit $U1$ deals with image pre-processing and segmentation. The pre-processing includes important image transformations and corrections nec-

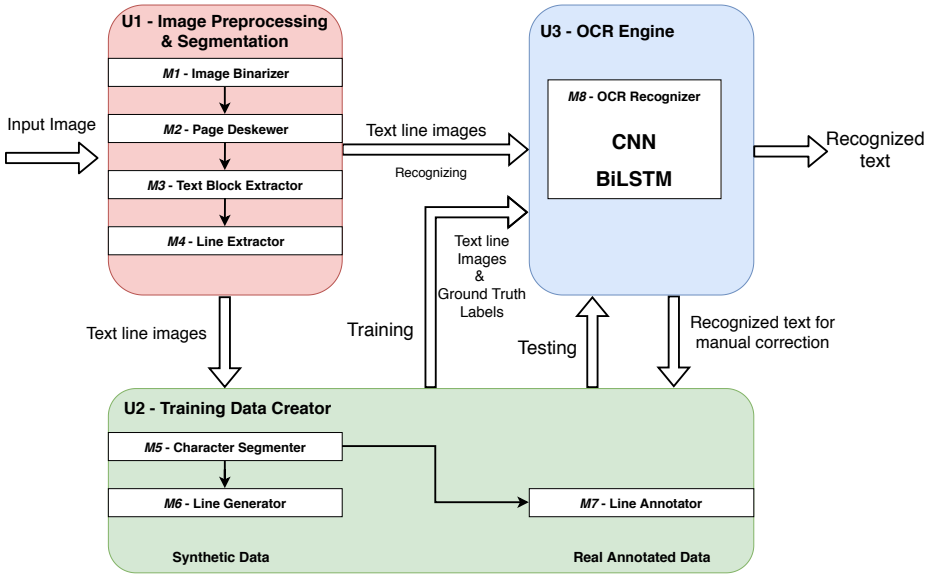



Fig. 1: Architecture of the HDPA framework

essary for a successful layout analysis and segmentation. The goal of this module is to prepare line images that can be fed to the OCR engine for text recognition.

This functional unit comprises four modules. The first module $M1$ converts the document image (color or gray-scale) into a binary representation. Unfortunately, a significant amount of the scanned pages is not horizontally aligned, which would negatively influence the segmentation and consequently also decrease the OCR accuracy. Therefore, the following module $M2$ performs page deskewing.

Modules $M3$ and $M4$ are devoted to the segmentation. We perform the segmentation in two steps. First, we detect and extract text blocks ($M3$) within the straightened page and determine the reading order of the blocks. Then it is significantly easier to detect and extract text lines ($M4$). The output of this module is a list of extracted line images which will be used by the other two functional units.

### 3.2 Functional Unit $U2$ – Training Data Creator

The functional unit $U2$ is used for the creation of training data for the OCR model and it is composed of three modules. In order to train an OCR engine, we need text line images with corresponding labels (ground truths). We can obtain such data in two ways: synthetic data generation and annotation of real extracted line images.

This framework allows creating two types of synthetic data. The first one, called *hybrid*, is created by putting together images of single characters (cropped from real document images) according to some meaningful text. We first segment a text line image into individual characters (module $M5$) to prepare several image

representations for each character. Then, according to a relevant historical text, we create text line images by concatenating the respective character images ($M6$).

The last module $M7$ belonging to this functional unit is used to create annotations (ground truths) for real line images. Initially, if no OCR model is available, it is just a simple annotation tool with GUI that allows the user to fill in the text rendered in the image. After annotating an initial set of line images, it is possible to train an OCR model and deploy it to the module. The module thus gets the ability to predict the text contained in the line image and suggests the transcription to the user who can correct it if necessary.

### 3.3 Functional Unit $U3$ – OCR Engine

This unit is the OCR engine itself. The engine utilizes a line-based approach that processes images of extracted text lines and outputs the estimated text sequence. We build on the combination of a CNN that is used as a feature extractor and an RNN which translates the sequence of image frames into a sequence of characters. In combination with the CTC loss, it can be trained to predict the text contained in the line images. The module $M8$ handles both the training and recognition phases of the engine.

## 4 Image Pre-processing and Segmentation

$M1$ - Image Binarizer

Image binarization can be considered as a pixel labelling problem [1]. It is defined as a function $f$ which maps the pixel intensities of the input image $I$ to values 0 or 1 into the binary output image. The simplest form of the function $f$ is a global thresholding defined as follows:

$$f(I) = \begin{cases} 1 & \text{if } I(x,y) >= T \\ 0 & \text{if } I(x,y) < T \end{cases} \tag{1}$$

where $T$ is the threshold value and $x$ and $y$ are the pixel coordinates. More sophisticated approaches use various variants of global or local thresholding.

Our binarization module employs the adaptive thresholding method proposed by Sauvola [28]. The method first classifies the image contents into several classes and then it applies two approaches for determining a threshold for each pixel. The approaches are combined to obtain the final result.

$M2$ - Page Deskewer

When a digitized document is scanned imperfectly and text lines are not horizontally aligned, the segmentation and the OCR becomes less accurate. The goal of this module is to find the document skew angle and rotate the image correspondingly. We have implemented a basic method based on horizontal projection profiles (HP) [25]. It rotates the image by different angles in a range $\theta_{min}$ to $\theta_{max}$ and

maximizes a criterion function calculated from the projection profile values. $HP$ of an image $I$ is defined as:

$$HP(y) = \sum_{x=0}^{w-1} I(x, y) \tag{2}$$

where $w$ is the width of the input image.

The criterion function for a particular angle is defined as follows:

$$c(\theta) = \sum_{y=1}^{h-1} (HP(y) - HP(y-1))^2 \tag{3}$$

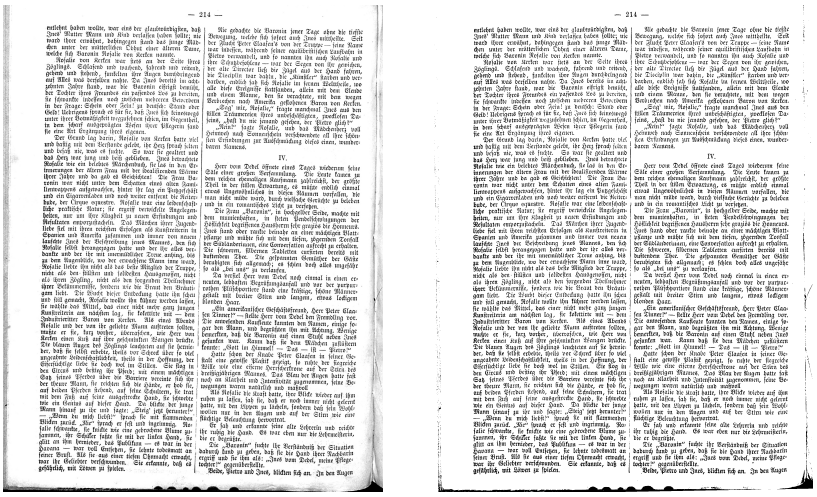
where $h$ is the height of the image.

The skew angle of the document is the angle $\theta^*$ that maximizes the function $c(\theta)$.

$$\theta^* = \arg\max_{\theta} c(\theta) \tag{4}$$

We have chosen this method because of its simplicity and efficiency.

The output of modules $M1$ and $M2$ is depicted in Figure 2.



(a) Input page (b) Binarized and rotated page

Fig. 2: Output of the pre-processing modules

$M3$ - Text Block Extractor

The goal of this module is to detect and extract text blocks. For this task, we have utilized the well-known fully convolutional network U-Net [27]. Although U-Net has been developed for semantic segmentation of medical images, it is possible

to apply this architecture to a different segmentation task such as text block extraction.

The FCN networks are composed of encoder and decoder parts. The encoder comprises a set of convolutional and pooling layers. In the decoder part, deconvolutions are used to upsamle the image to the original size. The convolution operation is defined as:

$$(f * h)[m, n] = \sum_j \sum_k h[j, k] f[m - j, n - k] \tag{5}$$

where $f$ is the input image and $h$ is the convolutional kernel. Indices $m$ and $n$ are concerned with the image matrices while $j$ and $k$ are the kernel indices.

We have trained the U-Net in two phases with slightly different training data. The first training dataset used for initial parameter settings was the Europeana Newspapers Project Dataset [7]. It is a set of historical newspapers created with the goal to provide all the challenges related to the historical document image processing task. The dataset contains more than 500 newspaper pages associated with ground truths containing full transcribed text, layout information and reading order. From this set we have selected a subset of 95 pages mostly written in German with varying layouts to address our real annotated dataset.

Within the second phase, we have utilized a dataset created from documents provided by the *Porta fontium*[3] project which aims at digitizing archival documents from the Czech-Bavarian border area. We have created an OCR and page layout analysis dataset from one selected newspaper, namely "Ascher Zeitung", printed in the second half of the nineteenth century. The dataset contains 25 pages. We have selected a representative set of pages with all types of layouts occurring in this newspaper.

Neither of these datasets have the appropriate ground truth in the form of the image mask (see Figure 5) which is necessary for U-Net training. The mask is a binary image, where the value 1 (white colour) represents a pixel that is within the text area and the value 0 (black colour) is a pixel beyond these areas. Properly trained U-Net should provide us with a predicted image mask similar to the ground truth image. Figures 3 and 4 show the U-Net architecture and the scheme of the learning process.

An example of the output mask provided by the trained U-Net is depicted in Figure 6. We can easily determine bounding boxes by a connected components analysis and render it to the original image (see Figure 7). Ideally, the model should exclude archive stamps and other irrelevant text or other elements. Unfortunately, it is very difficult to eliminate such a noise.

### *M*4 - Line Extractor

Although it is possible to extract text lines directly from the page image (see [14], [21] or [17]), we deploy a text line segmenter on smaller units i.e. text blocks. This works better than a line-segmentation applied on the full page because the text block detection already removes non-text regions.

[3] http://www.portafontium.cz/

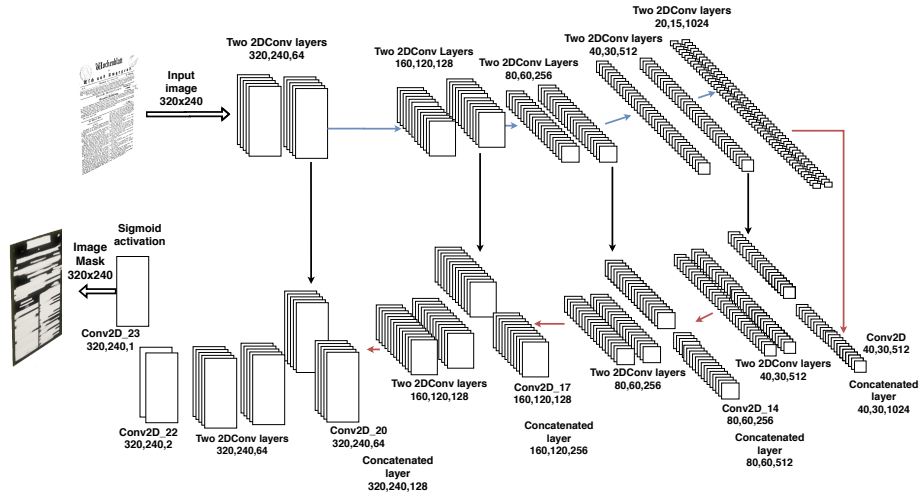



Fig. 3: U-Net architecture. Except of the last layer, ReLU activation functions are used. From the input ($320 \times 240$ pixel) image, an output image mask is produced. Blue arrows indicate `MaxPooling` with the pool size (2, 2) and the red arrows show `UpSampling` with the size (2, 2).

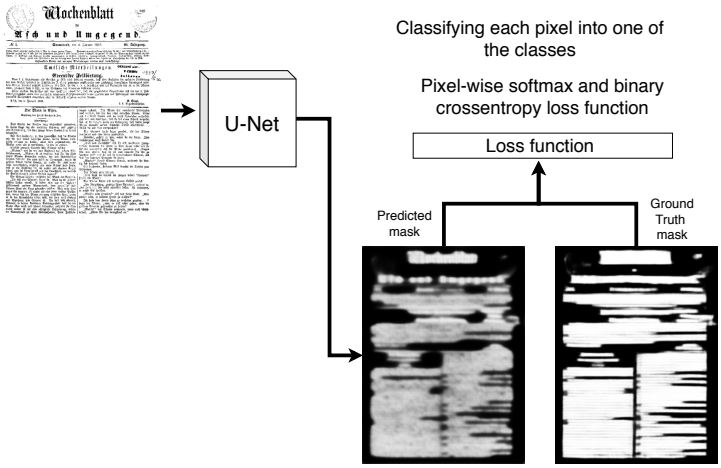



Fig. 4: Scheme of U-Net training [27]

For the line detection, we use ARU-Net, a deep neural network presented by Gruning et al. [14]. It was designed to detect baselines (line upon which the letters sit) in handwritten historical documents. ARU-Net extends the U-Net and it should provide a better line detection in pages with variable font size. It includes also an attention mechanism [3] which allows the ARU-Net to focus on image con-

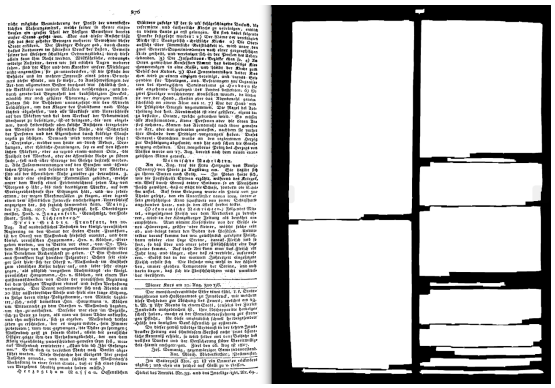


Fig. 5: Europeana image and its ground truth

Fig. 6: Visualised text region mask predicted by U-net

tent at different positions and scales [14]. As in the previous segmentation method, ground truths are represented as binary masks (Figure 8).

To obtain an image of a text line, we first need to determine the size of the font. We utilize an algorithm based on a projection profile of a region above the detected baseline. According to the profile we can approximately compute the x-height of the font. Then, adding the height of ascenders and descenders plus some margin, we can crop the line image. In some cases, also parts of neighboring lines are present in the line image. Therefore, we also apply a post-processing step, based on connected component analysis, that erases such parts of letters from surrounding lines.

A visualization of the line extractor together with text block extractor is shown in Figure 9.

## 5 Training Data Creator

The second functional unit deals with the training data creation. Its main purpose is to provide text line images and appropriate ground truth labels for the OCR engine.


Fig. 7: Visualised output from Text Block Extractor


Fig. 8: Text block and its ground truth

This unit can be divided into two data acquisition methods: synthetic data creation and annotation of real data. First, we present the Character Segmenter module which is an important part of synthetic data preparation process.

### *M*5 - Character Segmenter

This module is used for segmenting images of text lines into individual character images. This task can be achieved by finding character separator positions. The algorithm is based on projection profiles. The input image is first inverted and thresholded. Then we calculate the vertical projection profile of the image. This process is illustrated in Figure 10.

The white peaks indicate presence of characters. Values lower than a specified threshold are considered to be gaps separating the characters. The proposed segmentation of the above example is shown in Figure 11.

This example shows several segmentation errors that occur typically in letters *m*, *n* or *u*. Another issue is the presence of ligatures that are impossible to split


Fig. 9: Visualised output from Line Extractor


Fig. 10: Original text line and its vertical projection profile.


Fig. 11: Proposed segmentation of an example line image

using the projection profile method (*ch*, *tz* and *ck*). This was also a reason why we chose a line-based OCR approach in the first place.

The GUI allows manual correction of incorrectly segmented characters. The tool has options for merging or splitting incorrect segmentations. It is also possible to shift character borders. Extracted individual character images can be stored and used for further synthetic data creation. The output of this module is thus an annotated line image and optionally a list of pictures of individual letters (see Figure 12).

Creation of a ground truth label for the image is possible by typing appropriate letters on the keyboard. Since we work with old German documents, we provide buttons for some special German characters (e.g. ß) to speed up the annotation process. The Character Segmenter tool is depicted in Figure 13. It is worth noting that this tool is meant to be standalone and thus it is not part of the web framework.

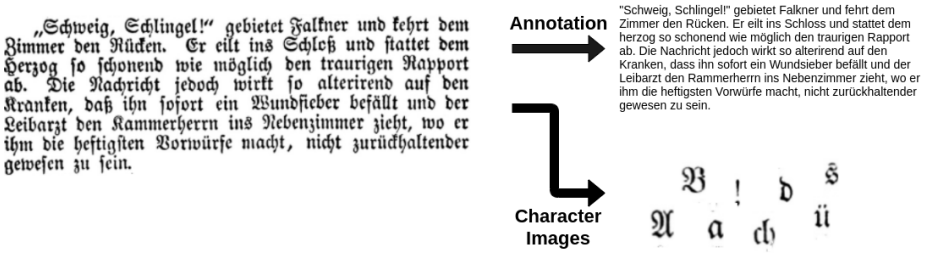



Fig. 12: Annotation and character segmentation


Fig. 13: Character Segmenter tool

*M6* - Line Generator

Synthetic line generation can be seen as an opposite process to the annotation. The process of generation is depicted in Figure 14.

To generate synthetic line images, we need to provide a text source and the way of rendering the text line image. We chose to use the simplest way of rendering such images by concatenating images of individual characters as depicted in Figure 15.

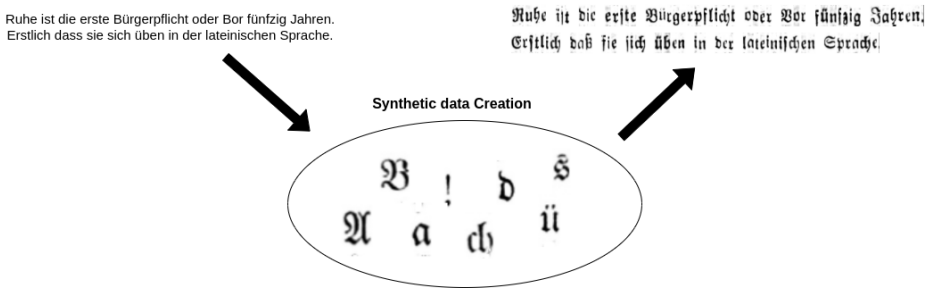



Fig. 14: Creating an image of a text line as a composition of character images


Fig. 15: Two examples of synthetic line images

*M*7 - Line Annotator

Similar to the case of the Character Segmenter (module *M*5), a text line image from the previous functional unit enters this module. The output is the text sequence (ground truth label) of the text line image. It allows the user to open a selected image of a text line. The user then simply transcribes the text into the appropriate text box. Once a trained OCR model is available (trained for example on synthetic data or subsequently fine-tuned on a small set of real images), we can load it into the tool and use it for prediction. The model predicts the text of the line image and suggests the transcription to the user who can correct the prediction if necessary (see Figure 16).

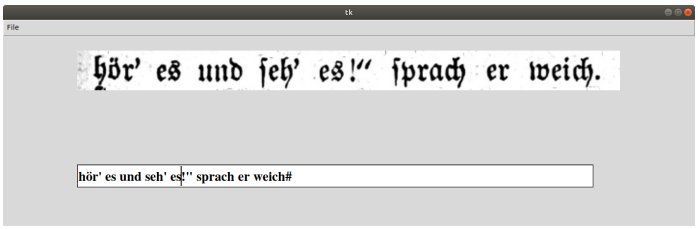



Fig. 16: Line annotator GUI

We briefly summarize the datasets used for the OCR training (module *M*8) and also for training of the U-Net (module *M*3).

*Synthetic Dataset*

Our synthetic dataset contains 25 000 line images and its only purpose is to train the OCR engine. The texts used for rendering synthetic images are based on old German documents to ensure that the language corresponds to the one used in the processed documents.

*Porta Fontium Dataset*

The real annotated Porta fontium dataset is used not only for training the OCR engine but also for training of the U-net model used for text block segmentation. It consists of 25 pages but only 10 pages are completely transcribed. All of them are accompanied by ground truths containing layout information and reading order stored in the PAGE format [24].

We divided the 10 fully transcribed pages into train, test and validation parts and we used it for OCR training. The remaining 15 pages are used for training the U-Net used for text block segmentation (see module *M3 Text Block Extractor*).

## 6 OCR Engine

The last functional unit contains only one module *M8*, the OCR recognizer. Our OCR engine utilizes a combination of a convolutional and a recurrent neural network. The CNN is used for feature extraction while the RNN is used for recognition itself. The architecture is a simplified version of the network proposed in [30]. Following Graves et al. [12] we use a bidirectional LSTM [15] architecture with CTC loss function [11].

The LSTM unit contains a memory part and three so called gates, namely input gate ($i$), output gate ($o$) and forget gate ($f$). The equations for the LSTM unit are as follows:

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\ h_t &= o_t \circ \sigma_h(c_t) \end{aligned} \tag{6}$$

Sigmoid function ($\sigma_g$) represents an activation function of each gate. Each gate has its own weight matrix($W_f$, $W_i$, $W_o$) and biases ($b_f$, $b_i$, $b_o$). The matrix $U$ in each gate represents the recurrent connection between the previous and the current steps. The current input (at timestamp $t$) is $x_t$ and $h_{t-1}$ is the previous output (at timestamp $t-1$). The fourth equation describes a cell state at timestamp $t$ ($\circ$ is the Hadamard product). The output at timestamp $t$ ($h_t$) is defined by the last equation where $\sigma_h$ is the $tanh$ function.

The input of our network are binarized line images. We resize the images so that their height is 40 pixels. The width is set to the maximum image width occurring in the training set. We keep the aspect ratio of the images and pad the rest of the image with white space.

The CNN creates feature vectors which are subsequently fed into the bidirectional LSTM. The output of the Bi-LSTM layer is passed to a set of dense layers followed by the softmax activation function. It represents a probability distribution of characters per each time frame.

The last part of the classifier is a transcription layer which decodes the predictions for each frame into an output sequence. To be able to distinguish each individual character, the blank-symbol (-) is present. It is also necessary to deduplicate the sequences of the same symbols. The architecture of the classifier is depicted in Figure 17.

Our network has two convolutional layers with 40 kernels of size $3 \times 3$. Each of them is followed by a max-pooling layer. The output of the convolutional layers is reshaped and connected to a fully-connected layer with 128 neurons and servers as input for the recurrent layers. We utilize two bidirectional LSTM layers with 256 units. We use ReLU [29] activation function after each layer.

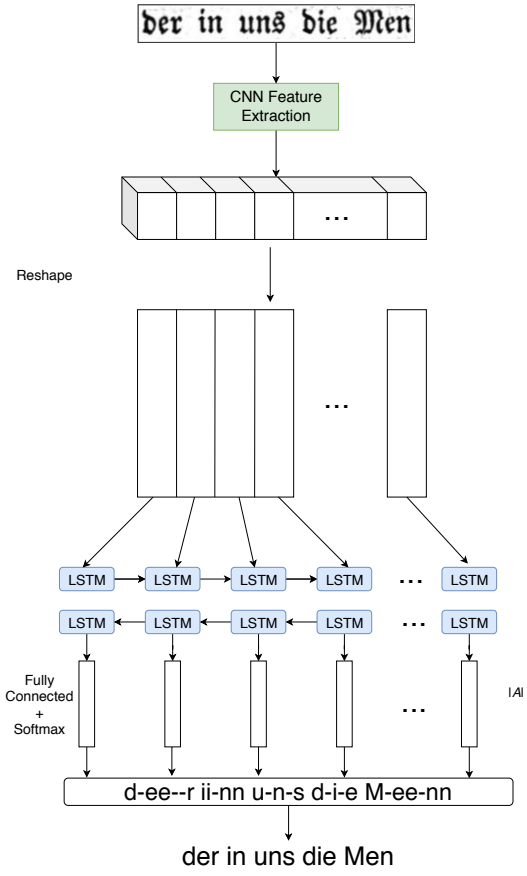



Fig. 17: Architecture of the OCR engine [23]

### 6.1 Classifier Training

| | # pages | # lines | # words | # chars |
|---|---|---|---|---|
| **Train** | 7 | 955 | 7653 | 50 426 |
| **Val** | 1 | 138 | 1084 | 6 669 |
| **Test** | 2 | 275 | 2163 | 13 828 |

Table 1: Statistics of the Porta fontium dataset

The model is trained in two phases. For the first phase, we used synthetic data (25 000 line images) for the initial training. The model is then fine-tuned on

text line images from the Porta fontium dataset, please refer to Table 1 for the statistics.

The model is trained using stochastic gradient descent (SGD) algorithm. The initial learning rate is set to 0.001. We applied early stopping based on the behaviour of the validation CTC loss.

Figure 18 shows the progress of training on synthetic data for 25 epochs. One page from the Porta fontium dataset is used for validation. The curves indicate that training the model longer than for 5 epochs is not beneficial.

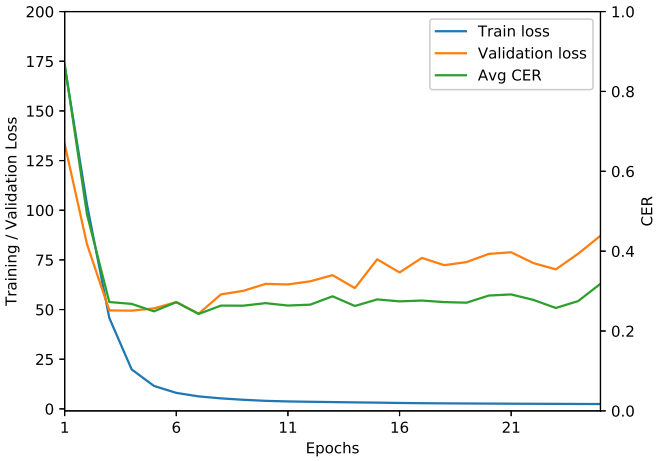



Fig. 18: Training progress on synthetic data

To fine-tune the model, we took the initially pre-trained model and continue the training for additional 100 epochs with real annotated line images from the Porta fontium dataset.

Figure 19 depicts training loss, validation loss, and CER for this fine-tuning.

In the next section, we present experiments and the evaluation of methods used within the framework.

## 7 Experiments

The goal of this section is to present and discuss experiments and results of the framework. We first present the segmentation experiments and then we compare our OCR results with other existing tools (Tesseract [31] and Transkribus [19,32]).

### 7.1 Segmentation

For segmentation, our test data consist of 10 pages from the Porta fontium dataset. We evaluated masks predicted by U-Net comparing it with ground truth binary

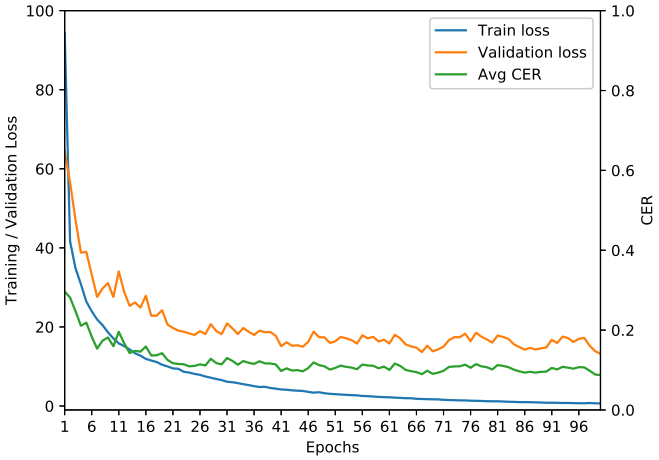



Fig. 19: Progress of the additional training on real data

masks (see Figure 20). The leftmost image (a) shows the ground truth of the page (for details see module $M3$). The middle image (b) shows the predicted mask from trained U-Net. The image on the right (c), shows visual evaluation and comparison of both masks. The green colour represents correctly assigned pixels (each green pixel is correctly predicted as a part of a text region - true positives). The red colour indicates that the model predicted this pixel to be part of a text region, but it is not (false positives). Finally, the blue (turquoise) coloured pixels are pixels that should be considered as part of a text region but the model omitted them (false negatives).

Averaged segmentation results (see Table 2) are computed and visualized using DIVA layout evaluator [2]. The results indicate that this step performs well obtaining an F1-score of more than 97%.

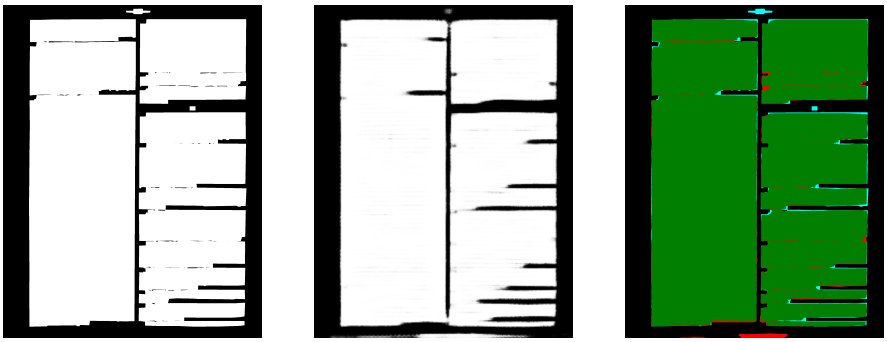
| Exact match | F1 score | Jaccard index | Hamming score |
|---|---|---|---|
| 97.67 | 97.67 | 95.50 | 97.67 |

Table 2: Results of text block segmentation in percent

## 7.2 OCR

### *7.2.1 Evaluation Metrics*

For presenting the OCR results, we use the following evaluation metrics. Firstly, we present the average accuracy (Avg ACC) results. In our case, the average accuracy



(a) Page 980 Ground Truth Mask (b) Page 980 Predicted Mask (c) Evaluation

Fig. 20: Visualisation of text blocks recognition by *U-Net*

indicates how many text line images were recognized correctly from all text line images:

$$Avg\ ACC = \frac{c}{n} \tag{7}$$

where $c$ is the number of correctly recognized text lines and $n$ is the number of the text lines in the test dataset.

We further measure how many corrections in the text line must be done by the user to obtain the correct output. This metric is reported by Levenshtein edit distance (ED) [20], which also indicates the number of insertions, deletions and substitutions:

$$Avg\ ED = \frac{1}{n}\sum_{i=1}^{n} ED(pr, gt)_i \tag{8}$$

$n$ refers to the number of total text lines in test dataset ($pr$ is the predicted output and $gt$ is the ground truth).

We report also character error rate (CER):

$$Avg\ CER = \frac{1}{n}\sum_{i=1}^{n} \frac{S_i + D_i + I_i}{N} \tag{9}$$

where $S$ refers to the number of substitutions, $D$ is the number of deletions and $I$ is the number of insertions in each text line ($i$) from test dataset, $N$ is the number of characters in each text line and $n$ is the number of total test lines.

The last reported metric is word error rate (WER) which is computed analogically as CER, however the characters are replaced by words.

### *7.2.2 Results*

Table 3 summarizes the five-fold cross-validation results on 10 pages and also provides a comparison with other OCR engines. The results show that the presented system outperforms three other models: two by Tesseract (Tess – deu_frak, Tess – Fraktur) and one provided by Transkribus.

Table 3: Results of the OCR systems on 10 annotated pages

| | CRNN | Tess – deu_frak | Tess – Fraktur | Transkribus |
|---|---|---|---|---|
| Avg ACC | **0.488** | 0.221 | 0.217 | 0.398 |
| Avg ED | **1.137** | 2.518 | 2.152 | 1.230 |
| Avg WER | **0.118** | 0.191 | 0.187 | 0.120 |
| Avg CER | **0.024** | 0.053 | 0.045 | 0.027 |

## 8 Conclusion and Future Work

In this work, we presented HDPA, a platform that provides a set of tools that facilitate the analysis and processing of historical document images. It was developed as a web framework based on Python and Django. Our main goal was to provide a basic functionality for the whole pipeline of document image processing. The system thus allows pre-processing, layout analysis, segmentation and finally the transcription of the text content. Another important part comprises utilities for training data preparation which are necessary if an OCR model is to be trained.

The framework can be easily extended. Our intent was to allow the users to implement their own modules for some specific needs and integrate them into the framework. We also performed a basic evaluation of the methods used for particular tasks. The results show that the set of tools can help the user to build a complete OCR system from scratch. Moreover, such a system performs better than available state-of-the-art OCR engines which is mainly due to the customization for the given historical document images.

Future development of the framework will head towards building an extension that will allow applying natural language processing methods on the transcribed data, which will bring tools such as named entity recognition, classification and intelligent full-text search in the documents. Another possible direction for future work is incorporating also methods for handwritten documents.

## Acknowledgement

This work has been partly supported from ERDF "Research and Development of Intelligent Components of Advanced Technologies for the Pilsen Metropolitan Area (InteCom)" (no.: CZ.02.1.01/0.0/0.0/17_048/0007267), by Cross-border Cooperation Program Czech Republic - Free State of Bavaria ETS Objective 2014-2020 (project no. 211). and by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications.

## References

1. Ahmadi, E., Azimifar, Z., Shams, M., Famouri, M., Shafiee, M.J.: Document image binarization using a discriminative structural classifier. Pattern recognition letters **63**, 36–42 (2015)
2. Alberti, M., Bouillon, M., Ingold, R., Liwicki, M.: Open Evaluation Tool for Layout Analysis of Document Images. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), pp. 43–47. Kyoto, Japan (2017). DOI 10.1109/ICDAR.2017.311

3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
4. Breuel, T.M.: The ocropus open source ocr system. In: Document Recognition and Retrieval XV, vol. 6815, p. 68150F. International Society for Optics and Photonics (2008)
5. Breuel, T.M., Ul-Hasan, A., Al-Azawi, M.A., Shafait, F.: High-performance ocr for printed english and fraktur using lstm networks. In: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, pp. 683–687. IEEE (2013)
6. Chernyshova, Y.S., Gayer, A.V., Sheshkus, A.V.: Generation method of synthetic training data for mobile ocr system. In: Tenth International Conference on Machine Vision (ICMV 2017), vol. 10696, p. 106962G. International Society for Optics and Photonics (2018)
7. Clausner, C., Papadopoulos, C., Pletschacher, S., Antonacopoulos, A.: The enp image and ground truth dataset of historical newspapers. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pp. 931–935. IEEE (2015)
8. Clausner, C., Pletschacher, S., Antonacopoulos, A.: Efficient ocr training data generation with aletheia. Proceedings of the International Association for Pattern Recognition (IAPR), Tours, France pp. 7–10 (2014)
9. Etter, D., Rawls, S., Carpenter, C., Sell, G.: A synthetic recipe for ocr. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 864–869. IEEE (2019)
10. Garz, A., Seuret, M., Fischer, A., Ingold, R.: A user-centered segmentation method for complex historical manuscripts based on document graphs. IEEE Transactions on Human-Machine Systems **47**(2), 181–193 (2016)
11. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on Machine learning, pp. 369–376. ACM (2006)
12. Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for unconstrained handwriting recognition. IEEE transactions on pattern analysis and machine intelligence **31**(5), 855–868 (2009)
13. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in neural information processing systems, pp. 545–552 (2009)
14. Grüning, T., Leifert, G., Strauß, T., Michael, J., Labahn, R.: A two-stage method for text line detection in historical documents. International Journal on Document Analysis and Recognition (IJDAR) **22**(3), 285–302 (2019)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
16. Kanungo, T., Lee, C.H., Czorapinski, J., Bella, I.: Trueviz: a groundtruth/metadata editing and visualizing toolkit for ocr. In: Document Recognition and Retrieval VIII, vol. 4307, pp. 1–13. International Society for Optics and Photonics (2000)
17. Kumar, V., Sengar, P.K.: Segmentation of printed text in devanagari script and gurmukhi script. International Journal of Computer Applications **3**(8), 30–33 (2010)
18. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks **3361**(10), 1995 (1995)
19. Leifert, G., Strauß, T., Grning, T., Labahn, R.: Citlab argus for historical handwritten documents (2016)
20. Levenshtein, V.: Binary codes capable of correcting spurious insertions and deletions of ones. Russian Problemy Peredachi Informatsii **1**, 12–25 (1965)
21. Likforman-Sulem, L., Zahour, A., Taconet, B.: Text line segmentation of historical documents: a survey. International Journal of Document Analysis and Recognition (IJDAR) **9**(2-4), 123–138 (2007)
22. Margner, V., Pechwitz, M.: Synthetic data for arabic ocr system development. In: Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on, pp. 1159–1163. IEEE (2001)
23. Martínek, J., Lenc, L., Král, P., Nicolaou, A., Christlein, V.: Hybrid training data for historical text OCR. In: 15th International Conference on Document Analysis and Recognition (ICDAR 2019), pp. 565–570. Sydney, Australia (2019). DOI 10.1109/ICDAR.2019.00096
24. Pletschacher, S., Antonacopoulos, A.: The page (page analysis and ground-truth elements) format framework. In: 2010 20th International Conference on Pattern Recognition, pp. 257–260. IEEE (2010)

25. Postl, W.: Method for automatic correction of character skew in the acquisition of a text original in the form of digital scan results (1988). US Patent 4,723,297
26. Rawls, S., Cao, H., Kumar, S., Natarajan, P.: Combining convolutional neural networks and lstms for segmentation-free ocr. In: Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on, vol. 1, pp. 155–160. IEEE (2017)
27. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, pp. 234–241. Springer (2015)
28. Sauvola, J., Pietikäinen, M.: Adaptive document image binarization. Pattern recognition **33**(2), 225–236 (2000)
29. Shang, W., Sohn, K., Almeida, D., Lee, H.: Understanding and improving convolutional neural networks via concatenated rectified linear units. In: international conference on machine learning, pp. 2217–2225 (2016)
30. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE transactions on pattern analysis and machine intelligence **39**(11), 2298–2304 (2017)
31. Smith, R.: An overview of the tesseract ocr engine. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, pp. 629–633. IEEE (2007)
32. Strauß, T., Weidemann, M., Michael, J., Leifert, G., Grning, T., Labahn, R.: System description of citlab's recognition & retrieval engine for icdar2017 competition on information extraction in historical handwritten records (2018)
33. Van Beusekom, J., Shafait, F., Breuel, T.M.: Automated ocr ground truth generation. In: Document Analysis Systems, 2008. DAS'08. The Eighth IAPR International Workshop on, pp. 111–117. IEEE (2008)
34. Zahour, A., Likforman-Sulem, L., Boussalaa, W., Taconet, B.: Text line segmentation of historical arabic documents. pp. 138 – 142 (2007). DOI 10.1109/ICDAR.2007.4378691