anyOCR: An Open-Source OCR System for Historical Archives

Abstract—**Currently an intensive amount of research is going on in the field of digitizing historical Archives for converting scanned page images into searchable full text. anyOCR is a new OCR system which mainly emphasize the techniques requires for digitizing a historical archive with high accuracy. It is an open-source system for the research community who can be easily applied the anyOCR system for digitization of a historical archive. The anyOCR system can also be used for contemporary document images containing diverse, simple to complex, layouts. This paper describes the current state of the anyOCR system, its architecture, as well as its major features. The anyOCR system supports a complete document processing pipeline, which includes layout analysis, training OCR models and text line prediction, with an addition of fast and interactive layout and OCR error corrections web-based services.**

I. Introduction

There has been a resurgence of interest in optical character recognition (OCR) in recent years mainly for digitizing historical archives all over the world. Commercial and open-source OCR engines (like OCRous [1] and Tesseract [2]) have traditionally been optimized for contemporary documents like books, letters, memos, and other end-user documents. However, OCR engines for large-scale digitization of historical archives differ in their requirements from such traditional OCR systems mainly becasue of complex layouts. In addition, OCR systems traditionally have usually been developed for specific scripts and languages, and it is difficult to train them for old scripts that can give high performance. There were some efforts from the reseach community to handel the challenging case of processing historical document images [3]–[5]. However, first of all these systems are not open-source. Secondly and more importantly these system are limited to give high performance for challenging historical document images. These issues limit the usefulness of such existing OCR systems for large scale digitization of historical documents.

The goal of the anyOCR system is to overcome these limitations.

- anyOCR is an open source OCR system allowing easy reuse of a complete OCR processing for any type of historical documents by the research community.
- The particular open source license used by anyOCR, the OpenContent License, simplifies usage and modifications.
- The system is designed from the ground up with multi-lingual and multi-script recognition exists. .
- The system can easily handel diverse, complex and irregular layouts of historical document images
- The system relies on only a small number of intermediate representations and interfaces, most of them image based, making it easy to integrate both existing and new algorithms.
- The system is extensible.

The rest of this paper will provide an overview of the architecture of the anyOCR system and the methods used in it, as well as some other information of interest to potential users or contributors to anyOCR. Please note that this paper is not a review of end-to-end OCR pipeline; for example, there are many worthwhile and competitive algorithms for the whole OCR pipeline including preprocessing, layout analysis and recognition, but this paper will focus on algorithms and techniques that are actually developed and used within the anyOCR system.

II. The Architecture of anyOCR System

The overall architecture of the anyOCR system is composed of four major components: anyBaseOCR, anyOCRModel, anyLayoutEdit, and anyOCREdit

- anyBaseOCR is responsible for a basic end-to-end OCR processing including the following steps: layout analysis (binarization, text and image segmentation and text-line extraction), text line recognition using a trained OCR model, and generation of text output in hOCR format.
- anyOCRModel is responsible for training a new OCR model for any script and language that can be used in the anyBaseOCR pipeline. The anyOCRModel training framework is based on an unsupervised sequence learning mechanism.
- anyLayoutEdit is responsible for interactively involving user in the anyBaseOCR pipeline to correct the text line segmentation errors as well as tagging logical labels with them.
- anyOCREdit is responsible for correcting errors in the OCRed text which is the output of a trained OCR predictions. The error correction here is done not only through involving user in a the correction process but also understanding user error correction behaviour through a machine learning approach for automating the post-correction process.

The complete architecture of anyOCR system is shown in the Figure 1. Each of the four major components of the anyOCR system is described in detail in the next four sections, respectively.

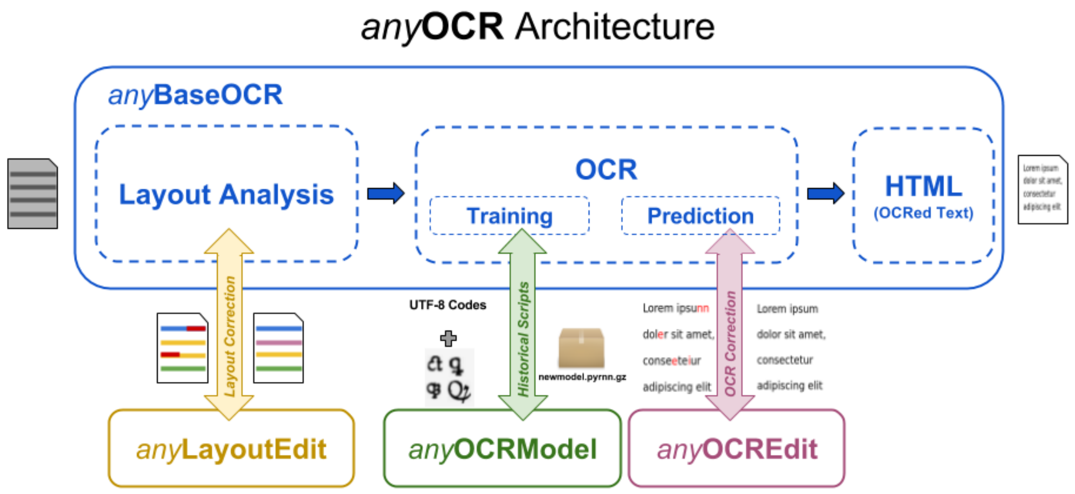


Fig. 1: anyOCR Architecture: a flow diagram of the anyOCR system. The anyOCR system consists of four major components: anyBaseOCR for end-to-end OCR processing from document image to text, anyOCRModel for training new OCR model in unsupervised manner for any script or language, anyLayoutEdit for interactively editing layout errors and labels, and anyOCREdit for interactively and automatically correcting OCR errors.

III. anyBaseOCR - End-to-End OCR Processing Pipeline

The anyBaseOCR component contains a set of document analysis methods that are usually required for a typical end-to-end OCR pipeline for extracting text form a document image. These methods includes binarization, text and image segmentation, text line segmentation, text line recognition, and producing OCRed text in hOCR format. The anyBaseOCR processing pipeline is shown in the Figure 2. The technical description of these methods are described here as follows.

A. Binarization

anyBaseOCR contains a percentile based binarization method [6] that is suitable for various different types of grayscale documents from properly scanned to camera-captured having non-uniform illuminations. The binarization method in anyBaseOCR takes into consideration the background statistics based on percentile filters. The binarization method starts with estimating the background at each location in the image,i.e., a whole new image is created having only the background of the image based on percentile.The threshold in this method is adapted in accordance with the background properties of the image. The original image has a domain of all gray level values, i.e., [0,255] and the background image estimated for each value based on percentile filters at every location has a domain of only two levels,i.e.,0,255. The thresholding is done in a way that if the pixel value in original image is less than ’t’ times the pixel value in background estimated image, then the corresponding pixel value in the output image is labelled one, where t is the parameter used to determine that whether a pixel is foreground or background, depending on the similarity of the pixel, and the background, which has been estimated using percentile filter; otherwise it is labelled zero. A sample binarization result can be seen in the Figure 2.

B. Text and Image Segmentation

Bloomberg [7] presented a multiresolution morphology based text and image segmentation method. It is a simple and script independent text and non-text segmentation method. It performs well for halftone mask segmentation, for which it was designed, but most of the time fails to accurately segment drawing type non-text elements such as line art, drawings etc. The anyBaseOCR contains the improved version of multiresolution morphology based text and non-text segmentation algorithm that was described in [8]. This improved version can handle halftones as well as drawing type non-text elements. A sample result after removing image part from the document using the above text and image segmentation method is shown in the Figure 2.

C. Text Line Segmentation

anyBaseOCR contains an improved version of Gaussian smoothing based text line segmentation method from OCRopus [1]. It first estimates the ”scale” of the text by finding connected components of individual letters in the binary image and calculating the median of their dimensions. Then column separators in binary image are found using convolution and thresholding with some post-processing steps like removal of two column separators that are too close to each other in the same horizontal line and extension of column separators to the first and last rows of the image with a condition that no character is crossed in between on the extended path. At first vertical white spaces on binary image are found and then the rest region is labeled in order to form smooth text region using filtering. Then using Gaussian and uniform filtering, the

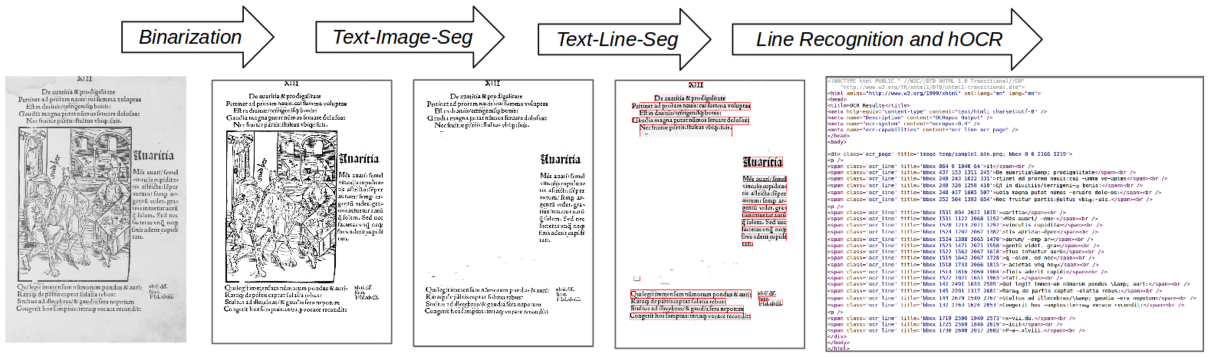


Fig. 2: anyBaseOCR Processing Pipeline: anyBaseOCR component is equipped with all the methods that are typically required for converting a document image into text: binarization, text and image segmentation, text line segmentation, text line recognition and output in hOCR format.

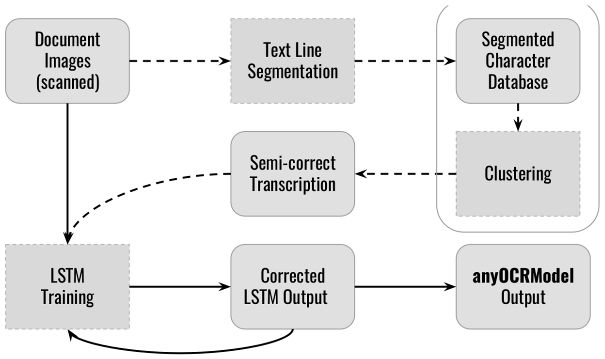


Fig. 3: anyOCRModel Training Pipeline: First text lines and unique symbols are extracted from the scanned data. These symbols are then clustered. After a language expert has identified the resulting clusters, semi-correct ground truth data for each text line is generated. In a second phase a LSTM-based OCR model is trained using the clustering output (using the OCRopus software suit). The trained model can then be used to generate better ground truth data for iterative retraining. When no better OCR model can be trained or gains become too small (e.g. less than 1%), training should be stopped.

column edges (gradients) are found in the binary image by setting a certain threshold in accordance with the scale of the image. The smoothened text region and the column edges are combined to get column separators. In the next step, out of the total column separators, only selected number of column separators with dimension greater than min value are selected. The finally obtained column separators are then combined with the initially obtained text region (through white space method) in order to find more precise text only regions in the binary image. All the gaps/holes within the text regions are filled up and thus final text only regions are obtained. Finally, in order to find text lines, at first, box-map (bounding box) is found by setting two thresholds. f the area of the slice list lies in between the threshold areas, then that slice is labeled one, otherwise it is labeled zero- it helps in removing noise. Then a clean image is obtained by multiplying the two image arrays of box-map and the given binary image, keeping only the desired text. On this cleaned image, the y-derivative of a Gaussian kernel is applied to detect the top and bottom edges of the remaining features. It then blurs this horizontally to blend the tops of letters on the same line together. The areas between top and bottom edges are blurred and treated as text line regions and termed as line seeds. A sample text line segmentation, where each segmented text line region is enclosed in a red-colour rectangle can be seen in the Figure 2.

D. *Text Line Recognition and hOCR*

For each segmented text-line image, anyBaseOCR uses a trained OCR model for predicting its text. The mechanism for training a line-based OCR model for any script or language is described in the anyOCRModel Section IV. After recognizing each text line, the results are composed in a “hOCR” (HTML-OCR) format, which does not only contain recognized text form a document image but also contain the layout information for each text-line in the document image. A sample OCRed output in hOCR format is shown in the Figure 2.

IV. anyOCRModel - Unsupervised Sequence Leaning Based OCR Training Framework

Institutes and libraries around the globe are preserving the literary heritage by digitizing historical documents. However, to make this data easily accessible the scanned documents need to be transformed into search-able text. State of the art OCR systems using Long-Short-Term-Memory networks (LSTM) have been applied successfully to recognize text in both printed and handwritten form. Besides the general challenges with historical documents, e.g. poor image quality, damaged characters, etc., especially unknown scripts and old fonts make it difficult to provide the large amount of transcribed training data required for these methods to perform well. Transcribing the documents manually is very costly in terms of man- hours and require language specific expertise. The unknown fonts and requirement for meaningful context also make the use of synthetic data infeasible. Recently an

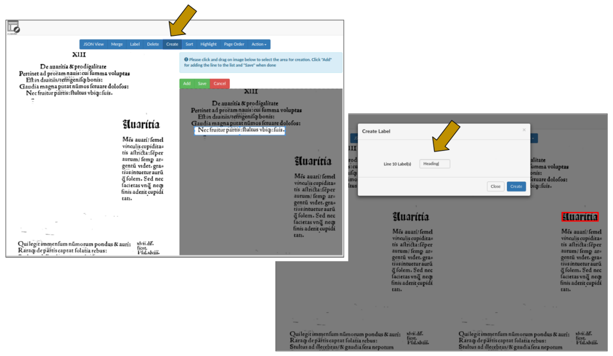


Fig. 4: anyLayoutEdit: A web-based tool for editing the text line segmentation results mainly to correct the segmentation errors, wrong reading order as well as assigning logical labels to the segmented elements. This tool provide a fast way of correcting layout segmentation errors through user interaction and using them in the anyBaseOCR pipeline.

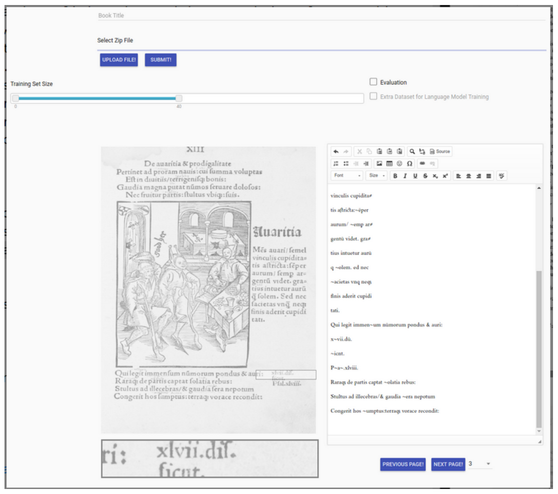


Fig. 5: anyOCREdit interactive Web-Based User Interface that facilitates user in fast and automatic ways of OCR error correction

anyOCR model training framework is proposed [9] that cuts the required input from language experts to a minimum and is therefore easily extendible to other documents. This approach combines the strengths of segmentation-based OCR methods utilizing clustering on individual characters and segmentation-free OCR methods utilizing a LSTM architecture. In this anyOCR model training framework unsupervised clustering for segmentation-based approach is used in tandem with a LSTM based segmentation-free approach to provide an OCR system for documents where no training data is available. The anyOCRModel uses that framework for training a high performance line-based OCR model for medieval historical documents without using any form of manually transcribed training data form LSTM. The anyOCRModel training framework is shown in Figure 3.

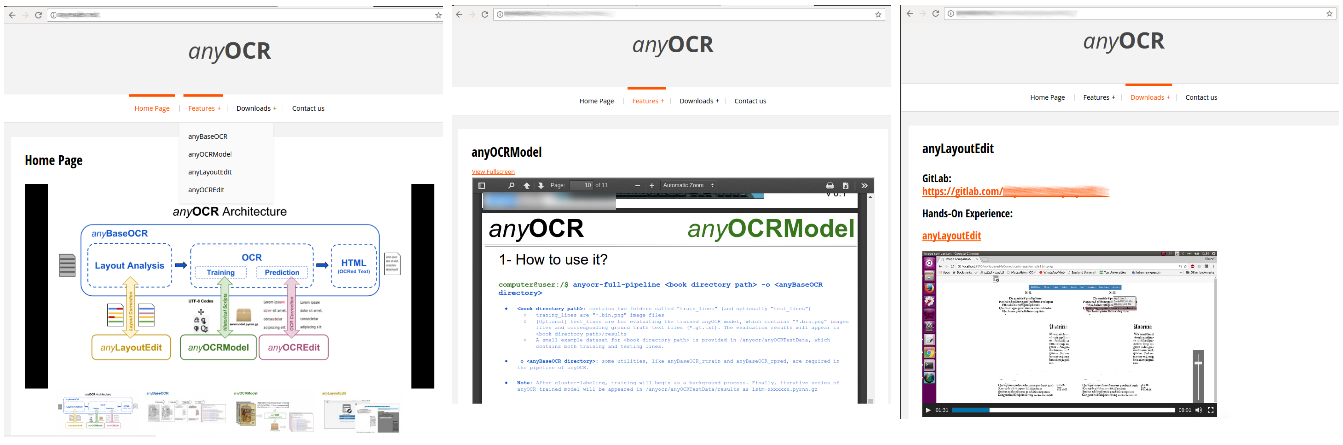
V. anyLayoutEdit - Web-Service for Interactive Layout Editing Tool

anyLayoutEdit has been developed to overcome the wrong-segmented lines from text line extraction step in anyBaseOCR. It has the ability to merge the segments of the same line, separate the merged lines, delete a line segment and recreate it. It could also used to start defining the lines from the scratch in a binarized image. We know that even if the text line segmentation is fully correct but with wrong reading order, then the final OCR text looks like a gibberish text. Therefore, another important feature in the anyLayoutEdit tool is that user can also re-defined the reading-order of page that might not be perfectly calculated by the anyBaseOCR text line segmentation utility. In addition, this tool gives the possibility to label the segmented lines so that each line would get a logical label, for example heading, main-body text, side-notes text, paragraph, image, drawing, etc., which definitely put a great positive impact on the final OCRed output in the similar layout as exist in the document image. This tool is designed as web-service so it could be used as interactive tool by the user to spot the elements in the document. After finishing the editing (i.e. mainly correcting text line errors in segmentation and reading order as well as labelling of different elements in the document image) the tool will generate correctly separated images for each line with their reading order and logical labelling The output from the anyLayoutEdit system is perfectly aligned with the formats used in the anyBaseOCR pipeline. So it clearly means that no extra work is required after editing text lines using the anyLayoutEdit web-service. Some of the features of anyLayoutEdit web-service are show in Figure 4.

VI. anyOCREdit - Web-Service for Interactive and Automatic OCR Error Correction

Currently an intensive amount of research is going on in the field of digitizing European cultural heritage for converting scanned page images into searchable full text. Researchers are mainly trying to adapt existing optical character recognition OCR tools for this purpose. However historical document images and their text differ significantly from modern book images and their text mainly because of following features (or one can also say degradations): historical fonts including ligatures, historical spelling variants, somewhat displaced characters (resulting from historical printing processes), fuzzy character boundaries due to ink creep into the paper over time, paper degradation resulting in dark backgrounds, blotches, cracks, dirt, and bleed through from the following page. Even the existing state-of-the-art OCR system (both commercial and open-source ones) result in mainly because of these features. Therefore the correction in OCRed text is needed to handle both historical spellings as well as true OCR errors.

We have developed anyOCREdit utility that contributes in two-folds for OCR error correction of historical text. Firstly a web-based interactive user interface (UI) is developed which helps user to perform manual error correction faster than traditional style. The interactive UI contains three panel: (1)



1- How to use it?

```
computer@user:/$ anyocr-full-pipeline <book directory path> -o <anyBaseOCR directory>
```

- `<book directory path>`: contains two folders called "train_lines" (and optionally "test_lines")
  - traning_lines are "*.bin.png" image files
  - [Optional] test_lines are for evaluating the trained anyOCR model, which contains "*.bin.png" images files and corresponding ground truth text files (*.gt.txt). The evaluation results will appear in <book directory path>/results
  - A small example dataset for <book directory path> is provided in /anyocr/anyOCRTestData, which contains both training and testing lines.
- `-o <anyBaseOCR directory>`: some utilities, like anyBaseOCR_rtrain and anyBaseOCR_rpred, are required in the pipeline of anyOCR.
- **Note**: After cluster-labeling, training will begin as a background process. Finally, iterative series of anyOCR trained model will be appeared in /anyocr/anyOCRTestData/results as lstm-xxxxxxx.pyrnn.gz

https://gitlab.com/[redacted]

Fig. 6: anyOCR System is made fully available for the research community. It contains documentation, source-code and hands-on experience videos. The snapshots of the anyOCR website containing all these details are shown in the figure.

the toolbar for uploading a zip folder that contains a set of scanned images and their erroneous OCRed text, (2) the image viewer where on hovering over the image a zooming lens appears underneath the image that contains an enlarged version of the surrounding area to the cursor to facilitate the viewing of small characters or damaged parts of the page, which facilitates the fast way of locating and correcting the OCR errors in the text editor, and (3) the text viewer for editing. Secondly, on top of the interactive UI, we have also integrate a Statistical Machine Translation (SMT) based automatic OCR error correction technique, which does not necessary require training data, with the interactive UI. The integrated SMT based automatic OCR error correction approach further accelerate the whole process. The interactive UI after integrating SMT based OCR error correction is show in Figure 5.

VII. The anyOCR System Availability, Documentation and Hands-On Experience

To access the tools that are developed in anyOCR system and practically know how to use them, a website has been created which contains the following:

- Documentations as PDF files for each of the tools anyBaseOCR, anyOCRModel, anyLayoutEdit, and anyOCREdit) that explains the complete logic behind them.
- Hands-On experience videos are recorded for each of these tool that clearly show how to install each them and how to use/run all of their functionalities.
- The source code of these tools as a single project is fully available on a GitLab repository.
- Additionally, an Ubuntu Virtual Machine is set up where all of these tools are pre-installed and ready which is also made available for research community.

pdf files that explain the the logic behind the tools. Videos snapshots are also available there, that shows some use cases of the tools. The source code of the project is wholly available on a GitLab repository.

The snapshots of the anyOCR system website that contains all of the above mentioned resources are shown in Figure 6.

VIII. Conclusion

The presented anyOCR open-source system is a collection of processes that are required for a complete end-to-end OCR pipeline. The system contains not only the state-of-the-art document layout analysis techniques (i.e. anyBaseOCR utility), but also contains a fully unsupervised OCR training framework (i.e. anyOCRModel) which can be trained easily for any script and languages, and on top of that interactive web-services for correcting layout and OCR errors (these are anyLayoutEdit and anyOCREdit utilities, respectively). These advanced features make it perfectly suitable for extracting text from historical document images together with their usage for contemporary document images. All the utilities are made available for reseach community with documentation and hands-on experience tutorials, which help the research community to use these utilities with ease. Furthermore, an Ubuntu based virtual machine (VM) containing all the pre-installed utilities is also provided to the community. Last but not the least, the research community can further developed these utilities.

References

[1] "OCRopus," https://github.com/tmbdev/ocropy.
[2] "Tesseract," https://github.com/tesseract-ocr.
[3] A. Garz, M. Liwicki, and R. Ingold, "HisDoc 2.0: Toward Computer-Assisted Paleography, Manuscript Cultures," vol. 7, 2015.
[4] M. Wuersch, R. Ingold, and M. Liwicki, "Sdk reinvented: Document image analysis methods as restful web services," in *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, April 2016, pp. 90–95.
[5] "Aletheia," http://www.primaresearch.org/tools/Aletheia.
[6] M. Z. Afzal, M. Krämer, S. S. Bukhari, M. R. Yousefi, F. Shafait, and T. M. Breuel, *Robust Binarization of Stereo and Monocular Document Images Using Percentile Filter*. Springer International Publishing, 2014.
[7] D. S. Bloomberg, "Multiresolution morphological analysis of document images," vol. 1818, 1992, pp. 648–662. [Online]. Available: http://dx.doi.org/10.1117/12.131480
[8] S. S. Bukhari, F. Shafait, and T. M. Breuel, "Improved document image segmentation algorithm using multiresolution morphology," vol. 7874.
[9] M. Jenckel, syed Saqib Bukhari, and A. Dengel, *anyOCR: A Sequence Learning Based OCR System for Unlabeled Historical Documents*, 2016.