TU WIEN Informatics

Extracting Retrievable Information from Archival Documents

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Josef Weber, BSc
Matrikelnummer 01363254

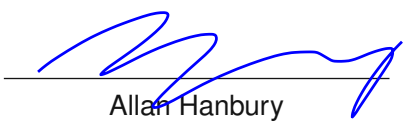
an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Allan Hanbury
Mitwirkung: Dipl.-Ing. Sebastian Hofstätter

Wien, 16. August 2021

Josef Weber



Allan Hanbury

TU WIEN Informatics

Extracting Retrievable Information from Archival Documents

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Josef Weber, BSc
Registration Number 01363254

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Allan Hanbury
Assistance: Dipl.-Ing. Sebastian Hofstätter

Vienna, 16th August, 2021

Josef Weber

Allan Hanbury

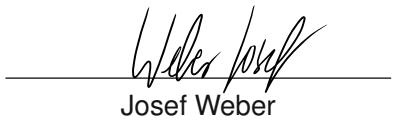
Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Josef Weber, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 16. August 2021



Josef Weber

Acknowledgements

I would like to thank my advisors Sebastian Hofstätter and Prof. Allan Hanbury for their valuable assistance and expertise. Sebastian encouraged my interest in a variety of other research fields and he has always supported me in my research activities. I would like to acknowledge Benedikt Fuchs for providing innovative contributions and ideas that have been applied to my work.

Finally, I would also like to thank my girlfriend Veronika for the patience and encouragement she has always shown me throughout this work. Her support kept me motivated and focused.

This research was undertaken in the course of the project "Visual History of the Holocaust: Rethinking Curation in the Digital Age" (www.vhh-project.eu). This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822670.

Kurzfassung

Die Digitalisierung großer Mengen von Bilddaten im Hinblick auf die Textextraktion wird im Bereich der Archivforschung zunehmend gefordert, um die Dokumente digital aufzubewahren und computergetrieben verarbeiten zu können. Bei der Digitalisierung großer Dokumentenmengen stellt die automatisierte Verarbeitung aufgrund der Vielzahl von Schwierigkeiten, die Archivdokumente mit sich bringen, eine Herausforderung dar. Während sich einige Studien darauf fokussieren, dokumentenspezifische Schwierigkeiten wie die Auflösung oder Ausrichtung der Dokumente zu korrigieren, um die Dokumente zunächst extrahierbar zu machen, fokussieren sich andere Studien auf Methoden zur Verbesserung und Optimierung des Textextraktionsprozesses. Andere Digitalisierungsprojekte haben gezeigt, dass das Zusammenspiel von Dokumentenkorrektur- und Dokumentverbesserungsmaßnahmen essentiell ist um ein gutes Digitalisierungsergebnis zu erzielen.

Wir haben Anforderungen von Historikern, die im Forschungsprojekt „Visual History of the Holocaust“ arbeiten, an ein Digitalisierungssystem extrahiert und analysiert, um diese mittels technischer Lösungen umzusetzen. Um der Forderung nach einem minimalen Interaktionsdesign und dem daraus resultierenden Automatisierungsgrad gerecht zu werden, setzten wir unüberwachte Metriken auf der Basis von Textextraktionsmetadaten ein, die es dem System ermöglichen, Entscheidungen über Dokumententransformationen zwischen und innerhalb von Verarbeitungsschritten zu treffen. Wir evaluieren unser Digitalisierungssystem, die OCR-Pipeline, an einem von Historikern definierten und transkribierten Datensatz im Vergleich zu einem im Projekt verwendeten kommerziellen Digitalisierungswerkzeug und messen die Überlegenheit unseres Systems anhand einer Wortfehlermetrik.

Wir erweitern die extrahierten Klartextdaten, indem wir Maßnahmen zur Korrektur von Rechtschreibfehlern sowie Maßnahmen zur Extraktion von Zeitkontexten anwenden, um die Anwendbarkeit der extrahierten Daten zu erweitern. Im Hinblick auf die Korrektur von Rechtschreibfehlern evaluieren wir unsere Methode anhand eines aufgabenspezifischen Datensatzes und messen die Auswirkungen der auf den extrahierten Text angewandten Korrekturvorschläge. Im Hinblick auf die Erkennung von Zeiteinheiten stellen wir die Auswirkungen auf die Erkennungsraten von Zeiteinheiten zwischen der Textausgabe der OCR-Pipeline und der extrahierten Ausgabe ohne Korrekturmaßnahmen auf der Grundlage eines weiteren aufgabenspezifisch annotierten Datensatzes dar.

Abstract

The digitisation of large volumes of image data with regard to text extraction is increasingly demanded in the field of archival research in order to preserve the documents and process them in a computer-driven manner. With regard to digitizing large volumes of documents, automated processing presents a challenge due to the multitude of difficulties that archival documents entail. While some studies focus on correcting document-specific difficulties such as document resolution or document orientation to make them extractable initially, other studies focus on methods to improve and optimise the text extraction process. Other digitisation projects have shown that the interaction of both document correction and document improvement measures is essential and can lead to a good digitisation result.

We extracted and analysed requirements of historians working in the research project "Visual History of the Holocaust" for a digitisation system in order to translate and implement requirements into technical solutions. To meet the requirement for a minimal interaction design and the resulting level of required automation, we employed unsupervised metrics based on text extraction metadata that enables the system to make decisions about document transformations between and within processing steps. We evaluate our digitisation system, the OCR pipeline, on a dataset defined and transcribed by historians, against a commercial digitisation tool used in the project, and measure the superiority of our system using a word error metric.

We further augment the plain text data extracted by applying spelling error detection and correction measures as well as time context extraction measures to expand the applicability of the extracted data. With regard to the spelling error correction workflow, we evaluate our method on the basis of a task-specific dataset and measure the effects of the correction suggestions applied to the extracted text. In terms of time entity detection, we present the impact on time entity recognition rates between the text output of the OCR pipeline and the extracted output without pre-processing and correction measures applied based on another task specifically annotated dataset.

Contents

CHAPTER 1

Introduction

Due to the increasing demand for digitization of documents, both in the interest of document preservation and subsequent analytical processing, the need for reliable and accurate systems for this digitization process is greater than ever. Especially for documents with a historical context, the digitization of documents is important for advancing research and preserving our cultural heritage. Today, computer systems enable the implementation of such processes without the need for human interaction. Being able to process and analyze archival data efficiently and effortlessly on a computer is an attractive option to consider as it saves time and human resources. The application of computer-driven solutions enables automated transcription of documents from entire archives, providing historians with more capacity to research the digitized contents.

Optical character recognition (OCR) is a field of research that enables the translation of different types of documents or images into machine-encoded text to be used for further analysis or digital preservation. Since the 1980s, OCR has been a highly researched topic and in recent years it has again made great strides through the use of artificial intelligence and self-learning systems which enables the possibility of generating domain-specific recognition models. Modern OCR engines are able to recognize different fonts and languages and are therefore well suited for the digitization of archive data.

However, with historical documents the precision of these engines can vary considerably and therefore measures to improve the recognition of the text must be applied in advance. With regard to the text extraction process, historical texts present numerous challenges that are reflected in the typeface used, the text arrangement on the document, and various contaminants such as grime or faded letters. In order to address these challenges a series of image preprocessing steps for quality enhancement is required. Furthermore, in the digitisation process in the field of archival research, it is often indispensable to comply with the preservation of the digital facsimile files. A digital facsimile is a photocopy replica of a printed document that represents the original as accurately as possible and is available in the form of a digital image file. Considering the preservation of the

digital facsimile files, the documents can be enriched with additional information and annotations. With respect to the text extraction process and the objective of recognising the text content on the image as accurately as possible, facsimile preservation presents an additional challenge, since document transformations must be designed in such a way that there must be no visual alteration on the facsimile document.

The concern to expand and sustain cultural heritage through digitization is being promoted by various stakeholders, from libraries to private and government organizations. Projects such as Improving Access to Text (IMPACT [1]) focus on large scale digitization of historical documents and address important aspects of the archival research domain such as knowledge sharing, cost factor and quality of results. The impact of a digitization system, intended to process historical documents, is far-reaching, affecting and supporting the research of historians and offers potential for automating processes along the digitization procedure.

1.1 Motivation and Problem Statement

The project "Visual History of the Holocaust (VHH): Rethinking Curation in the Digital Age" [1] aims to preserve historical material and to reconstitute its original context in a way that it can be presented to specific target audiences but also to help professionals to dig deeper into history. The demand for a comprehensive solution to automate the digitization of archived facsimile documents derives from the requirements associated with supporting historians' workflows. In order to effectively map requirements to domain-independent technical solutions, an iterative development and evaluation process and the process of gaining insight into the historian's workflow is essential. The main requirement components include the automation factor of the digitization procedure and the design for a non-technical user audience, the capability of spelling error correction and the output of digital facsimile documents with embedded text. Existing solutions are often designed domain- or project-specific, which limits their general usability. Further, existing solutions may require user interaction during the digitisation process or expertise to use the respective tool. The open-source tool OCR4all developed by Reul [21] adopts a semi-automatic approach in which the user can make task-specific input efforts to improve the resulting accuracy. Due to the currently exclusive local installation of the docker containerized software and the optional effort required from the user during the OCR workflow, intuitiveness and ease of use of the tool are only partially given. The Origami OCR Pipeline [15] specializes in transforming historical newspaper documents into machine-readable data and supports large scale data processing. The classification of different region types detected by layout and contour detection models is an important aspect which is responsible for the subsequent quality of the dewarp and extraction steps. A strong focus is given to the extraction of column based table data and to the read flow determination.

[1] Improving Access to Text (IMPACT) http://www.impact-project.eu/ (accessed 16th August, 2021)

Archival documents that have never been reviewed or corrected after the initial capture present challenges, such as incorrect document orientation or suboptimal document resolution, both of which must be addressed through corrective measures to perform a successful OCR process. As mentioned by Liebl and Burghardt [15], further document transformations to enhance the basic OCR text quality are necessary in order to be able to carry out post-processing measures such as spelling error correction successfully. In recent years, several studies [4, 11] have shown which techniques can be used to improve OCR results, also in the area of archival research. Many studies refer to document thresholding techniques known as document binarization that aim, in the context of OCR, to separate document regions into text and background to enhance the text recognition. The thresholding approaches that are being researched involve traditional [19, 24], hybrid and, in recent years, machine learning-based [11] techniques and models. The determination of the binarization method as well as the corresponding algorithm parameters represents a document-specific challenge which can be assessed, for example, on the basis of document-specific decision criteria such as the respective document exposure. The presented preprocessing framework [4] employs a hybrid binarization approach that uses local average contrast to decide for a binarization method. The framework has proven that it is able to binarize heavily degraded documents efficiently, but due to the model complexity there is a high number of algorithm parameters to be evaluated which strongly increases the runtime of the preprocessing measure.

1.2 Contributions of the Work

The contributions in this thesis are:

- **Analysis of the requirements of historians for research tasks on archival documents.** For us as computer scientists, gaining an understanding of the requirements of the historians and insight into their working environment creates the basis for our planning and implementation process. Therefore it is important to understand each process including input and output components and to enable reproducibility in order to be able to map the technical solution to the process and the corresponding requirement. In addition to the outlined requirements and concerns of the historians, the archival research domain and its regulations constitute additional requirements that must be taken into account.

  We have recorded and analyzed project-specific as well as project-independent requirements for the archival text research domain in order to subsequently implement the required measures. Through close collaboration and continuous evaluation with historians, we have developed methods that can be applied project-independently in the archival research domain.

- **Optical Character Recognition (OCR) pipeline.** In this thesis we develop a comprehensive multi-modal Optical Character Recognition (OCR) pipeline that

connects the domain-specific transformations and corrections for facsimile documents, digital photocopy replica documents from historical archives. The analysis of the requirements of historians for research tasks on archival documents, as stated in our first contribution, as well as policies and frameworks of the archival research domain constitute the basis for the development of the pipeline. The pipeline is capable of processing thousands of documents and therefore is designed to be robust against a large landscape of varying document features such as different lighting conditions, image resolution and document rotation. By addressing these difficulties in individual processing-steps in the OCR pipeline, researchers have the ability to digitize documents without the requirement to make manual corrections to the data. The resulting automated optical character recognition (OCR) pipeline requires no technical knowledge on the part of the user to use the tool for document digitization.

Figure 1.1 illustrates the difference in text recognition shown by the red rectangles between the unprocessed original document on the left and the output of the OCR pipeline shown on the right. As can be observed visually from the difference between the two images, our pipeline recognizes 30% more text for this particular document in relation to the length of the total text by applying our automated image preprocessing methods. We set up the mean line confidence (mlc) as an unsupervised evaluation metric which represents the average value of all text recognition confidence values of an OCR extraction result. By applying the mean line confidence and further document specific metrics such as the text line height or the retrieved text length, decision making for algorithm parameter selection and document transformation applications are determined along the OCR pipeline. As can be seen in Fig 1.1 on the original unprocessed document, the unequal exposure ratios are responsible for the poor text recognition as the text areas are exclusively recognized on the brighter left document regions. On the processed document on the right side, thresholding transformations were applied based on document specific attributes which led to a bypass of the uneven lighting conditions and thus further text regions to be recognized. Due to automated and reliable pre-processing steps, no human interactions or correction procedures are required to be performed manually before data can be processed by the OCR pipeline.

- **Recognition and Normalisation of Dates - Building Searchable Data.** To achieve best results in the text extraction process, computer vision and image processing methods followed by Optical Character Recognition (OCR) techniques are applied. In order to extend the applicability of the raw basic OCR text, we apply natural language processing (NLP) techniques to add additional task specific context to the data. The term searchable refers to the extended applicability of data through the addition of information and annotations. We apply post OCR spelling error detection and correction methods which creates the prerequisite text to detect and extract time expression entities that enable us to build enhanced data that matches the original document content as closely as possible. After

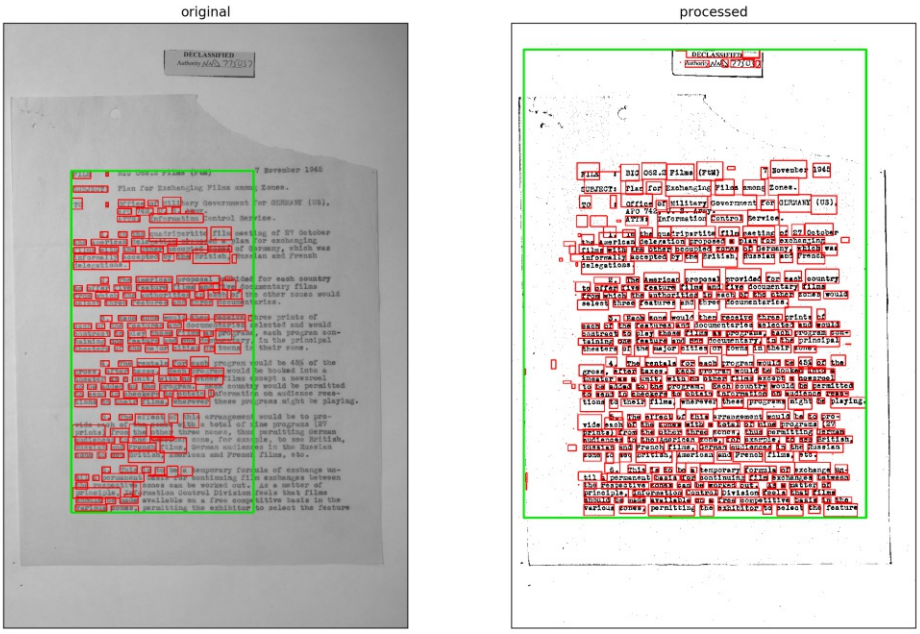


Figure 1.1: Unprocessed and processed documents and their impact on OCR recognition.

recognition of time expressions, date normalization is applied in order to retrieve a standardized date format. A uniform data format of the time entities provides searchable data, which is characterised by its applicability to data-driven analysis, filtering and search processes.

The contribution of our post OCR text enhancement methods includes our spelling error correction workflow and the time entity recognition which operate together and are responsible for enriching the base OCR text with extracted time units and error correction input information. In order to measure the performance of our time entity detection and normalization, we have annotated a task-specific dataset of documents, which on average contains three time entities to be detected per document. Based on the evaluation test dataset, the average hit rate of correctly identified and normalized entities per document was 70%.

- **Evaluation.** To measure the extent to which our optical character recognition, post OCR error correction and entity recognition solutions meet the requirements of the historians, we perform evaluations on the basis of task specific datasets and datasets selected by the historians.

  In terms of quality of the digitised output text, a benchmark comparison is carried out between the text output of the OCR pipeline and the output of the ABBYY fine

reader[2], a commercial text extraction engine. The performance of the spelling error correction workflow is evaluated for the base OCR text, the corrected OCR text, and also for the ABBYY Fine Reader output using a recall metric to determine the similarity to the respective manual transcribed ground truth text.

To enable intuitive user interaction with the implemented solutions, a prototype web-based application called Bulk OCR Webservice (BOW) was developed. Within this application, users can upload data to be digitized as well as inspect and retrieve the results. Through usability evaluations of the prototype as well as direct communication with the target audience, further requirements for a comprehensive interaction interface were evaluated and identified.

1.3 Structure of the Thesis

In Chapter 2 we analyze the requirements of the historians which represent the challenges we address with our research. In Chapter 3 we discuss the background and related work we researched on computer vision methods, spelling error correction strategies and named entity recognition approaches.

The OCR pipeline is presented in Chapter 4, where the text extraction process including the image pre-processing steps are explained in detail. This chapter also includes experiments, the evaluation against another OCR tool and the metrics we used for evaluation. Chapter 5 covers our spelling error detection and correction model and model evaluation. Chapter 6 then describes how date recognition and normalization is performed on the extracted data resulting from the OCR pipeline. In Chapter A the Bulk OCR Webservice together with its use cases is introduced. Technical details on the implementation of the OCR pipeline are provided in Appendix B.

Chapter 7 concludes the thesis ideas for future work are pointed out.

[2]ABBYY FineReader https://www.abbyy.com/ocr-sdk/ (accessed 16.08.2021)

CHAPTER 2

Analysis of the Requirements of Historians

In order to support the working environment of historians with technical solutions, it is particularly important to first become familiar with the work processes of researchers, the challenges and the data archives in the field of archival research. Therefore the close cooperation between computer scientists and the historians is essential to work out these requirements and to build a final application that meets the needs of the historians. In Table 2.1, the requirements are presented with a short description and then described and analyzed in detail in the following sub-sections.

Initially, the concept of the digitisation tool to be developed was presented to the stakeholders involved in the project at a meeting held in Vienna. The historians' requirements listed in Table 2.1 emerged from the first conversation that took place in a virtual conference between the historians and the computer scientists to develop a tool for digitising archival text documents. After the presentation of the elaborated concept and after the first test iteration, additional requirements and refinements arose which were elicited in the course of further feedback and evaluation meetings, which were each held virtually.

2.1 Language Support

Since several countries and thus several data archives are involved in the archival research project, it was necessary to support several languages in the text extraction and correction procedures. While the majority of the documents to be digitized are in English and Russian, there are still some archives with German and French documents. The participation of Russian historians in the project led to the demand for the possibility of digitising Russian documents. Language selection is the only user-supplied parameter

Requirement	Description
Language Support	Language Independence and expandability.
Input and Output Formats	Support for a variety of input formats and output of multiple file formats
Searchable Data	Provision of high-quality, machine-readable data from analogue archive data.
Graphical User Interface	Providing easy interaction to the OCR pipeline.
Facsimile	Digital and true-to-original replication of original archival document.
Transparent Extraction History	Traceability and documentation of the progress of the final data output.
Minimal Interaction	No human pre-processing interaction required before data can be extracted.
Large Scale Processing	Capable of scalable processing of large amounts of data.
Archival Documents	Addressing digitisation challenges that arise with historical archival documents.
Raw Image Format	Conversion of raw image input files.

Table 2.1: Table of requirements of the historians.

that ensures that the OCR process and the subsequent language specific spelling error correction workflow are carried out correctly.

For extracting text from archival documents we utilize Tesseract, an out-of-the-box Optical Character Recognition (OCR) engine that is developed and supported by Google. Tesseract offers language support for over 150 languages and script types, four different OCR engine modes that vary in accuracy and speed, and features such as blacklisting of characters to be ignored during the extraction process. The selection of the language can be done in a graphical user interface such as the Bulk OCR Web Service (BOW), presented in Appendix A, by selecting a language option from a drop-down menu or via the command line interface by specifying the language in the configuration text file which is described in detail in Appendix B.2.

New archives can introduce new languages and therefore the ability to adapt to new languages is an important requirement for the application. Consequently, the application has been designed so that additional languages can be added in the form of language-specific trained text recognition models to be obtained from the tesseract model repository[1]. Regarding the post-OCR spelling correction workflow, as described in Section 5.1, we offer support for English, German, Russian and French text. Analogous to the expansion of the language scope of the OCR process, additional languages are made possible in the

[1]tesseract tessdata repository https://github.com/tesseract-ocr/tessdata (accessed 16.08.2021)

spelling error correction workflow by installing language-specific dictionaries provided by the MySpell spell checker[2].

2.2 Input and Output Formats

The various archives from the different countries usually do not have uniform strategies for archiving or preserving their documents. This wide spectrum of different archiving types is reflected, among other aspects, in the different data storage format types and in the way the data is being managed. Therefore, being able to process a wide range of different types of data was also a requirement to make the tool more independent against file formats in its application. Besides the different types of *raw* formats, common image file formats are also supported. More information about the supported formats can be found in the file transformation processing step of the OCR pipeline described in Section 4.4.

To meet the requirement to be able to import and use the extracted data in any research application, a variety of output formats must be supported to ensure compatibility. The output formats include standardized and structured formats like JSON as well as unstructured plain text files and additionally annotated PDF facsimile files. This provides historians with more opportunities to take advantage of third party research tools that require specific input data types.

2.3 Searchable Data

The text content of the archival data to be extracted is initially available stored as image data in the various file types as described in Section 2.2. The primary objective and requirement is to convert the text content, which is available in the form of image data, into machine-encoded and processable text as error-free as possible. Obtaining a high quality OCR text that is reflected by the fidelity of the text on the original image is the basis for further improvement measures and influences their respective performance. By applying image processing methods, document-specific problems such as incorrect document orientation or unequal exposure ratios within the image are resolved in order to be able to carry out a subsequent successful text extraction process. In addition, the plain OCR text is then enriched with suggested corrections for spelling errors and enhanced through an annotation process that extracts temporal contexts.

Providing correction suggestions based on a post-OCR spelling error detection and correction workflow supports for the creation of an absolute accuracy through human post-correction. Correction suggestions can also be used to expand the context in addition to the plain OCR text to provide better results for a text search scenario. We employ several tools that work independently of each other for spelling error classification and

[2] MySpell Spell Checking and Dictionaries https://www.openoffice.org/lingucomponent/dictionary.html (accessed 16.08.2021)

subsequent correction. A detailed description of our workflow for spelling error correction can be found in Chapter 5.

After text spelling corrections have been retrieved, temporal contexts are marked as so-called time entities on the basis of the corrected text. These time entities include dates as well as partial dates, year-only dates and timestamps. To find and assign time entities in documents, we employ tools for recognizing and normalizing time entity data to form a uniform notation and to facilitate searching and filtering. This allows the researchers to leverage a more complex criterion to filter documents or to establish temporal relations among documents. The time annotation process is described in Chapter 6.

2.4 Graphical User Interface

The demand for a graphical user interface emerged before the first test iteration to enable the interaction with the OCR pipeline as interactive and user-friendly as possible. Platform independence and constant availability led to the decision to opt for a web browser-based service.

The web service offers two main functions which are described in Appendix A. The interaction with the OCR pipeline includes the upload of files to be processed, the state management of the OCR pipeline and the reception of the results. The document preview functionality comprises the presentation of the extracted text embedded on the original image file, suggested spelling error corrections and annotated time entities.

With respect to the target group with little or no technical knowledge, we have continued the minimal interaction requirements as for the OCR pipeline in order to enable user acceptance and intuitive handling. To start the digitisation process, the user selects the document language from the drop-down menu and then loads the data to be digitised onto the application using the drag-and-drop functionality. When the digitisation process is complete, the user can either retrieve the results or interactively examine them in preview mode while the time entity detection and spelling error correction functionalities are presented along descriptive guidance.

2.5 Facsimile

The output of digital facsimile, exact copies of the original document represents the requirement that relates to the preservation of the historical value of the documents. The creation of the digital facsimile is one of the most relevant concerns since it is supposed to give the researcher the feeling of being in the archives directly in front of the physical original, however combined with the convenience of the extracted and embedded OCR text as an additional layer on top of the original image.

For us, this is one of the most challenging tasks because no visible deviations or modifications are allowed on the original document. This includes the preservation of the original document resolution and the background on which the embedded OCR-text is

placed must remain unchanged and also correspond to the original. Still, we are capable of applying corrective measures and document transformations in a way that we do not violate any facsimile guidelines and are thus able to provide facsimile documents with embedded text. The creation and export procedure for facsimile documents is explained in the data extraction processing step of the OCR pipeline described in Section 4.4.

The output facsimile documents are created as PDF files and contain an upright image which may have been corrected by the rotation correction and the embedded OCR text. The background image is not affected by any transformation such as exposure correction and therefore represents the original image. In the case of a multi-page input document, the facsimile also contains the equivalent number of pages.

2.6 Transparent Extraction History

For historians, it is necessary to keep track of applied transformations along the OCR pipeline processing steps to make the final result transparent and traceable and to capture the digitisation history of the respective document.

In order to realize a transparent extraction history solution, meta data is stored along the individual processing steps, explaining the action or change that was made to the data. In addition to the metadata, layout information and extended information such as correction suggestions resulting from the spelling error correction workflow and time entities are extracted for the representation of text recognition results or for other purposes such as manual correction of spelling mistakes.

In order not to destroy the catalogued archive data context, the metadata capture also ensures that file names retain their designation for all associated output files. The preservation of the file names ensures simple accessibility and identification of the data for further applications within the project.

2.7 Minimal Interaction

The requirement for minimal interaction refers both to the interaction with the graphical user interface that communicates with the OCR pipeline as well as the OCR pipeline used as standalone application and to the fact that documents must not require human pre-processing or correction. Corrections such as rotation correction for initial text recognition, determining the optimal document resolution or adjusting various transformation parameters are carried out automatically. Therefore, the OCR pipeline is designed for the scenario where an archival document that has never been corrected is read and then processed by the pipeline. Automated decision making procedures on the basis of document specific evaluation metrics, described in Section 4.2 along the OCR pipeline are employed to perform autonomous actions and keep human interaction low. The minimum resulting interaction with the OCR pipeline is to be reduced to uploading data

and providing the associated document language via a web service, such as the Bulk OCR Webservice, and retrieving the results.

2.8 Large Scale Processing

The processing of large volumes of data is a requirement that is related to the minimal human interaction requirement, described in Section 2.7. A semi-automatic, non-minimal interaction digitisation process would require a considerable amount of interaction from the user in the case of thousands of documents to be digitised, whereby the machine processing the data would have to wait for the user input and process it in parallel. Thus, a minimal human interaction design based on process automation enables the basis for large-scale data processing, so that data can initially be processed without human interaction. Therefore the automatic decision-making processes in the OCR pipeline must function efficiently and reliably as due to the different archiving strategies mentioned in Section 2.2, a variety of challenges emerges, such as the existence of several file formats, that need to be addressed. The objective is that the target user group can rely on a robust system in which documents can be digitised and retrieved without restrictions on the amount of data and without the need to pre-process the data.

The processing of large volumes of data implies a high consumption of computing resources and a high demand on computing time. Performance is a subset requirement which must not be a tradeoff with the output quality of the OCR pipeline. The data and results should be returned within a minimum time according to the quality requirements. The objective is to maximize quality while minimizing computation time and the consumption of resources. Among the performance-critical measures are the use of different text recognition models, the capability running multiple processes at the same time known as multiprocessing, and the avoidance of repetitive calculations through the caching of metadata. Detailed and technical descriptions of the performance optimizing measures mentioned can be found in Appendix B.

To enable distribution of workloads we employ a container-based design with the intention to enable horizontal scaling of the OCR pipeline and thus higher processing capacity with little effort. Containers offer a possibility to package an application or code within an isolated and minimal requirement environment. They perform consistently and show predictable behavior, regardless of the underlying hardware they are deployed on. This enables us to provide the application with high availability and scalability to meet growing requirements or high data volume. In addition, the Docker container technology enables us to efficiently package the OCR pipeline, including dependencies and operating system to seamlessly integrate it into third party services such as the Bulk OCR Webservice. Another reason for the Docker Technology is that containers are suitable for Continuous Integration and Continuous Deployment (CI/CD). This allows developers to integrate their code into one common repository at an early stage and in frequent iterations and then deploy the code quickly and efficiently. This aspect turned out to be especially useful in the test and evaluation phase when we needed to quickly implement and publish

new requirements.

2.9 Archival Documents

The fact that archival research involves working with historical documents implies special document characteristics which must be recognized and addressed as a requirement. Our task is to deal with digital document copies as they come out of the archives after they were originally recorded. Often these digital document copies have never been opened, reviewed or corrected in any way before.

In the archive research domain, all documents to be archived must initially be recorded by either a scan or a photograph. Even if no information seems to be visible, valuable or detectable on the original physical document or it is heavily contaminated it must be kept in the archive. For this reason, documents such as blank back pages, which do not contain any text to be extracted or pages containing mirrored text that shines through from the back side of the document are kept in the archive.

Another problem that clashes with modern OCR practices is that the original recording was made by cameras instead of scanners. Even though these cameras provide sufficient resolution for the purpose, more human error is introduced such as an incorrect document orientation that is not set upright and therefore often cannot be processed by the OCR engine. Uneven lighting conditions on documents caused by regional shadows also complicate the text extraction process and require corrective measures.

Further challenges with archive documents arise from various contaminations such as: handwriting over the text to be recognized, dirt and stains, typos, the use of historical typeface, document layout structuring and poorly printed letters. The awareness of challenges including the handling of special archival documents such as blank pages, problems initiated by humans in the recording process and document-specific difficulties with historical documents are taken into account in order to be compliant with the guidelines of the archival research domain and to achieve a successful digitisation.

2.10 Raw Image Format

Archival documents are often available as a raw file format when they are initially captured in order to store as much information as possible in addition to the image. This additional information includes unprocessed sensor data and other parameters that are helpful for post-processing the documents. The disadvantage of raw files is the file size, which results from the additional information and the very high image resolution, and the fact that these documents are not yet ready for processing or previewing.

In order to make the documents accessible and portable for historians, currently different copies are created manually for various applications. For example, an access copy is created which is designed for fast opening and viewing of the document. For a historian, this manual work is inefficient and very time-consuming when a large number of raw

documents have to be transformed. It is therefore obvious to replace this task of manual raw file transformation, which is indispensable for historians, by a machine.

For a given raw file input document, multiple copies are created and additionally returned to the user with different document properties including file format type, level of image compression and applied document transformations such as white balance correction. Detailed descriptions of the raw output variant specifications can be found in the first processing step, raw file conversion of the OCR pipeline described in Section 4.4.

CHAPTER 3

Background and Related Work

In this chapter, we discuss methods and software solutions that form the foundation of the knowledge we have built on to develop the OCR pipeline. Since the OCR pipeline requires knowledge from several areas of computer science, this chapter is divided into three sections. The order in which the sections are presented corresponds to the order in which the OCR pipeline is arranged and constructed.

In Section 3.1 we explain image processing methods consisting of geometric image transformations and document separation transformations that have corrective and enhancing effects on the text extraction process. The relevance of post OCR correction is outlined in Section 3.2 along methods for detecting and correcting spelling errors. Lastly, we describe the named entity recognition and normalisation process in Section 3.3 with a dedicated focus on temporal contexts to be extracted and refer to the effects of erroneous OCR text.

3.1 Image Processing

The performance of a text recognition system can be influenced and improved by applying document transformations to the input images. The application of document enhancement measures for subsequent text extraction is therefore a heavily researched topic and spans the fields of computer vision and, in recent years, machine learning. Image transformations employed in this thesis include affine or geometric transformations to perform image scaling and rotation as well as thresholding techniques to enable image segmentation. The following subsections explain concepts for affine transformations and thresholding techniques related to document enhancement for the OCR process.

3.1.1 Orientation Detection and Correction

The determination and correction of the document rotation is a crucial geometric transformation procedure that initially can enable successful text extraction. We exclusively reference the two page orientations landscape and portrait as well as their 180 degree flipped transformations for document rotation determination and the subsequent correction. The multiples of 90 degrees thus represent 0 (upside up), 90 (upside right), 180 (upside down) or 270 (upside left). Except for the upside up document orientation, OCR tools have limited ability to extract text efficiently in any of these rotations and therefore the identification of the rotation and the subsequent correction is indispensable. On the basis of a correct page orientation and in further consideration of the orientation correction, the skewness detection deals with the estimation of the fine-grained rotation in relation to the text that is located on the document. In the following, researched techniques and approaches for document orientation detection and correction are explained.

Lu et al. [28] propose a technique for automatic script and orientation detection of printed text documents by utilizing the character stroke density and distribution to map each document image to a document vector. For this, reference vectors are formed for each orientation and script configuration to be evaluated. Finally the font and orientation of a document is determined calculating the similarity between the document vector and the pre-constructed reference document vectors using the K-nearest neighbor algorithm. The proposed identification method is capable of determining page orientation and script while being robust against document skew.

Van Beusekom et al. [29] present a brief overview of skew and orientation detection methods and propose a one-step approach for combined skew and orientation detection using the geometric text-line model by Breuel [5]. The text-line model employed extracts Latin script text lines by optimizing parameter triplets according to a target function using a branch-and-bound search algorithm. To measure and evaluate the parameter configurations, a quality function determines the line-specific matching criteria of the text line model to a given set of reference points. The determination of each text line found in the document is carried out as a parameter optimization problem in a robust least square sense. The number of bounding boxes matching the model are to be maximized while the distance of the reference points to the text baseline are to be minimized. The baseline represents the imaginary line that defines the lower limit for uppercase letters and descenders are letters that may contain information below that baseline in Latin scripts. The line model is applied to all four orientations (0, 90, 180, 270) of the image and among the two best performing orientations according to the quality function, the correct orientation is determined based on the lower number of descenders. The presented method has outperformed techniques of the two open source software tools Leptonica[1] and Tesseract[2] based on a publicly available dataset.

[1] Leptonica http://leptonica.org/ (accessed 16.08.2021)

[2] Google Tesseract OCR https://opensource.google/projects/tesseract (accessed 16.08.2021)

The determination of the correct document orientation and the subsequent corrective measures based on a quality metric also serves as a decision criterion for our approach. However, this metric must not be dependent on the script type or the language.

3.1.2 Scaling

In the scaling process of raster graphical image data, pixels are reconstructed onto a new raster and therefore made smaller or larger. The image scaling transformation modifies the image resolution and, analogously, the text characteristics of the text present on the image document. A relevant text specific attribute is the text height also known as font size. The scaling process thus influences the performance and the runtime of the OCR process. OCR models trained on certain annotated data also perform very well on similar data. The scaling factor of the training documents related to document specific properties such as text height thus influence the training process and the resulting OCR model. Confronting the OCR model with documents that contain properties the model has not encountered in the training process, the OCR output will likely not be good. Conversely, if we want to achieve good results with the selected OCR model, we need to adjust the document scaling accordingly. Determining the text height as a crucial factor for OCR quality first and then using it for document scaling is an important processing step to achieve the highest text extraction performance from the OCR model.

Javed et al. [12] present a novel approach for training and detecting font size recognition directly from run-length compressed text documents exploiting simple line height features. The relevance of text height information for the intelligent OCR process as well as in image analysis is presented as a pre-knowledge extraction stage. The two-stage model first learns to segment each line of text to classify each compressed line of data on the basis of the resulting regression model. Due to the focus on determining the line height of compressed documents, only binary documents were considered for the research in order to save the decompressing process and the additional computing resources it requires. Using two datasets of 35 and 15 noise-free and skew-free documents with different font size properties, the proposed method was tested. The test was carried out on the basis of defined ground truth information for the data sets on which also the models to be tested for predicting the font size of text lines were trained. Two experiments were conducted using different model features to demonstrate the aim of this research work, the idea of font size recognition directly in compressed documents. Comparisons with other methods were not carried out.

3.1.3 Binarization

Document binarization, also known as image thresholding, is an indispensable pre-processing step and contributes to a major extent in improving the character recognition process. In the context of OCR, binarization classifies a document into two sets, text and background. Research on document binarization has been ongoing since the late 1970s, and new methods have also been explored in recent years due to the availability of

machine learning. Since the binarization process has a key function in our approach to document enhancement, we provide a brief background on the various image thresholding techniques, which can be categorized as follows:

- Global Thresholding Methods
- Adaptive Thresholding Methods
- Hybrid Thresholding Methods
- Machine Learning based Methods

Global Thresholding Methods

Global thresholding methods are based on a single threshold input parameter responsible for dividing an assumed bimodal histogram of a grayscale input document, which contains two peaks, using comparison operations. Given an input grayscale image $g(x, y)$ the thresholded output image $b(x, y)$ is defined as follows:

$$b(x,y) = \begin{cases} 1, & \text{if } g(x,y) > T \\ 0, & \text{if } g(x,y) \leq T \end{cases} \tag{3.1}$$

Each pixel value of the input image is compared at location (x, y) to a threshold value T. The global binarization classifies each pixel on the document as text if the value is greater than the threshold and otherwise as background if the value is less than or equal to the threshold. In the following, global thresholding approaches are presented which are capable of determining an automatic threshold value.

A method that uses the global threshold is the Otsu thresholding technique [19], one of the most popular and best performing global thresholding methods. The Otsu method iteratively determines for all available grayscale values of the grayscale the so-called inter-class variance between the two binarization categories and finally selects the maximum value. The maximum inter-class variance value subsequently defines the biggest spread of foreground and background and the threshold which is then applied to the image according to equation 3.1. This approach performs well on images with constant lighting conditions, recognizable by a corresponding bimodal histogram. However, Otsu's as well as the regular global thresholding method perform poorly on documents with inconsistent exposure.

To compensate for the uneven light conditions on the document, the Otsu thresholding method is often combined with exposure corrections such as contrast limited adaptive histogram equalization (CLAHE) which is applied before binarization. Harraj et al. [10] present an approach for a pre-processing workflow that includes lighting correction measures, the luminosity grayscaling method, text details sharpening and the use of Otsu's binarization method targeting to improve OCR accuracy. Based on a standard data set, an

improvement in OCR results of 2% up to 6.8% was evaluated. In our approach, exposure compensation became redundant due to the application of an adaptive thresholding technique.

Local Thresholding Methods

Local thresholding methods, unlike global thresholding methods, are capable of dealing with varying and inconsistent lighting conditions on documents. The local thresholding algorithm determines a so-called local threshold for each individual pixel based on a surrounding neighbourhood region whose size can be defined. Based on the histogram data of the pixel specific surrounding neighborhood region, different lighting conditions located on the document are thus bypassed. Furthermore, it is important to point out that local thresholding methods, in contrast to global thresholding methods, have more than one parameter to be defined which implies an additional effort of determining these additional parameter values. The selection of method parameters impacts the binarization result as well as further processing steps with the thresholded output image. In Figure 1.1, the binarization effect of the adaptive thresholding technique is demonstrated on the basis of a document that shows unequal light conditions due to a sharp-edged shadow on the right side. In the following, different approaches to determine the local thresholds based on a neighborhood context are presented.

J. Bernsen's method [4] uses the mean value of the maximum and minimum intensitiy data points within the respective sliding window values to determine the pixel specific local threshold. This method proves to work well particularly with high contrast images.

Niblack's local thresholding technique is a widely adopted and referenced algorithm and is discussed in [27] and [13]. Niblack's algorithm calculates an individual threshold $T_{Niblack}$ for each pixel which is defined as follows:

$$T_{Niblack} = m + k \cdot \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}(p_i - m)^2} \tag{3.2}$$

Based on the local mean of the pixel values m, the addition of a constant value k, and the multiplication with the standard deviation of the current sliding window of size NP, the local threshold $T_{Niblack}$ is defined.

Based on Niblack's technique, Sauvola and Pietikainen [24] present an improved method in which rapid image surface analysis is applied before binarization, classifying regions into text and background. Based on the assigned regional content type, a corresponding algorithm for threshold determination is selected. This method addresses and overcomes the problem of noise generation, unwanted document details that exceed the respective threshold value and are thus classified as foreground text elements. Additionally, the method of Sauvola and Pietikainen features a fast option in which the threshold is calculated only for every n^{th} pixel and the intervening pixels are classified by interpolation.

For more local thresholding techniques, detailed background and benchmark evaluations we refer to [4, 27].

Challenges associated with local thresholding methods involve the determination of function parameter values for a specific input document. Parameter selection strongly influences the binarization result and may even make it worse with regard to OCR output quality. Furthermore, the choice of parameters can have a significant impact on the processing time of the entire thresholding procedure, considering that an individual threshold is calculated for each pixel. The OCR pipeline makes use of the adaptive threshold technique and we address its drawbacks accordingly to enable an automated, reliable and computationally resource efficient application of this method. We have decided to employ the adaptive threshold technique as it is able to cope with uneven lighting conditions and further document characteristics and degradations of archival documents as described in Section 2.9 if the parameters are selected appropriately. Therefore, we present an approach to parameter selection that makes use of document-specific properties.

Hybrid Thresholding Methods

Hybrid thresholding methods imply the combination of local and global thresholding methods. This allows to combine the best of both worlds, namely the short runtime of global thresholding and the robustness against inconsistencies such as non-uniform illuminated image regions of local thresholding methods.

Boudraa et al. [4] present a hybrid thresholding method consisting of Otsu's global thresholding method, NICK method, a multi-level Otsu as well as a document pre- and post-processing step. With regard to the NICK method, the sliding window based local thresholding technique [13] is a derivative of the local thresholding technique of Niblack that makes the k value adjustable to control binarization noise and the thickness of the characters. Preprocessing includes grayscale transformation and CLAHE for low contrast images. The post-processing step comprises connected component analysis and morphological operations to eliminate noise elements generated during binarization. A contrast based metric was determined as the decision criterion for the respective thresholding algorithms to be applied as well as for the application of CLAHE in pre-processing.

Lund et al. [17] present an approach that uses multiple independent binarizations on the same input image to complement the final text from the thresholded output images. This kind of ensemble technique is used to extract the initial text and perform a simultaneous post OCR correction. The threshold values to be applied for the binary transformation are predefined and lead to parallel OCR outputs which are subsequently merged by an alignment process to a final hypothesis text. A word error rate metric and an annotated dataset were used to evaluate the extent to which the information content of ground truth texts is distributed across different binarization configurations and how combined transformations lead to the best output.

Farrahi Moghaddam et al. [8] present an unsupervised ensemble of experts (EoE) framework which is capable of combining the output of multiple thresholding methods. Each of the 23 thresholding methods from the H-DIBCO'12 competition is designated as an expert and is capable of performing feature extraction and subsequent pixel class voting. The ensemble of experts framework first defines an endorsement graph to map the confidentiality of the experts among themselves. Subsequently saturated opinions are consolidated and schools of experts are identified and selected by thresholding the endorsement graph. From each cluster of experts only those are consolidated that have a certain level of endorsement as determined by a threshold parameter. The EoE framework applies the principles of saturated opinion consolidation and expert school selection, and has been successfully evaluated against the individual performing methods that participated in the H-DIBCO'12 competition on the same dataset.

Machine Learning based Methods

In contrast to the traditional thresholding methods discussed previously, approaches that use machine learning to support the image thresholding process are trained to solve a defined task related to binarization. A wide variety of tasks can be trained and learned, such as the decision of a local or global threshold value, algorithm-specific parameters or document degradations and irrelevant document features to be eliminated. Using data augmentation and ground truth information, training data for the respective model to be trained and created is provided.

He and Schomaker [11] propose an iterative deep learning approach for document enhancement for subsequent document binarization, built on the U-Net neural network. Using Recurrent Refinement (RR) and Stacked Refinement (SR), the correction model or a variation of it is applied iteratively to the document to be corrected and restored. The iterative approach thereby allows to capture degradations that occurred infrequently during model training and are therefore difficult to process. The proposed method showed good results evaluated against three public benchmark data sets and also provides an enhanced version of the degraded documents for visualization purposes and binarization applications.

Thresholding Evaluation

To measure the extent to which a binarization transformation meets the expectation of document separation with respect to a given task, it is critical to make configurations measurable to determine performance and method differences. In general, a distinction is made between supervised and unsupervised methods, which differ in the information on the basis of which the evaluation is performed.

Supervised evaluation metrics are based on a ground truth which represents the optimum result to be achieved. Regarding the evaluation of thresholding, the correctness value is determined for each binarized pixel on the basis of the corresponding ground truth pixel and finally summarized into a metric. The often used F-measure as described in [11] is a

composite metric consisting of precision and recall to measure the accuracy for a given task. Precision is defined from the ratio of correctly positively classified pixels and all positively classified pixels. The recall metric is defined by the ratio of correctly positively classified pixels and all misclassified pixels that should have been classified as correctly positive. In relation to the evaluation of the OCR process, the ground truth refers to the transcribed text on the document. The binarized document is intended to support the OCR process and the resulting extracted text should correspond as closely as possible to the ground truth text. Word distance based metrics such as the Word Error Rate (WER) are used in [17] to determine the performance of an OCR system by comparing hypothesis texts resulting from different binarization settings to a ground truth text.

The advantage of *unsupervised* over supervised methods is that they are able to assess the quality of the respective binarization configuration and its impact on the subsequent OCR process without the requirement of external input information such as a ground truth. With regard to the thresholding parameter selection process, unsupervised methods allow decisions to be taken on the basis of document-specific metrics in order to achieve a binarization result that is optimised for the OCR process. In [20], a mechanism for benchmarking unsupervised metrics for parameter selection of thresholding methods is presented. The experiment conducted shows the relationship between an unsupervised evaluation metric in terms of its respective best binarization configuration. The resulting binarizations are then evaluated according to a ground truth based OCR accuracy metric to determine the strengths of the unsupervised metrics.

3.2 Post OCR Correction

Due to the still imperfect OCR results, the field of automatic spelling error correction, including post-OCR error correction, is a very active research area, just like optical character recognition itself. Post OCR Correction is an indispensable processing step that not only improves the extracted final result of the OCR process but also facilitates and reinforces subsequent analysis activities and information extensions. A comprehensive description of the history and all methods would exceed the scope of this thesis. Therefore, we give a brief background on the methods that contributed to our approach to post OCR error correction.

In [22], the results of the ICDAR 2019 post OCR text correction competition are presented, in which 34 teams registered and 5 methods were submitted. The competition comprised first the error detection task and subsequently the error correction task, which was conducted independently of the first task.

Given only the raw OCR text data without the input image, the error detection task was performed and the index positions of the erroneous words were evaluated using the F-measure. For the second task, based on the given index positions of the wrong words, the correction of the OCR errors was performed which includes a list of suggested corrections for each wrong word. The highest ranked proposals were evaluated using the Levensthein distance similarity metric, which was calculated against the ground truth.

For both outputs, the Clova AI team performed best using their presented context-based character correction (CCC) method. By employing the pretrained language model BERT [7] and its context awareness property, the error detection classification was performed. The subsequent error correction of each erroneous token is conducted by an LSTM model which uses the erroneous token and text context from BERT to perform a character-level correction. To select the most promising candidate, beam search is finally applied.

In contrast to supervised OCR error correction models that have learned to detect and correct errors based on ground truth data, *unsupervised* methods offer a further possibility to perform corrections without the requirement of annotated data.

Hämäläinen and Hengchen [9] present an unsupervised method for OCR error correction that exploits the advantages of Neural Machine Translation (NMT) models. The unsupervised use of NMT is enabled by extracting the required parallel data employing natural language processing (NLP) methods. The extraction of the parallel corpus for the OCR errors is done by utilizing text-similarity measurements in a given text corpus to find possible candidates for corrections. Similarity refers to the semantic meaning of the erroneous word as well as to the similarity of the individual characters of the word to be corrected. The automatic retrieval of the parallel data as a preprocessing step is performed with the help of NLP methods including text lemmatization and the use of word embeddings to standardize the process of correction candidates. For a OCR token marked as erroneous, ten corrections are suggested by the model and then the first candidate that is perceived as valid in the respective language is returned as the correct word. The proposed method is language independent and with an optional input of error-free corpus the results can be improved further. Word segmentation detection and correction is not supported with the presented approach.

The multi-modular domain-tailored OCR post-correction system was developed by Schulz and Kuhn [25]. To cover the variety of different error types that can occur in OCR text with regard to detection and correction, the combined use of several unsupervised methods including statistical machine translation (SMT) and spell checking mechanisms is presented. The system consists of two stages, the suggestion modules which are separated into word and sentence level suggestions, and the decision modules which are responsible for the final decision of the correction suggestion. The word level suggestion modules use spell checkers, word splitters, compounders that merge two tokens into one word, and text internal vocabulary to preserve named entities. In the sentence level suggestion modules, statistical machine translation (SMT) models are trained on parallel corpus data. At token and character (unigram) level, four models are trained on domain specific data and general data respectively. The decision modules are responsible for choosing the best fit suggestion candidate for each word. Therefore a ranking mechanism evaluates correct suggestion to be proposed as the final correction. To solve the decision problem of determining the appropriate candidate for a word, the Moses toolkit [14] is used which makes use of a language model and a phrase table that is formed from all input words and corresponding suggestion words. In addition, weighting of the different modules is taken into account as the reliability of the respective suggestions varies.

To continue the digitization process after the OCR process and automatic post OCR correction for further improvement, manual post-correction is enabled by exporting a standardized format as described in [25]. The application of manually corrected data for the reinforcement of machine learning models also enables an improvement of supervised post OCR correction models.

3.3 Entity Recognition and Normalisation

Named Entity Recognition (NER) is an information extraction subtask that identifies mentions of real world entities in unstructured text and classifies them into predefined categories such as people, places, organizations, or temporal context.

Strien et al. [26] assess the impact of spelling errors introduced by the OCR process on subsequent analytical tasks in the natural language processing domain. The NLP tasks used to evaluate the effects of OCR quality include sentence segmentation, dependency parsing and named entity recognition which is particularly relevant for us. For each linguistic processing task *spacy*[3] was used, a software tool for performing NLP tasks. To quantify the impact, each task is performed on a human transcribed text used as ground truth and on the OCR text output. The information extracted from the ground truth of the task determines the highest performance that can be achieved with the OCR text. To evaluate the OCR quality, the Levenshtein similarity is calculated between the ground truth and the OCR text for a document. The test results were visualized in scatterplots using the Levenshtein distance in relation to the respective NLP task accuracy metric. While sentence segmentation shows a high impact due to low segmentation accuracy, the impact on named entity recognition tasks is relatively low, with the geopolitical entity category showing the highest impact. Strien et al. [26] have conducted a large-scale analysis of OCR errors in terms of their impact on NLP tasks and indicate their impact levels. Among the future work areas mentioned, the relevance of post OCR correction was highlighted in order to exploit the full potential of all NLP tools.

Rodriguez et al. [23] have conducted and analyzed an experiment in which they evaluate the performance of four different Named Entity Recognition tools using the output of the open-source OCR tool tesseract 3[4]. The historical documents to be extracted were obtained from the Wiener library consisting of 17 files and King's College London's Serving Soldier archive consisting of 33 files and provide the basis for the experiment. To evaluate the impact of erroneous words in the NER process an evaluation ground truth was created on the basis of manually annotated named entities. With regard to the NER process, uncorrected OCR text and manually corrected OCR text was compared against the ground truth annotated entities by performing precision, recall, and F1 score comparison. It is important to note that named entity types in this work refer only to people, places, and organizations. From the Wiener library's dataset, one of the four

[3] spaCy https://spacy.io/ (accessed 16.08.2021)

[4] Google Tesseract OCR https://opensource.google/projects/tesseract (accessed 16.08.2021)

NER libraries showed a significant difference between the corrected and uncorrected data, and from King's College London's dataset, two libraries had a significant difference. The overall results show that manual correction of OCR results does not significantly improve the performance of the evaluated entity types in the NER process. According to Rodriguez et al, the low performance difference is due to the NER models being trained on non-historical data, which often resulted in misclassification. Also mentioned by Rodriguez et al. [23] is the importance of post OCR correction in terms of the text quality required by other information retrieval tasks.

The detection and normalization of temporal expressions in OCR texts has been evaluated very little to date as most common NER tools do not provide time as entity type category. However several approaches exist to recognize the wide variety of different notations of temporal contexts. Machine learning-based approaches incorporate data-driven techniques, semantic features [16] or dictionary based features [2] to build a time expression recognition model. Normalization of the temporal expression is the second step in which a standardised value according ISO 8601 is assigned to a recognized temporal context. Commonly, rule-based approaches are used for normalization, which interpret individual text segments by means of syntactic rules.

SUTime [6] offers a monolithic approach in which both tasks of temporal context extraction are combined. It employs a deterministic rule-based system which is built on the basis of regular expressions (regex) and applies multiple types of rules that can be extended. Additionally, the system is able to extract a reference date from the document to allow expressions like *last friday* to be resolved as a date. In order not to treat words from a non-temporal context as temporal expressions, they are detected and removed by filtering rules which make use of part of speech tagging. Performance was evaluated on the tasks of identifying and providing the correct standardised TIMEX3 type tag using precision, recall, and F1 scores. A performance comparison of SUTime was performed against three other tools and showed that SUTime outperforms its competitors in terms of the highest overall F1 score and the highest overall recall score for temporal expression recognition. The highest precision was achieved by HeidelTime[5], which is also a rule-based system.

3.4 Summary

In this chapter we have presented theoretical foundations along several studies that form the basis of the techniques we use in this work. We discuss methods and indispensable measures to potentially determine and correct the document orientation as well as the document resolution to enable a successful OCR process initially. We point out different document thresholding methods and their strengths and make a firm decision to employ the local thresholding method to support the subsequent OCR process. To overcome the disadvantage of the adaptive thresholding method, the large number of parameters to be

[5] Heideltime temporal tagger https://dbs.ifi.uni-heidelberg.de/resources/temporal-tagging (accessed 16.08.2021)

defined, we use unsupervised methods to determine an optimal binarization configuration in order to enhance the subsequent OCR process and resulting output text.

To further improve the extracted OCR text in terms of text quality and integrity to the original document, we discuss approaches for post OCR spelling error correction. We present supervised as well as unsupervised methods for error detection and correction and emphasise the importance of post OCR correction in relation to subsequent information retrieval tasks. Cited studies referring to the impact of post-OCR corrections on the information retrieval task, the named-entity recognition extraction process show that corrected OCR results do not significantly improve the NER extraction process with regard to the entity types person, location and organization. In our work, we refer to the extraction of temporal expressions which, to our knowledge, have been little evaluated with respect to corrected and uncorrected OCR data. We point to methods that incorporate machine learning for extraction, rule-based approaches, as well as a monolithic solution that combines recognition and normalisation of the recognised entities.

CHAPTER 4

Multimodal OCR Pipeline for Facsimile Documents

The OCR pipeline prepares archival data to support the subsequent text extraction process in order to retrieve output that satisfies the requirements of historians. The OCR pipeline does not aim to replace expertise but to reduce manual and repetitive work. Beyond the pre-processing of the document and the subsequent text extraction, the generation of an annotated facsimile document, the application of a post OCR spelling error correction workflow as well as a time entity recognition procedure are part of the solution.

This chapter is structured as follows. In Section 4.1, we explain the chosen order of the individual processing steps of the OCR pipeline and why this order was designed in this way. Section 4.2 presents metrics that are utilised between and within each processing step to make automated decisions for document transformations and corrections. Next, in Section 4.3 we explain what types of metadata are being processed and stored along the OCR pipeline steps. A detailed description of the seven individual processing steps can be found in Section 4.4, which provides a full insight into the functionality of the OCR pipeline through additional illustrations and descriptions. Finally, an OCR quality benchmark evaluation is presented in Section 4.5 and a summary of the chapter is given subsequently.

The workflow responsible for handling spelling errors after the OCR process is described and evaluated in detail in Chapter 5. The time entity recognition procedure that takes place at the end of the OCR pipeline to extend document searchability is described in Chapter 6. Implementation details can be found in the appendix where details of performance measures, a description of the container architecture and configuration options of the OCR pipeline are provided.

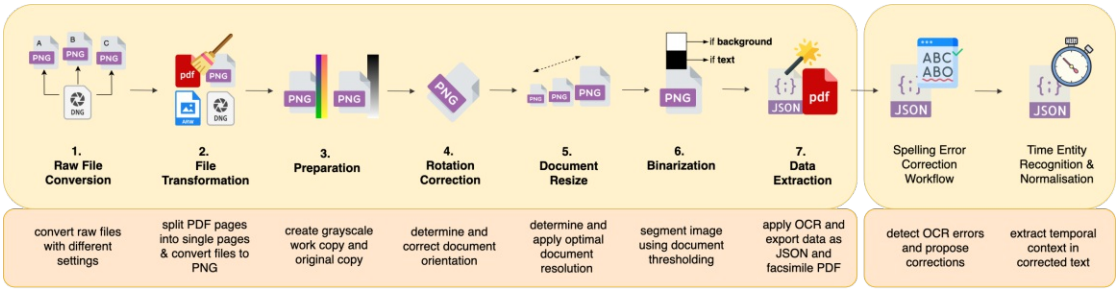


Figure 4.1: OCR pipeline illustrated showing the seven processing steps and the two additional text enhancement steps.

4.1 Processing Order

In this Section we explain how and why the chosen sequence of the individual processing steps was selected and decided upon in this way. The full description of the individual steps of the OCR pipeline can be found in Section 4.4. In the following, the OCR pipeline steps and their functions are summarized briefly. Subsequently, design decisions and considerations are discussed. A visual representation of the processing order of the individual processing steps of the OCR pipeline as well as corresponding brief descriptions are shown in Figure 4.1.

- The first processing step converts input files into a file format that the OCR pipeline can operate with in the subsequent steps. This enables predictable behavior of the software tools and methods we apply along the OCR pipeline.
- Next, the correct rotation of the documents is determined using the mean line confidence metric and corrected if necessary. This is the most crucial and essential step in the pipeline which, in case of an incorrect document rotation, can initially enable text recognition for all further processing steps.
- The document resize step determines the optimal image resolution leveraging the extracted document mean text line height, mean line confidence and the word length. Resulting metadata including the determined document width and height dimensions as well as the the mean text line height is captured to be utilized for subsequent processing steps. The data necessary for evaluating the optimal document resolution is collected through several document resize transformations and text extractions.
- In the subsequent binarization step, the best parameter configuration for the local thresholding algorithm is evaluated by means of the two metrics mean line confidence and text length. Three parameter value series to be evaluated are determined on the basis of the previously extracted mean line height metric as well as static values.

The best configuration of the evaluated parameter values is determined and the transformation is subsequently applied to the document.

- The penultimate step makes use of the previously collected information to apply document specific transformations prior to the final text extraction. The obtained OCR text results are exported to several output formats including the creation of annotated facsimile documents as well as the standardized JSON data format.

- Further enrichment of the raw OCR data with additional information takes place in the ultimate processing step. A post OCR spelling error correction workflow is applied and a time entity recognition is performed. The spelling error correction enables the export of an automatically corrected OCR text and reinforces the quality of the entity recognition performed afterwards.

Sequence Analogy – When we designed the OCR pipeline and the sequence of the processing steps, the analogy to the human behaviour of a manual document correction process was simulated.

The first document transformation, the rotation correction, is analogous to making a document initially readable for humans or respectively for the OCR engine. Then the human will set the optimal distance between her/his eyes and the document to make the document sharp and readable. This process is equivalent to the document resize transformation in the OCR pipeline. Uneven lighting conditions on the document caused by shadows can be avoided by making adjustments according to the present lighting conditions, e.g. by changing the position of the document to the interfering light source. Document transformations to improve readability are applied in the final processing steps to the already readable and resized document and is analogous to the binarization step which can, among other things, compensate for unequal light conditions.

Equally, the evaluation metrics we used contributed to the arrangement of the processing sequence since transformations are often based on them. For instance, the binarization process depends on the mean line height, which in turn is determined in the document resize process and optimized based on the evaluation metrics. The effect of the metrics on the order is nevertheless consistent with the order of the manual correction analogy. Therefore the resize transformation takes place before the binarization step.

4.2 Evaluation Metrics

This section explains the metrics we use to evaluate within the individual processing steps in the OCR pipeline. We distinguish between supervised and unsupervised metrics. Supervised metrics are used exclusively for performance evaluations and require a manually created text transcription, a ground truth. Unsupervised metrics, in contrast, are used at runtime of the OCR pipeline and are derived on the basis of the data of an OCR text extraction process. They enable automated decision making by turning document

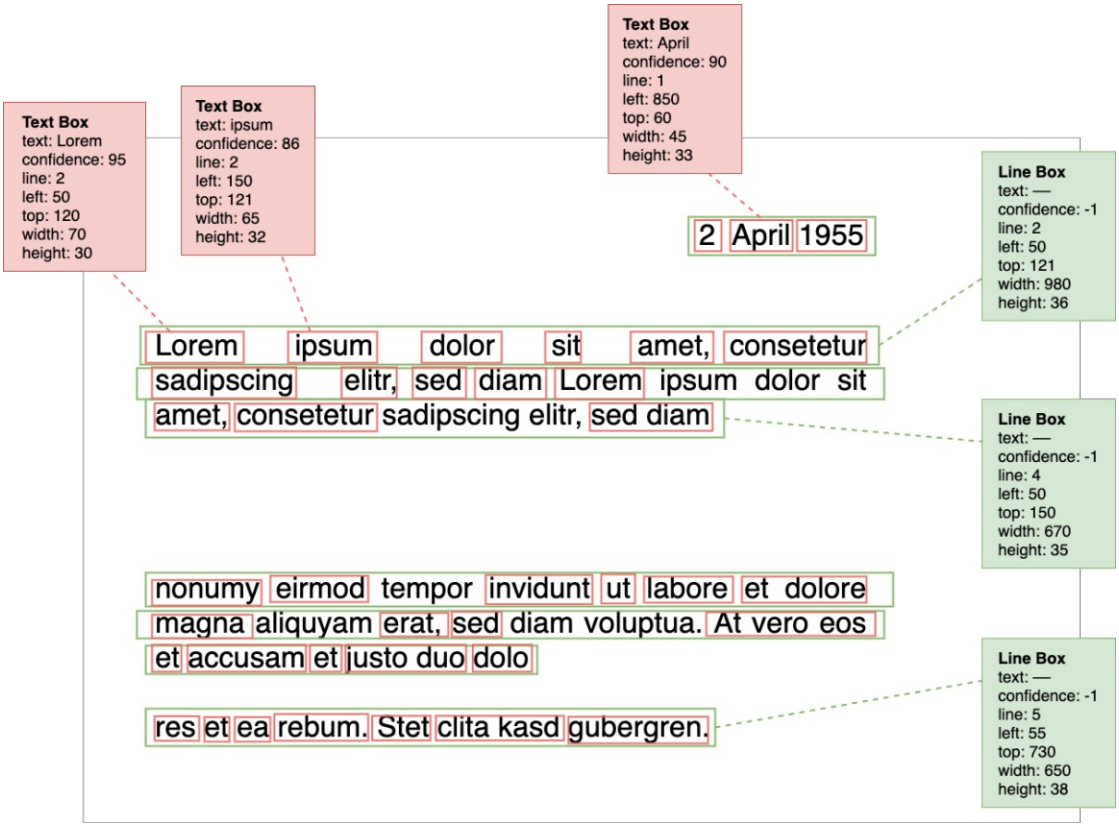


Figure 4.2: Visually explained OCR output metadata showing text box and line boxes with associated attributes.

transformations along the OCR pipeline into measurable and evaluable information in the document pre-processing steps.

The choice of the right set of evaluation metrics for a given task is crucial in decision making and determines the final outcome of the respective process. Each metric has a defined value range and a significance to a problem context to be assessed. Therefore, each metric will yield unique results on a given problem that we would like to evaluate. To obtain the best evaluation result for a given task, we choose evaluation metrics based on the underlying problem context relative to the properties of the evaluation metric itself. In the following subsections, we explain the selected evaluation metrics that are capable of addressing task-specific evaluation problems and thus fulfil the requirements of automated decision making within the OCR pipeline and OCR text quality.

In order to get a better understanding of the following evaluation metrics and their calculation, the concepts of the text box and the line box in relation to OCR output data are presented visually in Figure 4.2 with three example attribute value lists each. A text box represents a recognised text and associated coordinate information that has been

classified by the OCR engine at a specific region on the input document. A text line box represents a single line in which several text boxes can be located and does not contain any text information.

Mean Line Confidence (mlc)

The first unsupervised evaluation metric is based on confidence values that are assigned to each classified text box resulting from the OCR extraction, as illustrated in Figure 4.2. This confidence in the context of optical character recognition defines the amount of certainty about the respective region of the document to being equivalent to a certain word or accumulation of characters. The confidence value range spans from zero to one hundred, lower values indicating a less confident recognition and high values a more confident recognition of the textual content. Values outside this value range, namely *-1*, indicate line boxes or abstract boxes and specify a unique identifier value for these identified entities. Depending on the task to be evaluated we would consider these confidence values either individually in case of a post OCR spelling error correction or all at the same time. To estimate the overall document text extracted confidence, we consider the mean of all confidence values while excluding unique line box confidence values and text boxes that only contain white space character text as such elements distort the correctness of the metric.

Word Length

The word length represents the second unsupervised evaluation metric and indicates the number of recognised text boxes from an OCR text extraction for a single document. As with the mean line confidence calculation, line box confidence values and text boxes that only contain white space character text are filtered in advance. The OCR models we utilize have been trained to identify single words and generally a data row which we refer to as a box contains a single word at a time. Due to line breaks, a box field may contain a split word, and due to narrow spacing, multiple words may occur within a text box.

For documents featuring inconsistent lighting conditions, the word length metric is especially supportive in estimating the correct parameter configuration for the binarization step, as for this type of document this value may vary significantly. In such cases the word length is thus an indicator for the completeness of a text extraction process. Nevertheless, the word length metric is never used solely as a decision-making metric along the OCR pipeline since in the case of binarization, for example, the certainty about the confidence is additionally relevant. By incorporating the word length metric, exceptionally high or low word lengths can be detected and, if required, the corresponding transformation candidate can be excluded from the respective process.

Mean Line Height

The mean line height evaluation metric is calculated from the average value of the height parameter of all line boxes and thus represents the average text line height for a document.

Empty line boxes that do not contain text boxes or text information are excluded from the calculation. The value is numeric and represents the average document text height of the recognized text in pixels. In the document resize step, the mean line height metric is captured on the basis of various document scalings. In the subsequent decision process of determining the optimal document resolution, the mean line height is used in addition to the mlc and the word length to exclude scaling candidates with mean line height values lower than a defined threshold. Furthermore, the determined mean line height value is used in the binarization step to support the parameter selection process for the adaptive thresholding algorithm.

Word Error Rate

In addition to the unsupervised evaluation metrics, the word error rate (wer) is used as a measure of similarity for the evaluation of two texts, the hypothesis and the respective ground truth. The ground truth which was transcribed manually by human hand represents the target result to be achieved. The hypothesis is obtained by an OCR process and represents the text to be evaluated against the ground truth. In our scenario, word error rate is calculated by the minimum edit distance between the human-transcribed ground truth text and the machine-extracted hypothesis text. The minimum edit distance is computed by the number of editing operations required to transform the hypothesis into the ground truth. Under the condition that ground truth data is available the word error rate can therefore only be utilized outside the OCR pipeline for performance evaluations and benchmarking experiments.

4.3 Capturing Metadata

The metadata generated and captured along the OCR pipeline includes evaluated algorithm parameters and document specific metrics that have been determined for each processing step. Furthermore, binary records are kept of whether document transformations have been performed and applied successfully, to be used as logging feature, for error handling and to ensure that document transformations such as the rotation correction are only applied once. Stored metadata comprises the corrected rotation angle, document height and width scaling values, binarization parameters as well as the final OCR text, spelling error corrections, extracted temporal contexts and all output file paths.

With respect to the runtime performance requirement as described in Section B.3, computing-intensive processes must be performed in a time-efficient manner, respecting the output quality to be maintained. This makes it all the more important to store information that has already been obtained in such a way that it can be retrieved in order to avoid redundant and additional computationally intensive calculations.

The continuation and maintenance of unique document identifiers is not only an important requirement for the historians but also allows information from different processing steps

Document Label	Extension	Bit Depth	Processing & Compression	Specific Attributes
Master Copy	TIFF	16	unprocessed & uncompressed	Adobe (1998), dpi: 300 px/inch
Production Master	TIFF	16	processed & uncompressed	Adobe (1998), dpi: 300 px/inch
Access Copy	JPEG	8	processed & compressed	created from Production Master, Quality: 50, ICC profiles: sRGB, scaling: fixed 50%
Access Copy	PDF	8	processed & compressed	Facimile access copy version annotated as in OCR pipeline step 7

Table 4.1: Table of raw file conversion properties.

to be merged and reused along the OCR pipeline. Metadata is stored and passed as dictionary object between the internal processing steps to avoid overhead by executing read and write memory operations. With reference to the requirement from Section 2.6 for a transparent extraction history, the collected metadata is added to the output as *JSON* format file and represents the solution to enable traceability of applied document transformations for the historians.

4.4 Pipeline Steps

In this section we describe the functionality of each processing step in the OCR pipeline, associated experiments, findings and outcomes. It includes explanations, evaluations and considerations for design decisions we have implemented along the OCR pipeline processing steps. A graphical overview of the processing steps and their sequence in the OCR pipeline is illustrated in Figure 4.1[1].

1. **Raw File Conversion**

   The first step in the OCR pipeline refers to the raw file conversion requirement mentioned in Section 2.10 and deals with the conversion of raw type files, image files that are uncompressed and have not been processed in any way, into project specific image output formats. Since the manual process of file type conversion and file re-naming requires a considerable amount of time and computing resources, this optional step is integrated into the OCR pipeline, under consideration of the requirements discussed in Section 2.2 and 2.6.
   The defined image output formats contain specific file properties such as image

[1] icons taken from https://www.flaticon.com/ (accessed 16.08.2021)

compression, color bit depth and file type. It is important to mention that this step is optional, project-specific and therefore has no connection or dependency to any other steps of the OCR pipeline. In order to extend usability and accessibility of input raw files for further applications, they are converted to the specific file types described in Table 4.1.
Excluded was the conversion to the raw file type Digital Negative (DNG) for an archival copy due to the data loss of a raw to raw file type conversion. For the file conversion we used the python library rawpy[2] which allows to configure the raw sensor data with more than 20 parameters to produce the expected output files. The difficulty was to determine the parameter values in order to achieve the above mentioned document properties while avoiding inconsistencies in contrast, exposure ratios and others. Output documents resulting from various parameter combinations were tested and evaluated together with the historians who requested the raw file conversion feature, as the output document should have similar properties to those obtained by transforming conventional raw file document viewers. Different conversions resulting from different parameter settings were presented to the historians and the configuration that satisfied the historians was finally applied.

2. **File Transformation**

Due to different data archiving and preservation strategies, archived documents often have different characteristics, such as file type intended for data preservation. In this first and essential step of the OCR pipeline, we build an equal file format foundation by creating a work copy of the documents in order be able to apply and perform analysis activities and transformations. The target is to create a uniform file format basis by transforming any input document into the *PNG* file format. Due to uniformly created file type properties based on a common file type basis, the converted documents have a more uniform behaviour with respect to document transformations that are applied in the following processing steps. Valid input file formats include PDF, JPG and PNG, as well as the two raw formats ARW and DNG. We chose PNG as the base format for the pipeline because its lossless properties and quality preservation are ideal for text extraction and long-term image archiving. Additionally JPG can be configured as base format if required. PDF files consisting of single or multiple pages are also extracted into single PNG files and are merged[3] back into a single document in the data export step.

3. **Preparation**

In the preparation step, a grayscale work copy and an original work copy are defined and generated for each input document, which are used exclusively in the

[2]rawpy: https://github.com/letmaik/rawpy (accessed 16.08.2021)
[3]PyPDF2: https://github.com/mstamy2/PyPDF2 (accessed 16.08.2021)

subsequent processing steps. The dot per inch (DPI) value can be configured and is set to 300 by default whenever a document save-process is executed.

Original Work Copy – The PNG document resulting from the previous stage represents the so-called original work copy and is used, among other things, as a background image for the annotated PDF document produced at the final export stage in the OCR pipeline. The generated original work copy PNG file is additionally returned at the end of the OCR pipeline due to the transparent extraction history requirement mentioned in Section 2.6.

Grayscale Work Copy – Also part of the file transformation step is the creation of the work copy document as grayscale transformation which is generated from the previously exported original work copy. Input documents may appear in color, grayscale or in a binarized color scale consisting only of black and white pixels. To transform a color image to a one channel grayscale image we apply luma transformation which leverages a matrix of luma coefficients that specify the color conversion for each respective color channel (RGB). The luma transformation Y with the specified coefficients according to the standard BT.601 is defined as follows:

$$Y = (R \times 0.299) + (G \times 0.587) + (B \times 0.114) \tag{4.1}$$

The positive side-effect for the subsequent steps in the processing pipeline is the reduced amount of data to be processed and the resulting shorter processing time. The work copy is used to collect document-specific meta-information about transformations and to perform text extraction.

The status after the preparation stage is a uniform file basis of all input documents as well as the existence of an original work copy in color and a work copy as grayscale document.

4. Rotation Correction

Tesseract 4 is able to handle two directions of rotation, the human readable format (upside up) and the rotation flipped 90 degrees clockwise (upside right). Hence, we need to perform rotation determination to verify if the actual document rotation is either of those two valid options and subsequent rotation correction if required. Based on a task specific dataset of documents to be corrected in their rotation, we came up with a cooperation of the following rotation correction methods to minimize the rate of rotation misclassification. Furthermore, we make the assumption that the archival documents have been captured in one of the four multiples of 90 degree rotations (0, 90, 180, 270) so we do not consider any fine grained rotation angles for skewness-correction.

Tesseract OSD Detection – The first level of rotation correction was originally done by the built-in orientation and script detection (OSD) feature. We discovered

that this technique showed fluctuating results with regard to the accuracy of the proposed correction. There were also problems with handling large files, in terms of document resolution (length-width ratio), that could not be processed. We first considered and verified only the high confidence values as being correctly identified document rotations. At a later stage of the project we decided to discard this method to be substituted with a more robust rotation correction solution.

2-Rotate Rotation Correction – We developed a rotation correction strategy to determine and correct any input document rotation. Tesseract is able to process an input image in any orientation, but only two orientations provide text results that are human readable. Based on the fact that tesseract is able to extract two orientations with regard to human readability, we have to extract the document twice to determine its current orientation. We first extract the text content of the document in its original rotation as well as in the form rotated by 180 degrees. The 2-rotate strategy returns two results to evaluate, which always includes one correct rotation that can be successfully processed by tesseract and one incorrect rotation that resolves in an unreadable text. If a text extraction fails because the input document is too large, the resolution of the document is adjusted. The document resolution is then resized by a percentage value and then the 2-rotate method is applied again.

For the original document as well as the rotated document we perform a text extraction and calculate the mean line confidence (mlc). Therefore, the result consists of one mlc value for each rotation, usually one lower and one higher value while the higher value indicates the valid rotation and the lower the invalid rotation. To determine which value represents the correct rotation, the difference between the two mlc values is calculated and evaluated. If this difference is greater than 20, the result is considered significant and the corresponding rotation represents the correct rotation.

If there is still no evaluable numerical or significant result, for example, in the case that there is no extractable content on the document at all, the document is transformed and evaluated by the further rotations 90 and 270. The two additional rotation data entries complement the previously collected information and can support to determine the correct rotation if the previously collected information was insufficient. The significance threshold value, which determines a sufficient difference between two mlc values is set to 20 on default because this determined value represents a clear and sufficient difference for deciding the correct document orientation. If no significant result is available, the highest measured mlc respectively its corresponding rotation value is used for subsequent rotation correction.

Gathering initial Meta-Information – Another important aspect in this stage are the initial mean line confidence and text length metrics which are necessary for decision-making in the subsequent processing steps. In order to avoid wasting compute resources and time we exclude the processing of files that do not contain any extractable content, indicated by an undefined (not a number) mlc value.

Determining upside up Document Rotation – After the successful detection and correction of the rotation as a basis for the text extraction for Tesseract we also need to determine the human readable upside up rotation to avoid manual rotation correction for the end user. We use the fact that text words and lines usually have a higher pixel value in their width than in their height. Therefore we use the text-box data from the previously corrected rotation to obtain the sum of text boxes that have more width than height and the sum of text boxes that have more height than width. The higher value finally decides whether the image is already aligned in an upside up position or needs to be rotated 90 degrees to counter clock wise in order to be upside up.

Mean Line Height – Furthermore, the mean line height which can also be referred to as the text size is determined on the basis of the text-box height data of the upside up readable rotated document. The mean line height carries document specific information and is a crucial decision parameter for the subsequent processing steps to perform further document transformations accurately.

The status after the rotation correction step includes correctly rotated documents that are machine and human readable and the extraction of document-specific meta information to be used for further processing.

5. **Document Resize**

Input documents may differ in resolution and therefore the character size of the printed text will be affected accordingly. For reasons of document preservation, archival documents are usually recorded in high resolution in order to capture as much information or detail as possible. Therefore, it is important to mention that a higher image resolution does not imply that the expected OCR text quality will be better. If the font-size of the characters, measured in pixels, is too large or too small the OCR model might misclassify the word or not even detect the text segment at all. This scenario occurs due to the fact that a OCR model is trained on a specific annotated dataset that might not have included documents with such font or character height properties. Therefore, we resize the image to different resolutions and evaluate the best performing setting on the basis of the metric data collected. For image resize operations[4] we use the bicubic interpolation algorithm which considers the closest 4×4 neighbourhood of known pixels to estimate an unknown pixel value. If an improvement or an equivalent result can be determined with a shorter processing-time, the resize operation will be applied to the document. The reduced processing time due to the equally reduced amount of data is a positive side effect of the document resize process in regard to further document transformations in the OCR pipeline.

Figure 4.3 consists of four independent graphs, each representing a relationship between the document scaling, expressed in percentage of the original image size,

[4] pillow resize method: https://pillow.readthedocs.io/en/stable/reference/Image.html?highlight=resize#PIL.Image.Image.resize (accessed 16.08.2021)

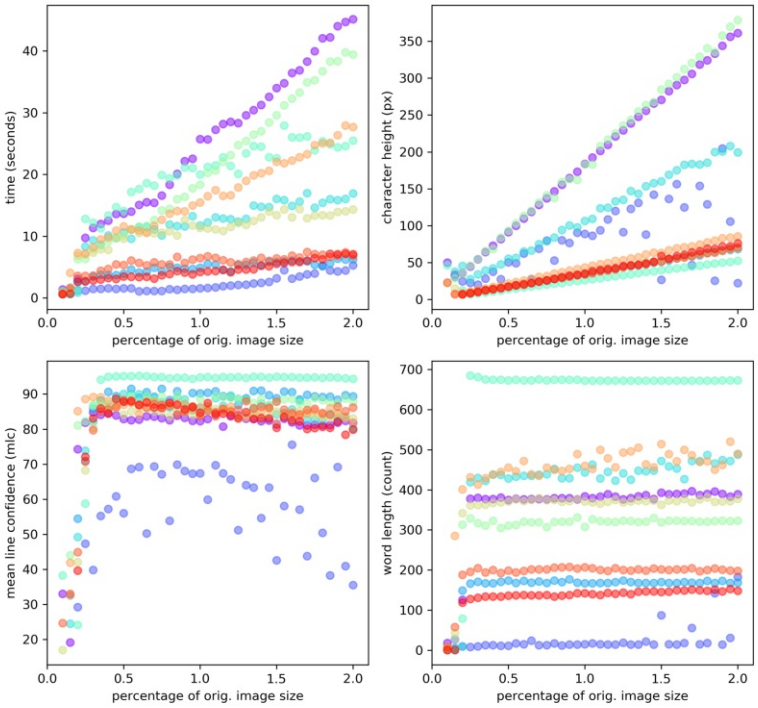


Figure 4.3: Four individual plots that illustrate the effect of document resolution on OCR text extraction with regard to the processing time, character height, mean line confidence and word length. Each colour corresponds to a unique document which has been scaled 40 times between the values 0.1 and 2 of the respective original document size.

and an evaluation metric on the Y-axis. These graphs support the following text on why and how the collected metrics contribute to the evaluation of the optimal resolution. The linear relationship between the document resolution and the text extraction runtime is shown in the upper left plot in Figure 4.3 and thus influences any further text extraction and document transformation in the OCR pipeline.

Resolution Settings – The resolution settings to be evaluated are based on percentages and calculated from the document specific height and width attributes. By default the resolution settings consist of the values 0.2, 0.3, 0.4, 0.5 and 0.8 and can be adjusted via the configuration object of the OCR pipeline. We only scale to smaller resolution formats, based on the original document resolution, as otherwise inconsistencies in the expected results may occur. Such an inconsistency can be observed in Figure 4.3 in the lower right graph where one document unexpectedly multiplies its word count at scales higher than the original document. Figure 4.4 illustrates the effect of the respective scaled up image, which shows that when scaled too high, the OCR engine does not recognise the text content, but many

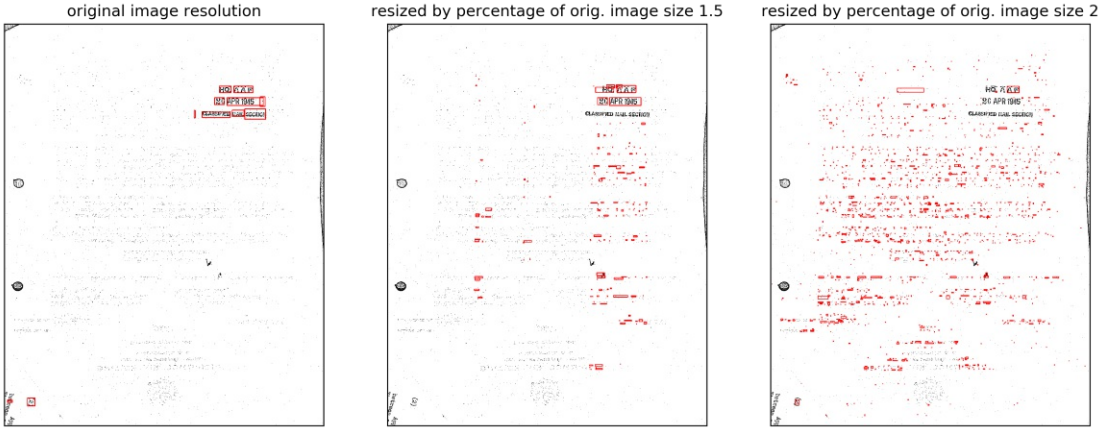


Figure 4.4: Up-scaled resolution causing inconsistencies in a thresholded input image.

irrelevant and single letters, resulting in the exceptionally high word count. The red rectangles visualise regions where text was recognised. It is important to note that this particular image has been previously binarized resulting in some black pixels which, at high document resolution, are recognised as text symbols by the OCR engine. Subsequently in the shape determination, we have to deal with the detection and exclusion of such cases.

Evaluation Metrics – In order to enable the subsequent document resolution evaluation, the following metrics are required to be captured for each shape transformation. The mean line confidence, together with the word length metric, provide information to enable the identification of outlier shapes that deviate significantly from the average results. The mean line height, which was already calculated in the previous pipeline step on the basis of the original scaling, is an essential decision factor for the shape evaluation process and will be decisive for the quality in the following binarization step. In the upper right graph in Figure 4.3, the mean line height metric illustrates its linearity and thus represents a robust and reliable parameter that can be effectively used as a filter parameter to exclude resolution settings below or above a certain threshold[5]. Some documents show a lower linearity which is due to documents from which text is difficult to extract, reflected in the mean line confidence of the respective scaled document.

Shape-Determination & Application – The collected metric data represents the foundation for the determination of the optimal document specific resolution. In the lower left plot of Figure 4.3 we see that document resizing operations below 0.2 percent of the original image size yields a low mean line confidence metric and in the lower right plot we can observe the equivalent phenomena for the word length

[5] inspiration for document resize https://groups.google.com/g/tesseract-ocr/c/Wdh_JJwnw94/m/24JHDYQbBQAJ

metric. This knowledge gives us certainty to remove exactly these transformation outliers in the first filter iteration. We want to remove transformations for which the OCR engine was not confident in the text extraction process. Therefore, candidates with a low mean line confidence value are removed by applying a lower threshold that is derived from the average of the mlc distribution minus one single standard deviation. Next, z-score is used to eliminate shapes that deviate from the word length distribution. Finally, we use the mean line height to filter shape candidates with too low character heights using a threshold that defaults to 18 pixels. From the remaining candidates, the transformation with the fastest runtime or respectively the lowest resolution is selected and applied to the document. In case a filter operation fails or there is insufficient or no data available, for example if no candidates remain after the outlier removal process and the subsequent mean line height threshold filter, the process decides for the default unscaled candidate shape.

The status after the document resize step includes potentially resized documents which have adequate or improved properties for text extraction. Under the condition that the resize operation has been applied the runtime will be lower for further extraction iterations in the subsequent OCR pipeline.

6. Binarization

Document-binarization narrows down the range of colors of a grayscale document to two values, black and white, and therefore will provide support for the OCR model to classify document-regions as text and background[6]. The effect of the binarization document-transformation effect can be observed in Figure 1.1. The binarization processing step is the main contributor in document improvement for the final OCR extraction and is the most time and resource intensive due to the constant quality requirement to be ensured. Document specific difficulties underlying the historical archival documents, as explained in Section 2.9, require the selection of a robust binarization method. The adoption of non-static and document specific binarization parameters is essential to counteract diverse disturbing factors such as changing background illumination or contents that shines through from the back side of the document.

In order to justify the application of a binarization transformation, the document-specific parameter values for the selected binarization method must be tested and evaluated. For the estimation and evaluation of the binarization results, the mean line confidence metric was applied, which is computed from the OCR-text resulting from the binarized document.

Thresholding Methods – For document binarization, we exclusively use adaptive thresholding methods which are robust against archival document specific contaminations. In contrast to global thresholding methods, where none or at least one

[6] opencv-python: https://docs.opencv.org/2.4/index.html (accessed 16.08.2021)

parameter has to be set, the adaptive mean and adaptive Gaussian thresholding methods[7] we use require three parameters to be configured. The adaptive mean thresholding method defines a local threshold $T_{mean}(x, y)$ by calculating the mean of its surrounding $s \times s$ blocksize neighborhood pixel gray values p_{ij} minus a constant C value and is defined as follows:

$$T_{mean}(x, y) = (\sum_{i=1}^{s} \sum_{j=1}^{s} p_{ij})/s^2 - c \tag{4.2}$$

The adaptive Gaussian thresholding method defines a local threshold $T_{gaussian}(x, y)$ by calculating a weighted sum of its surrounding $s \times s$ blocksize neighborhood pixel gray values. Each weight for the corresponding neighbourhood pixel is obtained from the Gaussion filter function based on the distance from the center pixel (x, y). The adaptive Gaussian thresholding method is defined as follows:

$$T_{gaussian}(x, y) = (\sum_{i=1}^{s} \sum_{j=1}^{s} w_{ij} \cdot p_{ij}) - c \tag{4.3}$$

On the one hand our selected thresholding algorithms are advantageous because the method can be set precisely to obtain an exact separation, but on the other hand it is also time consuming and computationally intensive because it is necessary to choose from a large number of possible values for each parameter in order to find the best document-specific configuration.

The c-value c represents a constant value parameter that is subtracted from the mean or weighted sum of the neighborhood pixels. The valid value range for the c-value is -255 to 255. If the subtraction of the c value results in exceeding the grayscale value limits, the threshold is set to zero (black) in case of a negative value and if the value exceeds 255 it is set to 255 (white).

Parameter (Range) Tuning – In order to optimize the process runtime as well as the quality of the resulting binarization, the value range for each input parameter is selected document specific. In the following list, the approach for the parameter value range determination is described for each of the three input parameters for the adaptive thresholding binarization method.

- **Blocksize** – The blocksize parameter s determines the size of the local neighborhood area centered around each pixel. The reference to local pixel neighborhoods makes the blocksize parameter mainly responsible for the robustness against different lighting conditions on the same document. If this parameter is set too high or too low, there is too little or too much image information for the classification, which leads to a poor overall binarization result. Because the blocksize parameter is only limited by its document-specific

[7] adaptive mean thresholding & adaptive Gaussian thresholding: https://docs.opencv.org/4.5.1/d7/d4d/tutorial_py_thresholding.html (accessed 16.08.2021)

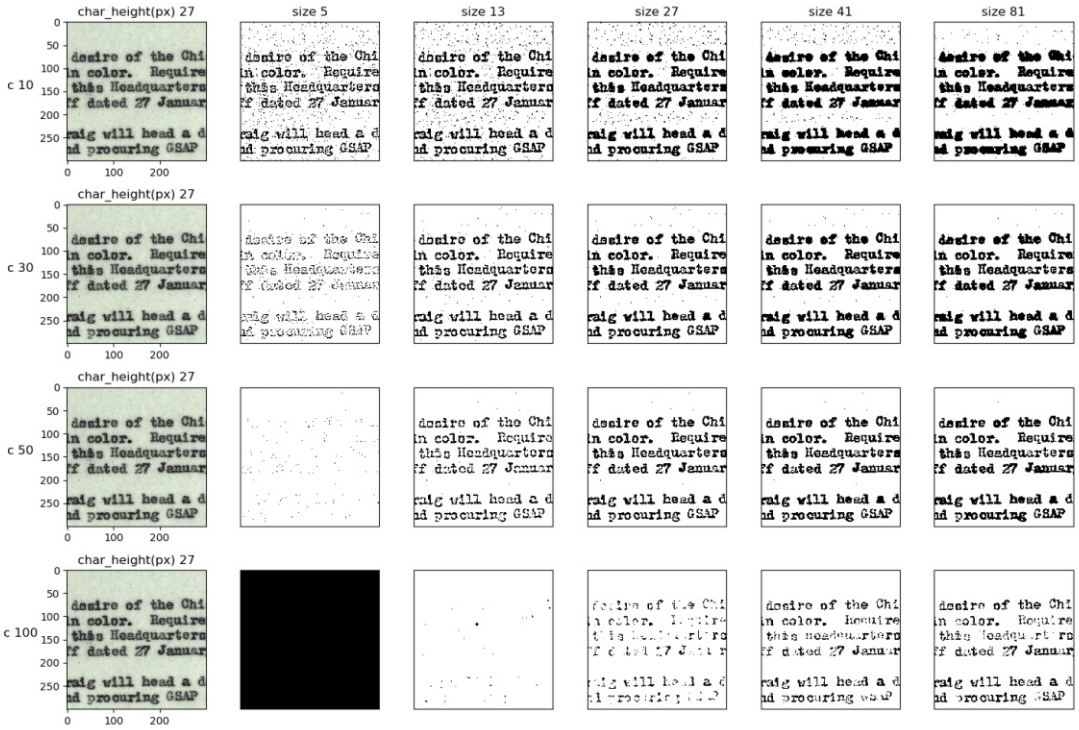


Figure 4.5: Center cropped binarization parameter selection.

resolution, its height and width attributes, a wide range of values is possible. We have tested and evaluated the following two methods for selecting blocksize value ranges and implemented the second variant due to the potential for a precise selection of test values which lead to a lower duration while maintaining the threshold output quality.

a) **Static Parameter Range** – The first method of selecting a parameter range refers to a static subset of values. For testing and evaluating a static blocksize parameter range, we used 6066 documents from an archive provided by the historians. Approximately half of these documents were not suitable for the test because they did not contain any text content to be extracted. To define a static value range, we first defined 15 blocksize values between 3 and 153 to be evaluated. For each document, subsequently all blocksize values were applied using the adaptive mean thresholding method and the adaptive Gaussian thresholding method. For each thresholded output document, a text extraction was carried out and the mean line confidence and the word length was calculated. Then, for each document and its resulting transformations, outliers with respect to the word length metric were removed and, of the remaining

transformations, the candidate with the highest mean line confidence was chosen as the best blocksize value. Finally, it was possible to define a global optimal blocksize parameter range leveraging extracted evaluation data, but due to individual document properties such as document resolution, this static range is unacceptable with regard to runtime performance due to its size of values. Due to the inflexibility of a global blocksize parameter value range, qualitative binarization results are achievable only with respect to a high runtime, which is not suitable for our application in the real scenario.

b) **Dynamic Parameter Range** – Due to the uniqueness and variety of types of document contamination, statically determined values might not work for all documents as they were only obtained from a selected set of documents. Applying a *one fits all solution* works well on the data based on which the globally determined blocksize value range was determined but brings unpredictable results on documents with different features.
Our second approach aims at choosing the blocksize parameter range based on document specific properties. Therefore, we minimize the number of combinations to be tested and are still able to test under the best parameter constellations. The dynamic blocksize range is determined on the basis of the mean line height determined in the *Document Resize* processing step, which expresses the average height of the printed letters in pixels. By default, two additional values are derived from the document-specific mean line height. The final range consists of three values, a negative offset from the mean line height value, the mean line height value itself and a positive offset from the mean line height value. The offset percentage range can be configured if required, e.g. to have more or fewer binarization options examined with respect to the parameter blocksize. The offset percentage range list contains 0.5, 1 and 1.5 by default. We have chosen the document-specific mean line height value as the decision parameter for the blocksize parameter value selection in order to achieve good binarization results regarding a text extraction process. In addition, the dynamic approach allows the process runtime to be configured by making the blocksize offset percentage range list configurable.

Figure 4.5 illustrates 20 different binarization parameter settings applied to the original document which is shown in the first column of each row as reference. Each applied binarization illustrated consists of a value combination of the blocksize (on the x axis) and the c-value (on the y axis). The dynamic blocksize range was determined from the mean line height, which for this document amounts to 27 pixels, and calculated for the additional percentage factors [0.2, 0.5, 1, 1.5, 3] to demonstrate the effect on the binarization result of different combinations of blocksize and c-value from the initial reference value of the mean line height. The images of Figure 4.5 have been center cropped and zoomed in to illustrate the binarization differences in detail.

We can observe that extreme values regarding the blocksize, which deviate too much from the initial character height value (the factors 0.2 and 3), produce faded or no longer recognizable results. When performing a text extraction on the two binarized images with a blocksize of 0.2 and 3, the calculated mean line confidence results accordingly in the lowest values of all other percentage factors. In contrast, the blocksize parameters factors 0.5, 1 and 1.5 in combination with a c-value below 100 provide readable and clearly separated binarization results. Identical observations were made on ten other documents.

- **C-value** – The purpose of the c-value is to avoid noise elements in small blocksize windows by shifting the threshold value by the constant defined value. If, for example, a blocksize window contains only background information, small noise elements would be classified as foreground, which is to be avoided by the c value.

  The c-value range was defined utilizing the mean line confidence metric in order to exclude extreme values and to narrow down the range of parameter candidates. Furthermore, the well performing value range of the c-values can be determined empirically by observing the binarization results as in illustrated in 4.5 where extreme values above 100 did not lead to an improvement but to a deterioration of the binarization.

- **Adaptive Method** – The adaptive method parameter defines the algorithm that determines the pixel specific binarization threshold $T(x, y)$ to classify the pixel white or black. The two types of adaptive thresholding *ADAPTIVE THRESH MEAN C* and *ADAPTIVE THRESH GAUSSIAN C* of adaptive methods are available. Both methods require the same parameters to be defined. With the adaptive Gaussian thresholding method, no additional parameters need to be defined for the calculation of the Gaussian filter, as the defined blocksize is used as the kernel size.

  Both thresholding methods proved to be important and complementary, especially with regard to the attribute of image sharpness, which refers to the blurriness or clarity of the printed characters. The Gaussian window filter operation of the *ADAPTIVE THRESH GAUSSIAN C* method softens the resulting binarized image, benefiting especially images with sharp or thin printed letters. In case of poor quality documents with blurred and fuzzy characters, the *ADAPTIVE THRESH MEAN C* thresholding type is advantageous as no additional filtering option is applied resulting in clear and sharp characters.

Configuration Evaluation – In order to determine the optimal document binarization, an evaluation of all parameter combinations is carried out on the basis of the determined parameter ranges.

With regard to determining the best binarization setting we investigated the relation of the applied mean line confidence and word length metric to OCR accuracy based

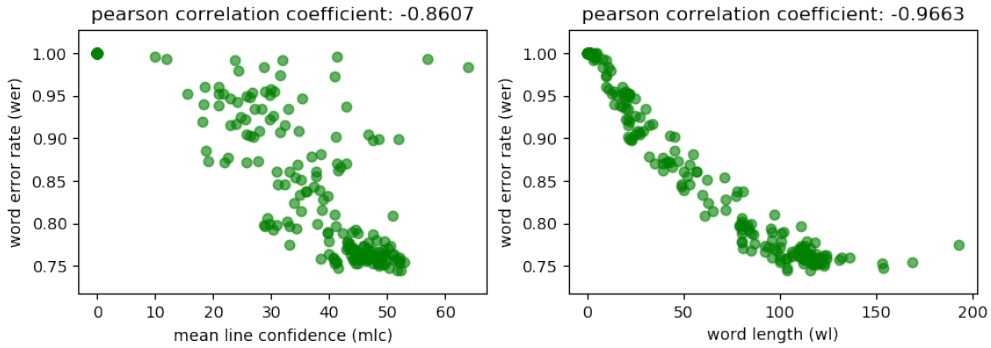


Figure 4.6: Showing the relation between the mean line confidence and the word length to the word error rate to demonstrate the relation of the metrics to OCR accuracy. Each data-point represents a text extraction based of a thresholded version of one document.

on ground truth data and the word error rate evaluation metric. Nine documents, for each of which we had a ground truth text available, were binarized 240 times with different parameter configurations and subsequently a text extraction was carried out on the basis of the thresholded document. Based on the resulting text, the word error rate to the ground truth, the mean line confidence and the word length were calculated for each configuration. The relation of the calculated evaluation values for one of the documents is shown in Fig 4.6. Finally, we observed an average Pearson correlation coefficient of 0.84 between the word error rate and the mean line confidence and 0.94 between the word error rate and the word length.

In the selection process, binarization candidates with a word length value lower than the average value minus one standard deviation are excluded. From the remaining configurations, the transformation setting with the highest mean line confidence value is selected and applied to the document. In the case that no binarization configuration leads to an improvement of the document with respect to the mean line confidence and word length that we obtained from the resize step, the transformation is not applied to the document.

7. Data Extraction

In the final stage of the OCR pipeline we apply the knowledge we have gained about document transformations to retrieve the OCR text output. For the final text extraction we use the so called *tessdata_best*[8] models for the tesseract 4 engine which provide the most accurate OCR result while having a longer runtime. The data is extracted and exported to the following file types to serve different purposes of application.

[8]traindata best: https://github.com/tesseract-ocr/tessdata_best

- **hOCR** – hOCR is a form of representation of text data obtained from the OCR process that additionally captures layout information, coordinates and metrics such as the recognition confidence. This additional information about text positions and confidence values extends the application possibilities of the data and is the foundation for interactive applications. To mention one use case, our Bulk OCR Web Service (BOW) [A] employs the processed hOCR data and additional entity annotations as an interactive result preview. With simple interactions, filter options based on the confidence values can be implemented. In addition, the hOCR output provides the basis for the subsequent PDF generation.

- **Standard Output** – The standardised output in *JSON* data format contains, in addition to the obtained hOCR data, meta-information on the document transformations that were acquired within the OCR pipeline and applied to the document. This purely textual and structured output is particularly suitable for machine-driven tasks or data preservation as the format is accessible and standardised.

- **PDF** – The Facsimile PDF output is primarily intended to represent a true-to-original document with embedded text annotations. For this reason, neither the image resolution may be altered by enlarging or reducing it, nor the background image, for example, by the binarization may differ from the original document in the output document. The PDF file can be generated directly by Tesseract with the required facsimile properties, but the document transformations and knowledge previously gained in the OCR pipeline are not utilized and incorporated.

  To avoid further text extraction iterations and to ensure consistent quality among the different output formats, the facsimile document is created from the previously extracted hOCR text and the original document. The bounding boxes given by the hOCR output, which are provided with coordinates, are used to create text annotations on the PDF. The length and height of the PDF document page to be created is determined by the properties of the original document. For each text box, the text inside is stretched to fill the full width of the box to make the positioning of the text as accurate as possible to the word below it on the document, to enable for precise text selection. The output of multi-page annotated PDFs is supported if the input document consists of several pages.

  An additional phantom-character-removal method removes text fields from the hOCR data that have oversized box properties or contain only irrelevant text content such as exclusively whitespace characters. This method is crucial for the usability of the output PDF documents, as too large, non-information-bearing fields can affect the precise text selection of the document. In summary, the creation of annotated PDF documents from previously extracted hOCR data enables the application of filter operations or the precise use of positional

information and allows the creation of high quality and usable facsimile documents.

File ID	*Ground Truth Length*	*WER ABBYY*	*WER BOW*	*ABBYY-BOW WER difference*
a	694	57.22	15.98	41.24
b	1136	43.63	30.9	12.74
c	1837	3.0	3.26	-0.26
d	1217	36.66	33.4	3.26
e	673	52.93	31.79	21.14
f	659	93.89	71.48	22.41
g	1570	4.97	11.3	-6.32
h	2190	4.64	6.12	-1.48
i	1927	35.69	14.4	21.29

Table 4.2: BOW & ABBYY FineReader benchmark dataset showing word error rates (WER) for each document.

mean ABBYY Length Offset	*mean BOW Length Offset*	*mean-error ABBYY*	*mean-error BOW*	*ABBYY-BOW difference*
63.77	17.66	36.95	24.29	11.75

Table 4.3: BOW & ABBYY FineReader benchmark results presenting mean length offset values as well as mean word error rates for BOW & ABBYY FineReader.

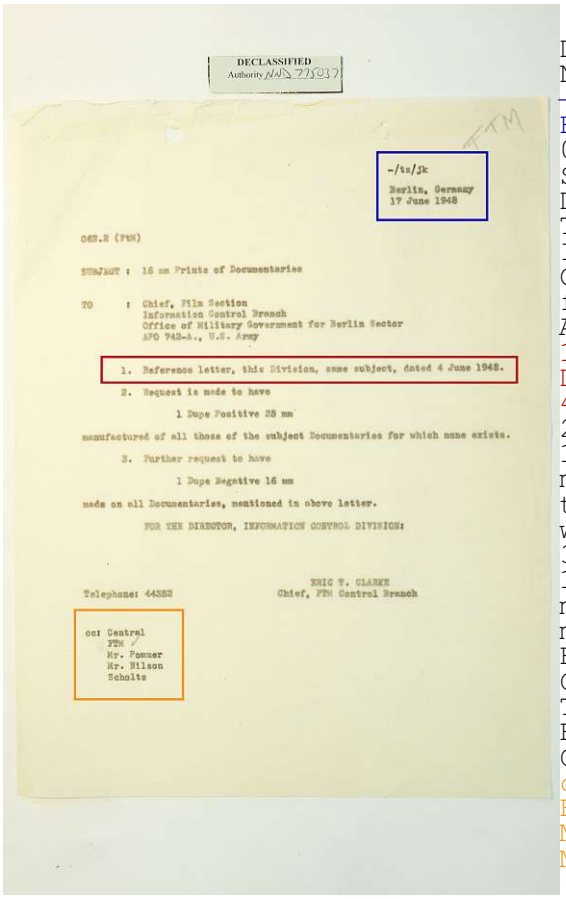
4.5 OCR Quality Benchmarking

Evaluating the text output quality of our BOW compared to a current in use tool is a prerequisite to justify its application. In our scenario, the OCR text results are evaluated against those of the ABBYY FineReader[9]. The evaluation dataset, consisting of nine documents, was selected by historians to target the digitisation of critical or difficult-to-extract documents. In Table 4.2, the evaluation dataset consisting of nine documents is labelled with unique identifiable letters (from a to i) in the *File ID* column. To access the full name resolution of the documents or to preview the documents, we refer to the Appendix C.

To evaluate the output texts of the two OCR tools, we created a ground truth text for each document which defines the text quality to be achieved for the two OCR engines. Information such as handwritten annotations were not considered in the ground truth.

[9]ABBYY FineReader https://www.abbyy.com/ocr-sdk/ (accessed 16.08.2021)

Preprocessing steps of the output text before comparison include transformation into lowercase characters and removal of multiple white-space characters. The Word Error Rate (WER) metric is then used to determine the text similarity of the texts from the BOW and the ABBYY fine reader engine to the corresponding document ground truth.



(a) Document: NARA_RG-260_HMSMLR-A1-260_Box-265_Documentary-Production-Unit-NAID-7550505_029

(b) Ground ground truth text

Figure 4.7: Document (File ID: a) with ground truth text next to it.

The evaluation results are presented in Table 4.3. The mean length offset value indicates how close the number of recognised words is to the number of the words of the ground truth text, regardless of their correctness to the ground truth value. The error refers to the evaluated word error rate and the error difference is calculated by the document specific difference between the BOW error and the ABBYY FineReader error. A high positive difference indicates superior quality of the BOW, while negative values indicate inferior quality compared to the ABBYY FineReader. In summary, the BOW has a mean error rate of 24.29 and the ABBYY FineReader a mean error rate of 36.95 on the

```
| DECLASSIFIED
Authority NAY 775037)
-/t3/ 3x
Barlin, Germany 17 June 19548
082.2 (Fem)
SURJAUT 18 mm Prints of
Documentaries
70 Chief, Film Section
Information Control Franch
Office of Hilisary Goverment
for Berlin
APQ T42-A., U.6.  Aray
1.  Reference letter, this
Divisicn, same subject, dated
4 Jume 1948.
2.  Request is mede $0 have
1 Pupe Positive 25
papufactured of all those of
the subject Documentaries for
which none sxists.
3.  FPuarther request to huve
1 Dupe Negative 16 um
msde on all Documentaries,
mentioned in adove letter.
POR THX IRFORMATION CORTROL
DIVISIONS
ERIC Ⓡ.  CLARKE
Telephone:  44352
Chief, FT! Contrel Branch
cct Central
FIM
Mr.  Pommer
Mr.  Nilson Scholts
```

(a) BOW Text

```
DECLASSIFIED
Authority AIaJsS "77 7
/xn a
Bwrlia* Oaraauy XT <ftmo 1949
049.8 (UK)
fWS0l i 16 am Prlats of
SrnmilafiN
TO t Chi«ft Fils Steitta
Xmforvati«* Cntnl imd
Offic* of Kllll«rf mrn«at fw
Barita Sootor
APO 743-A., 0.8.  Anqr
I. lifirtMi Uttir, this PMilia,
itai M>tj«ct, d*ti<
4 tat 1048.
8.  8t*ost 1« aal« I« heft
1 Jtop« fHlilT« 38 ms
aMOfftotarti if all tkaii if
ihm tafcjact ômiialirlM far
which aoaa «xltti.
3.  further r#û»t le hvo
X Dttp« 1*|«IIti 16 m
«mit on all Do6ua«at*rl«a»
moattoaed in ikn latter.
Y08 TKB DIA80T08, XVVQIMATI09
008T80X DITISIOI1
8816 9* 6IA1X8
talaphaaai 44388
Ctlif, TfH Oaatrol Iraaak
cot C@at.ral
im /
Nr.  PoEiMer
Nr.  Vilaon Seholts
```

(b) ABBYY FineReader Text

Figure 4.8: BOW & ABBYY extracted text side by side comparison.

given dataset. The word error rate of the BOW is 40% lower than that of the ABBYY Finereader. The results refer to the data-specific environment determined by historians working in the archival research domain and may not be generalised to all data and document types.

Figure 4.7 shows a document from the evaluation dataset and its corresponding ground-truth as text. The three text passages marked in blue, red and yellow in the ground truth text refer to the text content of the regions marked in the document image on the left side.

In the Figure 4.8 the OCR output texts of the BOW and the ABBYY FineReader are shown side by side to allow a direct comparison of the reading quality. The *BOW minus*

ABBYY word error rate difference of the document with the file identification *a* as to be observed in Table 4.3 amounts to *41.24* and the word error rate difference is also evident when reading the two texts. To better illustrate the comparison with the Ground Truth text shown in Figure 4.7, the same marked passages are also highlighted.

4.6 Summary

The multimodal OCR pipeline combines seven individual processing steps for correcting archival documents to subsequently perform text extraction. By means of a defined sequence of processing steps to be applied and several selected evaluation metrics, we have designed a method to make automated decisions within and between the individual processing steps. In detail, we present the seven processing steps and point out the respective evaluation metrics used in relation to the problem context of each processing step. In an OCR quality benchmark evaluation, we compare the text output of our OCR pipeline against the text output of the commercial product, the ABBYY fine reader, and perform superior on a dataset defined and transcribed by historians with a 40% lower word error rate.

CHAPTER 5

Spelling Error Correction

The post correction of spelling errors in OCR text is a crucial task that enhances document searchability, to enable efficient annotation of entities and to support or reduce the subsequent manual correction of errors. We have developed a four-stage spelling error correction workflow that entails text-preprocessing, error detection, error correction and result parsing processes.

In Figure 5.1 the architecture of the spelling error correction workflow with its four processing steps is illustrated along a text input example to demonstrate the processing state in each stage. In the upcoming section we describe the processing steps along their functionalities, methods and software libraries included. Finally, the performance evaluation in terms of document improvement is presented.

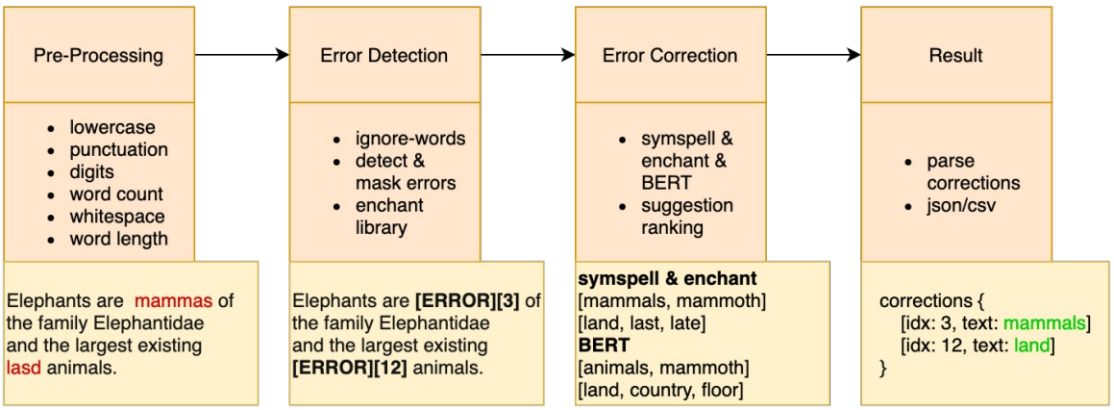


Figure 5.1: Spelling Error Correction Workflow illustrated showing the processing steps.

5.1 Text Preprocessing

Before any corrective measures are applied to the text, preprocessing steps are performed to enable detection of misspelled words, to avoid the masking of correct words and to be able to write each corrected word back to its original position in the index based *json* format.

- **Punctuation** – Punctuation marks are special characters which provide structure in the sentence structure. Punctuation marks can cause misclassification during error detection, causing correct words to be marked as incorrect due to the presence of punctuation marks. Therefore, in the first preprocessing step, punctuation marks are removed from each word using a regular expression (regex) which only allows letters and numbers.

- **Invalid Values** – Invalid values are *nan* (not a number) numeric data type values which occur in data records which are responsible for the structure of the document. These data entries are not to be corrected and are excluded through a simple filter operation.

- **Extract & Remove Digits** – Similar to punctuation marks, digits do not contribute to the correction of words and can cause misclassification in the error recognition process. However, the difference here is that characters are often interpreted as letters, and vice versa, as there are often similarities in the appearance of the script. For instance, the capital letter "G" can be recognized as the number "6" and vice versa. Text and numbers could also be written together as for example the incorrect the potential time entity *January1942*. A correction of this word would not lead to a successful recovery of the context. As a solution, all numbers are removed to make the respective word initially classifiable as misspelled or correct. In order not to eliminate the potentially correct numerical context, it is written in a separate column.

- **Lowercase** – The lowercase transformation of the text to be corrected ensures that exclusively capitalized or single uppercase letters that occur not at the beginning of a word are transformed to lowercase words. The lowercase transformation can misspell the beginning of an entity such as a name, but it primarily avoids the misclassification of spelling errors.

- **Whitespace** – Blank characters or multiple blank characters which are either caused by previous preprocessing steps or by the OCR process are cleaned up in the last step of the text preprocessing. We replace plain white-space and plain punctuation words with a special token that will be ignored in the subsequent correction process to be able to preserve the original document length and structure.

- **Word length** – Words consisting of less than two letters are excluded from the correction because these words are often stop words, thus contributing little

relevance to the document content and there are many correction measures which would all have a similar high similarity to the wrong word. Like empty words, they are marked as such with a token to be ignored and not corrected.

5.2 Spelling Error Detection & Correction

Our solution for spelling correction combines the collaboration of multiple strategies and libraries for error-detection and the subsequent correction.

- **Term Frequency** – To avoid the correction of potential named entities, words that express the names of people, locations and others, we build a list of words to be ignored in the subsequent error detection and correction procedure. We build the list of words to be ignored leveraging the term frequency (TF) to determine how many times each word from the extracted text occurs in it. Words counted more than three times in the text corpus are added to the ignore words list and are protected from a potential error correction.

- **Detect & Mask Errors** – We utilize the spell checker functionality of the enchant library[1] to iterate over the word tokens to detect and mark incorrect words. We store the words classified as incorrect and the corresponding index positions of the words in the given text corpus.

- **Correction Suggestions** – For the first part of the spelling error correction process, we utilize two different dictionary based word similarity-based libraries to retrieve the first correction suggestions for each word classified as incorrect. At first, we apply the symspell library[2,3] for spelling correction to retrieve suggested corrections for each word previously marked as incorrect. The same procedure of retrieving suggestions for each misspelled word is performed with the enchantment library.

- **Ranking & Thresholding** – After both dictionary-based correction suggestions have been collected, they are merged into one list, duplicates are removed and the remaining candidates are ranked according to a similarity metric. The similarity metric used for the ranking process is based on the Levenshtein distance[4] metric which measures the minimum number of characters to be edited (edit-distance) necessary to form the misspelled word.

  In order to further limit the correction suggestions that deviate too much from the word to be corrected, half the word length is calculated for the respective incorrect word, which represents the threshold that is used to filter correction suggestions

[1] pyenchant: https://pyenchant.github.io/pyenchant/ (accessed 24.06.2020)
[2] symspellpy: https://github.com/mammothb/symspellpy (accessed 24.06.2020)
[3] jiwer: https://github.com/jitsi/jiwer (accessed 24.06.2020)
[4] jellyfish Levenshtein distance: https://jellyfish.readthedocs.io/en/latest/ 16.08.2021

based on their individual word length. Often archival documents contain additional meta information for instance indexes such as page index, numerical text content and domain specific abbreviations that lie outside of the core document content. Another type of distortion which we do not want to correct and rather filter out is noise text that results from interference sources on the respective image region which were mistakenly interpreted as text during the extraction process.

As our method is not able to correct numerical data, abbreviations, index information or similar, we aim to avoid applying a spelling error correction on this kind of data. The majority of this type of data is excluded in advance in the text preprocessing stage and, if exclusion was not possible, the proposed correction candidates are restricted on the basis of a word length threshold.

- **BERT Suggestions** – Furthermore we leverage the pre-trained BERT[5] [7] language representation model to the correction process to perform context-based spelling error correction. At this point, we use the index positions extracted at the beginning of the error detection phase to access the respective text content before and after the word to be corrected. To receive context-based correction suggestions from the BERT model, we pass the contextual text phrases before and after the word that is labelled as incorrect. Based on the two contexts, the model returns k suggestions ($k = 80$ by default) to connect the two text phrases.

- **Suggestion Decision** – The BERT model will most likely suggest a word that would fit the context but may not match the word from the original document. To retrieve the most appropriate suggestions among the BERT suggestions, we incorporate the incorrect word, for which we want to find the best correction, by calculating the word similarity between each suggestion and the word to be corrected. BERT suggestions are treated separately from the dictionary based and ranked suggestions because in the case of a high word similarity to the incorrect word, the BERT suggestion is given priority due to its context based expressiveness. For the word similarity, Python's built-in Sequence Matcher[6] is used to determine a string of letter sequence similarity ratio between 0 and 1. If all k suggestions proposed by BERT show similarities below the threshold ratio of 0.75 to the incorrect word, we discard the suggestions from the BERT model and select from the three highest ranked dictionary based suggestions.

- **Result Parsing** – In order to make the correction suggestions available for each word classified as incorrect, all those suggestions are integrated into the JSON output data format in a separate column. Single and multiple correction suggestions are saved as a list and must be parsed accordingly when retrieved.

[5] pytorch-pretrained-bert==0.6.2: https://github.com/huggingface/transformers (accessed 24.06.2020)

[6] difflib sequence-matcher: https://docs.python.org/3/library/difflib.html 16.08.2021

File ID	*Ground Truth Number of Words*	*Recall RAW*	*Recall ABBYY*	*Recall CORR*	*Recall Difference CORR-RAW*	*Recall Difference CORR-ABBYY*
a	72	0.611	0.167	0.708	0.097	0.542
b	115	0.513	0.435	0.583	0.070	0.148
c	169	0.988	0.994	0.994	0.006	0.0
d	143	0.629	0.580	0.699	0.070	0.119
e	76	0.566	0.289	0.632	0.066	0.342
f	91	0.923	0.912	0.934	0.011	0.022
g	97	0.959	0.938	0.959	0.0	0.021
h	174	0.770	0.954	0.810	0.04	-0.144
i	192	0.786	0.693	0.849	0.062	0.156

Table 5.1: Spelling Error Correction evaluation dataset.

mean recall ABBYY	*mean recall RAW*	*mean recall CORR*	*mean recall difference RAW-CORR*
0.6624	0.749	0.796	0.046

Table 5.2: Spelling Error Correction evaluation results presenting mean recall values as well as mean differences of two respective mean recalls.

5.3 Spelling Error Correction Evaluation

Setup

To measure the effect of our spelling error correction workflow, we employ a recall metric that uses the bag-of-words vector representation of words. The bag-of-words model uses key-value pairs to represent unique words and the number of occurrences in a given text. First, the ground truth bag-of-words is created from the manually transcribed documents and thus represents the vocabulary of words to be matched by the OCR text. Then, the bag-of-words transformation is created from the plain OCR text and from the corrected text to determine the performance of the post spelling-correction.

For our evaluation, only the word occurrence is taken into account, but not the number of occurrences and therefore we classify binary whether a word occurs in the ground truth vocabulary or not. For a text to be evaluated, all words are classified according to whether they occur in the associated ground truth vocabulary. The result of the classification is represented as a bag-of-words $\vec{B_n}$ vector which describes which word occurs in the ground truth bag-of-words vocabulary $\vec{G_n}$. The sum of all words occurring in the text to be evaluated in relation to the sum of all words in the ground truth vocabulary determines

the ratio of the two vocabularies. This ratio is defined as our recall metric r as follows:

$$r = \frac{\sum \vec{B_n}}{\sum \vec{G_n}} \tag{5.1}$$

G_n = Ground Truth Vocabulary vector of size n of values [1]
B_n = Bag of Words Evaluation vector of size n of values [0, 1]
r = Recall Metric

A 1:1 ratio or a recall of one is the best result to achieve, as the vocabulary of the text to be evaluated contains all words from the ground truth.

Dataset

We used the same and already annotated documents from the OCR Quality Benchmarking evaluation from Section 4.5 as dataset, as these also show diversity in the context of spelling error correction.

In our evaluation we want to determine to what extent an automatic post-OCR spelling error correction based on the proposed error corrections is able to improve an erroneous OCR text. For this, we compare the plain OCR text and the OCR text that has been corrected on the basis of the proposed corrections against the ground truth text. To create the corrected text, we expand the plain OCR text with the first correction suggestion for each word classified as incorrect. The ground truth text was transcribed by historians and the plain OCR text was extracted by the OCR engine.

Findings

We developed the the spelling correction workflow with regard to minimizing the number of corrections per document, but with the highest possible text improvement, expressed by a high recall value. Through specific text preprocessing measures, words such as named entities and numerical expressions can be prevented from being recognised as errors and subsequently corrected, which ultimately keeps the number of false corrections low and the recall high.

Table 5.1 shows aggregated data of the evaluation dataset and related summative metrics are shown in Figure 5.1. The abbreviation *RAW* refers to the plain OCR text produced by the OCR pipeline, *ABBYY* refers to the plain OCR text produced by the ABBYY fine reader OCR engine and *CORR* refers to the text after applying spelling correction on the *RAW* text. We included the ABBYY fine reader text output in the evaluation to show that similar behavior to the *RAW* text produced by the OCR pipeline is observed in the OCR quality benchmarking from Section 4.5 in terms of the recall metric.

The number of unique words in each document is shown in the ground truth number of words (Bag-of-Word) column. How many words from the ground truth occur in the RAW text in the CORR text and in the ABBYY text is shown in the corresponding Recall columns. Each recall column reflects the recall metric as defined in Equation 5.1 and represents the ratio of the bag-of-word matches to the ground truth for the respective document text. In the column *Recall Difference CORR-RAW*, the document-specific recall differences show the improvement between the automatically corrected OCR text due to the post-OCR text correction and the plain OCR text.

The mean recall difference RAW-CORR value given in Table 5.2 amounts to 0.046 and represents the average percentage improvement of a RAW OCR text of the OCR pipeline. Textual improvements are most evident in documents that have a low recall value, while documents that are nearly identical to the ground truth show little or no improvement.

5.4 Summary

The Spelling Error Correction workflow provides potential correction suggestions to the OCR text produced by the OCR pipeline. Preprocessing measures which include, among others, the removal of punctuation marks and a lower-casing operation of the text are first carried out on the plain OCR text to prepare the text for subsequent error detection. After classifying the errors, the error correction procedure follows, which makes dictionary-based and context-based correction suggestions and recommends a ranking sequence of them. To test the impact of the applied correction suggestions, we conducted an evaluation of corrected OCR text and uncorrected OCR text with regard to a ground truth texts using a recall metric.

CHAPTER 6

Time Entity Recognition and Normalisation

Time expressions appear in almost every document, whether in the form of a full date consisting of day, month and year, a mere year or as a time of day. Extracting and annotating such temporal contexts in OCR texts enhances the searchability and information content of the raw text data and expands data applicability. Therefore, we focus on time as an entity and aim to recognise, normalise and classify it as such in the respective document.

Further named entity types such as people, places or organisations are also relevant in the archival research domain, but these have often been professionally catalogued and can therefore easily be matched and classified with the OCR text. For this purpose, dictionaries exist for each entity type, which are continuously expanded and maintained by the researchers.

6.1 Time Entity Extraction Procedure

The classification of time entities requires the application of a robust mechanism that is able to recognise, normalise and eventually parse time entities into a given data format. In the following we explain how our Time Entity Recognition and Normalization functionality is integrated into the OCR pipeline and applied to the extracted data followed by a performance evaluation.

1. **Time Entity Recognition** – The recognition step is concerned with detecting the temporal contexts in the OCR text. There are a variety of notations that time entities entail and, in addition, OCR spelling errors can complicate the process of recognition. The infinite spectrum of notations of dates and times requires an OCR

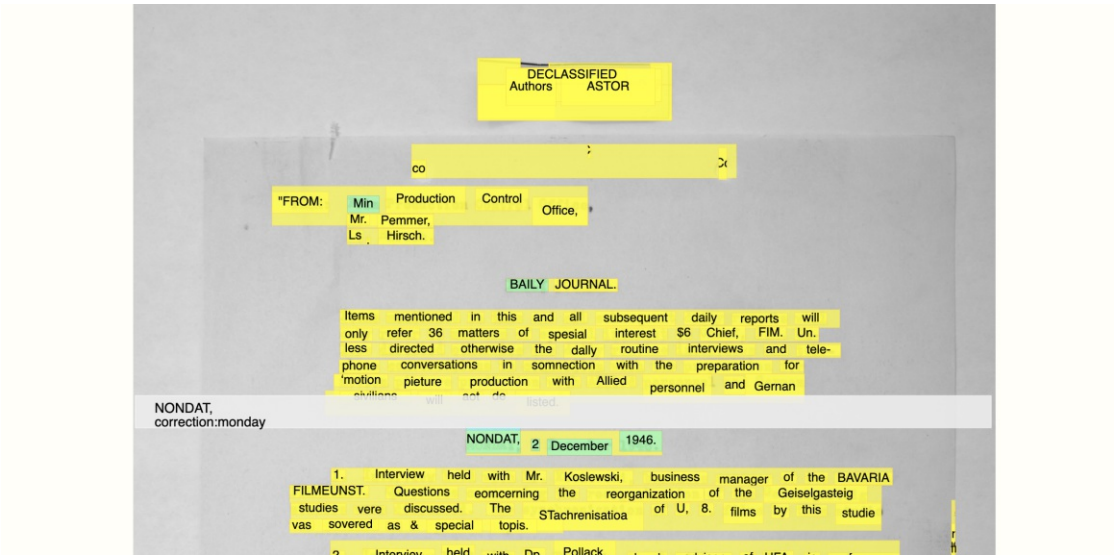


Figure 6.1: BOW Showcaser illustrating a recognised time entity due to applied spelling correction.

text that is as true to the original as possible, which is indeed a major objective of the OCR pipeline.

The combination of a text extracted as error-free as possible and a post OCR error correction pipeline is necessary to enable the recognition of the time entities. With regard to the correction of words where a whitespace character was mistakenly omitted during text extraction, there is the possibility that these words represent a temporal expression which can be recognised as a time entity after successful correction. Written out months or days of the week that are misspelled may also be corrected by the spelling error correction workflow and thus support time entity recognition.

2. **Normalisation** – In order to bring different time expression notations or incompletely recorded time entities into a uniform notation, they are normalised. Recognised dates consisting of year *YYYY*, month *MM*, day *DD* and times consisting of hours *hh*, minutes *mm* and seconds *ss* are parsed in the following order: *YYYY-MM-DDThh:mm:ss*. Time entities such as *12 November*, which are only partially complete without year information, are parsed as *XXXX-11-12* replacing the year context with a placeholder value. The time entity normalisation process ensures that recognised entities are typed according to the standardised TIMEX3 standard.

3. **Data Integration** – The integration of the found and normalised entities into a specific data structure or data format is another important step, for example to visualise the entities. In Figure 6.1, the data showcaser functionality of the BOW

demonstrates how OCR results as well as correction suggestions and extracted time entities are displayed on the underlying original document. In addition to the yellow standard OCR text boxes, annotated temporal contexts are highlighted in turquoise. In the figure, a hover interaction is performed on the time entity with the text *NONDAT,*, which results in the display of additional correction suggestion information above the hovered box as a grey window. The grey window shows that the suggested correction *monday* was provided by the spelling error correction workflow which resulted in a fully extracted time annotation in the course of the time entity recognition.

Used Software

The software library we use for the detection and normalisation of time entities is the Stanford Temporal Tagger [6], for which there is also a Python wrapper[1]. It is a deterministic rule-based system that can be configured with additional regular expression rules.

BOW correct detected	*RAW correct detected*	*Total Entities*
51	38	61

BOW Text Word Count	*RAW Text Word Count*	*Word Count Difference*
4795	3708	1173

BOW Detection Ratio	*RAW Detection Ratio*	*Difference Ratio*
0.7050	0.4889	0.2161

Table 6.1: Time Entity Detection Evaluation results for BOW and unprocessed RAW text output – showing number of detected entities in relation to the total entities, the respective detection ratios and OCR text word counts.

6.2 Time Entity Detection Evaluation

The evaluation of our Time Entity Detection procedure is primarily intended to demonstrate the effect of the BOW measures through improved recognition of the entities in the extracted text corpus. Therefore, the Named Entity Recognition performance of the *BOW* will be compared against the OCR results of the unprocessed documents, where only the document orientation was corrected, which we refer to as the *RAW* OCR text output in the following tables and figures.

[1]Stanford Temporal Tagger - SUTime: https://pypi.org/project/sutime/ (accessed 16.08.2021)

Dataset

For the evaluation, we created another task-specific dataset of 18 documents with 60 entities to be recognised. The entities to be recognised and the resulting total number of entities per document were annotated by ourselves as ground truth and validated for reliability by a historian. The type of time entities relevant for annotation refers to complete dates consisting of day month and year, fractions of complete dates, and excludes purely verbal references to the present or the future such as 'today' or 'tomorrow'. The completeness of an entity refers to the information-carrying text part of the recognised time entity. For instance, if the entity 'Tuesday, 7 March 1944' is only partially recognised as '7 March 1944', it is still recognised as complete because the day of the week can be inferred. If, on the other hand, the year aspect is not captured, the time entity is recognised as incomplete or incorrectly recognised.

Findings

The factor that accounts for the consistent performance of the BOW is reflected in the fact that archival documents are often only partially or not at all extractable without the application of image correction measures. By applying spelling correction, spelling errors in the extracted text are additionally corrected and can contribute to the improvement of the Named Entity Recognition result quality. In Section 6.1 we can observe how the post OCR spelling error correction was able to expand the information content of a misspelled entity by successfully correcting the day of the week *NONDAT,* to *monday.*

In the case of the document with the filename *Entry6B-Box20323* presented in Table 6.2, the individual time entity to be determined could not be recognised from the BOW output, but could be recognised from the RAW text. Examining the extracted text output of the BOW of the associated file, we find that the year to be extracted, *1954*, consists only of the last three numbers and has therefore not been matched as time entity. When applying document improvement measures, there is a risk that not all document areas are improved, as the focus of the individual processing steps of the OCR pipeline aims at a holistic improvement of the document for the respective task to be solved. The document with the filename *NARA-21.20.20190226* contains the most temporal contexts per document in the dataset, with a total of 22 entities to be identified, which are represented in a tabular form on the document.

The work of Rodriguez et. al. [23] demonstrates that manual correction of OCR text does not lead to a significant improvement in NER performance with regard to the named entity types person, location and organization. In our experiment we focus on the temporal entity type. Considering all documents in our dataset, the BOW recognises an average of *70%* of all time entities, while only *48%* are successfully recognised from the unprocessed RAW text. Using the word count difference between the BOW and the RAW outputs as shown in the corresponding column in Table 6.2, we determined three documents in the dataset that feature word count difference values over 150 which is related to incomplete text extraction on the part of the RAW text. The fact that the text

File Name	*Total Entities*	*BOW word-count*	*RAW word-count*	*word-count Difference*	*BOW detection ratio*	*RAW detection ratio*
8-6-2013-02-13-40-3-1	0	133	130	3	1*	1*
8-6-2013-10-24-41-0...	5	278	306	28	5/5=1.0	2/5=0.4
Entry6B-Box20323	1	136	138	2	0/1=0.0	1/1=1.0
Entry260-Box2632404	9	325	322	3	7/9=0.78	6/9=0.67
NARA-16.10.20191662	1	411	388	23	1/1=1.0	0/1=0.0
Entry260-Box2632454	2	382	388	6	2/2=1.0	2/2=1.0
NARA-RG-260-HMSMLR-...	1	188	134	54	0/1=0.0	0/1=0.0
Entry260-Box2632416	3	279	266	13	3/3=1.0	3/3=1.0
8-6-2013-02-13-40-1-1	3	425	422	3	3/3=1.0	3/3=1.0
NARA-RG-208-HMSMLR-...	2	141	143	2	2/2=1.0	2/2=1.0
NARA-21.20.20190226	22	228	202	26	20/22=0.91	16/22=0.73
8-6-2013-02-13-40-2-1	1	13	13	0	0/1=0.0	0/1=0.0
Entry260-Box2644678	2	476	4	472	2/2=1.0	0/2=0.0
Entry260-Box2644176	2	337	168	169	2/2=1.0	0/2=0.0
NARA-RG-260-HMSMLR-...	1	358	0	358	1/1=1.0	0/1=0.0
50-SKIRBALL-SkebaFu...	3	673	672	1	3/3=1.0	3/3=1.0
Entry260-Box2644657	2	6	1	5	0/2=0.0	0/2=0.0
8-6-2013-02-13-40-4-1	1	6	11	5	0/1=0.0	0/1=0.0

Table 6.2: Time Entity Detection evaluation dataset. * The value amounts to 1 because none of the entities to be recognized on the respective document was recognized.

has only been partially extracted means that the recognition of time entities can only be performed on the corresponding subset of the text, resulting in less or no recognition of time entities. If we omit these three partially extracted documents, the mean recognition rate would shift to *66%* for the BOW and to *55%* for the non-preprocessed RAW text. Measures for the complete extraction of documents by using our OCR pipeline as well as applied post-OCR spelling corrections support the time entity extraction process according to our assessment.

CHAPTER 7

Conclusion

In order to map the requirements of historians researching archival documents from World War 2 to technical solutions we extracted and analysed the requirements accordingly. By gaining knowledge and understanding of the workflows and procedures to be supported, as well as the environment of archival research, ten requirements were elaborated and specified in close cooperation with the historians. The resulting requirements of the historians represent the demand for a comprehensive solution to automate the digitisation of archived facsimile documents. With the aim of not replacing expertise but supporting the historians' workflows, the requirements were translated into technical solutions and implemented through an iterative development and evaluation process in our digitisation tool, which we refer to as the OCR pipeline. In the following, we summarise how we have implemented the requirements of the historians and to what extent our solutions have met the expectations of the historians.

- Archival Documents & Language Support – Awareness of document specific characteristics that historical archival documents exhibit including contaminations, human error and special document types, was essential for us to develop resilient document correction and enhancement methods. To achieve language independence within the OCR pipeline we approach language expandability by using Tesseract 4 as an OCR engine that supports different language and script types, and by using myspell language dictionaries to recognise and correct spelling errors.

- Input and Output Formats & Facsimile & Raw Image Format – The support for a variety of input formats is implemented in the first dedicated processing step in the OCR pipeline, in which different input formats are first converted into a uniform file format that is used for further processing. All requested output formats were implemented and include plain text output as an unstructured text file, the standardised and structured JSON output format, and the annotated

digital facsimile PDF format. The digital facsimile output, which represents a true replica of the original archival document, is created by mapping extracted text onto the original document so that no previously applied document transformations are visible. In a separate processing step, we implemented the requirement to convert raw type files, image files that are uncompressed and have not been processed in any way, into project-specific image output formats.

- Searchable Data – In addition to providing high-quality, machine-readable data from analogue archive data, we extend the applicability of the extracted data through additional information enrichment procedures. Using a task-specific dataset defined and transcribed by the historians, we compared the quality of the output texts of our solution with that of the commercial product ABBYY-finereader, showing that the average word error rate of the OCR pipeline is 40% lower than that of ABBYY-finereader compared to the ground truth transcription.

  We developed a post OCR spelling error correction workflow that detects potential spelling errors based on the extracted text and provides correction suggestions to support the creation of an absolute transcription accuracy through human post-correction. In an evaluation, we demonstrate to what extent our spelling error correction workflow achieves improved transcription accuracy in an auto correct scenario, where correction suggestions are applied to the extracted text.

  Based on the text with the applied error correction suggestions, temporal contexts are subsequently extracted and normalised, which enables a temporal classification of documents or the establishment of temporal relationships between documents. The time entity detection method was evaluated on a task-specific dataset with a total of 60 annotated entities. The performance of the detected time entities was 70% for the text extracted by our OCR pipeline, and 40% for the extracted text with no pre-processing measures being applied to the documents.

- Graphical User Interface – We developed a web-based interface prototype, the Bulk OCR Webservice, to enable easy interaction with the OCR pipeline and the input and output data. The graphical user interface allows OCR results, spelling error correction workflow data and extracted time entities to be displayed and for historians to interact with them. BOW continues and meets the minimal interaction requirement of the OCR pipeline so that no technical expertise is required for use either.

- Transparent Extraction History – The traceability and documentation of applied document transformations that result in the extracted text is fulfilled by collecting metadata along the OCR pipeline processing steps. Stored metadata is available for each document processed by the OCR pipeline and includes information on document transformations, such as document height and width scaling values or a corrected rotation angle.

- Minimal Interaction – We implemented a minimal interaction design that enables a target group with little or no technical knowledge to operate the digitisation

tool. Furthermore archival documents must not require human pre-processing or correction in order to be processable by the OCR pipeline. To realise the minimal interaction design, automatic decision-making processes are necessary within the OCR pipeline processing steps in order to be able to carry out document corrections and document transformations without human supervision.

- Large Scale Processing – The requirement to process large amounts of data continuously to the minimal interaction requirement also demands a high degree of automation throughout the digitisation process. Scalability and portability capacities of the application are demanded, which are realised by means of a container architecture of the OCR pipeline. To enable automation along the OCR pipeline, we have successfully applied unsupervised evaluation metrics as decision making tools to perform document transformations and corrections. The proposed metrics are calculated document-specifically from text extraction process metadata and are able to support the determination of the optimal document resolution, decide on the correct document orientation and support the determination of the best threshold configuration. Furthermore, before determining the best threshold configuration, the metrics are used to define the parameter values to be evaluated. With regard to runtime performance optimization as part of the large scale data processing requirement, the targeted selection of the parameter values to be evaluated enables a reduced duration of the evaluation process and thus of the total runtime for each document to be digitised.

7.1 Future Work

More interesting adaptations, methods and experiments have emerged during the work on this thesis but are beyond the scope of this dissertation. In the following, we summarize our future work concerns for deeper investigation for particular mechanisms and further ideas categorised as future scientific work and future implementation work.

Future Scientific Work

- *Improve output quality* – We discovered potential for improvement of the digitised output in the area of OCR text quality and in post OCR error correction. With regard to the OCR process, in our document thresholding parameter search procedure several text extractions are performed and only the best evaluated text is considered for the final result. Lund et al. [17] presented an ensemble method which creates a composite text by applying several global thresholding methods and text extractions. Following up on this idea we consider multiple output texts and corresponding confidence values resulting from different adaptive thresholding configurations to use them not only for aggregating a best text result but also to support and reinforce the post OCR error correction workflow and named entity recognition process. In terms of correcting spelling errors, additional texts provide

another source of suggested corrections and can also be used to extend the word occurrence dictionary to prevent named entities from being accidentally corrected.

- *Improve runtime performance* – As the project progressed, the historians increasingly made additional demands on the output text quality and the integration of further features which often extended the overall runtime of the application. To counteract the increased runtime of the application, we have already implemented and documented runtime optimisation measures in this work. Further potentials for improving runtime are as follows:
  - In our approach, most text extraction processes take place in the thresholding step and therefore require the most time and computing resources. Applying a simple global thresholding method in combination with a previously applied contrast correction such as CLAHE (Contrast Limited Adaptive Histogram Equalization) instead of a parameter rich and more complex adaptive thresholding method may also have potential in runtime improvement, while maintaining output quality, but was not tested due to time constraints.
  - Determining the average text-line height based on run-length compressed documents as presented by Javed et al. [12] would result in several iterations of text extractions being omitted in the search process for the optimal document resolution, since the text-line height decision parameter is known in advance and can be utilized.
  - Furthermore, metrics that express information about the document exposure could determine the choice of the thresholding method, so that in the case of a uniformly distributed exposure, a much faster algorithm can be applied. For instance, Boudraa et al. [4] uses Michelson's contrast formula to decide on the thresholding algorithm based on the local contrast value of a given document.

Future Implementation Work

- *Increase automation aspect* – By eliminating the only user-definable parameter of our OCR pipeline, namely the language, absolute automation can be achieved. These measures, in turn, require language recognition mechanisms which, related to the OCR pipeline approach, must be applied individually prior to any document transformation or correction in order to perform further preprocessing, OCR and NLP tasks with the best transcription performance available using the appropriate models. An incorrect classification of the languages present in the document means that inappropriate language models would be used subsequently for text extraction and post spelling error correction. Particularly documents which, due to severe degradations, achieve a partially complete and error-free text during OCR recognition, the initial language classification may be difficult. Barlas et al. [3] present a multi-layered approach for language and script type recognition for printed and handwritten documents, where language recognition is based on statistical analysis of bi-grams of OCR text.

- *Expand data applicability* – In addition to the method presented in this paper for recognising and normalising time entities in OCR texts, other entity types from the named entity recognition task, such as places, personalities or organisations, can also be extracted in an automated manner. Identified entities can subsequently serve as supporting components for further downstream NLP tasks such as relationship extraction (RE) or event extraction to continue to build structured and contextual information. Ma et al. [18] present a novel method for extracting named entities and relations from unstructured text by improving existing table-filling approaches to identify entities and relations and were able to outperform state-of-the-art methods on two datasets. A comprehensive introduction to the various techniques for extracting named entities and relations is additionally presented in [18].

- *Implementation of a comprehensive user interface* – The requirement to implement a comprehensive and feature rich user interface to interact with the OCR pipeline and its generated output is beyond the scope of this work and the presented webservice prototype serves to illustrate the digitisation output potentials as a test environment. The additional implementation requires the design and implementation of a manual post correction opportunity that incorporates the suggested corrections provided by the OCR pipeline. Furthermore, a search mechanism that can incorporate the temporally normalised and recognised contexts, among other things, is desired and remains for future work.

APPENDIX A

Bulk OCR Webservice (BOW)

We implemented the Bulk OCR Webservice (BOW) as a prototype test environment graphical user interface to enable interaction with the OCR pipeline and its output data for the historian target group. We implemented the GUI using the open source python web framework django[1] and the[2] JavaScript library which is responsible for data-based document transformations within the service. The web-based interface comprises two usecases which are discussed and illustrated in the following.

A.1 OCR pipeline interaction

The handling of the OCR pipeline refers to the providing of the input data as well as the receiving of the digitised output data. The OCR pipeline interaction and at the same time landing page is divided into three sections or use cases, illustrated in Figure A.1, and includes an upload drag and drop area, an area that provides information about the current OCR pipeline status and a download section to obtain results.

1. **Document Upload**

   We have designed a workflow for the document upload which gradually demands information from the users without overwhelming them with many parameters to be specified at once. First, the user loads files from his local computer into the BOW either by using the drag and drop functionality or via file-select field. After defining the set of files to be uploaded, the user can review the files, reset all or start the file-upload procedure. Next, the language of the documents as the only parameter to be defined by the user must be selected as shown in Figure A.2 via drop-down menu and subsequently the user can lanuch the digitisation process.

[1] django: https://www.djangoproject.com/ (accessed 16.08.2021)
[2] D3.js: https://d3js.org/ (accessed 16.08.2021)

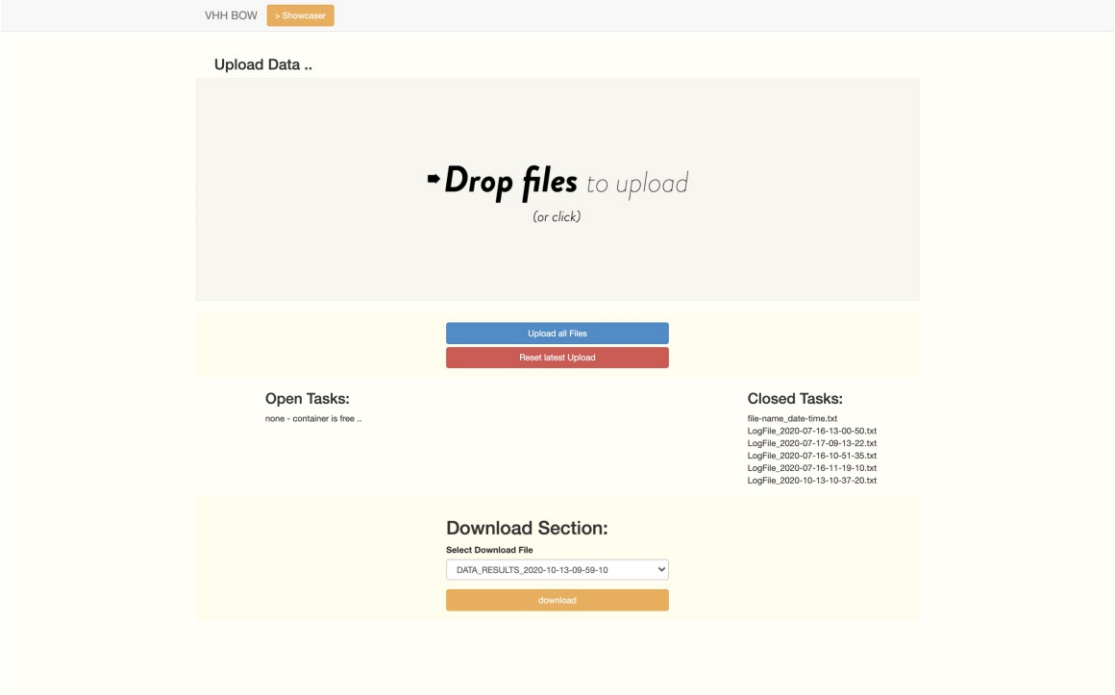


Figure A.1: BOW data upload page including the upload, container state and download section.

Within the upload section of BOW we integrated *dropzone.js*[3], an open source drag and drop javascript libary which offers drag and drop functionality, configuration options, file preview functionality and upload state feedback. The following configurations were made to the drag and drop window.

- *autoProcessQueue: false* - Set to false allows us to enable the user to review the uploaded files. The upload is triggered on a button click event.
- *addRemoveLinks: true* - Append a remove option to each document thumbnail preview.
- *parallelUploads: int* - Speed up the upload process by defining this parameter machine and computing power specific (default value is 2)
- *acceptedFiles: ".jpeg,.jpg,.dng,.arw,.pdf,.png"* - Limit the upload exclusively to file types which can subsequently be processed by the OCR pipeline to avoid errors.
- *on addedfile* - Implement dynamic action that compares each newly added file to avoid duplicate uploads.

[3]dropzonejs: https://www.dropzonejs.com/ (accessed 16.08.2021)

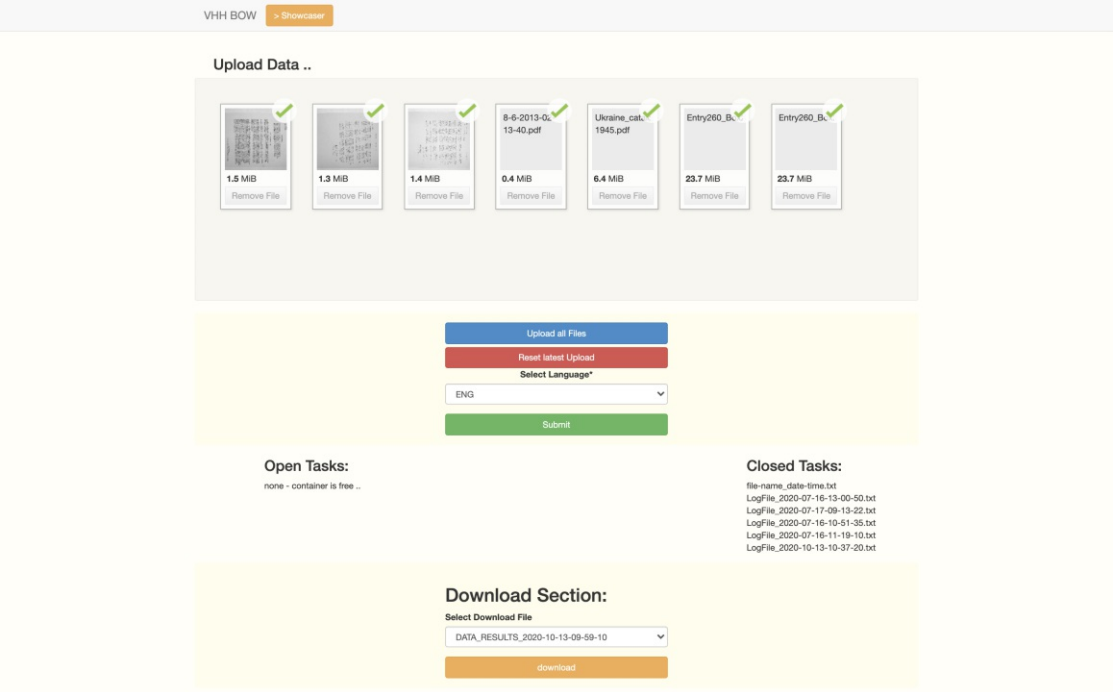


Figure A.2: BOW data upload page demonstrating successfully uploaded documents, showing the language configuration drop-down menu and the container launch submit button.

- *on complete* - After successful completion of the upload process, container settings and the container launch button are being displayed.
- *on reset* - Resets the latest uploaded files and removes all files from the server.

To provide additional parameters such as the user defined language to the OCR pipeline which is wrapped as a docker container, a configuration file is created and passed to the container. The configuration is defined as a dictionary object and passed to the subsequent OCR pipeline task whenever the user launches the digitisation process.

2. **Container State & Logging**

To enable the BOW application to track the status of the OCR pipeline docker container the application console output of the container is written to logfiles. These logfiles, which we refer to as tasks, are categorized into closed tasks and open tasks. Based on whether an OCR pipeline run has been completed or not, the logfiles either end up in the open tasks folder or, if completed successfully, are moved to the closed tasks folder. Both open and closed tasks are displayed in the

middle area as shown in Figure A.1 and provide the user with information about the availability of the container.

One logfile corresponds to one OCR pipeline run and contains the complete application console output, including errors and exceptions. As soon as a container run is finished the last line of the latest logfile is used to determine the current state of the OCR pipeline. The status is set to available whenever there is no open task logfile and set to busy whenever an open task exists. The way an OCR pipeline run was completed, with or without errors, is not relevant for the web server, it only matters whether the container is available or whether it is currently occupied by an open task. As soon as the last line of the latest in progress run output log file corresponds to *task finished* the container state is interpreted as available and therefore the container is able to process the next task. The corresponding logfile is archived, moved to the closed tasks and the container status is set to available

3. **Result Retrieval**

   In the download section at the bottom of the OCR pipeline interaction page, results can be downloaded as compressed zip files after selecting them from a list in a drop-down menu. Each zip file corresponds to an OCR pipeline run and contains the complete extraction history as well as potential errors, the data in the formats specified and exported by the OCR pipeline.

A.2 Data Showcaser

The data showcaser aims to demonstrate text extraction results, document transformations and further annotation and enhancement of the raw text data generated by the OCR pipeline. The user has the opportunity to interactively research and investigate the data results leveraging extraction confidence values, recognised time entities or suggested corrections for potential spelling errors. Alternatively, it can also be used as a file browser and preview tool.

1. **Data Selection**

   To present different processed documents and corresponding data results in the showcaser we treat each OCR pipeline task separately and we refer to them as independent datasets. A dropdown button enables the selection of a dataset which is identified by an unique timestamp. The results relevant for the showcaser include the hOCR data for representing the layout formatted OCR text, the original as well as the binarized image documents and an index file containing meta information about all documents in the dataset.

   After selecting the dataset the user has the option to investigate a document from the grid preview selection view. For the grid view shown in Figure A.2 we use the data indexing file to list all image documents and corresponding result data

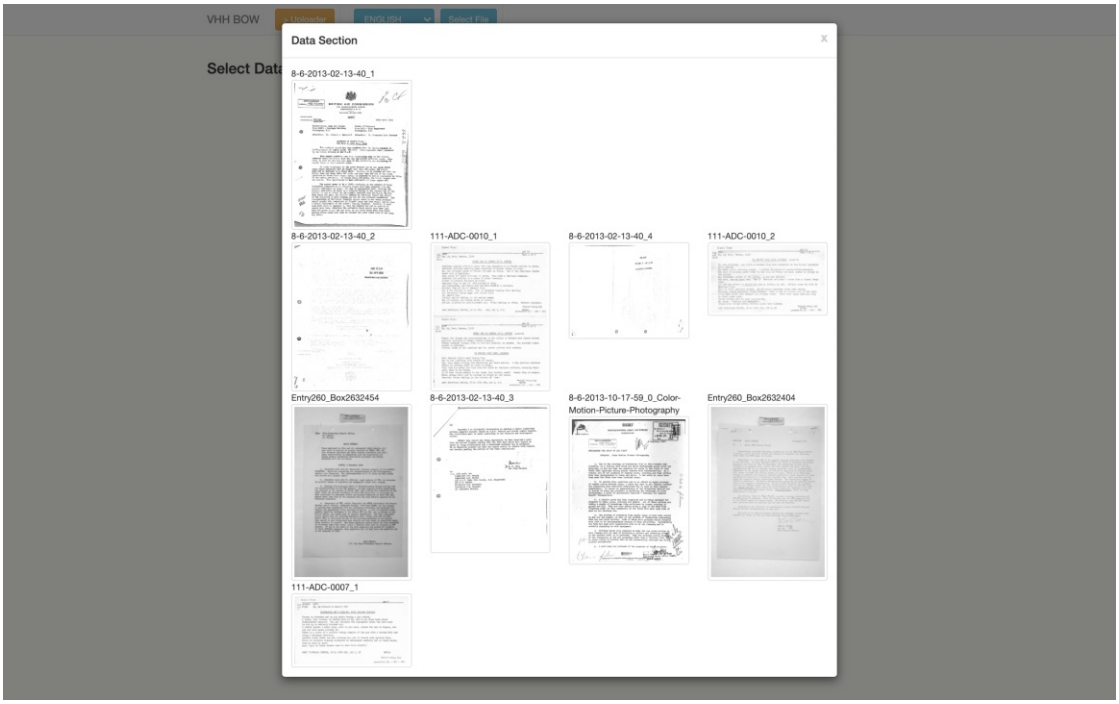


Figure A.3: BOW showcaser illustrating the thumbnail preview document selection.

within the dataset and show a image preview thumbnail. After document selection, the user can always select another document or change the dataset to investigate another subset of documents.

2. **Representation of text recognition results**

   Based on the underlying document image, the associated extracted text is displayed on the region on which it was detected utilizing available coordinate information as shown in Figure A.4. Available configuration options to filter and modify the recognized text are defined in the header section of the detailed showcaser document view. The header remains at the top of the page when navigating through a document and is always accessible. The following options are available.

   - *Show/Hide Boxes* – Enables to show and hide the text boxes in which the respective recognized words or characters are displayed. When hovering the text boxes, the content and, if available, language specific correction suggestions are displayed. In addition to the normal yellow text boxes, showing OCR text and potential correction candidates, there are blue boxes which indicate that the content contains has been identified as a time context. This time context is annotated as a separate column in the corresponding result files and allows us to annotate the affected text boxes accordingly.

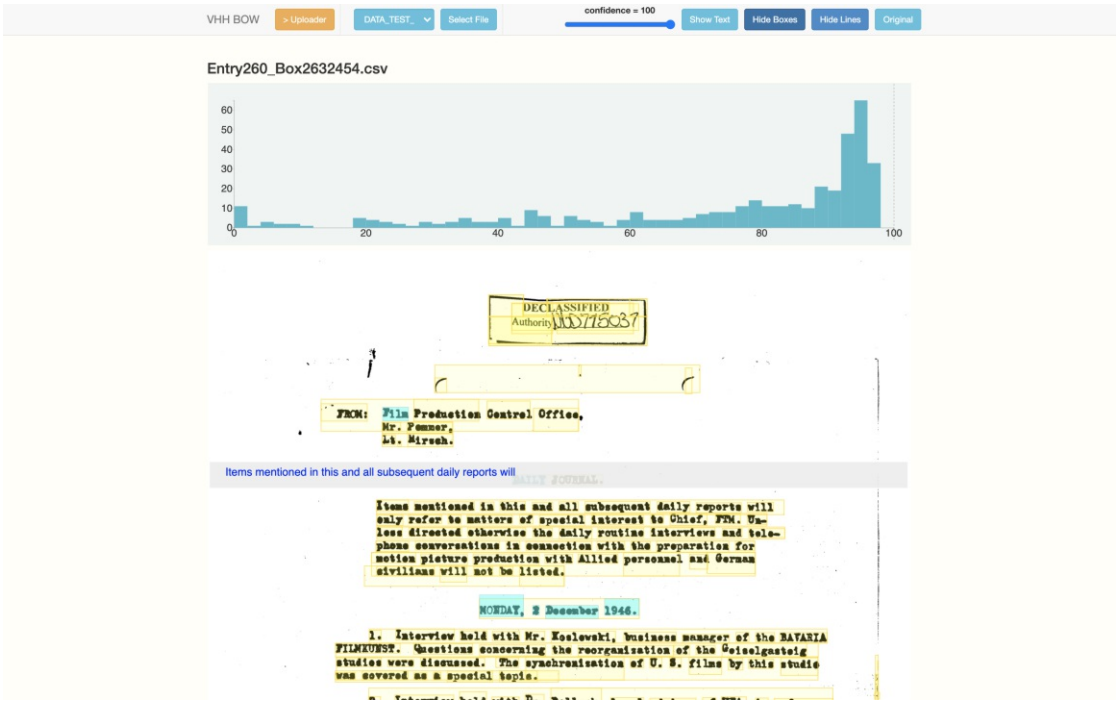


Figure A.4: BOW showcaser document view illustrating a user line hover action to investigate merged line contents.

- *Show/Hide Text* – Enables the OCR text display option to display the underlying background image or to show the text. To get a better differentiation of the OCR results and the background in color rich images it is recommended to hide the text and use the text box hover option to display the desired OCR text content.
- *Show/hide Lines* – Enables the option to show and hide the text lines that span horizontally containing multiple text boxes. The hover interaction over line boxes displays the content of all summarized boxes which are located on the respective line. Each box has a value which defines the association to a line. The merged content created by hovering the line is visible in Figure A.4 as a gray field which is created above the cursor position.
- *Confidence* – Each text box and each line has an associated confidence value. This allows us to provide this numeric value as a filter option. The confidence value takes values between 0 and 100 and indicates the extent to what the OCR engine considers the response text correct, whereas 100 corresponds to correct and 0 incorrect. Because line boxes always have a confidence value of -1 as a special value, they are excluded from the filter and can only be turned off by the show/hide lines feature. This special confidence value allows us to make line boxes filterable as such or to make them visible or hidden as

described above.

- *Binarize/Original* – This option allows to switch the background image between the original unprocessed document and the document affected by the thresholding transformation namely the binarized document and vice versa.
- *Confidence Histogram* – All confidence values of a selected document are displayed above the image in a histogram graph to present an indication of the distribution of the value. The further the distribution is shifted to the right towards the maximum value, the better the estimation of the total text recognized. To configure a confidence threshold filter we implemented a range slider also located in the header section. When the slider is moved to the far right, all text boxes are displayed including their text contents. The further the slider is moved to the left, the lower the confidence text values will be displayed exclusively and values higher than the set threshold will be hidden. The current filter threshold value is displayed above the slider and also visually as a dividing line in the histogram.

APPENDIX B

Implementation Details

This chapter is intended to provide additional insight into the implementation details of the OCR pipeline. In Section B.1 the container technology and architecture components in use are discussed. Subsequently in Section B.2 configurability options of the OCR pipeline are explained. Details about performance measures, considerations and optimizations can be found in Section B.3.

B.1 Container Technology

To make the OCR pipeline as software solution portable, accessible and configurable we use the container technology provided by the docker platform[1]. While the docker platform enables effortless creation and deployment of containers, the containers themselves provide standalone images that include the application, all dependencies, the runtime and the system specific settings such as environment variables. With the OCR pipeline application packaged as a container, the processes run in isolation using the computing resources allocated by the underlying IT infrastructure. With minimal effort, the OCR pipeline can be migrated and deployed to new environments regardless of the underlying complexity of the application components and dependencies.

In our Docker configuration we use *python:3.7* as base image which already has the correct python version and the package manager pre-installed. The main components and dependencies include the OCR engine tesseract 4 including dependencies[2], the installation of various language dictionaries for the spelling correction workflow and the OCR language models.

[1] Docker Container `https://www.docker.com/` (accessed 16.08.2021)

[2] Google Tesseract OCR `https://opensource.google/projects/tesseract` (accessed 16.08.2021)

B.2 Configuration

In order to extend the adaptability of the OCR pipeline to different infrastructure environments and fields of application, we have implemented a configuration possibility. The way the configurations are passed to the OCR pipeline is through a python dictionary object which can either be passed as a string or optionally in a text file. For the management and documentation of configurable parameters within the OCR pipeline we use the everett[3] library. As soon as the container is invoked, the OCR pipeline is initialised with the additional configuration passed as input parameter.

Among the configurable parameters are adaptation options for refining the OCR pipeline to the input data, such as the blacklist of characters to be ignored within the OCR process or the language models to be used. The most common user-adapted parameter is the language configuration, which is responsible for the OCR models to be used and, in the post OCR spelling error correction workflow, for the selection of the correct language dictionaries. Parameters relevant for the domain specific environment and IT infrastructure settings include the number of *CPU* cores to be used for multiprocessing, the paths where input and output data should be stored and the paths of the tesseract models to be used.

B.3 Performance

Supplementary to the historians' performance requirement which belongs to the large scale data processing requirement 2.8, this subsection explains what measures were investigated and subsequently applied to the OCR pipeline for performance regarding runtime optimization. In the following aspects we were able to influence the runtime performance while maintaining OCR output quality.

- **Multiprocessing** - By default, Tesseract works with multiprocessing and utilizes up to four cores to in the OCR process which is the most performant setting with only one input document. In order to be capable of processing several documents in parallel, one computing core should be allocated for each text extraction. Thus, with four available cores, four documents can be extracted side by side. We disable the multiprocessing feature of tesseract and set the number of threads to be utilized to one. This is done by configuring the system environment variable *OMP_THREAD_LIMIT*. Deactivating the multiprocessing feature allows us to run several instances of tesseract in parallel to process one OCR process per processor core simultaneously. If multiprocessing is not disabled and multiple instances of tesseract are started and used simultaneously, the processes will block each other, causing the process to be slower than with traditional serial processing.

[3] everett: https://pypi.org/project/everett/ (accessed 16.08.2021)

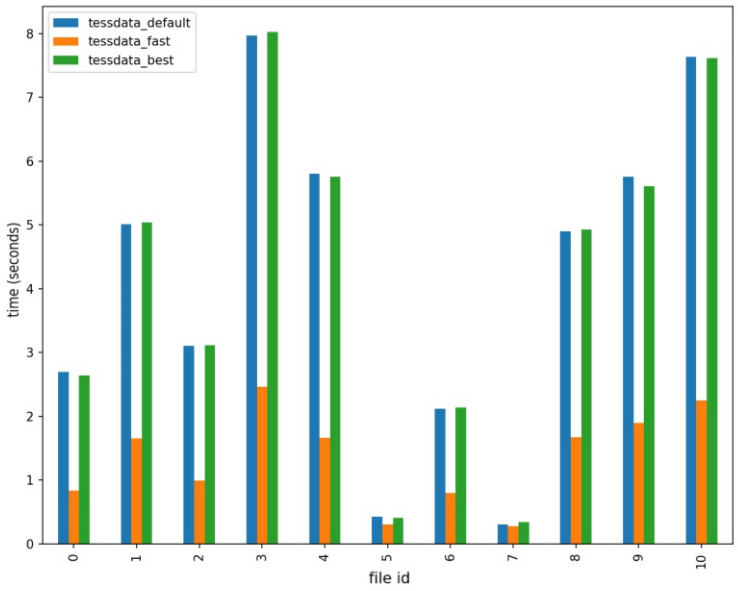


Figure B.1: Tesseract 4 tessdata model runtime performance measure for 11 documents.

- **Tesseract Traindata** - The tesseract traindata (*tessdata*)[4] models contain the language specific machine learning trained networks for text recognition and work exclusively with the *LSTM OCR engine* of Tesseract 4. Tesseract 4 offers one fast (*tessdata-fast*) and one best (*tessdata-best*) model option for each available language. These differ in terms of file size of the model type, the accuracy of the OCR results and internal complexity of the model resulting in a different processing time required for the text extraction procedure. In Figure B.1, ten grouped bar stacks, each representing a document and its respective three OCR extraction runtimes resulting from the application of three different model types. As can be observed the tessdata-best model takes significantly more time than tessdata-fast, consuming up to more than twice as much processing time. It can also be seen that the tessdata-default models used by default are equivalent to tessdata-best in terms of quality and runtime performance. On average, tesseract fast models can reduce the extraction time by up to 55%.

  Without compromising between quality and runtime, the objective becomes to increase runtime performance without sacrificing the quality of the results whenever possible along the OCR pipeline steps. In order to determine the optimal traindata applications along the OCR pipeline and to measure the effect of the selected

[4] Tesseract Traindata https://github.com/tesseract-ocr/tessdata (accessed 16.08.2021)

models on the overall runtime, we have tested the following variants for the use of traindata combinations.

1. **fast+fast**: utilize fast models for preprocessing tasks and fast models for final data extraction and export
2. **fast+best**: utilize fast models for preprocessing tasks and best models for final data extraction and export
3. **fast+best(+bin)**: utilize fast models for preprocessing tasks and best models for binarization stage and final data extraction and export
4. **best+best**: utilize best models for preprocessing tasks and best models for final data extraction and export
5. **best+fast**: utilize best models for preprocessing tasks and fast models for final data extraction and export, resulting in a poor runtime performance and text quality

After evaluating the OCR text output quality with regard to the associated runtime of each configuration, we decided to implement variant *fast+best* as it provided constant text quality with a total runtime acceptable to the user target group.

- **Document Resize** - Referring to the processing step 5, the document resolution is a crucial factor when it comes to the OCR quality but also to the runtime of the extraction process. The resolution of a document determines through the height and width parameters of the document the size of the file and thus the amount of data to be processed by the OCR engine. Especially high resolution documents contain more data, which results in more computing effort for text recognition and binarization and thus increases the runtime. In addition to the primary objective of the resizing step, which is to improve text recognition, the runtime of any further computation on the document is also reduced.

- **Relation finding** - The discovery of relations between metadata of documents and functional parameters to be used within the OCR pipeline allows us to set these parameters precisely to save processing time, especially in an iterative process. For example, with regard to the document-specific parameters to be determined in the binarization step 6, a search and evaluation procedure determines the best configuration from a series of parameter values.

  However, if we have a prior knowledge of the relationships to the parameters we are looking for, we can define parameter rows more efficiently by using narrower ranges of values, and therefore save time by avoiding additional iterations. The key document specific derived metric is the mean line height which is determined in the resize stage. In the subsequent binarization step, the man line height is accessed to generate a document-specific set of values for the block size binarization parameter. Relation finding processes took place during the development of the OCR pipeline to meet performance requirements for both text quality and runtime.

- **Avoiding Overhead** - Since our OCR pipeline is built in python we utilized several python wrappers for the tesseract engine written in C++. Besides the overhead caused by the python wrapper, there is also overhead caused by loading the tess-data language models and overhead caused by the handling of multiprocessing along the OCR pipeline.

  Why use a wrapper for C++ tesseract at all? - A python wrapper acts as a binding for the tesseract OCR engine, which is originally written in C++, for python and therefore adds a certain overhead due to programming language translation. The wrapper, in contrast, allows us to integrate methods and functions from NLP and computer vision in a unified language along the entire OCR pipeline. The tesseract wrapper also enables reliable error handling and object-based handling of the output data.

  Choosing the right library for the task - Depending on the task in the respective processing step in the OCR pipeline, two different wrappers are employed for the tesseract OCR engine.

  With respect to iterative search tasks like those found in the binarization or resizing step in the OCR pipeline, the re-instantiation of the language models would create a considerable amount of overhead that must be avoided. For this scenario, *tesserocr*[5] is used, also a python wrapper that integrates directly with Tesseract's C++ API via *cython*. By instantiating and loading the tess-data models into memory once per CPU, we are enabled to process multiple documents in a time efficient manner. The tess-data loading overhead amounts to approximately 0.4 seconds per instantiation.

  For tasks where single images are extracted as for instance in the final extraction process the *pytesseract*[6] wrapper is considered. Compared to the tesserocr wrapper the data extraction functionality is more extensive in terms of output detail information and the runtime performance difference is marginal since the final extraction process is not an iterative process.

  Overhead Measurement - The tess-data loading overhead was determined as follows. First we created a 10 by 10 pixel plain white background image without textual content to decouple the language model loading time from the OCR extraction time. Using the generated image we measured the time needed for the two wrapper libraries, pytesseract and tesserocr, to complete ten OCR text extraction processes each. Pytesseract takes on average 0.4 seconds and tesserocr 0.003 seconds which corresponds to the plain time needed for the text-extraction process of an empty 10 by 10 image.

[5] tessract wrapper - tesserocr https://github.com/sirfz/tesserocr (accessed 16.08.2021)
[6] tessract wrapper - pytesseract https://github.com/madmaze/pytesseract (accessed 16.08.2021)

APPENDIX C

Evaluation Documents from the OCR Quality Benchmarking Dataset

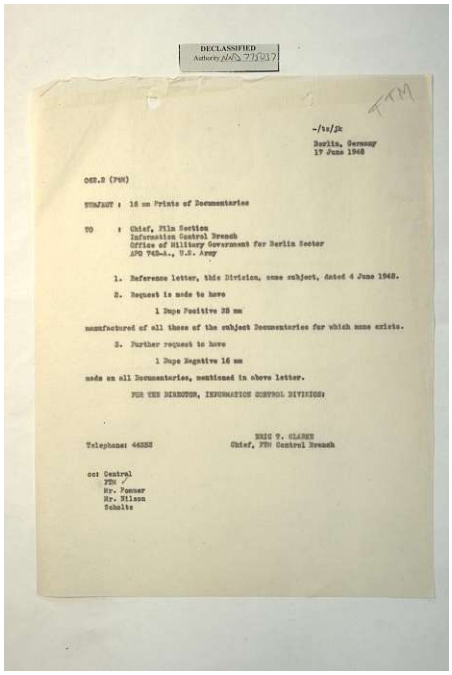
The digital images referenced in Table C.1 as electronic records were retrieved from the "National Archives and Records Administration (NARA)", College Park, MD (USA) to conduct the OCR quality benchmark evaluation from Section 4.5 as well as the spelling error correction evaluation from Section 5.3.

The Table C.1 is intended for referencing and resolving the individual name-space components of the electronic records used in the evaluation. Therefore, the column *File ID* in Table C.1 matches the identification values of the column *File ID* in Table 4.2. The nine electronic records and their *File ID* labels in Figure C.1 also correspond to the *File ID* columns of the two Tables C.1 and 4.2.

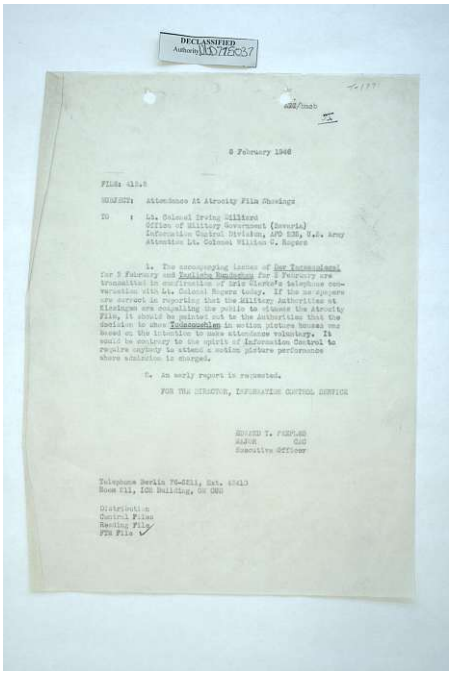
The NAID (National Archives Identifier) found in table C.1 allows for accurate identification of the documents in the National Archives Catalog online search `https://catalog.archives.gov` (accessed 16.08.2021)

File ID	*Archive abbreviation*	*Record Group Number*	*Series Entry ID*	*Box number within series*	*Folder number and name*	*NAID*	*Image number*
a	NARA	RG-260	HMSMLR-A1-260	Box-265	Documentary-Production-Unit	7550505	029
b	NARA	RG-260	HMSMLR-A1-260	Box-262	Atrocity-Film	7550483	18
c	NARA	RG-208	HMSMLR-NC148-6G	Box-03	German-Operation	4726574	30
d	NARA	RG-260	HMSMLR-A1-260	Box-264	Film-meetings-quadripartite	7550504	046
e	NARA	RG-260	HMSMLR-A1-260	Box-265	Documentary-Production-Unit	7550505	019
f	NARA	RG-208	HMSMLR-NC148-355	Box-1711	Atrocities-German-General	4734014	02
g	NARA	RG-208	HMSMLR-NC148-6G	Box-02	Information-activities-in-Germany-Oct-1944-May-1945	4726567	058
h	NARA	RG-260	HMSMLR-A1-260	Box-263	Daily-Journal	7550484	62
i	NARA	RG-208	HMSMLR-NC148-404-NAID-722024	Box-803	German-Committee-Minutes-Oct-1944-May-1945		218

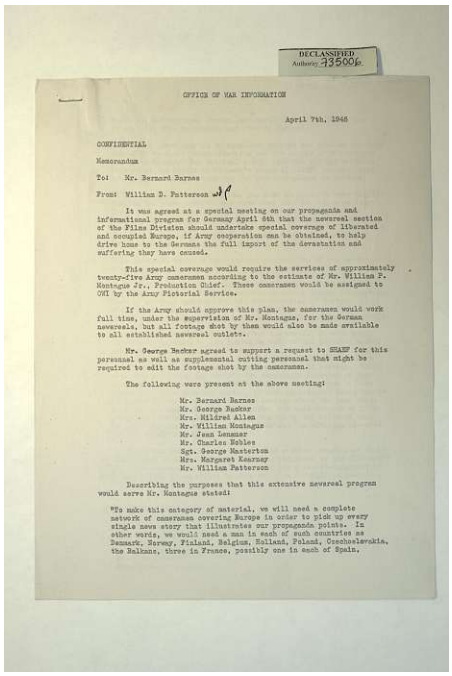
Table C.1: Document reference table for referencing and resolving individual name-space components of the electronic records used in the OCR quality benchmark evaluation.



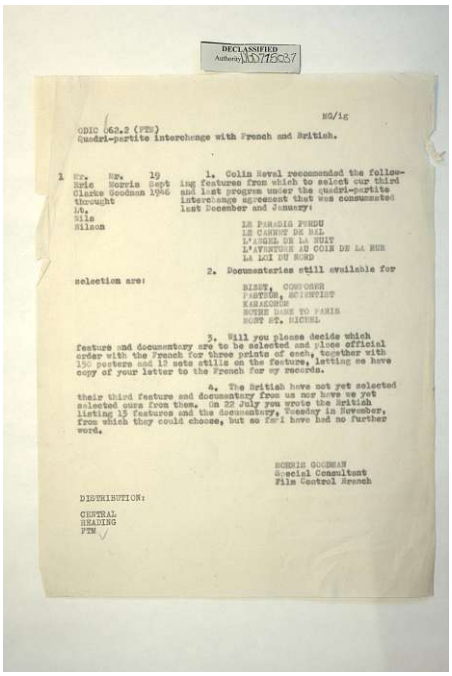
(a)



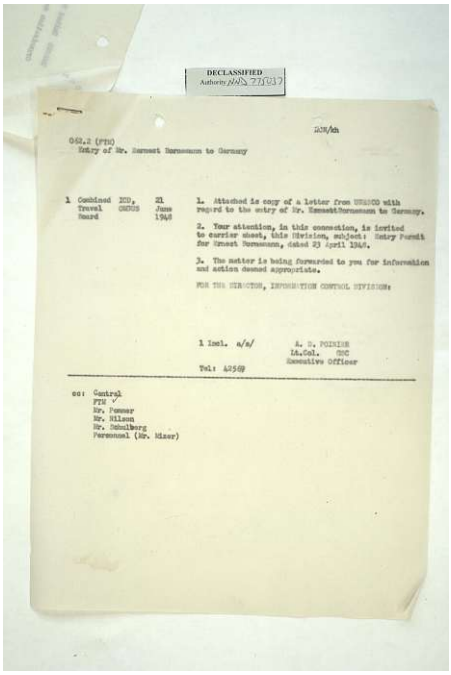
(b)



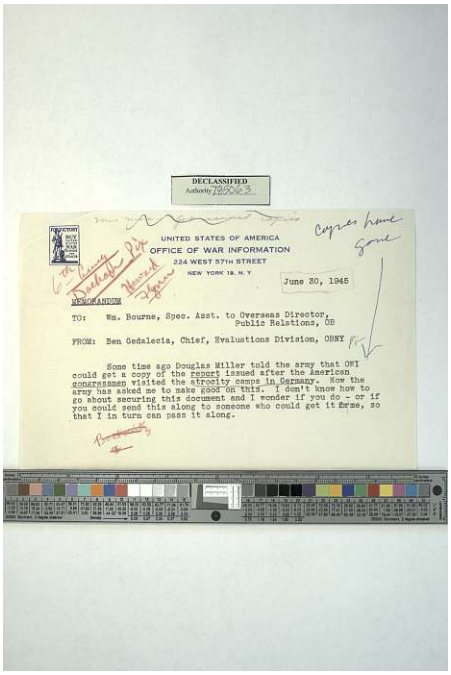
(c)



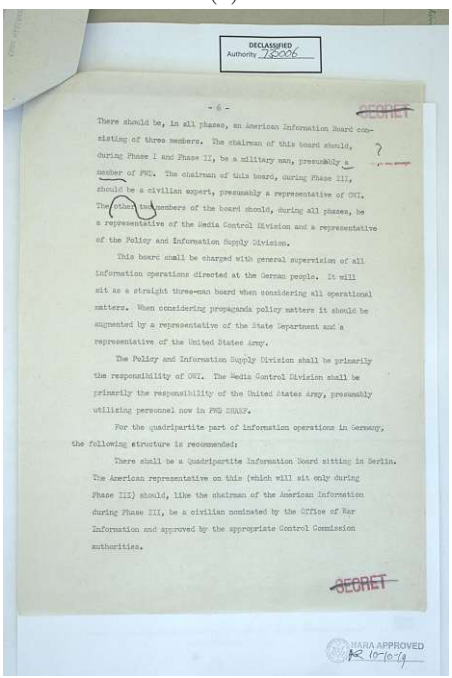
(d)



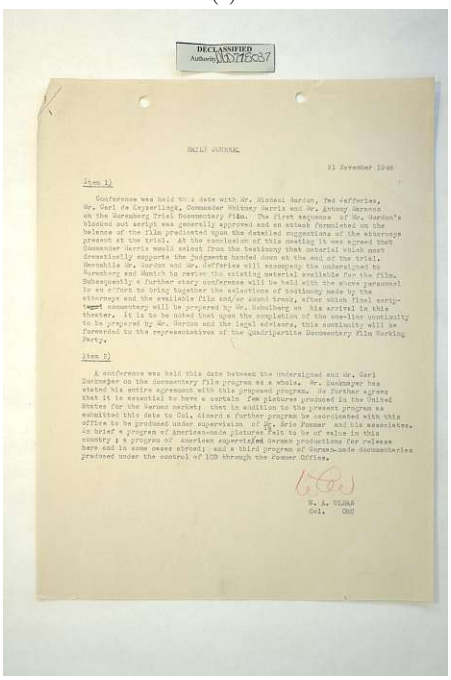
(e)



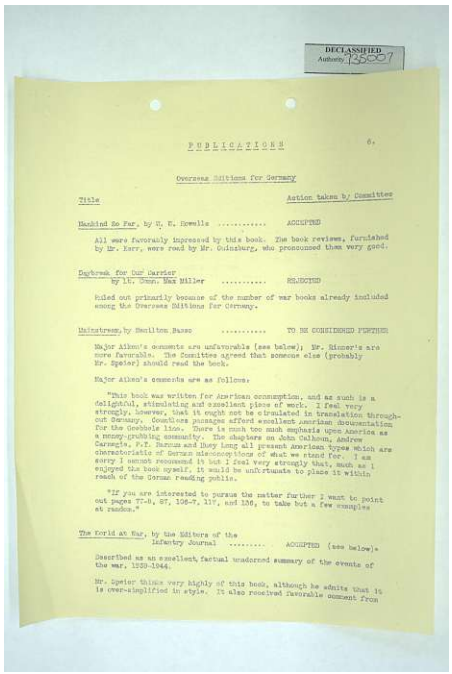
(f)



(g)



(h)



(i)

Figure C.1: Document preview of the nine electronic records retrieved from the National Archives and Records Administration (NARA) to conduct the OCR quality benchmark evaluation. Photos: VHH Project, Ulrike Koppermann, 2019.

List of Figures

List of Tables

Bibliography

[1] Visual history of the holocaust. https://www.vhh-project.eu/. Accessed: 16.08.2021.

[2] David Ahn, Sisay Fissaha Adafre, and Maarten de Rijke. Towards task-based temporal extraction and recognition. In Graham Katz, James Pustejovsky, and Frank Schilder, editors, *Annotating, Extracting and Reasoning about Time and Events*, number 05151 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. URL http://drops.dagstuhl.de/opus/volltexte/2005/315.

[3] P Barlas, D Hebert, Clément Chatelain, Sébastien Adam, and Thierry Paquet. Language Identification in Document Images. *Journal of Imaging Science and Technology*, 60(1):010407, 2016. doi: 10.2352/J.ImagingSci.Technol.2016.60.1.010407. URL https://hal.archives-ouvertes.fr/hal-01282930.

[4] Omar Boudraa, Walid-Khaled Hidouci, and Dominique Michelucci. Degraded historical documents images binarization using a combination of enhanced techniques. *CoRR*, abs/1901.09425, 2019. URL http://arxiv.org/abs/1901.09425.

[5] Thomas Breuel. Robust least square baseline finding using a branch and bound algorithm. *Proc SPIE*, 4670, 12 2001. doi: 10.1117/12.450735.

[6] Angel X. Chang and Christopher D. Manning. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, 2012.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[8] Reza Farrahi Moghaddam, Fereydoun Moghaddam, and Mohamed Cheriet. Unsupervised ensemble of experts (eoe) framework for automatic binarization of document images. pages 703–707, 08 2013. doi: 10.1109/ICDAR.2013.144.

[9] Mika Hämäläinen and Simon Hengchen. From the paft to the fiiture: a fully automatic NMT and word embeddings method for OCR post-correction. *CoRR*, abs/1910.05535, 2019. URL http://arxiv.org/abs/1910.05535.

[10] Abdeslam El Harraj and Naoufal Raissouni. OCR accuracy improvement on document images through a novel pre-processing approach. *CoRR*, abs/1509.03456, 2015. URL http://arxiv.org/abs/1509.03456.

[11] Sheng He and Lambert Schomaker. Deepotsu: Document enhancement and binarization using iterative deep learning. *CoRR*, abs/1901.06081, 2019. URL http://arxiv.org/abs/1901.06081.

[12] Mohammed Javed, P. Nagabhushan, and B. B. Chaudhuri. Automatic detection of font size straight from run length compressed text documents. *CoRR*, abs/1402.4388, 2014. URL http://arxiv.org/abs/1402.4388.

[13] Khurram Khurshid, Imran Siddiqi, Claudie Faure, and Nicole Vincent. Comparison of niblack inspired binarization methods for ancient documents. In Kathrin Berkner and Laurence Likforman-Sulem, editors, *Document Recognition and Retrieval XVI, part of the IS&T-SPIE Electronic Imaging Symposium, San Jose, CA, USA, January 20-22, 2009. Proceedings*, volume 7247 of *SPIE Proceedings*, page 72470U. SPIE, 2009. doi: 10.1117/12.805827. URL https://doi.org/10.1117/12.805827.

[14] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P07-2045.

[15] Bernhard Liebl and Manuel Burghardt. From historical newspapers to machine-readable data: The origami OCR pipeline. In Folgert Karsdorp, Barbara McGillivray, Adina Nerghes, and Melvin Wevers, editors, *Proceedings of the Workshop on Computational Humanities Research (CHR 2020), Amsterdam, The Netherlands, November 18-20, 2020*, volume 2723 of *CEUR Workshop Proceedings*, pages 351–373. CEUR-WS.org, 2020. URL http://ceur-ws.org/Vol-2723/long20.pdf.

[16] Hector Llorens, Estela Saquete, and Borja Navarro-Colorado. Applying semantic knowledge to the automatic processing of temporal expressions and events in natural language. *Information Processing Management*, 49(1):179–197, 2013. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2012.05.005. URL https://www.sciencedirect.com/science/article/pii/S0306457312000702.

[17] William B. Lund, Douglas J. Kennard, and Eric K. Ringger. Why multiple document image binarizations improve ocr. In *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*, HIP '13, page 86–93, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321150. doi: 10.1145/2501115.2501126. URL https://doi.org/10.1145/2501115.2501126.

[18] Youmi Ma, Tatsuya Hiraoka, and Naoaki Okazaki. Named entity recognition and relation extraction using enhanced table filling by contextualized representations. *CoRR*, abs/2010.07522, 2020. URL https://arxiv.org/abs/2010.07522.

[19] N. Otsu. A threshold selection method from gray level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:62–66, 1979.

[20] Marte Ramirez-Ortegon, Edgar Duenez-Guzman, Raul Rojas, and Erik Cuevas. Unsupervised measures for parameter selection of binarization algorithms. *Pattern Recognition*, 44:491–502, 03 2011. doi: 10.1016/j.patcog.2010.09.018.

[21] Christian Reul. *An Intelligent Semi-Automatic Workflow for Optical Character Recognition of Historical Printings*. doctoralthesis, Universität Würzburg, 2020.

[22] Christophe Rigaud, Antoine Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. ICDAR 2019 Competition on Post-OCR Text Correction. In *15th International Conference on Document Analysis and Recognition*, pages 1588–1593, Sydney, Australia, September 2019. URL https://hal.archives-ouvertes.fr/hal-02304334.

[23] Kepa J. Rodriguez, Mike Bryant, Tobias Blanke, and Magdalena Luszczynska. Comparison of named entity recognition tools for raw ocr text. 09 2012. doi: 10.13140/2.1.2850.3045.

[24] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognit.*, 33:225–236, 2000.

[25] Sarah Schulz and Jonas Kuhn. Multi-modular domain-tailored OCR post-correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2716–2726, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1288. URL https://aclanthology.org/D17-1288.

[26] Daniel Van Strien., Kaspar Beelen., Mariona Coll Ardanuy., Kasra Hosseini., Barbara McGillivray., and Giovanni Colavizza. Assessing the impact of ocr quality on downstream nlp tasks. In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence - Volume 1: ARTIDIGH,*, pages 484–496. INSTICC, SciTePress, 2020. ISBN 978-989-758-395-7. doi: 10.5220/0009169004840496.

[27] Romen Taiyenjam, Sudipta Roy, Oinam Imocha Singh, Tejmani Sinam, and Khumanthem Singh. A new local adaptive thresholding technique in binarization. *CoRR*, abs/1201.5227, 01 2012.

[28] C. Tan and S. Lu. Automatic detection of document script and orientation. In *2007 9th International Conference on Document Analysis and Recognition*, volume 2, pages 237–241, Los Alamitos, CA, USA, sep 2007. IEEE Computer Society. doi: 10.1109/ICDAR.2007.67. URL https://doi.ieeecomputersociety.org/10.1109/ICDAR.2007.67.

[29] Joost van Beusekom, Faisal Shafait, and Thomas M. Breuel. Combined orientation and skew detection using geometric text-line modeling. *Int. J. Document Anal. Recognit.*, 13(2):79–92, 2010. doi: 10.1007/s10032-009-0109-5. URL https://doi.org/10.1007/s10032-009-0109-5.