Converting scanned documents to TEI XML

Vaishali Burge
(Matr.No: 221202802)
vburge@uni-koblenz.de

Harshani Reddy
(Matr.No: 221202816)
harshani@uni-koblenz.de

Jyothsna Kalyanaraman
(Matr.No: 221202825)
jyothsnakalyan@uni-koblenz.de

March 27, 2024

Under the Guidance Of: Dr. Jens Dörpinghaus
Mathematical Institute, University of Koblenz, Germany
Federal Institute for Vocational Education and Training (BIBB), Bonn, Germany

Abstract

This research addresses the crucial task of information retrieval from historical documents, specifically those from the 18th and 19th centuries. Optical Character Recognition (OCR) technology serves as a foundation for this process, enabling the conversion of scanned documents into machine-readable text format. A prominent OCR solution, Tesseract, was employed in this study due to its capability of recognizing a vast array of languages. This facilitated the extraction of valuable information and metadata from the digitized historical materials. Furthermore, standardized OCR-D processors were utilized to convert the raw data from its original PDF format into a structured TEI XML format. This transformation ensures the confidentiality and controlled access of the historical data for future research endeavors and analytical purposes. The implemented methodology offers a systematic approach to information gathering and retrieval from historical documents. This approach aligns with the broader efforts of digitalization and archival preservation, fostering continued exploration and understanding of historical records.

Contents

List of Figures

1 Introduction

In this fast-paced technology, powerful OCR-D technologies and processors play a significant role in archiving historical records. OCR engines use models that have already been trained for recognition. Each engine has one or more internal model formats. It's still challenging to understand these models, despite the significant changes in techniques and methods. When the models are being used for crucial decision-making or projections, "ß" is identified as "B" and "umlauts" are occasionally deleted. These flaws might occasionally result in sentence misunderstandings. Understanding the models become essential when it comes to extracting text from documents or PDFs to uphold this confidence. This implies that OCR-D users shouldn't have to worry about implementation concerns like "where do I get models from and where do I put them"[2] instead, they should be able to focus on optimizing their OCR operations.

1.1 Background Work

Optical Character Recognition (OCR), is a technology that automates the conversion of captured images of text, both handwritten and printed, into a machine-readable format [3]. This process utilizes specialized software or mechanical systems to extract the text from documents such as receipts, paper forms, and other physical media. Essentially, OCR acts as a bridge, transforming the information contained within physical documents into digital data that can be easily processed, stored, and utilized by computers and other electronic devices. The development of OCR technology has been marked by significant advancements since its inception in the late 19th century. A notable milestone was achieved in 1935 with the issuance of US patent 2,026,329 for a "reading machine" invented by Gustav Tauschek[4]. Initially, the primary application of OCR technology was the digitization of printed text, allowing machines to process and analyze information previously confined to physical documents. As OCR technology matured, its applications expanded across diverse industries. Over the years, significant advancements have been made in OCR technology, leading to increased processing speeds and improved accuracy .

1.2 How does optical character recognition work?

An open-source application called OCR-D provides an extensive approach to full-text document recognition. This section tackles the complicated processes, breaking it into independent, thoroughly detailed steps. This framework allows for the creation of optimal workflows designed specifically for the management of historical printed documents, leading to full-text outputs that are highly valuable for research.

Full-text recognition is a multifaceted process that includes preprocessing and postprocessing procedures in addition to core text recognition. The first step is to preprocess the digital image so that it is better suited for text recognition. Methods like deskewing, dewarping, despeckling, binarization, and crop-

ping fall under this category. Layout recognition then locates text sections on the page, even down to the individual line. It is very important to recognize lines and baselines since they serve as the basis for later text recognition, which is mostly done with neural networks. Finally, the OCR results are post-corrected, if necessary, before being stored in repositories for long-term preservation.

The common steps involved are:

1. Installing necessary dependencies and the ocrd_all software to process the files.
2. Once the software is installed, we create a workspace to add files and process them separately:
   `ocrd workspace [-d path/to/workspace] init`
3. We add a unique ID to the workspace:
   `ocrd workspace [-d path/to/your/workspace] set-id 'unique ID'`
4. We add the files by the CLI:
   `cp -r path/to/your/images [path/to/your/workspace/].`
5. Now, we add those images to the empty METS created above by adding references for their path names. There are several ways to do this .
   (a) `ocrd workspace [-d path/to/your/workspace] add -g {ID of the physical page} -G {name of the image fileGrp} -i {ID of the image file} -m image/{MIME format of that image} path/to/that/image/file/in/workspa`
   (b) `ocrd workspace add -g P_00001 -G OCR-D-IMG -i OCR-D-IMG_00001 -m image/tiff images/00001.tif`
   (c) `ocrd workspace add -g P_00002 -G OCR-D-IMG -i OCR-D-IMG_00001 -m image/tiff images/00002.tif`[5]
6. If there are multiple pages, we utilize the CLI split command
   `convert combined_image.jpg -crop 50%x100% +repage page%d.jpg`.

1.3 Utilizing the OCR-D Processors

There exist various methods to employ the OCR-D processors, all of which utilize the following syntax:

- **-I** Input-Group: fileGrp containing the files to be processed
- **-O** Output-Group: fileGrp where the results will be stored
- **-P** parameter value: direct assignment of parameters for a specific processor

- **-p** parameter-file: file-based assignment of parameters for a specific processor
- **-g** page-range: range of physical pages to be processed [5]

1. **Executing a single processor:** If you intend to execute a single processor, then the navigation to the workspace can be achieved through the following command [5].

   ```
   ocrd-{processor name} -I {Input-Group} -O {Output-Group} [-p {parameter-file}]
   [-P {parameter} {value}]
   ```

   For instance:

   ```
   ocrd-olena-binarize -I OCR-D-IMG -O OCR-D-BIN -P impl sauvola
   ```

   The designated processor will read files in Input-Group, binarize them, and save the results in Output-Group within your workspace. It will also include information about this processing step in the METS metadata

2. **Executing multiple processors:** Consecutively running multiple processors on the same data is termed a workflow. For workflow processing, a workflow format and engine are necessary,as shown in the below command[5].

   For example:

   ```
    ocrd process \
   'cis-ocropy-binarize -I OCR-D-IMG -O OCR-D-SEG-PAGE' \
   'tesserocr-segment-region -I OCR-D-SEG-PAGE -O OCR-D-SEG-BLOCK' \
   'tesserocr-segment-line -I OCR-D-SEG-BLOCK -O OCR-D-SEG-LINE' \
   'tesserocr-recognize -I OCR-D-SEG-LINE -O OCR-D-OCR-TESSEROCR -P model Fraktur
   ```

3. **Obtaining more information about processors:** To access all available processors, can be accessed through following command[5]:

   ```
   ocrd-{processor name} --help
   ```

1.4 OCR-D Processor Models

OCR engines utilize pre-trained models for recognition purposes. Each engine may use one or more internal model formats. While some engines like Calamari require a full path to a model, others such as Tesseract, Ocropy, and Kraken allow centralized storage of models at a designated models[2].

1.4.1 Available Models

To retrieve a list of recognized or installed file resources, the command [2] can be executed as:

```
ocrd resmgr list-available
```

1.4.2 Resource Installation

Processor developers offer known resources in the `ocrd-tool.json` and are accessible by name to `ocrd resmgr download`. This can be achieved by installing resources using these commands [2]:

```
ocrd resmgr download <processor> <resource>
```

```
ocrd resmgr download ocrd-tesserocr-recognize '*'
```

Alternatively, can be also installed using `*` to download all known resources for a processor[2]:

```
ocrd resmgr download '*'
```

(In both cases, `*` must be in quotes or escaped to avoid errors.)

1.5 Guidelines to work with METS/PAGE

OCR-D has chosen METS as the foundation for its data transferring protocol. A set of guidelines and a method for utilizing METS files with OCR-D are described in this paper[6].

1.6 Metadata

1.6.1 A unique identification number for the generated document

Overall a unique identification for the each document must be included in the `mods:identifier` of METS documents. The preferred values for the `type` attribute are purl, urn, handle, or url.

1.6.2 Use relative filenames or URLs at all times.

Files not located in the METS file directory or any of its subdirectories should be accessed using the URL. There are supplied examples of both legal and invalid `mets:FLocat/@xlink:href` in the METS file.

1.6.3 Making sure Track of Processes Details in METS

Changes should be indicated in the METS metadata header by information added by OCR-D processors. To include agent details:

1. Locate the first `mets:metsHdr`.
2. Add a new `mets:agent` with attributes:
   - `TYPE` = "OTHER"
   - `OTHERTYPE` = "SOFTWARE"
3. Add a `mets:name` including processor name and version.

Example:

```
<mets:agent TYPE="OTHER" OTHERTYPE="SOFTWARE" ROLE="OTHER"
OTHERROLE="preprocessing/optimization/binarization">
  <mets:name>ocrd_tesserocr v0.1.2</mets:name>
</mets:agent>
```

1.7 Images

1.7.1 Explicit and sufficient pixel density of images is required.

Pixel resolution in the original input photos MUST be higher than 150 ppi. It is crucial to make sure that the density is explicitly adjusted during the digitalization process of every process that produces new pictures and alters their size. When images have missing or unusually insufficient pixel density metadata, processors should assume 300 ppi.

1.7.2 No Multi-page Images

Users MUST provide single-image TIFF files.

1.7.3 Images and Coordinates

Coordinates in a PAGE-XML are always absolute, When accessing the image of a layout element like a `pc:TextRegion` or `pc:TextLine`, the algorithm should resolve the image's location based on the specified rules.

1.8 File Groups `mets:fileGrp`

Referenced files must be organized into file groups without redundancy.

1.8.1 File Group `@USE` Syntax

All `mets:fileGrp` must have a unique `USE` attribute that hints at the provenance of the files and must be a valid `xsd:ID`.

1.8.2 File ID Syntax

Each `mets:file` must have an ID attribute. The ID attribute of a `mets:file` should be the `USE` of the containing `mets:fileGrp` combined with its corresponding page ID or a 4-zero-padded number. The ID must be unique inside the METS file .

1.8.3 MIME Type Syntax

Every `mets:file` element representing a PAGE file must have its MIMETYPE attribute set to `application/vnd.prima.page+xml` [6].

2 Related Work

Utilizing the OCR-D project website, the BSB (which was then a component of the coordinating project) conducted a survey in the spring of 2016 on how people make use of OCR texts, mainly aimed at humanistic individuals. A total of 139 scholars participated in the survey. Only a portion of the participants responded to some of the questions, and 39 of those responses were partially unreadable.

2.1 PHASE 1 : Insights from a User Survey

Here are the key takeaways:

1. Wide Use: A significant majority of participants utilize OCR texts in their research endeavors.
2. Dual Purpose: These texts serve two primary functions:
   (a) Search Tools: Researchers leverage them to search through large volumes of text data efficiently.
   (b) Text Analysis: They also employ OCR texts as a foundation for analyzing substantial datasets.
3. Tolerance for Errors: Interestingly, 60% of participants are willing to use OCR texts with errors ("dirty OCR") for research, while 40% consider such data unusable.
4. Historians' Preference: Historians stand out with a strong preference (87%) for using even error-prone OCR texts. They find them valuable as finding aids, helping to uncover information that might be missed otherwise.
5. Citing with OCR: OCR texts can facilitate the citation process by providing a starting point. Researchers can then correct any errors before the final citation. Source Preference: While OCR texts are used for citing, the original image remains the preferred source for librarians (61% vs. 39%).

6. Versioning Opinions: The concept of version control for OCR texts has received mixed reviews. Three-quarters of researchers acknowledge the importance of tracking changes, but only half see the need to access prior versions. The primary reasons for versioning relate to consistent quoting and replicating analyses based on the OCR text, although the burden of managing multiple versions is a concern. [7]

Overall, the survey highlights the widespread adoption of OCR texts in research. Even texts with errors can be valuable, particularly for historians. While the original image remains the gold standard for citing, OCR texts offer advantages in search and analysis and can streamline the citation process. The concept of version control for OCR texts requires further discussion to balance its benefits with its practical challenges.

2.2 PHASE 2: Module Project

This phase focused on improving the OCR-D workflow for full text recognition of historical prints. Here's a brief overview of the key projects:

- Image Preprocessing (DFKI): Enhanced tools for tasks like cropping, deskewing, and dewarping to improve image quality for OCR.
- Layout Recognition (DFKI): Extracted document structure for efficient text recognition, including text/non-text segmentation and block classification.
- Layout Analysis and Segmentation (U Würzburg): Developed a tool using Convolutional Neural Networks (CNNs) to separate text from images and segment pages.
- OCR Post-correction (U Leipzig): Implemented a system combining neural networks and finite-state transducers to improve OCR accuracy.
- Tesseract Integration (U Mannheim): Integrated Tesseract OCR engine into the OCR-D workflow and enhanced its stability, code quality, and performance.
- Automatic Post-correction (U Munich): Created a system for automatic post-correction of historical OCR outputs, with an optional interactive correction tool.
- Font Recognition and Model Repository (U Leipzig et al.): Developed tools for automatic font recognition and built a repository of font-specific OCR models for improved accuracy.
- Long-term Archiving (SUB Göttingen et al.): Created a prototype for long-term archiving and persistent identification of OCR results for historical prints. [7]

2.3 TEI Overview

TEI, established in 1987, develops guidelines for encoding texts in humanities and social sciences. Its guidelines, schemas, and the TEI Consortium (TEI-C) support ongoing development.

1. Encoding Scheme : The original TEI language (P1 through P3) used SGML syntax. With P4, users were given a choice of using SGML or XML; with P5, SGML is no longer an option. Furthermore, P5 relies heavily upon XML standards such as schema languages and programming tools like XSLT and XQuery. TEI Guidelines use XML syntax to define an encoding scheme, offering a tag set of nearly 500 elements for capturing metadata and structural features of documents.

2. Flexibility : TEI allows modular encoding, enabling users to select features tailored to specific genres.

3. Optimization Manual: TEI Guidelines provide detailed requirements and best practices for text encoding, emphasizing proper usage and examples. It includes an alphabetical list of TEI elements and explanations of concepts.

4. Quality Assurance: TEI conformance ensures documents are well-formed XML and validate against standard TEI schemas. Customizations must be correctly documented to maintain TEI conformance.[8]

3 Approach

The OCR-D procedure for obtaining whole text from scanned historical prints is described in this documentation. The guide explains each step and its purpose, while some may be optional based on the type of image. This guide gives an overview of the OCR-D processors that are available, as well as the parameters they require.
Important Note: Before diving into these workflows, ensure your images are properly prepared within an OCR-D workspace.

3.1 Final Workflow

1. `ocrd-preprocess-image`
   Input: OCR-D-IMG
   Output: OCR-D-PREP
   Parameters: output_feature_added binarized, command "scribo-cli sauvola-ms-split @INFILE @OUTFILE –enable-negate-output "

   Explanation: This processor enhances images for better OCR results by adding features and binarizing them using the Sauvola method with negation for foreground extraction [9] [10].

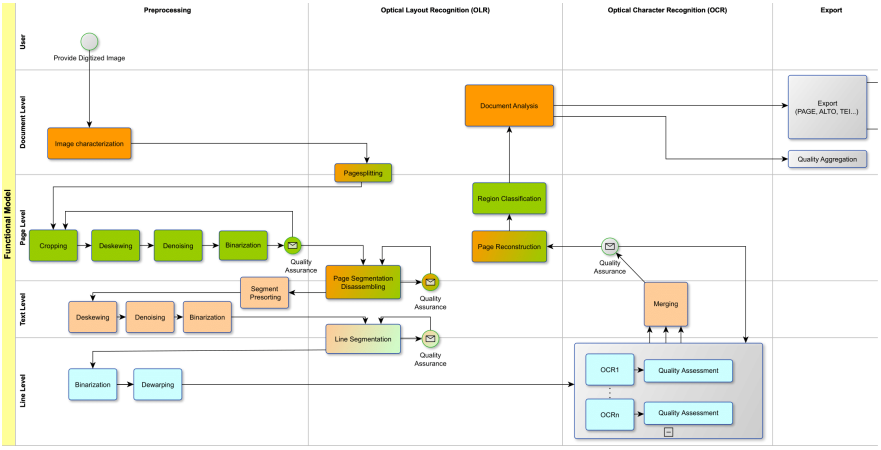


Figure 1: Funktional Model [1]

2. `ocrd-anybaseocr-crop`
   Input: OCR-D-PREP
   Output: OCR-D-CROP
   Parameters: marginTop 0.1, marginBottom 1.0, marginLeft 0.1, marginRight 0.9

   Explanation: This processor crops the image to remove margins and unnecessary content. It preserves essential elements while discarding irrelevant portions, based on specified parameters.

3. `ocrd-skimage-denoise`
   Input: OCR-D-CROP
   Output: OCR-D-BIN-DENOISE
   Parameters: level-of-operation page

   Explanation: Denoising the cropped image removes artifacts and noise, enhancing OCR accuracy, especially at the page level.

4. `ocrd-tesserocr-deskew`
   Input: OCR-D-BIN-DENOISE
   Output: OCR-D-BIN-DENOISE-DESKEW
   Parameters: operation_level page

   Explanation: This processor corrects skew or rotation in the image, focusing on the page level, crucial for maintaining text alignment.

5. `ocrd-tesserocr-segment`
   Input: OCR-D-BIN-DENOISE-DESKEW
   Output: OCR-D-SEG
   Parameters: shrink_polygons true

   Explanation: Segmentation of the image into lines helps improve OCR

accuracy, especially in multi-line documents, based on the specified parameter.

6. `ocrd-cis-ocropy-dewarp`
   Input: OCR-D-SEG
   Output: OCR-D-SEG-LINE-RESEG-DEWARP

   Explanation: This processor dewarps the segmented lines to rectify distortions incurred during scanning or image acquisition, ensuring accurate representation.

7. `ocrd-tesserocr-recognize`
   Input: OCR-D-SEG-LINE-RESEG-DEWARP
   Output: OCR-D-OCR
   Parameters: textequiv_level line, overwrite_segments true, model Fraktur+deu

   Explanation: Utilizing the Tesseract OCR engine, this processor recognizes text from the dewarped and segmented lines, facilitating language-specific recognition based on the specified model.

8. `ocrd-fileformat-transform`
   Input: OCR-D-OCR
   Output: OCR-D-HOCR
   Parameters: from-to "page hocr"

   Explanation: This processor transforms the recognized text into the HOCR format, suitable for subsequent processing or analysis.

3.1.1 Minimal Workflow

The minimal workflow utilizes the `ocrd-tesserocr-recognize` processor, which integrates various functions such as binarization (Otsu), region segmentation and text recognition into a single step. It serves as a fundamental single-step process to establish a baseline result that can be compared to more detailed workflows.And for less noisy and less disturbed images
For improved accuracy and flexibility, minimal workflow is recommended. Below is an example of such a workflow using the `ocrd-tesserocr-recognize` processor:

Processor: `ocrd-tesserocr-recognize`.
Parameters: - `-P segmentation_level region`.
- `-P textequiv_level word`.
- `-P find_tables true`.
- `-P model Fraktur_GT4HistOCR`.
Example using `ocrd-process`:

```
ocrd process "tesserocr-recognize -P segmentation_level region -P textequiv_level
word -P find_tables true -P model GT4HistOCR_5000000.997_191951" [1]
```

3.2 Conversion of METS File to HOCR

3.2.1 Step 1: Text Recognition and Segmentation

Utilize the `ocrd-tesserocr-recognize` processor to perform text recognition and segmentation at the specified level within the PAGE XML hierarchy. This step involves reusing or detecting segments and adding the TextEquiv results accordingly.

Command:

```
ocrd-tesserocr-recognize -I OCR-D-SEG-LINE-RESEG-DEWARP \
-O OCR-D-OCR -P textequiv_level line -P overwrite_segments true \
-P model Fraktur+deu
```

3.2.2 Step 2: XML to HOCR Conversion

Convert the resulting XML file to HOCR format using the `ocrd-fileformat-transform` processor through the command line interface.

Command:

```
ocrd-fileformat-transform -I OUTPUT -O output1.hocr \
-P from-to "page hocr"
```

3.3 HOCR to TEI XML Conversion

3.3.1 Preparation

Find out that HOCR files acquired by OCR are accessible when combined with a pre-established template for TEI XML encoding. This makes it easier to analyze and save data effectively.

3.3.2 Parsing HOCR Files

Take text content out of the HOCR files and get its coordinates. This is an essential step to preserve the quality of the document layout in later encoding procedures.

3.3.3 Mapping Text and Coordinates

Create a relationship between the text that has been extracted and the associated location in the document. This makes it possible for the encoded TEI XML output to accurately depict the original layout.

3.4 TEI XML Structure Design

To successfully arrange textual content, strategically create the TEI XML structure. Include components like sections, headings, and paragraphs to show the document's hierarchical structure.

3.4.1 Encoding Text

Convert the text that has been extracted into TEI XML format. Keep in mind that font styles, formatting options, and other textual elements are crucial for preserving data integrity and enabling additional analysis.

3.4.2 Handling Structural Elements

Include headers, paragraphs, and other textual divisions in the document's encoding. Use the proper TEI elements to represent the content's natural hierarchy and organization.

3.4.3 Incorporating Metadata

Include relevant metadata in the TEI header section, such as the title of the document, the author's identity, and the publishing date. This improves the context of the documents and makes thorough research and analysis easier.

3.4.4 Preserving Layout Information

Using TEI components, precisely capture the spatial arrangement of textual elements within the document by capturing and encoding layout characteristics like text alignment and page structure. This guarantees the integrity to the original design.

3.4.5 Workflow Summary

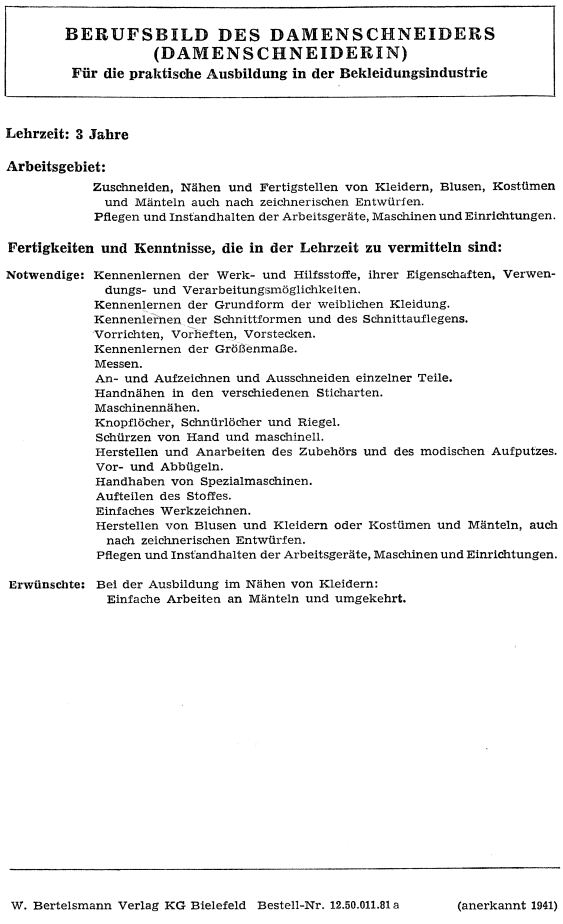
The conversion process in Python involves the following steps:

1. **Parsing HOCR File:** Read the HOCR file and parse the content using BeautifulSoup.
2. **Extracting Specific Information:** Find and extract text based on defined bounding boxes for different sections like title, date, place, organization name, etc.
3. **Writing TEI XML Document:** Construct a TEI XML document with the extracted information and write it to a file.

The workflow summary encompasses reading the HOCR file containing OCR data, parsing the content using BeautifulSoup to navigate through the document, extracting specific information like title, date, place, organization name using defined bounding boxes, constructing a TEI XML document incorporating the extracted details, and writing the TEI XML document to an output file for further processing and documentation.

4 Experiments

These operations of processors which include binarization, cropping, and line segmentation.The processed files are subsequently converted to Hierarchical OCR (HOCR) format, which provides a structured representation of the document's textual content. Finally, a Python script is employed to transform the HOCR files into Text Encoding Initiative (TEI) XML format, facilitating further analysis and manipulation of the document data. This process, is depicted in Figure 3 below.



BERUFSBILD DES DAMENSCHNEIDERS
(DAMENSCHNEIDERIN)
Für die praktische Ausbildung in der Bekleidungsindustrie

Lehrzeit: 3 Jahre

Arbeitsgebiet:

Zuschneiden, Nähen und Fertigstellen von Kleidern, Blusen, Kostümen und Mänteln auch nach zeichnerischen Entwürfen.
Pflegen und Instandhalten der Arbeitsgeräte, Maschinen und Einrichtungen.

Fertigkeiten und Kenntnisse, die in der Lehrzeit zu vermitteln sind:

Notwendige:
Kennenlernen der Werk- und Hilfsstoffe, ihrer Eigenschaften, Verwendungs- und Verarbeitungsmöglichkeiten.
Kennenlernen der Grundform der weiblichen Kleidung.
Kennenlernen der Schnittformen und des Schnittauflegens.
Vorrichten, Vorheften, Vorstecken.
Kennenlernen der Größenmaße.
Messen.
An- und Aufzeichnen und Ausschneiden einzelner Teile.
Handnähen in den verschiedenen Sticharten.
Maschinennähen.
Knopflöcher, Schnürlöcher und Riegel.
Schürzen von Hand und maschinell.
Herstellen und Anarbeiten des Zubehörs und des modischen Aufputzes.
Vor- und Abbügeln.
Handhaben von Spezialmaschinen.
Aufteilen des Stoffes.
Einfaches Werkzeichnen.
Herstellen von Blusen und Kleidern oder Kostümen und Mänteln, auch nach zeichnerischen Entwürfen.
Pflegen und Instandhalten der Arbeitsgeräte, Maschinen und Einrichtungen.

Erwünschte:
Bei der Ausbildung im Nähen von Kleidern:
Einfache Arbeiten an Mänteln und umgekehrt.

W. Bertelsmann Verlag KG Bielefeld Bestell-Nr. 12.50.011.81 a (anerkannt 1941)

Figure 2: The Original file

```
<TEI>
  <teiHeader>
    <fileDesc>
    <titleStmt>
        <title>BERUFSBILD DES DAMENSCHNEIDERS
(DAMENSCHNEIDERIN)
Für die praktische Ausbildung in der Bekleidungsindustrie
</title>
        </titleStmt>
        <orgName>KG Bielefeld</orgName>
      <publicationStmt>
        <publPlace>Bielefeld</publPlace>
      </publicationStmt>
    </fileDesc>
  </teiHeader>
  <text>
    <body>
      <div>
        <p>Lehrzeit: 3 Jahre</p>
        <head>Arbeitsgebiet:</head>
        <p>Zuschneiden, Nähen und Fertigstellen von Kleidern, Blusen, Kostümenund
        Mänteln auch nach zeichnerischen Entwürfen.
        Pflegen und Instandhalten der Arbeitsgeräte, Maschinen und Einrichtungen.</p>
        <head>Fertigkeiten und Kenntnisse, die in der Lehrzeit zu vermitteln sind:
        </head>
        <p>Notwendige: Kennenlernen der Werk- und Hilfsstoffe, ihrer Eigenschaften,
        Verwen-</p>
        <p>dungs- und Verarbeitungsmöglichkeiten.Kennenlernen der Grundform der
        weiblichen Kleidung.Kennenlernen der Schnittformen und
        des Schnittauflegens.Vorrichten, Vorlieften, Vorstecken.Kennenlernen der
        Größenmaße.Messen.An- und Aufzeichnen und Ausschneiden
        einzelner Teile.Handnähen in den verschiedenen
        Sticharten.Maschinennähen.Knopflöcher, Schnürlöcher und Riegel.Schürzen von
        Hand
        und maschinell.Herstellen und Anarbeiten des Zubehörs und des modischen
        Aufputzes.Vor- und Abbügeln.Handhaben von Spezialmaschinen.
        Aufteilen des Stoffes.Einfaches Werkzeichnen.Herstellen von Blusen und
        Kleidern oder Kostümen und Mänteln, auchnach zeichnerischen Entwürfen.
        Pflegen und Instandhalten der Arbeitsgeräte, Maschinen und Einrichtungen.</p>
        <head>Erwünschte:</head>
        <p>Bei der Ausbildung im Nähen von Kleidern:Einfache Arbeiten an Mänteln und
        umgekehrt.</p>
      </div>
    </body>
  </text>
</TEI>
```

Figure 3: The Final TEI Output

5 Conclusion

OCR-D offers a powerful framework for processing historical documents. While current implementations address many challenges, ongoing efforts to enhance character recognition accuracy, broaden processor options, and refine workflows will ensure continued improvement in the quality and efficiency of historical document digitization.

Through OCR-D, we have successfully processed images and converted them into structured formats like HOCR and TEI XML, marking a significant advancement in document digitization. However, minor issues persist, such as character identification accuracy, indicating areas for improvement.To address these challenges, further experimentation and optimization efforts are needed. By conducting additional experiments and refining algorithms, we can improve character identification accuracy and optimize processing pipelines. These efforts will contribute to enhancing the overall performance and reliability of OCR-D workflows.

These developments not only enhance document processing skills but also open up exciting possibilities for information retrieval and analysis in a variety of fields. OCR-D has the potential to significantly progress document digitization and open the door to more effective and accessible digital libraries and archives with further testing and development.

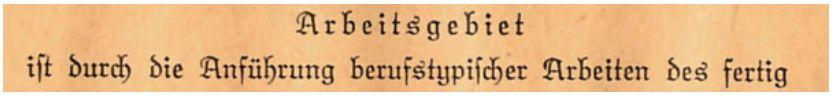
6 Limitations

6.1 Limitations of Tesseract:

- **Preprocessing Dependency:** Tesseract necessitates meticulous preprocessing for optimal performance, which is often challenging due to varying image quality.
- **Scanned Images:** Less effective with scanned documents due to artifacts and skewed text.
- **Complex Layouts:** Struggles with intricate layouts, multi-column text, and unconventional arrangements.
- **Handwriting Recognition:** Tailored for printed text, thus challenging for handwritten content.
- **Language and Fonts:** Performance fluctuations with less common languages and fonts.
- **Gibberish Output:** May generate gibberish, affecting data accuracy[11].

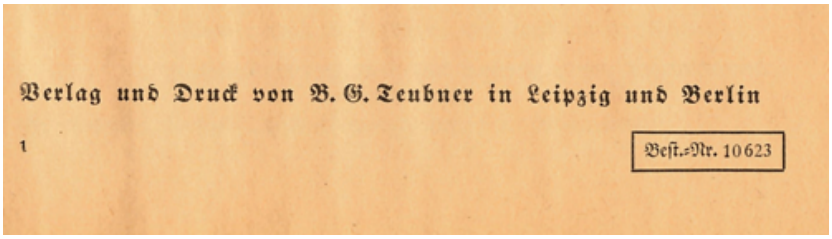
6.2 Identifying certain Letters

Such as letter "ch" fonts are being identified as "&alt" and some are interpreted wrongly, when applied to the Tesseract model hOCR conversions.




```
</span>ift dur&lt; die Anführung berufstypifcher Arbeiten des fertig</span>
```

Figure 4: Sometimes identifying "ch" as "&lt"




```
></span>Verlag und Deu&gt; von B. G., Teubner in Leipzig und Berlin</span>
```

Figure 5: Wrong identification of letters "Berleg" as "Verlag" and "Druck" as "Deu&gt"

References

[1] Ocr-d workflows. https://ocr-d.de/en/workflows.

[2] Ocr-d models. https://ocr-d.de/en/models.

[3] Herbert F. Schantz. *The History of OCR, Optical Character Recognition*. Recognition Technologies Users Association, Manchester Center, Vt., 1982.

[4] Katya Lopez-Nichols.

[5] Ocr-d user guide. https://ocr-d.de/en/user_guide.

[6] Library of Congress. METS: Metadata Encoding Transmission Standard. Technical report, Library of Congress, 2020.

[7] Ocr-d user survey. https://ocr-d.de/en/user_survey.

[8] Introducing the TEI guidelines. https://tei-c.org/support/learn/introducing-the-guidelines/.

[9] Mehmet Sezgin and Bulent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146, 2004.

[10] Maya R. Gupta, Nathaniel P. Jacobson, and Eric K. Garcia. Ocr binarisation and image pre-processing for searching historical documents. *Pattern Recognition*, 40(2):389, 2007.

[11] Docsumo. Introduction to tesseract ocr.