



Universiteit  
Leiden  
The Netherlands

# Computer Science & Economics

Enriching Historical Records:  
An OCR and AI-Driven Approach for Database Integration

Zahra Abedi

Supervisors:  
Richard van Dijk & Gijs Wijnholds

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)  
[www.liacs.leidenuniv.nl](http://www.liacs.leidenuniv.nl)

18/07/2024

## Abstract

This paper is part of the Linking University, City, and Diversity (LUCD) project, which aims to visualize the interactions between Leiden University and the city of Leiden since 1575 and capture the impact of students and professors on Leiden. Focusing specifically on the digitization of the 'Leidse hoogleraren en lectoren 1575-1815' dataset, originally compiled by A.A. Bantjes and L. van Poelgeest between 1983 and 1985, this research is dedicated to converting these typewritten records into a digital format and importing them to a centralized database created by the LUCD project. The dataset contains valuable information about professors and curators at Leiden University, such as birth and death details, education, and career history. The central research question is: 'How can we accurately extract and transform historical records data from scanned historical documents and map it into a centralized database?' This question is addressed through three sub-questions: the accuracy of OCR techniques, the use of AI for structured data extraction, and the mapping of this data into a centralized database which is developed by the LUCD project and contains high-quality data on professors and students associated with Leiden University.

The methodology begins with image preprocessing to enhance the quality of scanned documents for better OCR performance. Tesseract OCR is then trained on a customized training set to improve text recognition accuracy, especially for historical and language-specific nuances. For structuring the extracted text, GPT-3.5 Turbo with function calling and Pydantic is used, enabling us to generate valid JSON outputs aligned with our format requirements. The final step involves modifying the database structure and creating an algorithm for optimal person matching, considering various personal attributes to ensure accurate data linkage.

Results indicate that the OCR-generated text had a Character Error Rate (CER) of 1.08% and a Word Error Rate (WER) of 5.06%. Despite these errors, the generated text provided a workable basis for further processing. The AI model's JSON extraction performance achieved an accuracy of 65.04% for JSONs generated using correct text as input, and 62.72% for JSONs generated using OCR-generated text as input, indicating moderate accuracy that is subject to improvement. The record linkage algorithm demonstrated robust performance, linking correct JSON files created manually with an average accuracy of 93.67% and linking JSON files made from OCR-generated text files with an average accuracy of 80.95%.

This research contributes to the field of digital humanities by providing a structured approach to digitizing and analyzing historical records. The methodologies developed herein, including advanced OCR and AI techniques, offer a framework for similar projects. Our study highlights challenges such as page layout variability and terminology differences, suggesting future work in advanced layout recognition, volume-specific JSON schemas, and implementing more flexible linking algorithms using Levenshtein distance. Additionally, exploring multi-modal AI models such as GPT-4o could enhance data extraction, potentially bypassing OCR steps altogether.

## **Acknowledgements**

I am grateful to my thesis supervisors, Richard van Dijk, and Gijs Wijnholds, for their guidance and support throughout this project.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	LUCD Project . . . . .	1
1.3	Objectives . . . . .	1
1.4	Research Questions . . . . .	2
1.5	Thesis overview . . . . .	2
<b>2</b>	<b>Related Work</b>	<b>3</b>
2.1	Optical Character Recognition . . . . .	3
2.2	Information Extraction . . . . .	3
2.3	Record Linkage . . . . .	3
<b>3</b>	<b>Data</b>	<b>4</b>
<b>4</b>	<b>Methods</b>	<b>7</b>
4.1	Phase 1: PDF to PNG Conversion, Image Preprocessing, and OCR & Text Segmentation Per Person . . . . .	7
4.2	Phase 2: JSON Extraction from Text with AI . . . . .	10
4.2.1	Schema Definition . . . . .	12
4.2.2	Extraction Techniques . . . . .	13
4.3	Phase 3: Record Linkage & Database Enrichment . . . . .	14
4.3.1	Linking Algorithm . . . . .	14
<b>5</b>	<b>Evaluation</b>	<b>17</b>
5.1	Phase 1 Evaluation: Quality Assessment of Generated Text . . . . .	17
5.2	Phase 2 Evaluation: Quality Assessment of Generated JSON . . . . .	19
5.3	Phase 3 Evaluation: Quality Assessment of Linking Algorithm . . . . .	24
<b>6</b>	<b>Discussion</b>	<b>28</b>
<b>7</b>	<b>Further Work</b>	<b>29</b>
<b>8</b>	<b>Conclusion</b>	<b>30</b>
	<b>References</b>	<b>31</b>
<b>A</b>	<b>Example Page: Alphabetical List of Names</b>	<b>32</b>
<b>B</b>	<b>Example Page: Detailed Information About a Person</b>	<b>33</b>
<b>C</b>	<b>Example Page: List of Sources Used in the Volume</b>	<b>34</b>
<b>D</b>	<b>Example Person: Original Images</b>	<b>35</b>
<b>E</b>	<b>Example Person: Preprocessed Images</b>	<b>37</b>

F Example Person: OCR-Generated Text	39
G Example Person: AI-Generated JSON	42
H Example Person: Performance of the Record Linking Algorithm	49
I Regular Expressions Patterns for Information Extraction	50

# 1 Introduction

## 1.1 Background

Historical records serve as invaluable sources of information for researchers and historians. Significant research has been dedicated to converting these paper-based records into electronic formats, a process known as digitization [Pearce-Moses and Baty \[2005\]](#). Digitizing historical documents not only helps to preserve them but also offers additional benefits, such as improved accessibility for a broader audience and increased opportunities for discovering new links between the data [Ooghe et al. \[2009\]](#).

The process of digitizing documents involves several steps. Initially, the paper-based document is scanned to create a digital image. Subsequently, Optical Character Recognition (OCR) methods are used to extract text from the scanned document. Once the OCR-generated text is acquired, we can then apply information extraction techniques customized to meet our specific needs.

## 1.2 LUCD Project

This thesis is a part of the Linking University, City, and Diversity (LUCD) project<sup>1</sup>, which seeks to illustrate the interactions between Leiden University and city of Leiden since 1575 using data science techniques. One of the main focuses of this project is to capture the impact of students and professors on this city. The LUCD project is a collaborative work between researchers and students from LIACS and the Institute for History of Leiden University. The core team consists of Ariadne Schmidt (Professor specializing in the History of Urban Culture), Wessel Kraaij (Professor of Applied Data Analytics) Joost Visser (Professor of Large Scale Software and Data Science), and Richard van Dijk (Research Software Engineer). In 2022, a few Bachelor's projects, involving contributors like Rick Schreuder, Liam van Dreumel, and Michael de Koning, supported the LUCD project. Implementing a software architecture designed by Richard van Dijk, Liam van Dreumel concentrated on visualizing data, Rick Schreuder tackled the design of the database, and Michael de Koning took charge of software components, known as adapters, responsible for data extraction, transformation, loading, and linking. Consequently, they developed a website enabling users to discover more information on topics such as the origins of many of the students and professors at Leiden University, as well as how the academic community at the university has grown over time.<sup>2</sup>

## 1.3 Objectives

Building upon previous research, this thesis focuses primarily on enriching the centralized database, which contains high-quality data from various sources about students and professors of Leiden University. Specifically, we aim to integrate the 'Leidse hoogleraren en lectoren 1575-1815' dataset compiled by A.A. Bantjes and L. van Poelgeest in 1983. This dataset, originally typed using a typewriter, presents biographical data concerning professors and curators at Leiden University from

---

<sup>1</sup><https://www.universiteitleiden.nl/onderzoek/onderzoeksprojecten/wiskunde-en-natuurwetenschappen/liacs-linking-university-city-and-diversity>

<sup>2</sup><https://university.liacs.nl/dashboard/>

1575-1815 and is available in scanned format on the website of Leiden University<sup>3</sup>. More information about this dataset is provided in section 3. The objectives of this thesis include designing and implementing a pipeline to process scanned images, extracting highly accurate text, and converting it into a structured format. This structured data will be integrated with existing datasets from other sources, creating new entities and linking them with relevant entities in the database. Adapters will facilitate collaboration between incompatible interfaces without requiring modifications to their source code.

Given the significant research already conducted in the field of OCR, further extensive research is unnecessary. Instead, we will use an off-the-shelf OCR package and only make modifications to the parameters as needed. Consequently, the primary focus of this thesis is not centered on achieving flawless OCR results. Instead, the objective is to generate text of sufficient quality to effectively support information retrieval methods.

In case 'Leidse hoogleraren en lectoren 1575-1815' contains new information not covered by the centralized database and lacks corresponding tables, modification of the centralized database would be necessary to encapsulate all additional information.

## 1.4 Research Questions

The main research question is: 'How can we accurately extract and transform historical records data from scanned historical documents and map it into a centralized database?' Sub-questions include:

- How can we extract high-accuracy text from scanned historical documents using OCR techniques?
- How can AI be used to analyze the OCR-generated text and obtain a structured format?
- How can we map the structured data into a centralized database?

## 1.5 Thesis overview

Section 2 reviews existing literature and methodologies related to our project. The description of the dataset used and its historical context can be found in section 3. In section 4, we provide details of our three-phase methodology: image preprocessing and OCR, AI-based information extraction, and record linkage. Section 5 presents the evaluation metrics and results for each phase. Discusses of the findings, challenges, and implications of the research are mentioned in section 6. Section 7 proposes future research directions to enhance the methodologies. Section 8 summarizes the research contributions and outcomes.

---

<sup>3</sup><https://digitalcollections.universiteitleiden.nl/view/item/2078065/pages>

## 2 Related Work

### 2.1 Optical Character Recognition

Tesseract is a free and open-source OCR engine which is initially developed by Hewlett-Packard in the 1980s and is now maintained by Google<sup>4</sup>. Tesseract has been adapted to recognize a wide range of languages and scripts beyond its original design for English.

White [2012] has researched the process of training the Tesseract OCR engine to support Ancient Greek. The paper covers general procedures involved in training a new language for Tesseract, including training the script with common printed fonts and adding language-specific hints to improve recognition. It highlights particular challenges due to Tesseract’s English language heritage and describes strategies and programs developed to overcome these issues. In our project, we aim to use Tesseract as well and improve its outcomes for processing historical records. By incorporating additional training and introducing a wordlist of the most frequent words, we aim to enhance the accuracy of Tesseract in generating the text.

### 2.2 Information Extraction

Information Extraction (IE) is an important early stage in the pipeline for various high-level tasks such as question-answering systems Mollá et al. [2006]. Recent research has explored zero-shot IE, which aims to build IE systems from unannotated text with minimal human intervention. This approach significantly reduces the time and effort needed for data labeling. Wei, Cui, Cheng, Wang, Zhang, Huang, Xie, Xu, Chen, Zhang, et al. [2023] have investigated the use of Large Language Models (LLMs), such as GPT-3 and ChatGPT, for zero-shot IE. Inspired by this work, our project also employs zero-shot methods to extract information using prompts and GPT models. By using the capabilities of GPT, we aim to efficiently extract structured data from unstructured text with minimal manual intervention.

### 2.3 Record Linkage

Record linkage, also known as computerized matching, refers to the process of identifying and matching records that correspond to the same entities (such as individuals or businesses) using quasi-identifiers like names, addresses, and dates of birth Winkler [2014]. Winkler [2014] mentions that the key factor to effective record linkage is the preprocessing of raw data, which involves standardizing and parsing the data into a format suitable for applying statistical models.

Schraagen, Marijn and others [2014] research the complexities of record linkage within historical databases. His work incorporates methods from various scientific disciplines to tackle different aspects of record linkage, such as handling name variations and exploiting information from related records within the dataset.

---

<sup>4</sup>[https://en.wikipedia.org/wiki/Tesseract\\_\(software\)](https://en.wikipedia.org/wiki/Tesseract_(software))



### 3 Data

The 'Leidse hoogleraren en lectoren 1575-1815' dataset is divided into seven volumes. Volumes 1 to 5 contain information about professors who worked in various faculties of Leiden University, while volumes 6 and 7 contain information on other administrators and staff. For a detailed overview of the information about each volume, refer to Table 1.

Table 1: Overview of Volumes in the 'Leidse hoogleraren en lectoren 1575-1815' Dataset

Volume	Subject	Number of people	Year
1	Professors at The Faculty of Theology	64	1983
2	Professors at The Faculty of Medicine	55	1983
3	Professors at The Faculty of Law	57	1984
4	Professors at The Faculty of Mathematics and Natural Sciences	40	1984
5	Professors at The Faculty of Arts	73	1985
6	Addenda	51	1985
7	Curators	62	1985

Each volume provides information about approximately 57 individuals associated with Leiden University from 1575 to 1815. The structure of each volume is consistent, including the following sections: an introduction offering a general explanation of the professors and their respective faculties, an alphabetical list of names of the individuals mentioned in the volume (detailed in Appendix A), detailed information about each person in alphabetical order (detailed in Appendix B), and a list of sources used in the volume, organized numerically (detailed in Appendix C).

The information section about each person includes the following details:

- Date and place of birth and death
- Education
- Career history
- Additional positions
- Genealogical details (e.g., spouse(s), children, parents, grandparents)
- Special details (e.g., salary, memberships)

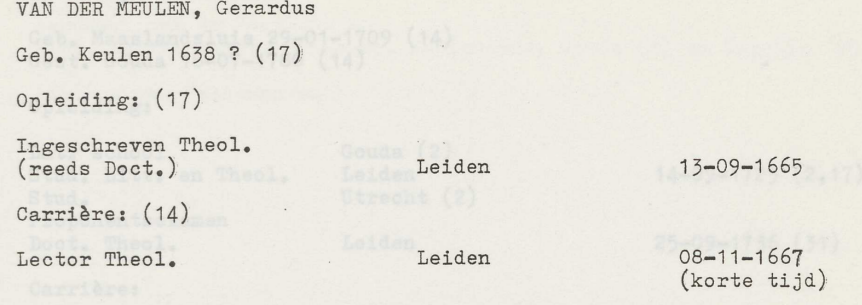
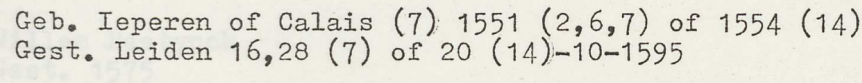
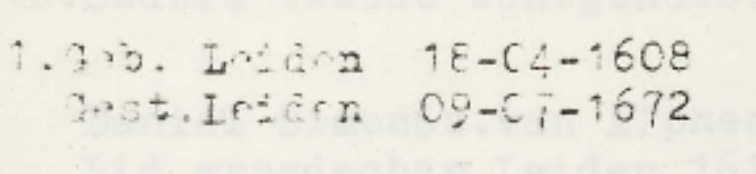
The dataset presents several inconsistencies across its volumes, which we have categorized into distinct types. One prominent issue is incomplete information, where data entries lack uniformity. While some records detail a person's birth place, birth date, and even death details, others are notably deficient in this regard. For instance, Figure 2a shows the birth year and birthplace of a person indicated after 'Geb.' ('Geb.' refers to 'Geboren', which means 'born' in Dutch.). However,

information regarding their death, which is normally mentioned after 'Gest.' ('Gest.,' short for 'Gestorven', meaning 'deceased' in Dutch), is absent.

Another challenge is the variability in formatting within the dataset, where the format of information differs significantly from one entry to another. For example, some individuals' birth dates are recorded precisely as '09-08-1673', while others vary. Figure 2b illustrates this difference, showing a birth year of either 1551 (as noted in sources 2, 6, and 7) or 1554 (as mentioned in source 14). Additionally, the death date in this example is either the 16th or 28th (according to source 7) or the 20th (according to source 14), with the death month and year being October 1559.

Furthermore, printing quality across the dataset can vary between volumes. Early editions were printed with lower quality, resulting in blurred or distorted text and images. In contrast, later volumes demonstrate significantly improved printing standards, ensuring clearer and more readable content. The differences in printing quality can impact accessibility, as presented in Figure 2c.

Table 2: Illustration of Inconsistencies in the Dataset

Inconsistency Type	Example Images
<p>Incomplete Information</p>	 <p>(a) Leidse hoogleraren en lectoren 1575-1815, Volume 1. This record demonstrates the inconsistencies in information completeness, providing details such as birthplace and birth date (mentioned after 'Geb.'), but lacking information about the person's death (mentioned after 'Gest.').</p>
<p>Variable Formatting</p>	 <p>(b) Leidse hoogleraren en lectoren 1575-1815, Volume 1. This record demonstrates the inconsistencies in data formatting, showing birth years as either 1551 (sources 2, 6, and 7) or 1554 (source 14), and death dates as the 16th, 20th, or 28th (sources 7 and 14), with a death month and year of October 1559.</p>
<p>Printing Quality</p>	 <p>(c) Leidse hoogleraren en lectoren 1575-1815, Volume 1. This record illustrates the low printing quality that might appear in the earlier editions.</p>

## 4 Methods

The research methodology comprises three phases, each addressing a sub-question to systematically and thoroughly examine the problem and evaluate it in detail. The phases are:

- **Phase 1:** PDF to PNG Conversion, Image Preprocessing, and Optical Character Recognition (OCR) & Text Segmentation Per Person
- **Phase 2:** Text to JSON Extraction with AI
- **Phase 3:** Record Linkage and Database Enrichment

These phases are designed to facilitate the accurate extraction and transformation of historical records data from scanned documents, ultimately. The process workflow diagram in Figure 1 illustrates all the steps involved in the methodology. The code used for these methods can be found in the GitHub repository.<sup>5</sup>

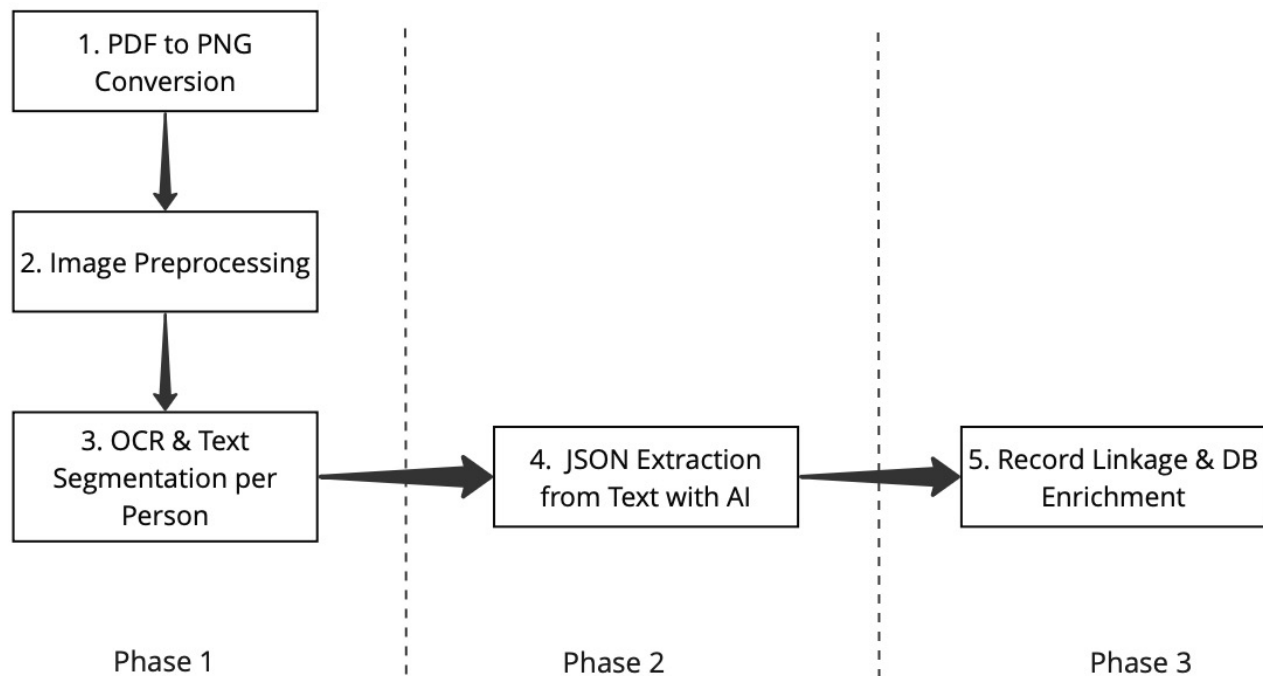


Figure 1: Process flow for the three-phase process, including text extraction, AI-based JSON generation, and record linkage for database enrichment.

### 4.1 Phase 1: PDF to PNG Conversion, Image Preprocessing, and OCR & Text Segmentation Per Person

This phase addresses the first sub-question: *How can we extract high-accuracy text from scanned historical documents using OCR techniques?* The focus is on preparing individual text files from

<sup>5</sup>GitHub repository: <https://github.com/Leiden-University-City-Lab/BantjesAdapter.git>

the original PDFs. The steps involved are:

1. **PDF to PNG Conversion:** The first step is to prepare the right format for our processing steps. We begin by downloading the PDF version of the scans from the website<sup>6</sup>. Since our intended OCR program is Tesseract, and PDF is not a compatible input format for Tesseract, conversion is necessary. The Tesseract OCR package supports various input formats, including PNG, JPEG, TIFF, JPEG 2000, GIF, WebP, BMP, and PNM<sup>7</sup>.

When choosing the conversion format for the PDF files, the decision was between a lossless and a lossy image format. Lossless means that the image compression preserves all original data, maintaining pixel-based quality even after modification or resizing. On the other hand, lossy compression removes some data during compression, potentially reducing image quality. The choice was made for the lossless image format because it preserves pixel-based quality. Among the lossless image options supported by Tesseract, TIFF, and PNG were considered. PNG format was selected for the images because PNG files are smaller compared to TIFF files. TIFF files tend to store a lot of information, while PNGs offer efficient compression without sacrificing quality<sup>8</sup>. For PDF to PNG conversion a Python module called pdf2image<sup>9</sup> is used.

2. **Image Preprocessing:** The next step is to preprocess the image. The raw scanned images might not be ideal for the OCR algorithm, and preprocessing can improve their suitability. This is especially important for historical documents, which are often degraded and cannot be accurately detected by OCR without prior processing [Badoiu et al. \[2016\]](#).

We will be using Tesseract OCR, which performs various image processing operations internally using the Leptonica library before doing the actual OCR. While most of the time Tesseract does a good job, there are cases where its built-in processing isn't sufficient, leading to a substantial reduction in accuracy<sup>10</sup>. Because we are working with scanned historical records, additional preprocessing is preferred. In Appendix D, we have included an example showing the original images from the dataset, and in Appendix E, the images after preprocessing can be seen. The preprocessing steps we plan to use are:

- (a) **Denoising:** Noise reduction is crucial in OCR due to its significant impact on the overall performance and accuracy of text recognition systems. Noise in images, such as specks, smudges, or distortions, can severely hinder the OCR software's ability to correctly identify and interpret characters, leading to a cascade of issues throughout the text processing pipeline [Badoiu et al. \[2016\]](#). Therefore, removing as much noise as possible is essential. For denoising, the OpenCV function `fastNlMeansDenoisingColored` was used with specific parameters to reduce noise while preserving the details of the image.

---

Listing 1: Denoising the images using OpenCV

---

<sup>6</sup><https://digitalcollections.universiteitleiden.nl/view/item/2078065/pages>

<sup>7</sup><https://tesseract-ocr.github.io/tessdoc/InputFormats.html>

<sup>8</sup><https://www.adobe.com/nl/creativecloud/file-types/image/comparison/tiff-vs-png.html>

<sup>9</sup><https://pypi.org/project/pdf2image/>

<sup>10</sup><https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html>

```
# Denoise image with OpenCV
denoise = cv2.fastNlMeansDenoisingColored(image, None, 5, 5, 7, 21)
```

- (b) **Grayscale:** Grayscale transforms a continuous tone image into varying shades of gray. By converting a colored image to grayscale, it can often reveal more details than the original colored version. This method is commonly applied in numerous real-time scenarios, including CCTV and traffic light cameras [Badla \[2014\]](#). During preprocessing the images, we converted the images to grayscale using OpenCV.

```
# Convert image to grayscale
gray = cv2.cvtColor(denoise, cv2.COLOR_BGR2GRAY)
```

- (c) **Binarization:** Binarization converts a grayscale image into a binary image, where each pixel is set to either black or white, usually by applying a threshold. Pixels above the threshold are turned white, while those below are turned black [Sezgin and Sankur \[2004\]](#). Choosing the right threshold is crucial since an incorrect one could either remove parts of the letters or create unclear boundaries between characters [Jindal \[2018\]](#). This technique simplifies images for tasks like text recognition and image analysis. Most modern techniques involve binarizing the image before extracting features, reducing computational demands, and allowing for more straightforward analysis methods [Boiangiu and Dvornic \[2008\]](#).

Listing 2: Binarizing the images using OpenCV

```
# Grayscale to binary with OpenCV threshold
th, image = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY)
```

3. **Optical Character Recognition (OCR) & Text Segmentation Per Person:** Extracting text from images and segmenting the text by individual persons to ensure clarity and accuracy. This step prepares individual text files that will be used in the next phase.

**Optical Character Recognition (OCR):** After properly preprocessing the images, the next step is to extract the text using the OCR tool. For this, Tesseract OCR was chosen. Originally developed at HP Labs between 1984 and 1994, Tesseract has evolved into a powerful open-source tool for extracting text from images [Smith \[2007\]](#).

Tesseract offers great flexibility with a large number of parameters that can be tweaked to optimize performance<sup>11</sup>. Experimenting with these parameters allows adaptation to different fonts, styles, and document layouts. Here are the essential aspects considered:

- **Training:** We trained Tesseract using a dataset created from screenshots of different lines of text from our dataset. For each screenshot, we generated a corresponding .txt file with the correct text. After training, a trained language data file, `.traineddata` was created to support the recognition of specific language elements and improve overall accuracy. However, the documentation notes that retraining Tesseract may not improve

---

<sup>11</sup><https://tesseract-ocr.github.io/tessdoc/tess3/ControlParams.html>

results unless a very unusual font is used<sup>12</sup>.

- **Language Configuration:** Tesseract can recognize text in multiple languages. Specifying the language(s) or script(s) is crucial for accurate recognition, as it adjusts the character set and dictionary accordingly. In this case, the language was set to Dutch.
- **Page Segmentation Modes (PSM):** Tesseract provides 14 different PSMs, each optimized for various layout scenarios. Choosing the right PSM ensures better text extraction results. For this project, we set the PSM value to 4.

**Text Segmentation Per Person:** To manage and track errors effectively, the text is split into separate .txt files for each person mentioned in the book. This approach allows for better control over the files. An algorithm using regular expressions in Python was developed for this purpose, based on specific characteristics of the books:

- The last name of each person is the only word in capital letters, and it appears on the first line of the page.
- Each new person starts on a new page, although not every page contains information about a new person.
- Each image corresponds to one page.
- Only personal images regarding the person's information are used; other images, such as sources or covers, have already been removed as explained in section 3.

The algorithm which is used in this step is shown in figure 2 and it operates as follows:

- (a) Sorts the PNG files numerically based on the page number in the filename.
- (b) Iterates through each image in the sorted list.
- (c) Performs OCR on the image to extract text.
- (d) Checks if the first few lines of the OCR text start with a last name (a sequence of three or more capital letters).
- (e) If a last name is found, it creates a new .txt file for the person to save the information there and then continues with the next PNG.
- (f) If no last name is found, it does not need to create a new .txt file. But it has to save the information in the latest .txt file made and then continue with the next PNG.
- (g) Once it has iterated through all the PNGs, the algorithm will stop.

See Appendix F for an example of a generated .txt file using this algorithm.

## 4.2 Phase 2: JSON Extraction from Text with AI

After creating separate text files for each individual, the next step is to extract all relevant information from these texts into a structured format. We chose JSON as our structured format because it is a key-value style lightweight data exchange format. Its simplicity makes JSON easy for humans

---

<sup>12</sup><https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html>

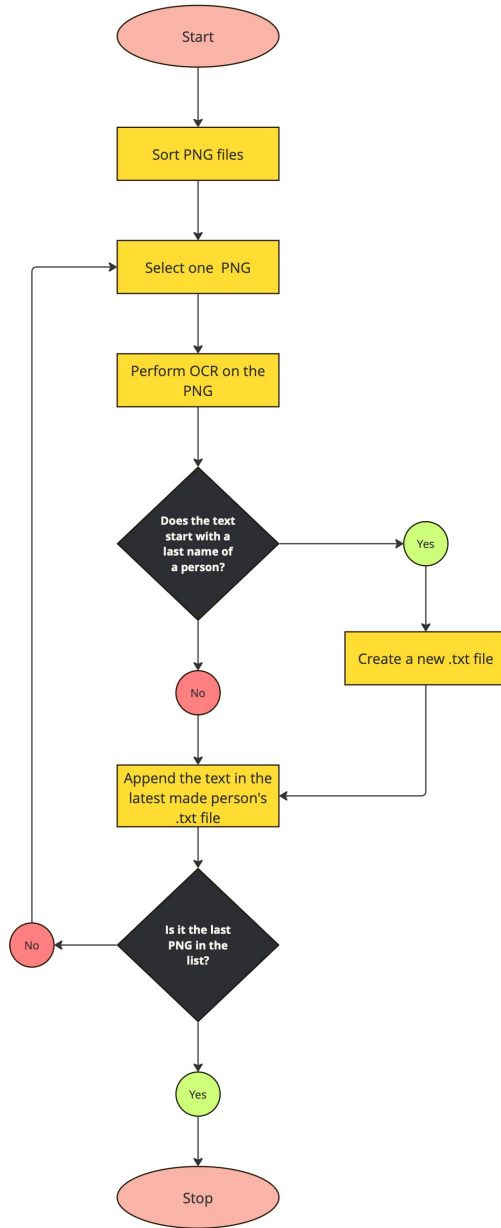


Figure 2: Algorithm for extracting and saving text per person from scanned book images using OCR.



to read and write, and for computers to generate and parse Peng et al. [2011].

Initially, we explored using regular expressions in Python to extract information following specific titles such as 'Opleiding' (Education) and 'Loopbaan' (Career) from the text. However, early experiments revealed significant challenges due to OCR-related errors. These errors made it difficult to reliably capture and process data using regular expressions. Therefore, we transitioned to using an AI model which is explained in the following sections. For further details on our initial experimentation with regular expressions, refer to Appendix I, where the corresponding code is documented.

#### 4.2.1 Schema Definition

In Section 3, we described the types of information present in our dataset. To generate a consistent output that contains all the necessary fields, we need to define a schema for our model. Given the complexity of our desired output, which involves highly nested and extensive JSON files, this task is challenging. To address this, we use Pydantic<sup>13</sup>, a Python library that allows us to define the keys of our JSON as classes.

Pydantic provides data validation through Python type annotations. It simplifies the handling of JSON responses, ensuring they match the specified schema without requiring intricate error-checking code. Additionally, Pydantic can automatically generate JSON schemas based on the defined models<sup>14</sup>. This approach significantly reduces complexity and makes the structure more manageable. The complete JSON schema for the output JSONs can be found in our GitHub repository<sup>15</sup>.

Below is an example of a Pydantic class from our schema, which is used to extract information related to the careers of the scholars in our dataset.

Listing 3: Example Pydantic Class for Career Information

```
class Career(BaseModel):
    """Identifying information about the persons career."""
    job: Optional[str] = Field(None, description='The type of job',
                               examples=['Hoogleraar Geschiedenis'])
    location: Optional[str] = Field(None, description='The location of the job',
                                     examples=['Leiden'])
    date: Optional[str] = Field(None, description='The date of the job.',
                                 examples=['1601-10-20', '1601'])
    source: Optional[str] = Field(None, description='The source of the info mentioned
                                                    in parentheses', examples=['6'])

class Person(BaseModel):
    FirstName: str = Field(..., description="The first name of a person",
                           examples=['Cornelis', 'Johannes'])
```

<sup>13</sup><https://pydantic.dev>

<sup>14</sup>[https://docs.pydantic.dev/latest/concepts/json\\_schema/](https://docs.pydantic.dev/latest/concepts/json_schema/)

<sup>15</sup>[https://github.com/Leiden-University-City-Lab/BantjesAdapter/blob/main/AI/json\\_schema.json](https://github.com/Leiden-University-City-Lab/BantjesAdapter/blob/main/AI/json_schema.json)

```

LastName: str = Field(..., description="The last name of a person",
    examples=['EKAMA'])
BirthDate: Optional[str] = Field(None, description="Birth date, Usually found
    after Geb.", examples=['1601-10-20', '1601', '1601-10'])
careers: List[Career]

```

## 4.2.2 Extraction Techniques

To extract the information, we use the GPT-3.5 Turbo model from OpenAI platform<sup>16</sup> as our Large Language Model(LLM). GPT-3.5 Turbo's function calling feature enables us to define functions in an API call and have the model intelligently output a JSON object with the necessary key-value pairs. Instead of executing the function, the Chat Completions API produces JSON that can be used to call the function in our code. This approach ensures that structured data is returned more reliably<sup>17</sup>. OpenAI's function calling capabilities, combined with Pydantic, allow for a more structured and reliable approach to output parsing. Pydantic's data models are defined with familiar Python-type annotations, making them intuitive and easy to use. Below is the GPT prompt and code used to generate the JSON objects:

Listing 4: GPT prompt for extracting structured data from OCR text using GPT-3.5 Turbo

```

def chat_completion(person_info):

    return client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
            {
                "role": "system",
                "content": '''You are an advanced data extraction system.
                    - You can identify each person by surname
                    - The surname is always in uppercase letters, followed by
                      the middle and/or first name
                    - If you can't determine the field value, refer to the
                      examples'''
            },
            {
                "role": "user",
                "content": f'Please extract the data for the following person:
                    {person_info}'
            }
        ],
        response_model=Person,
        max_retries=1,
        tool_choice="auto"
    )

```

<sup>16</sup><https://platform.openai.com/docs/models/gpt-3-5-turbo>

<sup>17</sup><https://platform.openai.com/docs/guides/function-calling>

---

In the prompt shown above, the `model` attribute specifies which version of the GPT model to use for generating responses. In this example, we use `gpt-3.5-turbo`. The `messages` attribute, a list of dictionaries, defines the conversation context, where the `system` message provides instructions and the `user` message contains the input data, including guidelines and examples for data extraction. The `response_model` attribute ensures the output adheres to the defined schema using a Pydantic model, in this case, the `Person` class. The `tool_choice` attribute, set to `auto`, controls the model’s decision to either generate a message or call a specified function. The `person_info` contains OCR-obtained input text about individuals, passed through the `user` message content. Lastly, the `max_retries` attribute is set to 1, specifying the maximum number of retry attempts if the API call fails. Appendix G shows an example of a generated JSON file using this technique.

### 4.3 Phase 3: Record Linkage & Database Enrichment

After obtaining the JSON files containing the extracted information, the next step involves enriching the centralized database with this data. The centralized database, developed by the LUCD project, already contains high-quality data on professors and students associated with Leiden University. However, the original database structure is insufficient to encapsulate all our data. Therefore, we modified the database schema by adding necessary tables and columns to accommodate the new information from our JSON files.

Figure 3 shows the Entity-Relationship (ER) diagram of the updated database version. The added tables are ‘education’, ‘career’, and ‘particularity’. We have added three new columns to the ‘person’ table: ‘AlternativeLastName’, ‘rating’, and ‘faculty’. The ‘rating’ column helps differentiate between newly added and original data, using a system of three ratings. Original data is given a rating of 3 for high quality, a rating of 2 is assigned if additional data matches an existing entity, and a rating of 1 is for entirely new entities. This approach allows us to effectively distinguish data quality levels, filter the newly added columns accordingly, and check the newly added data for possible faults or misspellings. If we find a linked record in the database with a high score, we will not overwrite the existing data of this instance. Instead, we will enrich this entity with new data if the corresponding tables are empty.

#### 4.3.1 Linking Algorithm

In implementing record linkage, quasi-identifiers play a crucial role. Quasi-identifiers are attributes like names, addresses, and dates of birth that can be combined to uniquely identify a person [Winkler \[2014\]](#).

To compare the values between two fields, we allow for partial matches. This technique ensures that first names such as ‘Casper Janszoon’ and ‘Casper Johannes’ are considered a match. The partial matching method used involves breaking down each name into substrings based on white spaces and checking if any part of the first name or last name appears in the corresponding field of the database records.

When linking two records together, we assess them against two specific conditions. If either



condition is satisfied, we consider the records as matching and proceed to integrate any additional information from our JSON files into these records. The conditions are as follows:

**First condition:**

- First name and last name match.
- Either the birth year is the same or the birth city matches.

**Second condition:**

- Last name matches.
- Birth year matches.
- Either the birth city or the birth country matches.

These conditions were chosen because they use identifiers that are present for most people in our dataset. Defining more matching conditions could result in fewer matches, as not all the people in our dataset may have all the required information.

We also introduce a condition for uncertain matches, where the data does not perfectly align but there is still a potential match. For these cases, the algorithm identifies records where the birth year and birthplace do not match, but the names match. If such a match is found, we create a new person in the database but introduce a relation between this newly added person and the person from the database who we thought might be the same individual. This approach allows us to easily identify uncertain cases and review them further.

## 5 Evaluation

We adopted a systematic evaluation approach to assess the performance of each phase in our methodology, ensuring a thorough analysis of our strategy’s effectiveness in achieving the expected results at every step. Conducting our assessment per phase allowed us to address each research subquestion effectively and gain insights into the performance of individual components of our methodology.

For our evaluation, we selected a sample comprising 10% of the total number of individuals in the dataset. This sample size was chosen to represent a diverse range of data across all volumes. By evaluating this sample, we aimed to draw meaningful conclusions about the overall performance of each phase in our digitization, information extraction, and enrichment process.

### 5.1 Phase 1 Evaluation: Quality Assessment of Generated Text

In our evaluation of Phase 1, we assessed the quality of the generated text. To ensure a comprehensive evaluation, we created a labeled dataset comprising 10% of the total text files from all volumes. This labeled dataset served as our ground truth for comparison with the OCR-generated text.

In evaluating the quality of the text recognition system, Character Error Rate (CER) and Word Error Rate (WER) have been chosen as the primary metrics. These metrics are widely accepted in the scientific community for assessing OCR systems, as they provide a clear and quantifiable measure of errors at both the character and word levels [Leifert et al. \[2019\]](#).

In contrast, metrics such as precision, recall, and F-measure, commonly used in information retrieval through the bag-of-words (BOW) model, were considered less suitable for this context. The BOW model suffers from several drawbacks: with the main drawback being that the model does not consider the reading order of words, failing to penalize permutations of recognized words [Leifert et al. \[2019\]](#).

Given this context, CER and WER have been used to provide a comprehensive evaluation of the OCR system.

To calculate these metrics, an algorithm based on the Levenshtein distance has been implemented. The algorithm processes pairs of files: one from the OCR-generated text files and one from the correct text files that we manually labeled. It computes the edit distance between the two texts and normalizes it by the length of the reference text, thus obtaining CER and WER.

The steps involved in the algorithm are as follows:

- Normalization: Both texts are stripped of leading and trailing whitespaces and converted to lowercase to ensure consistency in comparison.
- Levenshtein Distance Calculation: The edit distance is calculated using a dynamic programming approach that fills a matrix with the minimum number of edits (insertions, deletions, substitutions) required to transform one string into another.

- Metric Computation: Both CER and WER are derived from the Levenshtein distance. CER and WER normalize this distance by dividing the total number of edits by the number of characters and words in the reference text, respectively, providing a standardized error rate.

CER, defined as the ratio of the sum of insertions, deletions, and substitutions to the total number of characters in the reference text, offers an inverted measure of character accuracy. The CER is calculated as follows<sup>18</sup>:

$$\text{CER} = \frac{S_c + D_c + I_c}{N_c} \quad (1)$$

where:

- $S_c$  is the number of character substitutions,
- $D_c$  is the number of character deletions,
- $I_c$  is the number of character insertions,
- $N_c$  is the total number of characters in the reference.

WER extends this concept to the word level, providing insights into how well entire words are recognized. The WER is calculated as follows<sup>19</sup>:

$$\text{WER} = \frac{S_w + D_w + I_w}{N_w} \quad (2)$$

where:

- $S_w$  is the number of word substitutions,
- $D_w$  is the number of word deletions,
- $I_w$  is the number of word insertions,
- $N_w$  is the total number of words in the reference.

The evaluation involved comparing the OCR-generated text with our labeled dataset to compute the Word Error Rate (WER) and Character Error Rate (CER). We then averaged these rates for each volume. The results are presented in Figure 4.

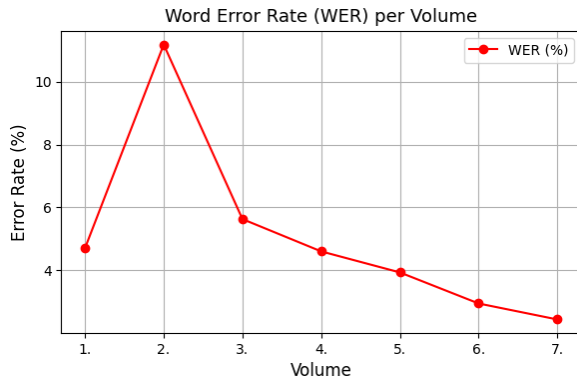
The WER is consistently higher than the CER across all volumes. This difference is expected because WER measures errors at the word level, which inherently accumulates more errors than CER, which measures errors at the character level. A single word error in WER can correspond to multiple character errors in CER, leading to higher overall WER values.

Among these volumes, volume 2 exhibits significantly higher error rates than the others, with WER and CER both notably elevated. This suggests that the OCR system encountered more difficulties

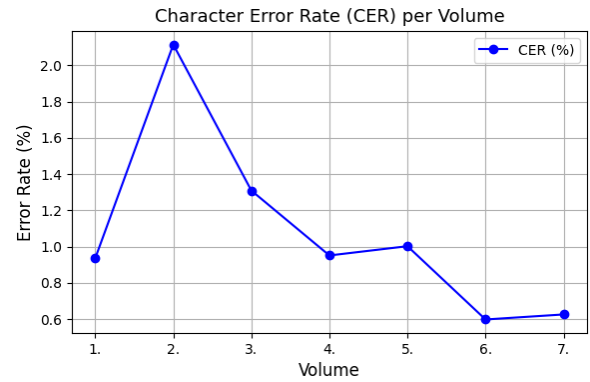
---

<sup>18</sup>[https://en.wikipedia.org/wiki/Word\\_error\\_rate](https://en.wikipedia.org/wiki/Word_error_rate)

<sup>19</sup><https://docs.kolena.com/metrics/wer-cer-mer>



(a) Average Word Error Rate (WER) per Volume.



(b) Average Character Error Rate (CER) per Volume.

Figure 4: Comparison of Error Rates

with the text in this particular volume. The primary factor contributing to the poorer performance observed in volume 2 is likely the quality of printing. Poor print quality can introduce noise and distortions in the text, making it harder for the OCR system to accurately recognize characters and words. For instance, faded ink, smudges, or uneven text alignment can significantly impact recognition accuracy. Figure 5 to 11 show images from all volumes, illustrating the varying printing quality.

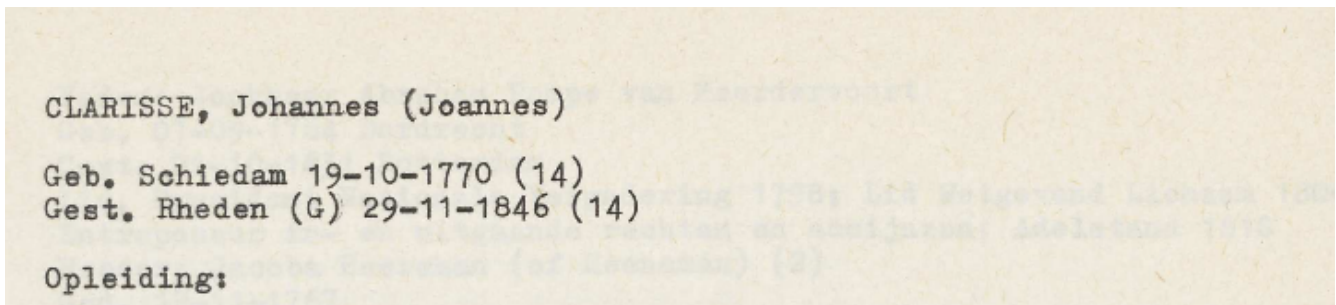


Figure 5: An image showing the printing quality of Volume 1

## 5.2 Phase 2 Evaluation: Quality Assessment of Generated JSON

In Phase 2 of our evaluation, we aimed to assess the quality of the JSON files created by our AI model. For 10% of the individuals in our dataset, we manually created correct JSON files that align with our predefined schema. These files were checked against the original images to ensure their accuracy.

To assess the impact of OCR-related errors and text quality on the generated JSON files, we created two sets of JSON files for accuracy evaluation using GPT-3.5 Turbo as our AI model. The first set used manually corrected text inputs. For each text file, we generated five JSON files to calculate average accuracy, ensuring a comprehensive assessment and avoiding non-representative outcomes.



FALCOBURGUS (VAN VALKENBURG), Adrianus

1. Geb. Leiden 01-11-1581 (19) (19)  
 Gest. Leiden 1650 (19) (19)

2. Opleiding:

† Stui. Phil., Geschiedenis, Leiden (2)  
 Theol. en Med.

Figure 6: An image showing the printing quality of Volume 2

VAN (DER) NIEU(W)STAD(T)(NEOSTADIUS), Cornelis

1. Geb. Brielle 1 Jun van 1549 (2) (2)  
 Gest. Den Haag 25-06-1606 (2) (2)

2. Opleiding:

- Stud. Iur. Douai (2)

Figure 7: An image showing the printing quality of Volume 3

SNELLIUS (SNEL VAN ROYEN), Rudolph

1. Geb. Oudewater 08-10-1546 (13) (13)  
 Gest. Leiden 01-03-1613 (13) (13)

2. Opleiding:

- Onderwijs Utrecht (3) (3)

Figure 8: An image showing the printing quality of Volume 4

VAN DER PALM, Johan Hendrik

1. Geb. Rotterdam 17-07-1763 (27) (27)  
 Gest. Leiden 08-09-1840 (27) (27)

2. Opleiding: (52)

Figure 9: An image showing the printing quality of Volume 5

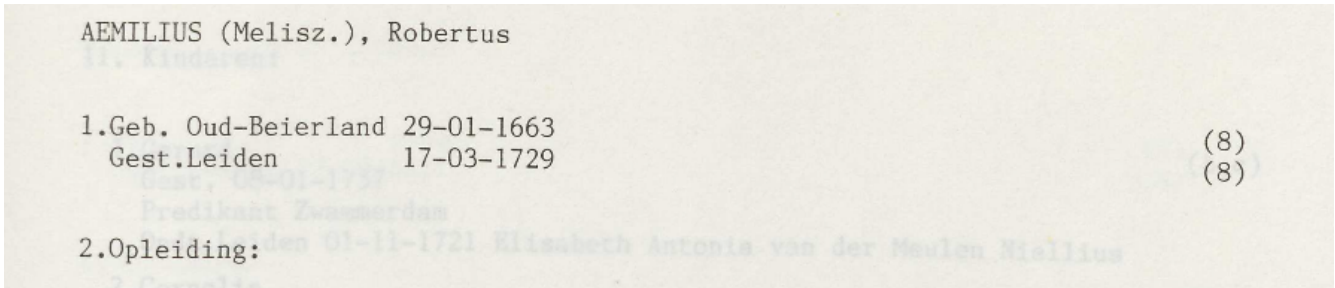


Figure 10: An image showing the printing quality of Volume 6



Figure 11: An image showing the printing quality of Volume 7

The second set was derived from OCR-generated text inputs. For each text file in this set, we also generated five JSON files. Here, the objective was to gauge accuracy considering potential spelling errors and OCR inaccuracies.

To evaluate these JSON files, we conducted a comparison between the AI-generated JSON files and the correct JSON files:

1. **Normalization by lowercasing:** To ensure a consistent comparison, we converted all text values to lowercase. This step helped in making the comparison case-insensitive, thereby focusing only on content accuracy rather than case differences.
2. **Value comparison:** We compared each key-value pair in the AI-generated JSON files with the corresponding pair in the correct JSON files. This comparison was performed separately for two sets of AI-generated JSON files: those based on Correct Text Files and those based on OCR-Generated Text Files.
3. **Accuracy assessment:** For each key, we calculated the accuracy by determining the percentage of correct values generated by the AI.
4. **Key categorization:** Keys in the JSON files were categorized into meaningful groups such as 'Main person', 'Education', 'Careers', 'Particularities', 'Spouses', 'Parents', 'Grandparents', 'In-laws', 'Children', and 'Far family'. This categorization allows us to perform a focused analysis of accuracy and helps us detect the error-prone areas in our JSON files.

Below is a comparison example of a small section from JSON files, focusing on keys related to the 'Main person' category. The first JSON is manually created and therefore is correct, while the

second JSON is generated using our AI model. In our dataset, the `type_of_person` attribute, set to 1, denotes that the individual is classified as a professor. Other types of persons that appear in the database include students and curators.

Listing 5: Example of a correct JSON file containing the key-value pairs related to the 'Main person' category

```
{
  "first_name": "Caspar Janszoon",
  "last_name": "COOLHAES",
  "affix": null,
  "gender": "Man",
  "alternative_last_names": ["KOOLHAES", "KOOLHAAS", "COELAES"],
  "type_of_person": 1,
  "faculty": "Theologie",
  "birth_country": "Duitsland",
  "birth_city": "Keulen",
  "birth_date": "1534-01-24",
  "death_date": "1615-01-15",
  "death_city": "Leiden"
}
```

Listing 6: Example of a JSON file, made using our AI model, containing the key-value pairs related to the 'Main person' category

```
{
  "first_name": "Caspar Janszoon",
  "last_name": "COOLHAES",
  "affix": null,
  "gender": "Man",
  "alternative_last_names": [],
  "type_of_person": 1,
  "faculty": "Theologie",
  "birth_country": null,
  "birth_city": "Keulen",
  "birth_date": "1534",
  "death_date": "1615",
  "death_city": "Leiden"
}
```

Table 3 shows the comparison results of the two JSON files above. The 'Accuracy' score represents the percentage of correct values generated by the AI for each key in the JSON file. For example, the `first_name` key has an accuracy of 100.00%, indicating that the AI correctly generated the first name in all instances. In our evaluation, if the AI extracts '1615' but the correct complete date is '1615-01-15', we consider it inaccurate even if the year information matches. This approach ensures precision in our accuracy assessment by mandating exact date formats where necessary.

Category	Key	Accuracy
Main Person	first_name	100.00% (1/1)
Main Person	last_name	100.00% (1/1)
Main Person	affix	100.00% (1/1)
Main Person	gender	100.00% (1/1)
Main Person	alternative_last_names	0.00% (0/3)
Main Person	type_of_person	100.00% (1/1)
Main Person	faculty	100.00% (1/1)
Main Person	birth_country	0.00% (0/1)
Main Person	birth_city	100.00% (1/1)
Main Person	birth_date	0.00% (0/1)
Main Person	death_date	0.00% (0/1)
Main Person	death_city	100.00% (1/1)

Table 3: Example of accuracy scores for the main person category in the JSON file.

Category	Average accuracy of JSON files made using correct text files	Average accuracy of JSON files made using OCR-generated text files
Main person	73.53%	72.29%
Education	68.29%	63.22%
Careers	66.84%	64.05%
Particularities	58.34%	53.05%
Spouses	63.23%	61.85%
Parents	70.13%	67.48%
Grandparents	66.09%	57.33%
In-laws	54.46%	59.16%
Children	69.61%	66.53%
Far family	59.85%	62.27%
Total	65.04%	62.72%

Table 4: Overall accuracy results per category for JSON files generated from correct and OCR-generated text files.

Table 4 displays the overall results per category, highlighting the differences in accuracy scores between JSON files generated using correct text files and those generated using OCR text files.

The average accuracy scores are generally higher across most categories for JSON files generated using correct text files compared to those created using OCR-generated text files. This difference is due to the inherent faults in the OCR-generated text, such as spelling errors or misplaced recognized words. These inaccuracies in the OCR text can impact the AI’s ability to generate correct JSON files, resulting in lower accuracy scores. Additionally, the total accuracy is higher for JSON files generated using correct text files, reflecting the overall better performance with accurate inputs.

In both sets of JSON files, the 'Main person' category, which consists of the keys shown in table 3 shows the highest accuracy scores. This high accuracy might be because this information primarily appears at the top of each page and thus at the beginning of each text file. The uniformity of these personal details makes it easier for the AI to accurately extract and generate the corresponding JSON values.

In contrast, the 'In-laws' category in the JSON files generated using correct text files has the lowest accuracy among all categories. This lower accuracy could be due to the variability and less consistent format of the information in this category, making it more challenging for the AI to accurately extract and generate the corresponding values. By variability, we mean that in some cases, the father and mother-in-law are mentioned within the information about the spouses, as shown in Figure 12. In other cases, they have their dedicated section, as shown in Figure 13. This inconsistency in format makes it difficult for the AI to consistently recognize and categorize the information correctly.

In the scenario shown in Figure 12, the AI may handle the information in several ways: 1) it may not include the in-law information at all, 2) it might place it under the 'In-laws' section, or 3) it could categorize it under 'Far family'. However, in our evaluation, we only consider the first approach as accurate. This is because the correct JSON files, manually created, set the in-laws values to null if not explicitly mentioned under 'ouders echtgenotes'. This scenario likely contributes to the low accuracy scores observed in these categories.

Furthermore, a possible reason why the 'In-laws' and 'Far family' categories scored lower in JSON files generated from correct text compared to those made using OCR-generated text could be due to the higher incidence of inaccuracies in OCR text. JSON files generated from OCR text, which often contain more errors, may lead the AI to leave the 'In-laws' and 'Far family' values empty. In contrast, JSON files made based on correct text, with fewer errors, may enable the AI to extract more of these relationships which are not necessarily correct.

Additionally, in the JSON files generated using OCR text files, the 'Particularities' category shows significantly lower accuracy scores. This category is particularly affected by OCR errors, which challenges the AI's performance.

### 5.3 Phase 3 Evaluation: Quality Assessment of Linking Algorithm

To evaluate the enrichment algorithm, we used sample data consisting of two sets of JSON files. The first set was generated using OCR text, and the second set was manually created to ensure correctness. By comparing these two sets of JSON files, we can better understand the impact of potential OCR errors and the errors introduced by the AI model on the record-linking process.

To measure the performance, we executed our algorithm on these two sets of JSON files and calculated the results for each volume. Below is the schema of the JSON file created specifically for our evaluation, which contains information on the performance of our database enrichment algorithm. Each JSON file contains objects corresponding to an individual from our dataset. If the linking algorithm finds a person in the database, the `new_person` attribute is set to false, and

Echtgenotes:

1. Anna Emerentia Musenhole (Muysenhol) (6,a)

Getr. Frankfurt a/d Main 1588 (a)

Gest. 1592 (54)

Vader: Gilles Muysenhol uit Antwerpen (a)

2. Jonkvrouwe Maria L'Hermite

Getr. Frankfurt a/d Main zomer 1593 (a)

Gest. 1621 (8)

Vader: Simon l'Hermite, Schepen Antwerpen (adellijk); Moeder: Johanna de

3. Anna Maria la Noye (Lannoy, de Lannoy) (6,a,54) Splijtere (a)

Getr. Middelburg 1622 (a)

Figure 12: Leidse hoogleraren en lectoren 1575-181, Volume 1, Page 44. The image shows an example in which the names of the in-laws are mentioned within the spouse's section (in Dutch, 'echtgenotes').

10.Ouders Echtgenote:

Thomas Siegfried Ring(s), (1644-1707),  
hoogleraar Rechten Frankfurt/Oder

(2,a)

Suzanna Maria von Scholtz von Hermensdorff

(a)

Figure 13: Leidse hoogleraren en lectoren 1575-181, Volume 2, Page 11. The image shows an example in which the father and mother-in-law have their sections in the document under 'ouders echtgenotes', meaning parents of spouses.

the `person_id` represents the unique ID of the identified person from the database. If no person is found, a new person will be added to the database and therefore the `new_person` attribute is set to true, and the `person_id` shows the ID of the newly added record. The `maybe_same_person` attribute is set to true when there is insufficient evidence for the linking algorithm to match two records definitively but identifies a potential match. In such cases, a new record is created under the ID specified by `person_id`.

Listing 7: JSON schema for evaluating database enrichment

```
{
  "name_of_file": {
    "person_id": int,
    "new_person": bool,
    "maybe_same_person": bool
  }
}
```

The accuracy results of the linking algorithm using correct JSON files are shown in Table 5, and Table 6 shows the results using OCR-generated JSON files. The 'Person\_ID Accuracy' measures how often the algorithm correctly identifies existing individuals. Using correct JSON files, accuracy ranges from 71.43% to 100%. With OCR-generated files, accuracy drops, ranging from 57.14% to 100%, indicating OCR errors impact the algorithm's ability to correctly identify individuals. The 'New\_Person Accuracy' measures how accurately the algorithm identifies new individuals. Using correct JSON files, accuracy is generally high (85.71% to 100%). With OCR-generated files, accuracy is lower, ranging from 57.14% to 100%. The 'Maybe\_Same\_Person Accuracy' measures how often the algorithm correctly flags uncertain matches. This metric consistently shows high accuracy (75% to 100%). The comparison shows the impact of OCR errors on the linking algorithm. While the algorithm performs well with manually created JSON files, its accuracy decreases with OCR-generated files, particularly for identifying existing individuals and new entries.

Volume	Accuracy Person_ID	Accuracy New_Person	Accuracy Maybe_Same_Person	Accuracy Average
Volume 1	71.43%	85.71%	100%	85.71%
Volume 2	80%	80%	100%	86.67%
Volume 3	100%	100%	100%	100%
Volume 4	100%	100%	100%	100%
Volume 5	87.5%	87.5%	100%	91.67%
Volume 6	100%	100%	75%	91.67%
Volume 7	100%	100%	100%	100%
<b>Total</b>	91.28%	93.32%	96.43%	93.67%

Table 5: Accuracy scores of the linking algorithm using correct JSON files created manually

Table 7 compares the counts of new persons generated by the linking algorithm across different volumes when using correct JSON files versus OCR-generated JSON files. The table includes the number of instances in each volume, the correct count of new persons that should have been created,

Volume	Accuracy Person_ID	Accuracy New_Person	Accuracy Maybe_Same_Person	Accuracy Average
Volume 1	57.14%	57.14%	100%	71.43%
Volume 2	60%	60%	80%	66.67%
Volume 3	83.33%	83.33%	100%	88.89%
Volume 4	66.67%	66.67%	100%	77.78%
Volume 5	62.5%	62.5%	100%	75%
Volume 6	100%	100%	75%	91.67%
Volume 7	100%	100%	85.71%	95.24%
<b>Total</b>	75.66%	75.66%	91.53%	80.95%

Table 6: Accuracy scores of the linking algorithm using JSON files made from OCR-generated text files

and the actual counts generated by the algorithm for both sets of JSON files.

In volumes 6 and 7, a significantly higher number of new people were generated compared to volumes 1 to 5. This difference is because the central database only contains information on professors and students of Leiden University while volumes 6 and 7 contain data on curators of Leiden University. As a result, the linking algorithm was unable to find matches for curators because they were not present in the centralized database, thus identifying more new individuals in these volumes.

Volume	Instance Count	Correct New_Person Count	Generated New_Person Count (Correct JSON)	Generated New_Person Count (OCR JSON)
Volume 1	7	0	1	3
Volume 2	5	0	1	2
Volume 3	6	1	1	2
Volume 4	3	1	1	2
Volume 5	8	2	3	5
Volume 6	4	3	3	3
Volume 7	7	7	7	7

Table 7: Comparison of New Person Counts between Correct JSONs vs. the JSON file made using OCR-generated text files



## 6 Discussion

In this study, our focus has been on enhancing the quality of OCR-generated text, optimizing information extraction using AI models, and refining the performance of our record linkage algorithm. However, several persistent challenges continue to impact the accuracy and comprehensiveness of our results, highlighting the need for ongoing refinement.

Despite efforts to enhance OCR text quality, residual issues impact downstream processes, particularly in JSON generation. Variability in page layouts across our dataset poses a challenge. Instances, where information from separate columns is interpreted as a single line, disrupt context continuity and coherence, potentially leading to inaccuracies in JSON outputs.

During information extraction, we employed a uniform JSON schema across all volumes, assuming document structure consistency. However, subtle variations between volumes, such as different terminologies ('Carrière' vs. 'Loopbaan' for job information), suggest that volume-specific JSON schemas might enhance model accuracy. Tailoring schemas to unique volume characteristics could improve the contextual relevance and reliability of extracted data.

In some cases, certain information appears in the data that we have not considered as separate JSON keys. For instance, a person's salary is sometimes mentioned in the 'Particularities' section ('Bijzonderheden' in Dutch). The reason for not including these as fixed JSON keys is that they are not consistently mentioned for all individuals, which would result in many null values. However, these specific details can be added to the JSON schema if they frequently appear and are deemed necessary.

The linking algorithm in this study uses specific criteria such as first name, last name, birth city, and birth year to link records. Our approach employs partial matching for names to accommodate variations within names. While first names such as 'Casper Janszoon' and 'Casper Johannes' are considered a match, it does not accommodate variations such as 'Coolhaas' vs. 'Koolhaes' due to stricter matching rules. This decision was made to reduce the number of incorrect matches while maintaining precision. Implementing techniques like Levenshtein distance could enhance the algorithm's flexibility, allowing for less strict matching criteria and potentially reducing false negatives.

For our evaluation, we selected a sample consisting of 10% of the total people in the dataset, amounting to 40 individuals. We did not choose a larger sample because manually creating the ground truths for the evaluations was an exhaustive and time-consuming task. However, from a statistical standpoint, to achieve a confidence level of 95% with a 5% margin of error, the ideal sample size for a population of 400 individuals would be 196. This represents approximately 49% of the total population. Calculations were performed using an online sample size calculator, which takes into account the population size, margin of error, and desired confidence level<sup>20</sup>.

---

<sup>20</sup><https://www.qualtrics.com/uk/experience-management/research/determine-sample-size/?rid=ip&prevsite=en&newsite=uk&geo=NL&geomatch=uk>

## 7 Further Work

This study introduces an approach to digitizing historical data using OCR and AI technologies. Moving forward, several avenues for future research can improve the sophistication and efficiency of these methodologies. Future research could explore using advanced multi-modal AI models, such as GPT-4o, developed by OpenAI<sup>21</sup>. GPT-4o can handle text and image inputs to generate structured text outputs, making it suitable for automated data extraction from historical documents. By using GPT-4o's capabilities, there is potential to simplify workflows and improve accuracy in generating JSON-formatted outputs directly from document images, potentially eliminating OCR steps.

Additionally, the study identified challenges in the current linking algorithm, particularly in accommodating variations within names. Implementing techniques like Levenshtein distance could enhance the algorithm's flexibility, allowing for less strict matching criteria and potentially reducing false negatives.

Future research should also consider the use of volume-specific JSON schemas to account for subtle differences in document structures across volumes.

Furthermore, exploring prompt engineering could lead to better results in information extraction. By refining and experimenting with different prompts, future research could optimize the performance of AI models in generating accurate and comprehensive JSON outputs.

---

<sup>21</sup><https://platform.openai.com/docs/models/gpt-4o>

## 8 Conclusion

To answer the main research question, 'How can we accurately extract and transform historical records data from scanned historical documents and map it into a centralized database?' we introduced a systematic 3-phase methodology, each designed to answer specific sub-questions.

Our first sub-question was: 'How can we extract high-accuracy text from scanned historical documents using OCR techniques?' We began by converting PDF files to PNG format to facilitate compatibility with Tesseract OCR. Subsequent image preprocessing steps included denoising, grayscale conversion, and binarization to optimize OCR accuracy for historical documents. The Tesseract OCR engine was then employed to extract text from the preprocessed images.

In the second phase, our attention turned to structuring the data into JSON format, addressing the sub-question: 'How can AI be used to analyze the OCR-generated text and obtain a structured format?' Using Pydantic, a Python library, we defined schema models to ensure consistency and facilitate data validation. Information about each individual was structured into nested JSON objects. Integration with OpenAI's GPT-3.5 Turbo model enabled intelligent function calling, providing structured JSON outputs aligned with our schema definitions.

In the final phase, we focused on database enrichment and record linkage, answering the sub-question: 'How can we map the structured data into a centralized database?' We enriched a centralized database originally developed by the LUCD project with extracted JSON data. Modifications to the database schema introduced new tables ('education', 'career', 'particularity') and columns ('AlternativeLastName', 'rating', 'faculty') to encompass comprehensive dataset attributes. The record linkage algorithm used quasi-identifiers such as names and birth details, incorporating partial matching conditions to accurately link records across datasets.

## References

- Sahil Badla. Improving the efficiency of tesseract ocr engine. 2014.
- Vlad Badoiu, Andrei-Constantin Ciobanu, and Sergi Craitoiu. Ocr quality improvement using image preprocessing. *Journal of Information Systems & Operations Management*, page 1, 2016.
- Costin-Anton Boiangiu and Andrei Iulian Dvornic. Methods of bitonal image conversion for modern and classic documents. *WSEAS Transactions on Computers*, 7(7):1081–1090, 2008.
- Khushneet Jindal. *Optical Character Recognition of Machine Printed Dogri Language Documents*. PhD thesis, 2018.
- Gundram Leifert, Roger Labahn, Tobias Grüning, and Svenja Leifert. End-to-end measure for text recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1424–1431. IEEE, 2019.
- Diego Mollá, Menno Van Zaanen, and Daniel Smith. Named entity recognition for question answering. In *Australasian Language Technology Association Workshop*, pages 51–58. Australasian Language Technology Association, 2006.
- Bart Ooghe, Heritage Cell Waasland, and Dries Moreels. Analysing selection for digitisation. *D-Lib Magazine*, 15(9/10):1082–9873, 2009.
- Richard Pearce-Moses and Laurie A Baty. *A glossary of archival and records terminology*, volume 2013. Society of American Archivists Chicago, IL, 2005.
- Dunlu Peng, Lidong Cao, and Wenjie Xu. Using json for data exchanging in web service applications. *Journal of Computational Information Systems*, 7(16):5883–5890, 2011.
- Schraagen, Marijn and others. Aspects of record linkage. *Diss. Leiden University*. <http://hdl.handle.net/1887/29716>, 2014.
- Mehmet Sezgin and Buğlent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–168, 2004.
- Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.
- Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*, 2023.
- Nick White. Training tesseract for ancient greek ocr. *Eiiruzov*, (28–29), 2012.
- William E Winkler. Matching and record linkage. *Wiley interdisciplinary reviews: Computational statistics*, 6(5):313–325, 2014.

# A Example Page: Alphabetical List of Names

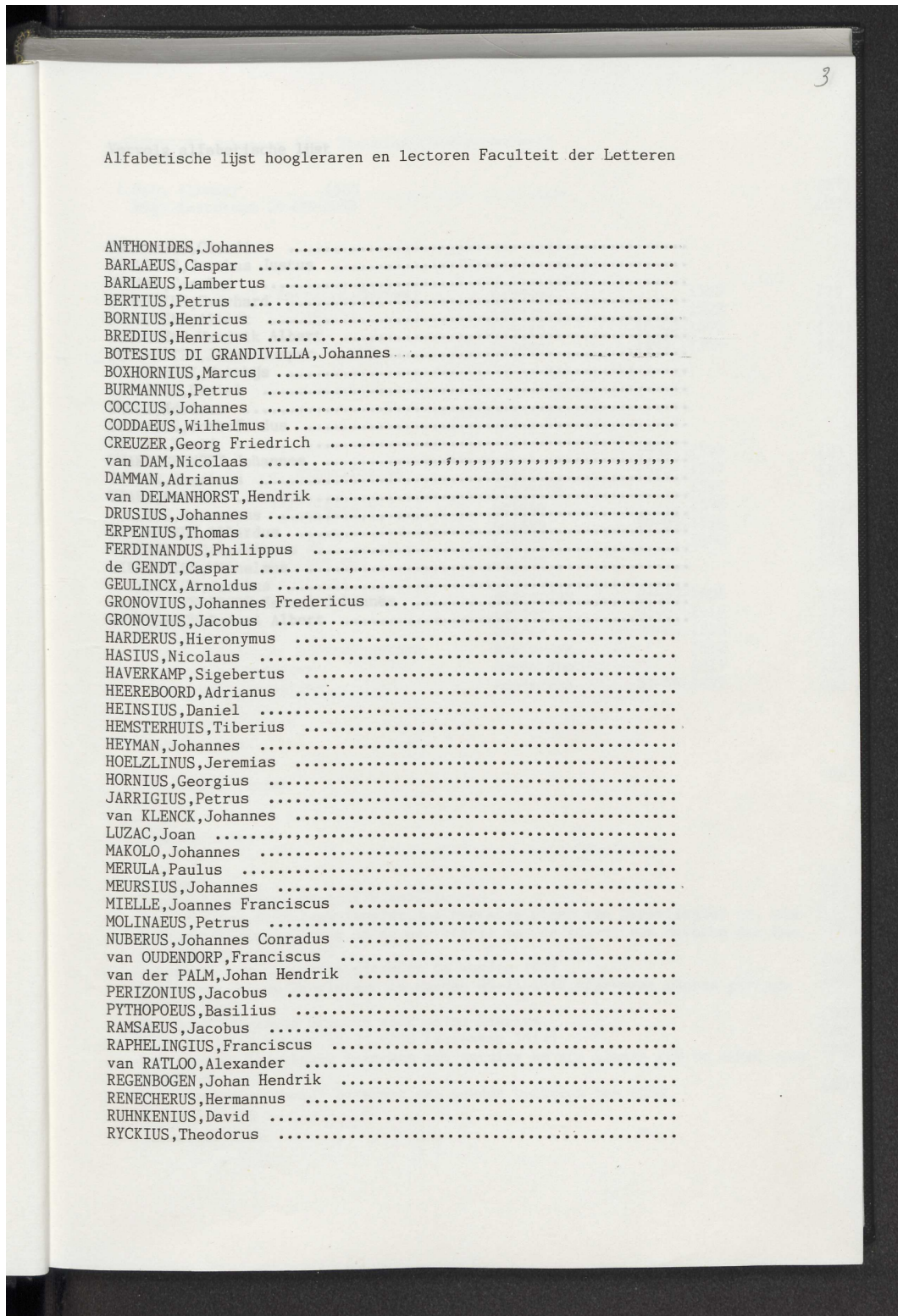


Figure 14: Leidse hoogleraren en lectoren 1575-1815. Volume 5. Page 9.

## B Example Page: Detailed Information About a Person

5

ANTHONIDES, Johannes (Jan Theunisz. (Barbarossius))

1. Geb. Alkmaar	1569		(2)	(27)
Begr. Amsterdam	29-09-1638			(80)
2. Opleiding:				
-Latijnse School (Potterus, Metius)	Alkmaar	1589	(2,80)	(2)
-Latijnse School (Stochius)	Leiden	1592		(80)
-Stud. Litt.	Leiden	16-02-1593		(28)
-Theses verdedigd	Leiden	1593, 1594		(29)
3. Loopbaan:				
-Garentwinder		1589 -1593		(2)
-Corrector bij prof. Raphelingius	Leiden	1595 -1597		(80)
-Reis naar Frankrijk	Frankrijk	1597		(80)
-Begin privé-studie Arabisch	Leiden	1598		(80)
-Poorter Leiden	Leiden	28-05-1599		(2)
-Drukker op Nieuwe Rijn	Leiden	1600		(2)
-Privaatdocent Arabisch met toestemming stadsbestuur	Leiden			(2)
-Poorter en drukker Amsterdam	Amsterdam	04-05-1604		(80)
-Lector Arabisch	Leiden	03-03-1612		(27)
- "Eerlijk afscheid"	Leiden	08/09-05-1613		(27)
-Brandewijnverkoper Oudenbrugsteeg	Amsterdam	1612		(2)
-Hoogleraar Hebreeuws	Amsterdam	1617		(2)
-Lid boekdrukkersgilde	Amsterdam	31-08-1627		(2)
4. Nevenfunctie:				
-Brandewijnverkoper				(80)
5. Bijzonderheden:				
-Doopsgezind				(2)
-Woont als student bij prof. R. Snellius				(2)
-Bevriend met prof. Raphelingius, Jan Paets (opvolger van Raphelingius jr. als academiedrukker), Sladus en de secretaris van de gezant van Marokko die hem in 1609 Arabisch leert				(80)
-Vergeefse poging academiedrukker te worden 1602				(80)
-Vergeefse poging hoogleraar te worden 08-11-1612, afgewezen wegens geringe kennis Latijn				(80)
-Vertaalt Arabische brieven voor prins Maurits				(80)
-Salaris: f150 (maart 1612), f200 (november 1612)				(80)
-Heinsius en Casaubonus bewerken zijn ontslag om zijn plaats vrij te maken voor Erpenius				(80)
-Het "eerlijk afscheid" betekent een gouden handdruk van f250				(80)

Figure 15: Leidse hoogleraren en lectoren 1575-1815. Volume 5. Page 11.

## C Example Page: List of Sources Used in the Volume

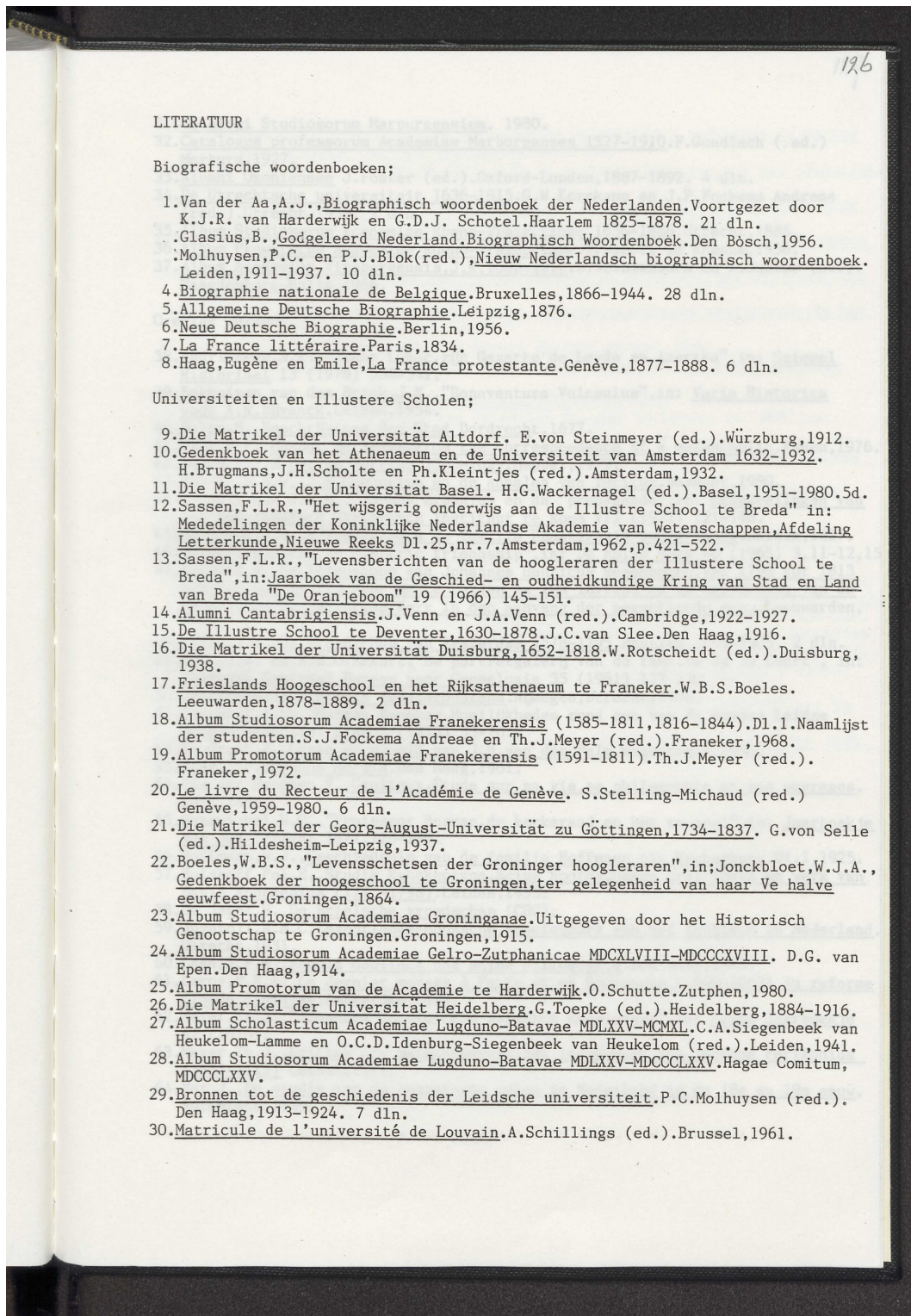


Figure 16: Leidse hoogleraren en lectoren 1575-1815. Volume 5. Page 133.

## D Example Person: Original Images

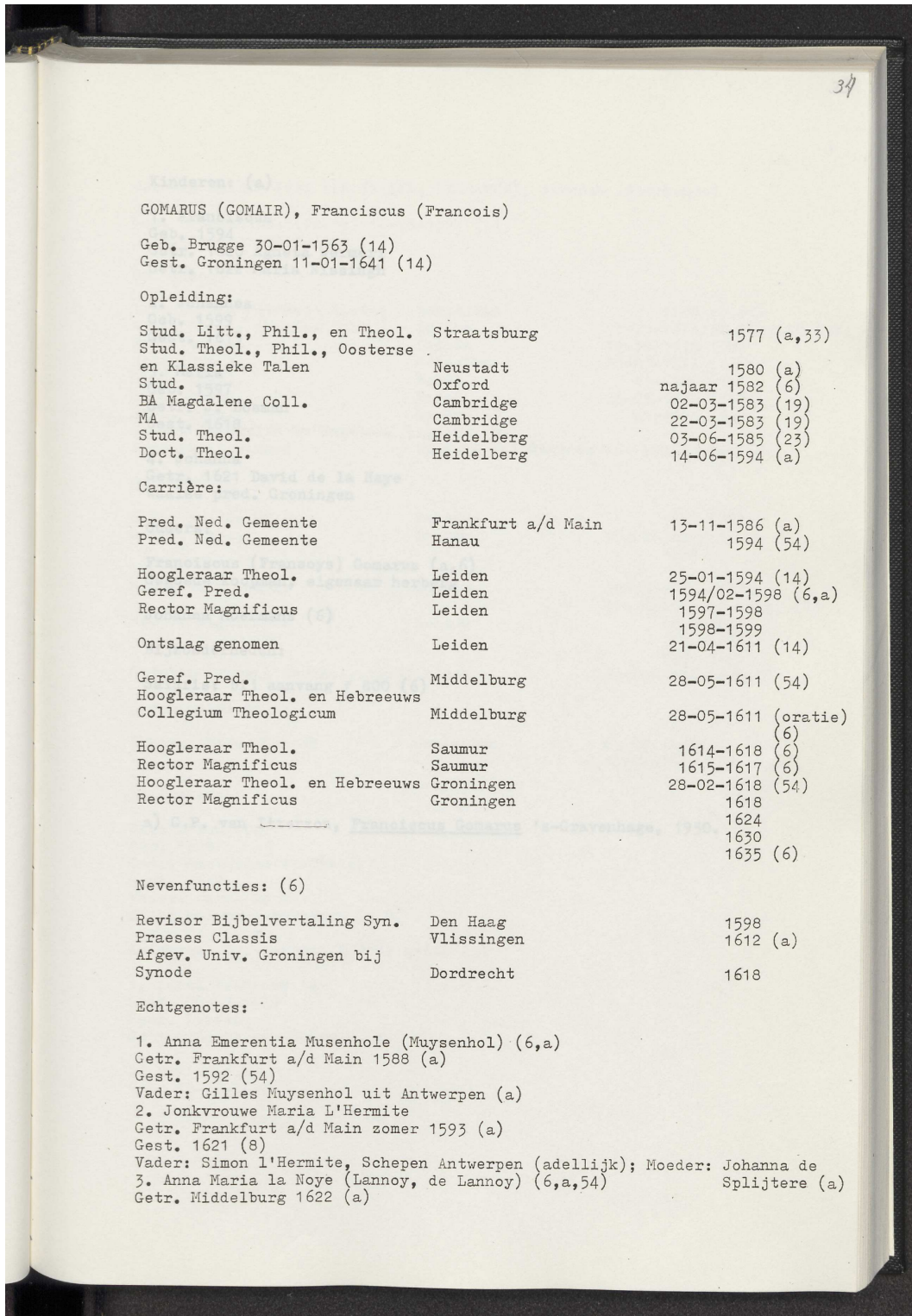


Figure 17: Leidse hoogleraren en lectoren 1575-1815. Volume 1. Page 45.



## Kinderen: (a)

## 1. Franciscus

Geb. 1594

Getr. 1620 Agneta Wermeri

Getr. 1622 Maria Nissingh

## 2. Johannes

Geb. 1599

Gest. 1613

## 3. Maria

Geb. 1597

Getr. J. Bosman

Gest. 1618

## 4. Johanna

Getr. 1621 David de la Haye

Waalse pred. Groningen

## Ouders:

Franciscus (Fransoys) Gomarus (a,6)

Gegoede koopman, eigenaar herberg

Johanna Moermans (6)

## Bijzonderheden:

Salaris: bij aanvang f 800 (6)

## Navenfuncties:

Process. Prov. Synode

Gorinchem

03-07-1609/

15-07-1609 (7)

## Schijngener:

a) G.P. van Itterzon, Franciscus Gomarus 's-Gravenhage, 1930.

Geb. 1600

Getr. Amsterdam 16-03-1627

Gest. 1649

Vader: Carel IJtken

Rijks koopman Amsterdam (a,6)

Kinderen: 10, waarvan 5 jong gest. (4)

## 1. Karel Heidanus (a)

Gest. 30-05-1679

Gebr. Leiden

Getr. Jacobs van Swannenburg

## 2. Maria (a,6)

Geb. 12-03-1628

Figure 18: Leidse hoogleraren en lectoren 1575-1815. Volume 1. Page 46.

# E Example Person: Preprocessed Images

37

GOMARUS (GOMAIR), Franciscus (Francois)

Geb. Brugge 30-01-1563 (14)  
 Gest. Groningen 11-01-1641 (14)

Opleiding:

Stud. Litt., Phil., en Theol.	Straatsburg	1577 (a,33)
Stud. Theol., Phil., Oosterse en Klassieke Talen	Neustadt	1580 (a)
Stud.	Oxford	najaar 1582 (6)
BA Magdalene Coll.	Cambridge	02-03-1583 (19)
MA	Cambridge	22-03-1583 (19)
Stud. Theol.	Heidelberg	03-06-1585 (23)
Doct. Theol.	Heidelberg	14-06-1594 (a)

Carrière:

Pred. Ned. Gemeente	Frankfurt a/d Main	13-11-1586 (a)
Pred. Ned. Gemeente	Hanau	1594 (54)
Hoogleraar Theol.	Leiden	25-01-1594 (14)
Geref. Pred.	Leiden	1594/02-1598 (6,a)
Rector Magnificus	Leiden	1597-1598
		1598-1599
Ontslag genomen	Leiden	21-04-1611 (14)
Geref. Pred.	Middelburg	28-05-1611 (54)
Hoogleraar Theol. en Hebreeuws Collegium Theologicum	Middelburg	28-05-1611 (oratie)
		(6)
Hoogleraar Theol.	Saumur	1614-1618 (6)
Rector Magnificus	Saumur	1615-1617 (6)
Hoogleraar Theol. en Hebreeuws	Groningen	28-02-1618 (54)
Rector Magnificus	Groningen	1618
		1624
		1630
		1635 (6)

Nevenfuncties: (6)

Revisor Bijbelvertaling Syn.	Den Haag	1598
Praeses Classis	Vlissingen	1612 (a)
Afgev. Univ. Groningen bij Synode	Dordrecht	1618

Echtgenotes:

1. Anna Emerentia Musenhole (Muysenhol) (6,a)  
 Getr. Frankfurt a/d Main 1588 (a)  
 Gest. 1592 (54)  
 Vader: Gilles Muysenhol uit Antwerpen (a)
2. Jonkvrouwe Maria L'Hermite  
 Getr. Frankfurt a/d Main zomer 1593 (a)  
 Gest. 1621 (8)  
 Vader: Simon l'Hermite, Schepen Antwerpen (adellijk); Moeder: Johanna de
3. Anna Maria la Noye (Lannoy, de Lannoy) (6,a,54) Splijttere (a)  
 Getr. Middelburg 1622 (a)

Figure 19: Leidse hoogleraren en lectoren 1575-1815. Volume 1. Page 45.

## Kinderen: (a)

1. Franciscus  
Geb. 1594  
Getr. 1620 Agneta Wermeri  
Getr. 1622 Maria Nissingh

2. Johannes  
Geb. 1599  
Gest. 1613

3. Maria  
Geb. 1597  
Getr. J. Bosman  
Gest. 1618

4. Johanna  
Getr. 1621 David de la Haye  
Waalse pred. Groningen

## Ouders:

Franciscus (Fransoys) Gomarus (a,6)  
Gegoede koopman, eigenaar herberg

Johanna Moermans (6)

## Bijzonderheden:

Salaris: bij aanvang f 800 (6)

a) G.P. van Itterzon, Franciscus Gomarus 's-Gravenhage, 1930.

Figure 20: Leidse hoogleraren en lectoren 1575-1815. Volume 1. Page 46.

## F Example Person: OCR-Generated Text

Listing 8: Text file generated using a trained Tesseract OCR engine, detailing the education and career of Franciscus (Francois) GOMARUS.

```
9
GOMARUS (GOMAIR), Franciscus (Francois)

Geb. Brugge 30-01-1563 (14)
Gest. Groningen 11-01-1641 (14)

Opleiding:
Stud. Litt., Phil., en Theol. Straatsburg 1577 (a,33)
Stud. Theol., Phil., Oosterse .
en Klassieke Talen Neustadt 1580 #
Stud. Oxford najaar 1582 (6)
BA Magdalene Coll. Cambridge 02-03-1583 (19)
MA Cambridge 22-03-1583 (19)
Stud. Theol. Heidelberg 03-06-1585 23)
Doct. TFheol. Heidelberg 14-06-1594 (a)
Carrire :
Pred. Ned. Gemeente Frankfurt a/d Main 13-11-1586 (a)
Pred. Ned. Gemeente Hanau 1594 (54)
Hoogleraar Theol. Leiden 25-01-1594 (14)
Geref. Pred. Leiden 1594/02-1598 (6,a)
Rector Magnificus Leiden 1897=-1598
1598-1599
Ontslag genomen Leiden 21-04-1611 (14)
Geref. Pred. Middelburg 28-05-1611 (54)
Hoogleraar Theol. en Hebreeuws
Collegium Theologdcm Middelburg 28-05-1611 (952t1e)
6
Hoogleraar Theol. Saumur 1614-1618 62
Rector Magnificus Saumur 1615.1617 (6)
Hoogleraar Theol. en Hebreeuws Groningen 28-02-1618 (54)
Rector Magnificus Groningen 1618
1624
1630
1635 (6)
Nevenfuncties: (6)
Revisor Bijbelvertaling Syn. Den Haag 1598
Praeses Classis Vlissingen 1612 (a)
Afgev. Univ. Groningen bij
Synode Dordrecht 1618
Echtgenotes:
1. Anna Emerentia Musenhole (Muysenhol) (6,a)
```

Getr. Frankfurt a/d Main 1588 (a)

Gest. 1592 (54)

Vader: Gilles Muysenhol uit Antwerpen (a)

2. Jonkvrouwe Maria L'Hermite

Getr. Frankfurt a/d Main zomer 1593 (a)

Gest. 1621 (8)

Vader: Simon l'Hermite, Schepen Antwerpen (adellijk)s Moeder: Johanna de

3. Anna Maria la Noye (Lannoy, de Lannoy) (6,a,54) splijtere (a)

Getr. Middelburg 1622 (a)

Kinderen: (a)

1. Franciscus

Geb. 1594

Getr. 1620 Agneta Wermeri

Getr. 1622 Maria Nissingh

2. Johannes

Geb. 1599

Gest. 1613

3. Maria

Geb. 1597

Getr. J. Bosman

Gest. 1618

4. Johanna

Getr. 1621 David de la Haye

Waalse pred. Groningen

Ouders:

Franciscus (Fransoys) Gomarus (4a,6)

Gegoede koopman, eigenaar herberg

Johanna Moermans (6)

Bijzonderheden:

Salaris: bij aanvang 7 800 (6)

a) G.P. van Ttterzon, Franciscus Gomarus 's-Gravenhage, 1930.

## G Example Person: AI-Generated JSON

Listing 9: This JSON file, generated using GPT-3.5 Turbo based on OCR-generated text as input, provides detailed information about Franciscus (Francois) GOMARUS, including his names, education, careers, particularities, spouses, parents, and children.

```
{
  "FirstName": "Franciscus (Francois)",
  "LastName": "GOMARUS",
  "Affix": "(GOMAIR)",
  "Gender": "Man",
  "second_names": [
    "Gomair"
  ],
  "alternative_last_names": [],
  "education": [
    {
      "subject": "Stud. Litt., Phil., en Theol.",
      "location": "Straatsburg",
      "date": "1577",
      "source": "33"
    },
    {
      "subject": "Stud. Theol., Phil., Oosterse en Klassieke Talen",
      "location": "Neustadt",
      "date": "1580",
      "source": null
    },
    {
      "subject": "Stud. Oxford",
      "location": "najaar 1582",
      "date": "1582",
      "source": null
    },
    {
      "subject": "BA",
      "location": "Magdalene Coll. Camnbridge",
      "date": "1583-03-02",
      "source": "19"
    },
    {
      "subject": "MA",
      "location": "Cambridge",
      "date": "1583-03-22",
      "source": "19"
    },
    {
      "subject": "Stud. Theol.",
```

```

    "location": "Heidelberg",
    "date": "1585-06-03",
    "source": "23"
  },
  {
    "subject": "Doct. TFheol.",
    "location": "Heidelberg",
    "date": "1594-06-14",
    "source": "a"
  }
],
"careers": [
  {
    "job": "Pred. Ned. Gemeente Frankfurt a/d Main",
    "location": "Frankfurt",
    "date": "1586-11-13",
    "source": "a",
    "is_side_job": 0
  },
  {
    "job": "Pred. Ned. Gemeente Hanau",
    "location": "Hanau",
    "date": "1594",
    "source": "54",
    "is_side_job": 0
  },
  {
    "job": "Hoogleraar Theol.",
    "location": "Leiden",
    "date": "1594-01-25",
    "source": "14",
    "is_side_job": 0
  },
  {
    "job": "Geref. Pred.",
    "location": "Leiden",
    "date": "1594-02",
    "source": "6,a",
    "is_side_job": 0
  },
  {
    "job": "Rector Magnificus",
    "location": "Leiden",
    "date": "1597-1598",
    "source": null,
    "is_side_job": 0
  }
]

```



```

    "job": "Geref. Pred.",
    "location": "Middelburg",
    "date": "1611-05-28",
    "source": "54",
    "is_side_job": 0
  },
  {
    "job": "Hoogleraar Theol. en Hebreeuws",
    "location": "Collegium Theologdcm Middelburg",
    "date": "1611-05-28",
    "source": null,
    "is_side_job": 0
  },
  {
    "job": "Hoogleraar Theol.",
    "location": "Saumur",
    "date": "1614-1618",
    "source": "62",
    "is_side_job": 0
  },
  {
    "job": "Rector Magnificus",
    "location": "Saumur",
    "date": "1615-1617",
    "source": null,
    "is_side_job": 0
  },
  {
    "job": "Hoogleraar Theol. en Hebreeuws",
    "location": "Groningen",
    "date": "1618-02-28",
    "source": "54",
    "is_side_job": 0
  },
  {
    "job": "Rector Magnificus",
    "location": "Groningen",
    "date": "1618-1635",
    "source": null,
    "is_side_job": 0
  }
],
"particularities": [
  {
    "particularity": "Salaris: bij aanvang 7 800",
    "location": "s 'Gravenhage",
    "date": null,
    "source": "6"
  }
]

```

```

}
],
"spouses": [
  {
    "FirstName": "Anna Emerentia",
    "LastName": "Musenhole",
    "Affix": null,
    "Gender": null,
    "source": "6,a",
    "second_names": [],
    "alternative_last_names": [],
    "BirthCountry": "Duitsland",
    "BirthCity": "Frankfurt",
    "BirthDate": null,
    "DeathDate": "1592",
    "DeathCity": null
  },
  {
    "FirstName": "Jonkvrouwe Maria",
    "LastName": "L'Hermite",
    "Affix": "(adellijk)",
    "Gender": null,
    "source": "8",
    "second_names": [],
    "alternative_last_names": [],
    "BirthCountry": "Belgi\u00eb",
    "BirthCity": "Antwerpen",
    "BirthDate": null,
    "DeathDate": "1621",
    "DeathCity": null
  },
  {
    "FirstName": "Anna Maria",
    "LastName": "la Noye",
    "Affix": "(de Lannoy)",
    "Gender": null,
    "source": "6,a,54",
    "second_names": [],
    "alternative_last_names": [],
    "BirthCountry": "Nederland",
    "BirthCity": "Middelburg",
    "BirthDate": "1622",
    "DeathDate": null,
    "DeathCity": null
  }
],
"parents": [
  {

```

```

    "FirstName": "Franciscus",
    "LastName": "GOMARUS",
    "Affix": null,
    "Gender": null,
    "source": null,
    "second_names": [],
    "alternative_last_names": [],
    "BirthCountry": null,
    "BirthCity": null,
    "BirthDate": null,
    "DeathDate": null,
    "DeathCity": null
  },
  {
    "FirstName": "Johanna",
    "LastName": "Moermans",
    "Affix": null,
    "Gender": null,
    "source": null,
    "second_names": [],
    "alternative_last_names": [],
    "BirthCountry": null,
    "BirthCity": null,
    "BirthDate": null,
    "DeathDate": null,
    "DeathCity": null
  }
],
"grand_parents": [],
"in_laws": [],
"children": [
  {
    "FirstName": "Franciscus",
    "LastName": "GOMARUS",
    "Affix": null,
    "Gender": null,
    "source": null,
    "second_names": [],
    "alternative_last_names": [],
    "BirthCountry": null,
    "BirthCity": null,
    "BirthDate": "1594",
    "DeathDate": null,
    "DeathCity": null
  },
  {
    "FirstName": "Johannes",
    "LastName": "GOMARUS",

```

```

    "Affix": null,
    "Gender": null,
    "source": null,
    "second_names": [],
    "alternative_last_names": [],
    "BirthCountry": null,
    "BirthCity": null,
    "BirthDate": "1599",
    "DeathDate": "1613",
    "DeathCity": null
  },
  {
    "FirstName": "Maria",
    "LastName": "GOMARUS",
    "Affix": null,
    "Gender": null,
    "source": null,
    "second_names": [],
    "alternative_last_names": [],
    "BirthCountry": null,
    "BirthCity": null,
    "BirthDate": "1597",
    "DeathDate": "1618",
    "DeathCity": null
  },
  {
    "FirstName": "Johanna",
    "LastName": "GOMARUS",
    "Affix": null,
    "Gender": null,
    "source": null,
    "second_names": [],
    "alternative_last_names": [],
    "BirthCountry": null,
    "BirthCity": null,
    "BirthDate": "1621",
    "DeathDate": null,
    "DeathCity": null
  }
],
"far_family": [],
"type_of_person": 1,
"faculty": "Theologie",
"BirthCountry": null,
"BirthCity": "Brugge",
"BirthDate": "1563-01-30",
"DeathDate": "1641-01-11",
"DeathCity": "Groningen"

```

}

---

## H Example Person: Performance of the Record Linking Algorithm

Listing 10: This JSON file demonstrates the algorithm's ability to link the example person to a person in the database. The file '21.vol1.txt' contains the OCR-generated text for this example person. The algorithm successfully identified a person with ID 485, indicating that the additional information found in our dataset about this person will be appended to this ID.

```
{
  "21.vol1.txt": {
    "person_id": 485,
    "new_person": false,
    "maybe_same_person": false
  }
}
```

# I Regular Expressions Patterns for Information Extraction

Listing 11: Regular expressions patterns used to extract information from text

```
# Define regular expressions for each part
name_pattern = r'^([\^,]+(?:\s+\([\^]+\))?)?:\s*(.*(?:=\s+\(|\|$\)))?'
birthdate_pattern = r'1\.Geb\..*?(\d{2}-\d{2}-\d{4})'
deathdate_pattern = r'Gest\..*?(\d{2}-\d{2}-\d{4})'
education_pattern = r'2\.Opleiding:(.*)3\.Loopbaan:'
career_pattern = r'3\.Loopbaan:(.*)4\.Nevenfunctie:'
extra_work_pattern = r'4\.Nevenfunctie:(.*)5\. Bijzonderheden:'
additional_info_pattern = r'5\. Bijzonderheden:(.*)6\.Ouders:'
parents_pattern = r'6\.Ouders:(.*)7\.Ouders Vader:'
parents_father = r'7\.Ouders Vader:(.*)8\.Ouders Moeder:'
parents_mother = r'8\.Ouders Moeder:(.*)9\.Echtgenotes:'
spouses = r'9\.Echtgenotes:(.*)10\.Ouders Echtgenotes:'
parents_spouses = r'10\.Ouders Echtgenotes:(.*)11\Kinderen:'
children = r'11\Kinderen:(.*)'
```