Digitization of Handwritten Ledgers

An Integrated Approach to Improve Handwritten Text Recognition

submitted in partial fulfillment for the degree of master of science

Roderick Majoor
12852724

Master Information Studies
data science
Faculty of Science
University of Amsterdam

Submitted on 12.04.2024

	UvA Supervisor
Title, Name	Dr. N.J.E. (Nanne) van Noord
Affiliation	MultiX Lab
Email	n.j.e.vannoord@uva.nl

ABSTRACT

Write your abstract here.

KEYWORDS

keywords, belong, here, with, commas, like, this

GITHUB REPOSITORY

https://github.com/roderickmajoor/Thesis

1 INTRODUCTION

The Bank of Amsterdam was an early bank established in 1609, described by some as the first central bank [19]. The bank played an important role in the financial world of the 17th and 18th century. Transactions made at the bank from 1650 to 1800 were recorded in ledgers and are still preserved to this day. These ledgers contain information about transactions, both debit and credit, of customers of the bank. The ledgers can thus be very important in analyzing the money flow at the time. Digitizing these ledgers may help in preserving them in a much easier to analyze format which would allow for more research into the contents of the ledgers. However, these ledgers are all handwritten making it hard to retrieve the information out of them on large scale. Current efforts to retrieve the information are done by using Handwritten Text Recognition (HTR) tools to extract the text in the ledgers. The problem is that the currently used HTR techniques have too many inaccuracies to be used effectively. Because of this, the handwritten text cannot be recognized correctly and can thus not be digitized properly. To improve digitization efforts, it is crucial to find out why some handwritten text cannot be recognized, what kind of errors are made during the HTR process and how the HTR process can be optimized to improve these inaccuracies. The research question we wish to answer is:

How can the categorization of errors in handwritten ledger analysis be used to enhance the identification and resolution of text recognition inaccuracies?

Subquestions belonging to this research question are:

- What factors contribute to text recognition errors in handwritten ledgers?
- What are the different types of errors encountered in handwritten text recognition?
- How can specific preprocessing methods improve text detection and segmentation in handwritten ledgers?
- What strategies can be implemented to mitigate common text recognition errors in handwritten ledgers?

2 RELATED WORK

The research gap being addressed in this project revolves around the challenges associated with the digitization of handwritten ledgers. The proposed research will contribute in closing the research gap within the HTR domain, specifically looking at historical handwritten documents. The digitization of handwritten ledgers contains several steps. In this section, we will look at key research papers that discuss different approaches to the steps of our problem.

2.1 Document Layout Analysis

The first challenge in our digitization task is being able to find the correct word order from the HTR output. Seuret describes layout analysis and finding the correct word order as one of the main challenges in HTR [21]. Several approaches to solve this problem have been tried. These approaches are typically divided in either deep learning methods or more classical computer vision methods. We will show recent advancements made in both approaches.

Kastanas et al. show the use of different deep learning architectures for layout analysis [8]. Transformer-based, graph-based models, and convolutional neural networks (CNN) are compared. The CNN-based model YOLOv5 and transformed-based model LayoutLMv3 both show promising results for identifying most elements on documents, with YOLOv5 performing slightly better [8].

Multiple other studies describe the use of transformer based methods to perform layout analysis and optical character recognition (OCR) [10, 13, 25]. Furthermore, Smock et al. show that transformer-based object detection models can produce excellent results for the tasks of detection, structure recognition, and functional analysis of tables [23]. While these approaches show promising results, they are not tested for our specific task at hand.

Oliveira et al. propose dhSegment, an open-source implementation of a CNN-based pixel-wise predictor for document segmentation. The approach aims to address various document processing tasks simultaneously, including page extraction, baseline extraction, layout analysis, and illustration and photograph extraction. They show that a single CNN architecture can be used across tasks with competitive results [2].

Droby et al. propose a holistic method that applies Mask R-CNN for text line extraction in historical documents. Their work achieved state-of-the-art results on well known historical datasets [6]. While their work does not directly transfer to our task, it does show the potential for document segmentation and possibly table recognition abilities of Mask R-CNN.

The aforementioned studies show that the use of deep learning methods can be used to perform layout analysis as well as table recognition tasks in historical documents. However, these methods often require massive amounts of training data to be used effectively. To use these methods, we should look whether pre-trained models transfer well to our dataset and are able to detect the layout of our documents. Other approaches use more classical computer vision techniques to segment a document in order to recognize the layout. These methods often require homogeneity of the documents to work effectively.

Lehenmeier et al. show a method that combines various state-of-the-art methods for OCR pro- cessing to detect layout on a document [9]. To perform table recognition, they make use of techniques such as binarization, denoising, and Hough line detection to find relevant parts of the tables. By taking the intersection of relevant horizontal and vertical Hough lines they are able to find table cells and thus determine the layout of the table [9].

Bulacu et al. introduce a method based on contour tracing that generates curvilinear separation paths between text lines in order to preserve the ascenders and descenders of text lines to determine the layout in handwritten historical documents [3]. With this approach they are able to determine seperate elements of tables in historical documents.

Liu et al. describe the use of several key steps, including skew correction, region segmentation, and page layout analysis on historical Tibetan documents [12]. Skew correction is achieved by leveraging baseline features and Hough transform to determine the document's skew angle. Subsequently, region segmentation is accomplished by identifying borders through a series of preprocessing steps, including median filtering, Gaussian smoothing, Sobel edge detection, and removal of small area regions. These processes collectively enable accurate positioning of document borders for further analysis and structure extraction [12].

Liang et al. describe the use of Hessian and Gabor filters in order to find table structures on a document [11]. They show the possiblity for column segmentation for the tables using the found lines.

Prieto et al. address the challenge of document image understanding in documents with complex layouts like tables. They compare two approaches and show promising results [18]. Their approaches take the text lines outputted by HTR systems and use machine learning methods to group these text lines and classify cells based on these grouped text lines in order to find the tabular structure.

It can be seen that depending on the dataset and the homogeneity of the documents, different approaches for layout analysis/table recognition may be suitable. When a similar table structure is clearly drawn on all the documents, approaches using basic image processing techniques could be suitable to determine the structure of the table. When the structure is not easily identifiable on the page, it might be better to look at approaches that use the output of HTR systems such as text lines or words locations to classify table cells. Machine learning based approaches could also be useful in these cases.

2.2 HTR Processing Techniques

Typical HTR pipelines consist of pre-processing an image to make it ready for HTR, running the image through the HTR system and post-processing the output of the HTR system [9].

Studies have shown that pre-processing images can lead to better results when using them as input for document layout analysis or OCR. Thus, we want to optimize our image material such that the document layout analysis and HTR can be performed more accurately. There are different methods for doing this such as adjusting the contrast to remove artifacts and noise or fixing the image alignment [1, 5, 9]. It is crucial to find out which pre-processing steps might be useful to our problem at hand. Chen et al. show that the use of several pre-processing steps including character segmentation, tilt correction, offset correction, size normalization and image thinning lead to better HTR output results [4].

Post-processing OCR and HTR output typically consists of trying to correct misidentified words or characters. One strategy to do this is to create a dictionary of candidate characters and use the Levenshtein distance to calculate the distance between predicted and dictionary characters [17]. The use of a dictionary for allowed characters is a shown to be a good strategy to eliminate HTR errors [9]. Other studies show the use of language models to correct word output and spelling in HTR systems [14, 17].

2.3 HTR Evaluation Strategies

Typically, HTR systems are evaluated using the word error rate (WER) and character error rate (CER). However, the evaluation of page-level HTR output faces challenges primarily due to the Reading Order (RO) problem [24]. In their study, they define a bag-of-words Word Error Rate (bWER) to accurately measure page-level RO-independent word errors as well as a regularized version of the Hungarian Algorithm (HA) to compute word- and character-level RO-independent recognition accuracy [24]. Another strategy is the use of the intersection over union (IoU) score which matches ground truth and HTR output boxes based on their coordinates on the page [16]. It is then possible to use the WER and CER to calculate the accuracy.

3 METHODOLOGY

This section describes the data and methods used to conduct this research.

3.1 Data Description

The dataset used in this study consists of pages from the collection of ledgers from the Bank of Amsterdam. The full collection can be found on the website of The Amsterdam City Archives. In the series of ledgers, the current accounts of the customers were maintained. Each customer had one or more pages, with smaller traders having a portion of a page. An alphabetical list of the traders and the page of their current account can be found in the index. The ledgers kept track of the amounts that were debited and credited from and to a customer. The ledger pages consist of several columns containing information such as the date, account holder name, account number and amount debited/credited. Figure 1 shows an example of what a page looks like. Our dataset is split into multiple subsets that all have slightly varying pages, while still having structural similarity. Figure 2 shows the amount of pages for each subset in the dataset. The total dataset consists of 68 pages. A word count for each page is shown in Figure 3

The ground truth and HTR system output both use a PageXML format to store values on the page. These PageXML store the coordinates for each item (e.g. textregion, tableregion, etc.) on the page. We can plot the values in the PageXML file over our image to better show this. Figure 4 shows the PageXML data of the ground truth and HTR system plotted on the original image for part of one page in our dataset.

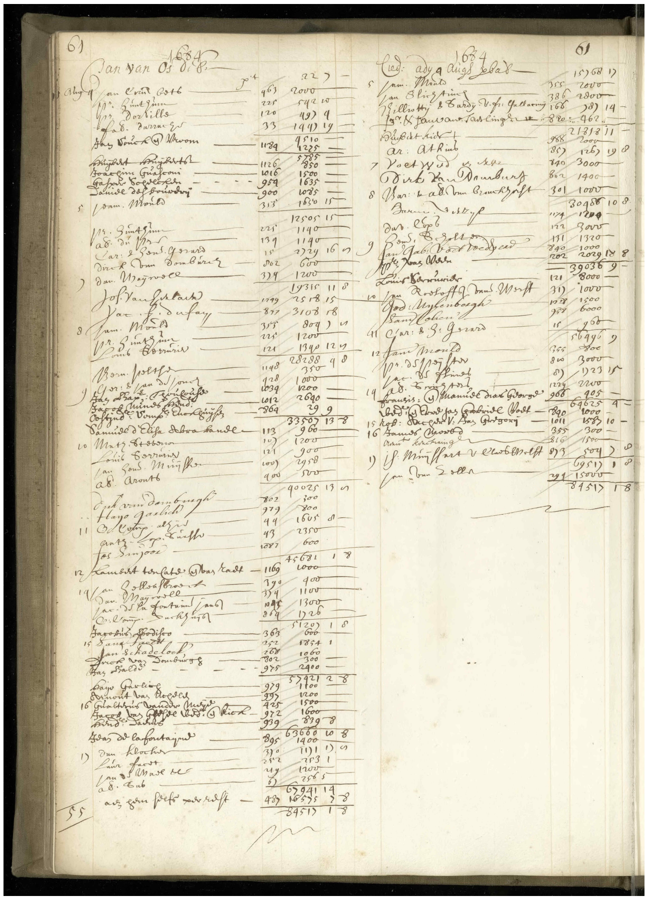


Figure 1: An example page from the ledger collection.

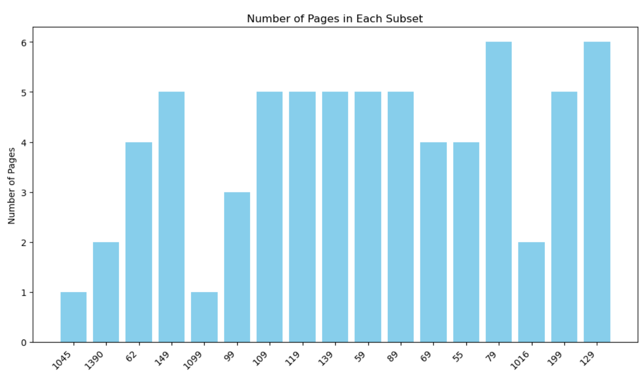


Figure 2: The amount of pages for each subset in our dataset.

3.2 Method

The methodology for this task is divided into two main parts: analyzing the current HTR system to identify its flaws, and performing steps to mitigate the errors made by the HTR system.

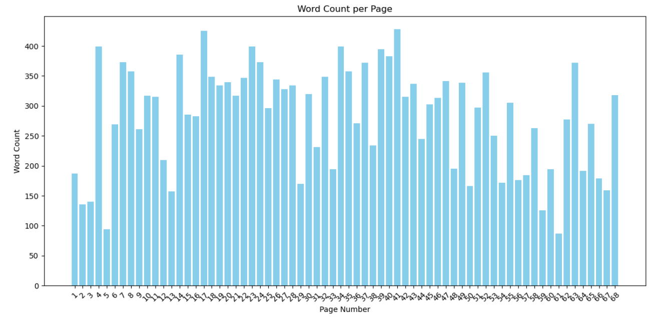


Figure 3: The word count for each page in our dataset.

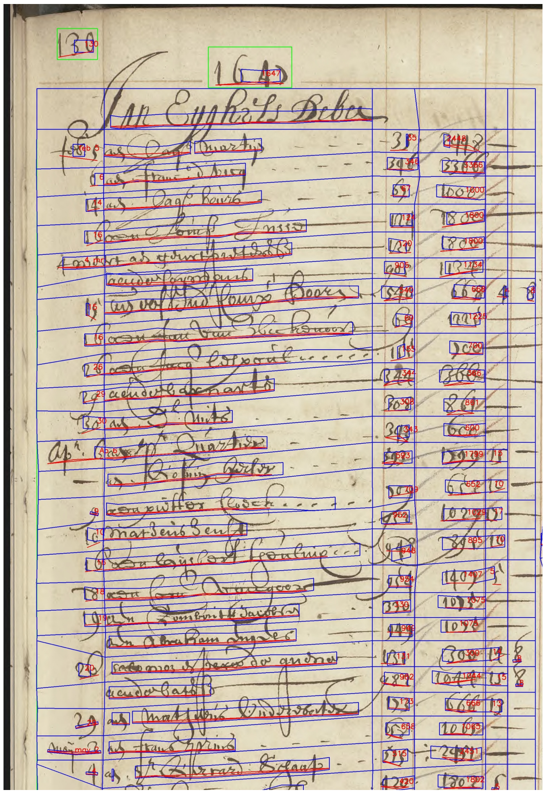
3.2.1 *Analyzing the HTR System.*

Before attempting to improve the HTR system, we will first conduct an extensive analysis to identify where the current system fails or has flaws. This will involve creating a categorization scheme that shows for each category the nature of the error and a possible solution. Based on this scheme, we can then try out various steps to improve the HTR system.

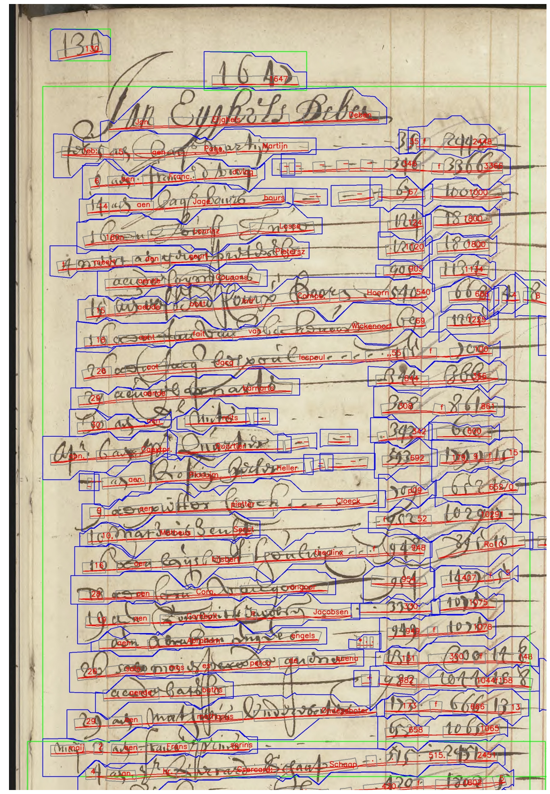
To obtain the accuracy of the HTR system, we will need to compare the HTR output to the ground truth values. This in itself is not straightforward, since the ground truth values do not contain all the words in the image while the HTR output does. Furthermore, since the layout of the page is not recognized by the HTR system, the order of the words is completely different in the ground truth and HTR output. To still compare the ground truth values and HTR output, we will use an approach that matches the words based on their bounding box coordinates on the image. To do this we make use of the IoU score. This is a technique that can be used to find how similar two bounding boxes are [16]. By matching each bounding box in the ground truth with a corresponding bounding box in the HTR output based on the highest IoU score, we can compare the groud truth values and HTR output on a word to word basis.

Once we have aligned the words in the ground truth and the HTR output properly, we can start to identify the flaws of the HTR system. We will make use of the Levenshtein distance, which can be used to find the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into the other [7]. By applying this on our ground truth values and HTR output on a word to word basis, we can track how many character insertions, deletions and substitutions are done in the HTR output for each word in the ground truth. We will show this for each subset of our data to also see whether some subsets are more difficult for the HTR system. We will also keep track of what characters are inserted, deleted or substituted so we can find what characters seem to be most problematic for the HTR system. The results of this analysis is shown Section 4.1.

By creating this analysis, we can also better inspect our image by visually showing the words that the HTR system can or cannot match. An example of this visual representation is shown in Figure 8. Using this visual representation, we can inspect the image to see whether there are any characteristics that might cause the HTR system to make errors. This could be noise on parts of the page, words being close together or even overlapping etc.



(a) The PageXML data of our ground truth plotted on part of the original image.



(b) The PageXML data of the HTR system plotted on part of the original image.

Figure 4: The PageXML data of the ground truth and HTR system plotted on the orginal image. The tabular structure of the ground truth values can be seen where this is not the case in the HTR output.

Upon completing our analysis, we can conduct a list of main errors made by the HTR system and decide on what steps could be taken to mitigate these errors. The findings of this analysis are shown in a categorization scheme in Table 2, showing the errors and possible solutions. The approach for these solutions is outlined in the next sections.

3.2.2 Layout Analysis. The second part of the methodology involves analyzing the layout of the documents. Two different methods will be used for this purpose:

(1) **Layout Parser Tool**: The Layout Parser is a tool for document image analysis that offers a range of pre-trained models and customization options. These models have been trained on diverse datasets, including historical documents, enabling users to select the most suitable model for their specific needs. Additionally, the Layout Parser allows for the training of custom layout models, allowing collaboration and knowledge exchange within the community. This also allows for the sharing of pre-trained models [22].

In our study, we will evaluate the Layout Parser by testing various models trained on historical documents. These models have different sizes and architectures, such as Faster R-CNN and Mask R-CNN, utilizing ResNet-50 or ResNet-101 backbones. The selection of backbone architecture influences the balance between model accuracy and computational costs [22].

(2) **Traditional Computer Vision Approach**: A more basic approach using OpenCV (cv2) will be tried to detect the layout of the documents. This approach will involve using image processing techniques to identify the borders and divisions of tables in the documents. This approach is based on the fact that the documents in our dataset show structural similarity. As can be seen in Figure 1, all of our pages contain the lines seperating the different columns of the table. By isolating these lines and extracting their coordinates, we will try to successfully seperate the data in correct columns. We do this by grayscaling our images and using Otsu's method to binarize our images [3]. This causes the text to be seperated from the more faint table lines. We then use morphological operations to seperate remaining noise on the page from the vertical table lines [15]. By then detecting Hough lines, we can extract the location of the column lines on the image [9]. To then identify the correct rows, a technique based on the coordinates found in the HTR output will be tried. This technique will involve analyzing the spatial distribution of

words in the documents to identify potential rows. By finding the centroids of the bounding boxes of words found in the HTR output, we can then try to group the words based on similarity of the vertical coordinate, by using hierarchical agglomerative clustering (HAC), thus creating seperate rows [20].

By performing this layout analysis, we could convert the found words by the HTR system into a dataframe for easier readability. The results of these methods will be compared to identify the most effective method for layout analysis.

3.2.3 Pre-processing. Pre-processing in our context involve the steps taken before the images go through the HTR system. It prepares the raw input images for further processing and analysis, with the goal of enhancing the image quality. This enhancement could potentially facilitate better recognition of text by the HTR system [5]. Common pre-processing steps include noise removal, Gaussian blurring, and binarization [1]. The main goal of these steps is to preserve the text while removing irrelevant artifacts of the image. Through experimentation, we aim to evaluate the impact of different pre-processing techniques on the performance of the HTR system. We will try out some common pre-processing steps and evaluate the performance of the HTR system to see whether they yield improved accuracy.

3.2.4 Post-processing. Post-processing in our experimental framework focuses on refining the output generated by the HTR system. One such post-processing step involves the removal of characters from the output strings that do not align with the ground truth dictionary [9, 17]. Analysis of the HTR output (as depicted in Figure 7a) reveals several insertions that are not valid letters or numeric characters. To address this, we plan to employ regular expressions (regex) to filter out invalid characters from the HTR output. By constructing a list of valid characters and using regex-based filtering, we aim to assess the effectiveness of post-processing techniques in improving the accuracy of the HTR output.

3.3 Evaluation

The evaluation of the methodology will involve calculating various metrics that measure the accuracy and effectiveness of the HTR system and the layout analysis methods.

For the HTR system, metrics such as WER, CER, and Levenshtein Distance will be calculated. These metrics measure the minimum number of operations (substitutions, insertions, deletions) needed to transform the HTR output into the ground truth. The formula for these metrics is:

$$WER = \frac{S_w + D_w + I_w}{N_w} \tag{1}$$

Where S_w, D_w, I_w are the number of word substitutions, deletions and insertions respectively and N_w is the total number of words in the ground truth.

$$CER = \frac{S_c + D_c + I_c}{N_c} \tag{2}$$

Where S_c, D_c, I_c are the number of character substitutions, deletions and insertions respectively and N_c is the total number of characters in the ground truth.

We can also calculate recall and precision scores. To calculate those for HTR output, we first need to define what constitutes true positives (TP), false positives (FP), and false negatives (FN) in our context.

- **TP (True Positive):** Characters correctly recognized by the HTR system.
- **FP (False Positive):** Characters incorrectly recognized by the HTR system (i.e., system identifies text where there is none or incorrectly identifies text).
- **FN (False Negative):** Characters that are present in the ground truth but not recognized by the HTR system.

The recall and precision are then:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

For the layout analysis methods, we will convert our ground truth PageXML table structure into a dataframe. We can then compare the cells for both the ground truth dataframe and the dataframe outputted by our layout analysis.

4 RESULTS

4.1 Analysis of HTR System Accuracy

The accuracy of the current HTR system is shown in Table 1. To calculate these accuracies we used the IoU method to match ground truth and HTR output words. A more in-depth analysis about the amount character insertions, deletions, substitutions and matches made by the HTR system can be seen in Figure 6 and Figure 5. Figure 7 shows the frequencies of characters inserted, substituted and deleted. Using these analyses, we can see more about the nature of the errors that the HTR system makes. We can see for instance that the characters '7' and '1' are often substituted. Furthermore, we can notice that there are ground truth annotations that contain spaces, where as this is not possible in the HTR output, because we select individual words. A selection of the most common errors and possible solutions is shown in Table 2.

Table 1: Baseline scores for the currently used HTR system. We matched the ground truth words with HTR output words based on highest IoU value.

Metric	Value
WER	0.4243
CER	0.3422
Recall	0.9475
Precision	0.7403
Total Edit Distance	18920

Table 2: Categorization of HTR System Errors

Error Type	Description	Possible Solution
Deletions	Characters not recognized at all (false negative)	Pre-process image, Improve HTR system
Insertions	Characters recognized that are not there (false positives)	Pre-process image, Improve HTR system
Substitutions	Characters recognized as another character	Pre-process image, Improve HTR system
Layout - Word Split	Single word is split into separate words	Layout analysis
Layout - Word Merge	Multiple words are recognized as one word	Layout analysis
Layout - Data Split	Data wrongly split (e.g., 'feb 5' in GT becomes 'feb' and '5' in HTR)	Layout analysis
Layout - Word Order	Word order not recognized	Layout analysis
HTR - Bounding Box	Wrong location bounding box vs actual location word	Improve HTR system
Case Sensitivity	Case sensitivity issues	Post-process string
Invalid Characters	Characters that cannot occur	Post-process string

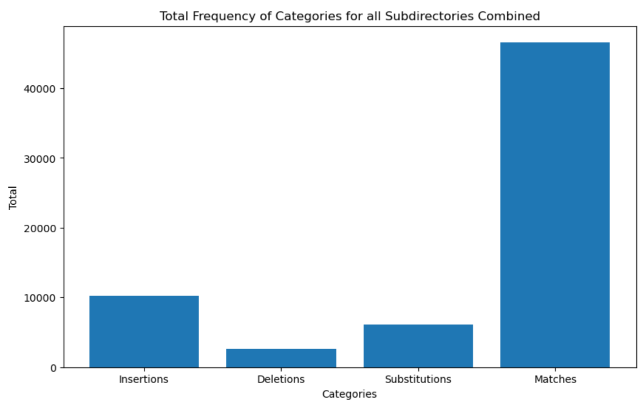
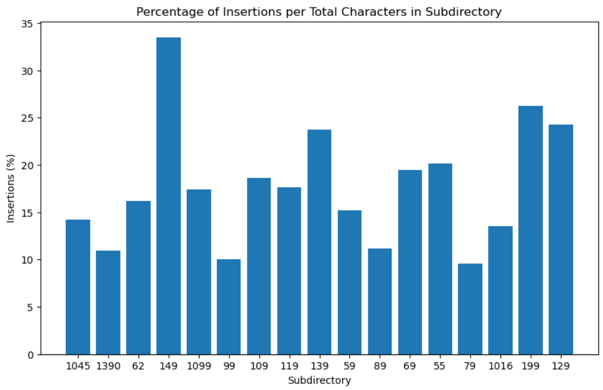
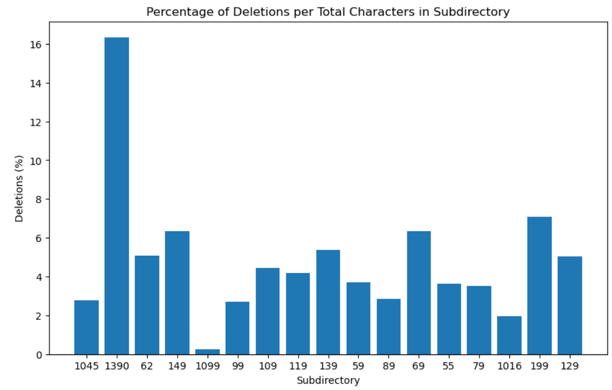


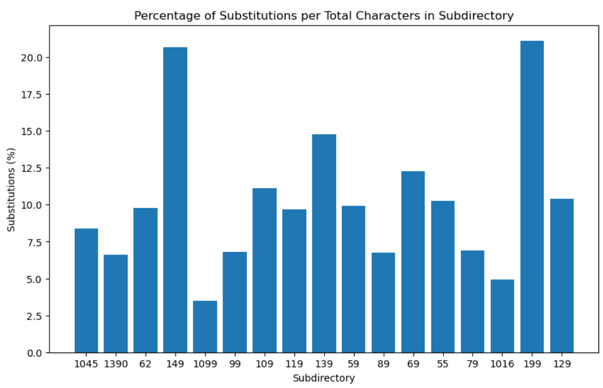
Figure 5: The total amount of insertions, deletions, substitutions and matches made by the HTR sytsem compared to the ground truth values. The values are on a character level and made word per word.



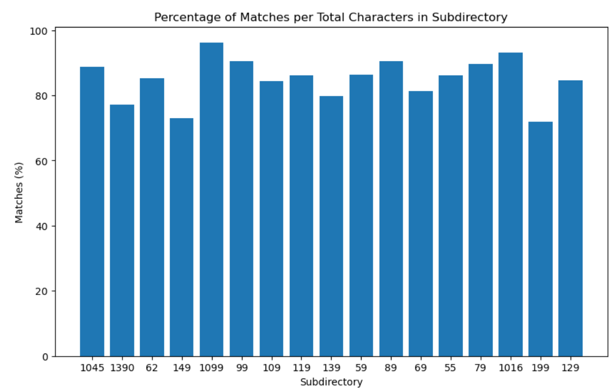
(a) Amount of insertions made by the HTR system per subdirectory.



(b) Amount of deletions made by the HTR system per subdirectory.

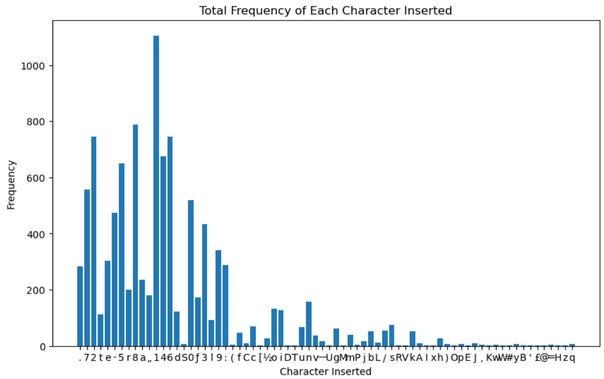


(c) Amount of substitutions made by the HTR system per subdirectory.

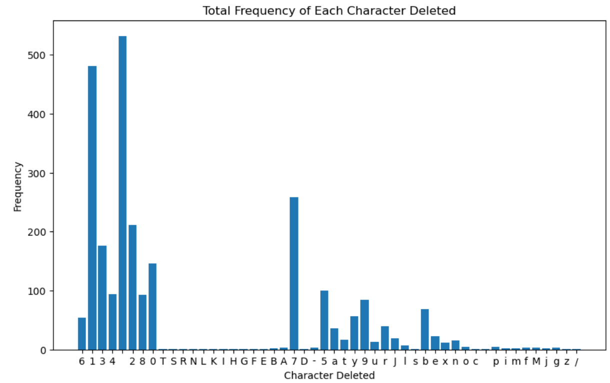


(d) Amount of matches made by the HTR system per subdirectory.

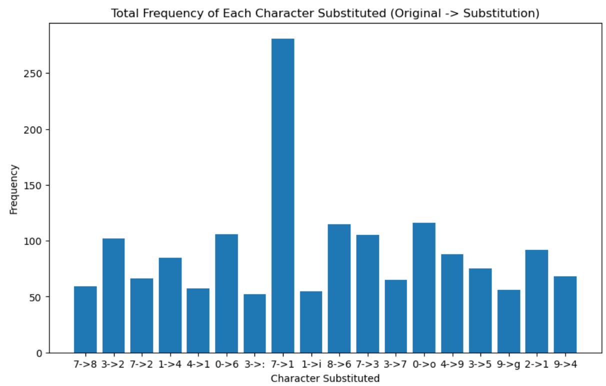
Figure 6: The amount of insertions, deletions, substitutions and matches made by the HTR system compared to the ground truth values. The values are on a character level and made word per word and as a percentage of total characters in ground truth.



(a) Frequency of inserted characters made by the HTR system.



(b) Frequency of deleted characters made by the HTR system.



(c) Frequency of substituted characters made by the HTR system. The substitutions are shown as (original character -> substituted character). We only show the substitutions with a frequency of 50 or higher.

Figure 7: The frequencies of inserted, deleted and substituted characters made by the HTR system.

5 DISCUSSION

Write your discussion here. Do not forget to use sub-sections. Normally, the discussion starts with comparing your results to other studies as precisely as possible. The limitations should be reflected upon in terms such as reproducibility, scalability, generalizability, reliability and validity. It is also important to mention ethical concerns.

6 CONCLUSION

Write your conclusion here. Be sure that the relation between the research gap and your contribution is clear. Be honest about how limitations in the study qualify the answer on the research question.

REFERENCES

[1] Yasser Alginahi. 2010. *Preprocessing Techniques in Character Recognition.* https://doi.org/10.5772/9776

[2] Sofia Ares Oliveira, Benoit Seguin, and Frederic Kaplan. 2018. dhSegment: A Generic Deep-Learning Approach for Document Segmentation. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE. https://doi.org/10.1109/icfhr-2018.2018.00011

[3] Marius Bulacu, Rutger van Koert, Lambert Schomaker, and Tijn van der Zant. 2007. Layout Analysis of Handwritten Historical Documents for Searching the Archive of the Cabinet of the Dutch Queen. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 1. 357–361. https://doi.org/10.1109/ICDAR.2007.4378732

[4] Tieming Chen, Guangyuan Fu, Hongqiao Wang, and Yuan Li. 2020. Research on Influence of Image Preprocessing on Handwritten Number Recognition Accuracy. In *The 8th International Conference on Computer Engineering and Networks (CENet2018)*, Qi Liu, Mustafa Mısır, Xin Wang, and Weiping Liu (Eds.). Springer International Publishing, Cham, 253–260.

[5] Sergio Correia and Stephan Luck. 2023. Digitizing historical balance sheet data: A practitioner's guide. *Explorations in Economic History* 87 (2023), 101475. https://doi.org/10.1016/j.eeh.2022.101475 Methodological Advances in the Extraction and Analysis of Historical Data.

[6] Ahmad Droby, Berat Kurar Barakat, Reem Alaasam, Boraq Madi, Irina Rabaev, and Jihad El-Sana. 2022. Text Line Extraction in Historical Documents Using Mask R-CNN. *Signals* 3, 3 (2022), 535–549. https://doi.org/10.3390/signals3030032

[7] Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications* 165 (2021), 113679. https://doi.org/10.1016/j.eswa.2020.113679

[8] Sotirios Kastanas, Shaomu Tan, and Yi He. 2023. Document AI: A Comparative Study of Transformer-Based, Graph-Based Models, and Convolutional Neural Networks For Document Layout Analysis. arXiv:2308.15517 [cs.CL]

[9] Constantin Lehenmeier, Manuel Burghardt, and Bernadette Mischka. 2020. *Layout Detection and Table Recognition – Recent Challenges in Digitizing Historical Documents and Handwritten Tabular Data.* Springer International Publishing, 229–242. https://doi.org/10.1007/978-3-030-54956-5_17

[10] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2022. TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models. arXiv:2109.10282 [cs.CL]

[11] Xusheng Liang, Abbas Cheddad, and Johan Hall. 2021. Comparative Study of Layout Analysis of Tabulated Historical Documents. *Big Data Research* 24 (2021), 100195. https://doi.org/10.1016/j.bdr.2021.100195

[12] Huaming Liu, Xuehui Bi, and Weilan Wang. 2019. Layout analysis of historical Tibetan documents. In *2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD)*. 74–78. https://doi.org/10.1109/ICAIBD.2019.8837040

[13] Tengchao Lv, Yupan Huang, Jingye Chen, Lei Cui, Shuming Ma, Yaoyao Chang, Shaohan Huang, Wenhui Wang, Li Dong, Weiyao Luo, Shaoxiang Wu, Guoxin Wang, Cha Zhang, and Furu Wei. 2023. Kosmos-2.5: A Multimodal Literate Model. arXiv:2309.11419 [cs.CL]

[14] Arthur Flor de Sousa Neto, Byron Leite Dantas Bezerra, and Alejandro Héctor Toselli. 2020. Towards the Natural Language Processing as Spelling Correction for Offline Handwritten Text Recognition Systems. *Applied Sciences* 10, 21 (2020). https://doi.org/10.3390/app10217711

[15] OpenCV. 2024. Morphological Transformations. https://docs.opencv.org/3.4/dd/dd7/tutorial_morph_lines_detection.html Accessed: 2024-04-18.

[16] Rafael Padilla, Sergio Netto, and Eduardo da Silva. 2020. A Survey on Performance Metrics for Object-Detection Algorithms. https://doi.org/10.1109/IWSSIP48289.2020

[17] Rémi Petitpierre, Marion Kramer, and Lucas Rappo. 2023. An end-to-end pipeline for historical censuses processing. *International Journal on Document Analysis and Recognition (IJDAR)* 26, 4 (March 2023), 419–432. https://doi.org/10.1007/s10032-023-00428-9

[18] Jose Ramón Prieto, José Andrés, Emilio Granell, Joan Andreu Sánchez, and Enrique Vidal. 2023. Information extraction in handwritten historical logbooks. *Pattern Recognition Letters* 172 (2023), 128–136. https://doi.org/10.1016/j.patrec.2023.06.008

[19] Stephen Quinn and William Roberds. 2009. *An economic explanation of the early Bank of Amsterdam, debasement, bills of exchange and the emergence of the first central bank.* Cambridge University Press, 32–70.

[20] Adrian Rosebrock. 2022. Multi-Column Table OCR - PyImageSearch. ^1^ Accessed: 2024-04-17.

[21] Mathias Seuret. [n. d.]. *Layout Analysis in Handwritten Historical Documents.* Chapter Chapter 4, 45–65. https://doi.org/10.1142/9789811203244_0004 arXiv:https://www.worldscientific.com/doi/pdf/10.1142/9789811203244$_0$004

[22] Zejiang Shen, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson, and Weining Li. 2021. LayoutParser: A Unified Toolkit for Deep Learning Based Document Image Analysis. arXiv:2103.15348 [cs.CV]

[23] Brandon Smock, Rohith Pesala, and Robin Abraham. 2022. PubTables-1M: Towards Comprehensive Table Extraction From Unstructured Documents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4634–4642.

[24] Enrique Vidal, Alejandro H. Toselli, Antonio Ríos-Vila, and Jorge Calvo-Zaragoza. 2023. End-to-End page-Level assessment of handwritten text recognition. *Pattern Recognition* 142 (2023), 109695. https://doi.org/10.1016/j.patcog.2023.109695

[25] Yiheng Xu, Tengchao Lv, Lei Cui, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, and Furu Wei. 2021. LayoutXLM: Multimodal Pre-training for Multilingual Visually-rich Document Understanding. arXiv:2104.08836 [cs.CL]

Appendix A FIRST APPENDIX

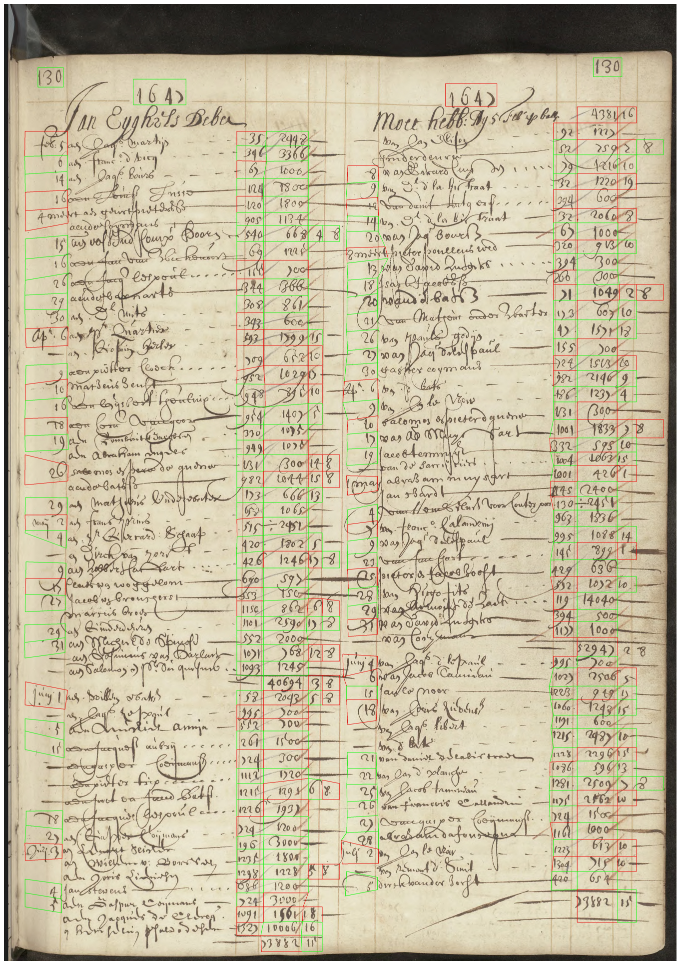


Figure 8: An example page showing the words correctly matched by the HTR system in green and the words wrongly matched in red. This can be helpful in visually inspecting the page to see whether there are common characteristics on parts of the page that could cause mistakes.

Put your appendices here.