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 17.06.2024

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

ABSTRACT

Handwritten text recognition is a crucial step in digitizing historical documents. In this work, we will explore the challenges associated with a handwritten text recognition model using historical ledger pages from the Bank of Amsterdam. We first conduct an error analysis of the model, and based on that make suggestions on where and how the performance of the model could be improved. The main concern is the model's inability to capture the layout of the pages. We perform post-processing steps to improve the output of the model and we create a layout analysis method, which aims to capture the distinctive tabular layout found in the ledger pages. While earlier research mainly focuses on capturing different elements on documents, such as text, tables and images, our work specifically focuses on capturing table cells and contents from a tabular structure. Our layout analysis method shows promising results, yet it faces challenges in accurately matching the ground truth bounding boxes. This discrepancy shows the difficulties in evaluating complex document structures. Lastly, we propose a method for training a deep learning model using pseudo-annotations which shows that the quantity of pseudo-annotations can compensate for the noise present in them and achieve promising results.

KEYWORDS

HTR, Layout analysis, Computer vision, Historical documents, Object detection

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 [18]. 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:

To what extent can the categorization of errors in handwritten ledger analysis be used to enhance the identification and resolution of HTR model 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 different processing methods improve HTR output in handwritten ledgers?

Since we perform a number of different experiments, our report follows the following structure: we start by comparing some related work. Then we will provide a description of the dataset. After that we look at the different experiments, where for each we provide our methodology including evaluation steps as well as results and discussion of that particular experiment. We end with a more general discussion and conclusion.

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 biggest 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 [19]. 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 [6]. 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 [6].

Multiple other studies describe the use of transformer based methods to perform layout analysis and optical character recognition (OCR) [8, 11, 23]. 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 [21]. 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 [1].

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 [4]. 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 processing to detect layout on a document [7]. 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 [7].

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 [2]. 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 [10]. 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 [10].

Liang et al. describe the use of Hessian and Gabor filters in order to find table structures on a document [9]. 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 [17]. 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 but often require many training data.

2.2 HTR Post-Processing Techniques

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 [16]. The use of a dictionary for allowed characters is a shown to be a good strategy to eliminate HTR errors [7]. Other studies show the use of language models to correct word output and spelling in HTR systems [13, 16].

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 [22]. A strategy to mitigate this 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 [15]. It is then possible to use the WER and CER to calculate the accuracy.

3 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 7 shows an example of what a page looks like. Our dataset is split into multiple subsets that come from different books and that all have slightly varying pages, while still having structural similarity. Figure 1 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 2

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 8 shows the PageXML data of the ground truth and HTR system plotted on the original image for part of one page in our dataset.

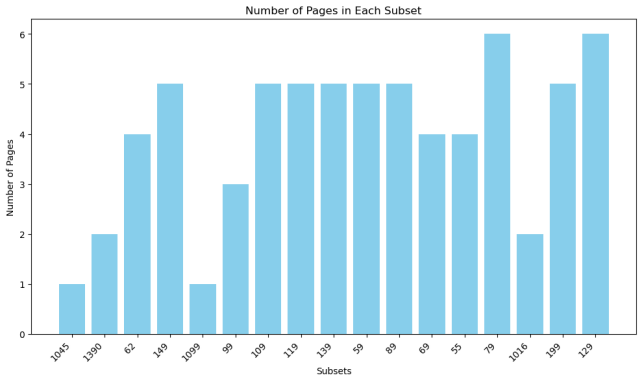


Figure 1: The amount of pages for each subset in our dataset.

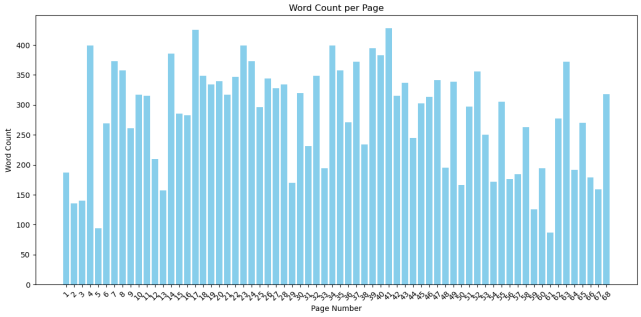


Figure 2: The word count for each page in our dataset.

4 HTR SYSTEM ERROR ANALYSIS

4.1 Method

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 [15]. 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 [5]. 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.

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 9. 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.

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 approach for these possible solutions is outlined in further sections.

4.2 Evaluation

For the evaluation of 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 formulas for these metrics are:

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

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 will 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:

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

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

4.3 Results

The accuracy of the current HTR system using the aforementioned metrics is shown in Table 1. As can be seen, the accuracy of the base model is not flawless. The recall is quite high, showing that not a lot of false negatives are found. However, the precision is lower, showing that the models incorrectly captures text. Figure 3 shows the total amount of character insertions, deletions, substitutions and matches made by the HTR system for the complete dataset. Here, we can clearly see that substitutions and mainly insertions of characters are quite high while deletions are relatively low. This further shows that the HTR system captures a relatively high amount of false positives, meaning that there are a lot of characters captured that are not actually on the page. Figure 4 shows this in more detail for each of our subdirectories. Here, we have 4 subfigures showing the insertions, deletions, substitutions and matches as percentage of total characters in each subdirectory. This shows that the accuracy of the HTR model can vary quite a lot per subdirectory. Furthermore, it shows that the amount of insertions and substitutions are consistently higher than deletions except for one outlier in subdirectory 1390, where there are a lot more deletions. In order to improve the model, it is thus crucial to find a way to mainly reduce the insertions and substitutions made by the model. Figure 5 shows the frequencies of different characters inserted, substituted and deleted. Here we can see that a lot of 'invalid' characters are inserted. These are non-alphanumeric characters that should not be part of our dataset. Furthermore, we see that many numbers and letters that look a like (such as i and 1) are substituted. We use these analyses and the fact that the HTR system is not able to capture the layout of the pages to conduct a list of errors made by the HTR system. A selection of the most common errors and possible solutions is shown in Table 2. In further sections, we will use the results of this section to improve the model.

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

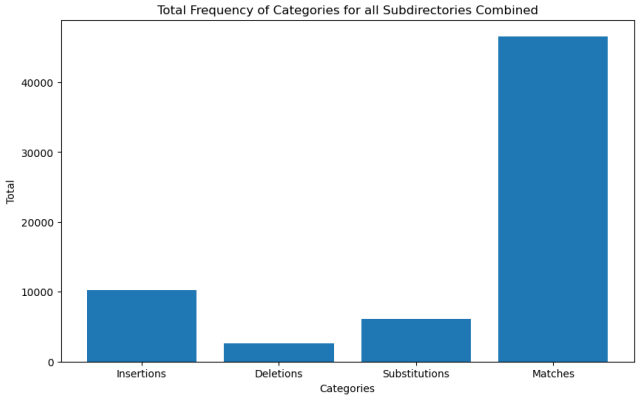
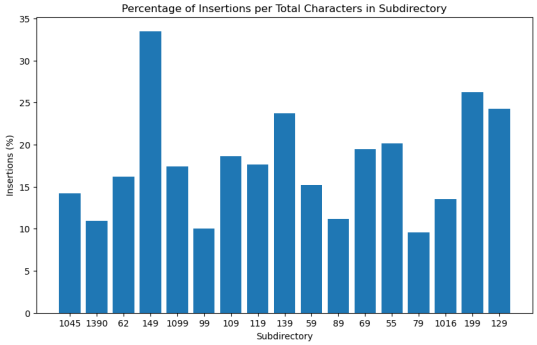
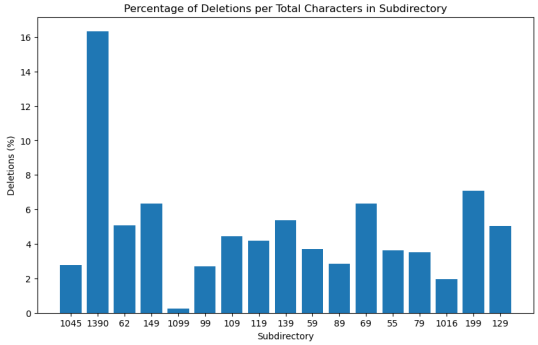


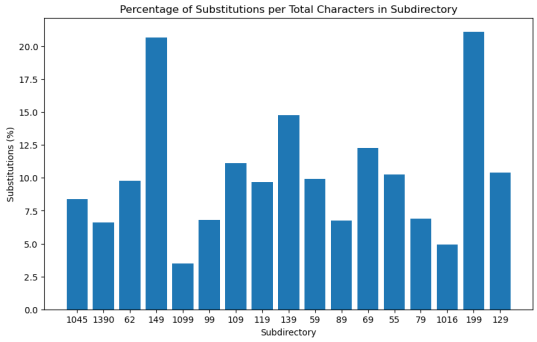
Figure 3: The total amount of insertions, deletions, substitutions and matches made by the HTR sytsem compared to the ground truth values. The values are on a character level and made word per word.



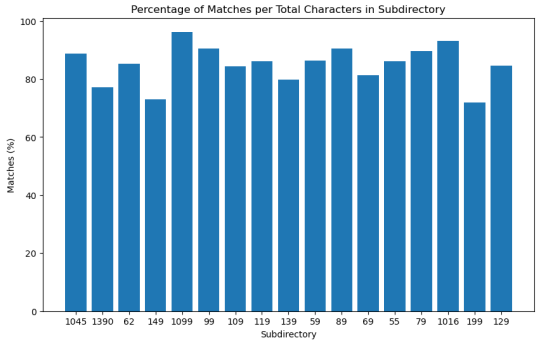
(a) Amount of insertions made by the HTR system per subdirectory.



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



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

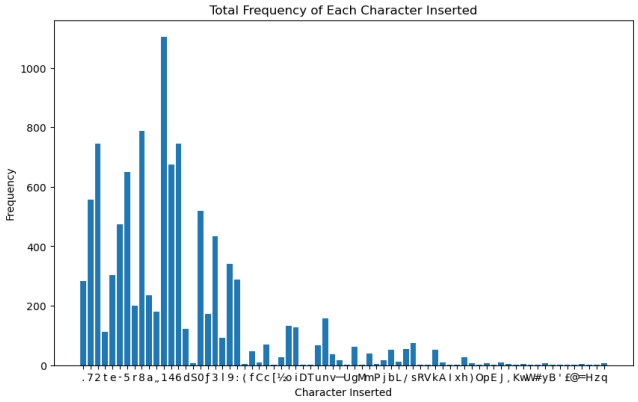


(d) Amount of matches made by the HTR system per subdirectory.

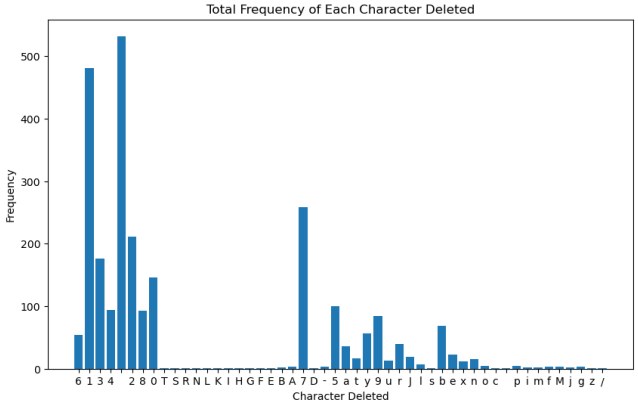
Figure 4: The amount of insertions, deletions, substitutions and matches made by the HTR system compared to the ground truth values per subdirectory. The values are on a character level and made word per word and as a percentage of total characters in ground truth.

Table 2: Categorization of HTR System Errors

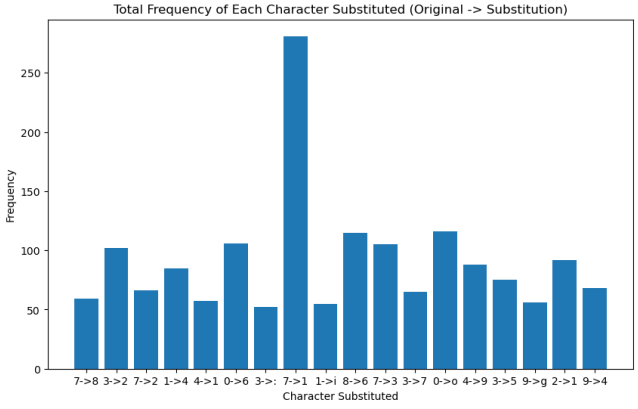
Error Type	Description	Possible Solution
Deletions	Characters not recognized at all (false negative)	Improve HTR system
Insertions	Characters recognized that are not there (false positives)	Improve HTR system, post-processing
Substitutions	Characters recognized as another character	Improve HTR system, post-processing
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-processing
Invalid Characters	Characters that cannot occur	Post-processing



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

5 POST-PROCESSING

5.1 Method

From our error analysis, we find that using post-processing techniques may lead to improved accuracy. 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 [7, 16]. Analysis of the HTR output (as depicted in Figure 5a) reveals several insertions that are not valid letters or numeric characters. To address this, we employ regular expressions (regex) to filter out invalid characters from the HTR output. We are then left with alphanumeric characters only. Furthermore, the HTR system often substitutes letters for numbers, as can be seen in Figure 5c. We can undo these substitutions in some cases. For instance, when a single character is found to be a letter by the HTR system, we can expect this to be wrong, since single character words are never letters in the ground truth. Also, numbers that contain a letter (such as '11i0') are probably wrong. We use these facts and the most common made substitutions as found in our earlier analysis to recover the right characters. We use the same evaluation metrics as before to compare our method to the baseline scores.

5.2 Results

We compare the post-processed HTR output to the original HTR output to see the different accuracy scores. The accuracy scores for both the baseline and post-processed HTR output are shown in Table 3. We see a very slight decrease in recall while all other metrics improve. This shows that our post-processing strategy is successful in eliminating some of the false positives (insertions and substitutions) of the original model while maintaining almost the same rate of false negatives (deletions). However, we can still see that the precision is relatively low compared to the recall, showing that there is still a good amount of false positives that our method did not catch.

Table 3: WER, CER, precision and recall scores for the HTR system output with and without post-processing techniques applied.

Metric	Baseline	Post-Process
WER	0.4243	0.3916
CER	0.3422	0.3169
Recall	0.9475	0.9418
Precision	0.7403	0.7622
Total Edit Distance	18920	17508

6 LAYOUT ANALYSIS

6.1 Method

Up until this point, we have been comparing the HTR output to the ground truth values by matching their bounding boxes based on the highest IoU score. We did this because the HTR system currently is not able to capture the layout of the the tabular structure of

the ledger pages and thus we could not just match the output directly, since the word order would be wrong. The final part of the methodology involves analyzing the layout of the documents. Recognizing the layout is a crucial step in digitizing the ledger pages, since without it there is no good way to use the output of the HTR system because the correct word order would not be known. We will perform the layout analysis by using image processing techniques as well as making use of the HTR output. For this, we will use the OpenCV (cv2) library to perform the necessary processing steps.

To detect the tabular structure on the ledger pages, we must find a way to detect the columns and rows of the table to get the table cells. The ledger pages contain physical lines, which separate the different columns. We start by trying to detect these lines.

The first step is to pre-process our image. We do this by converting our image to grayscale and applying a Gaussian blur on the grayscale image. This helps reducing the computational complexity and reduces high frequency noise [12].

Next, we apply two types of thresholding to the blurred image: Otsu's thresholding and Adaptive Gaussian thresholding. Otsu's thresholding only captures the most clear markings on the page, such as the text, while the more thin and faint column lines are not captured [2]. The Adaptive Gaussian threshold captures both the text as well as the column lines. By subtracting the first threshold from the latter, we are left with the column lines and some noise.

Following thresholding, we perform morphological operations, specifically dilation and erosion [14]. We use specific kernels to remove noise while retaining the vertical lines.

We then apply the Hough Line Transform to detect the lines in the image. This can be used to detect (nearly) straight lines in images [7]. We filter the detected lines based on their length and position. For each detected line, we calculate its slope and intercept, and then draw an extended line from the top to the bottom of the foreground area (which contains the ledger page) in the image. This is done using basic principles of line equations in coordinate geometry.

Finally, we perform additional dilation and erosion to merge close lines and remove final noise. We then find contours in the image and draw a line from the top to the bottom of each contour. We are then left with an image containing the contours of each column section, which can then be used to create bounding boxes for the different columns. The whole process is shown in Figure 11.

To detect the rows present in the ledger pages, we have to use a different strategy, since there are no physical markings for row lines on the pages. We will use the word bounding boxes which are outputted by the HTR system in order to find the row bounding boxes.

We start by extracting the bounding boxes of the words outputted by the HTR system. We will only use the word bounding boxes that are kept after performing the post-processing. Since the images in our dataset typically consist of two tables (debit and credit) we have to determine where both tables start and end. We do this by finding the middle column line from the list of column lines we found by our method above. We then separate the bounding boxes into two groups: left boxes and right boxes, based on their x-coordinates relative to the middle column line.

We then take a list of boxes (either left boxes or right boxes) and group them into rows. This is done by sorting the boxes by their x-coordinates (from left to right) and then grouping boxes with similar y-coordinates together. For each box, we find the best row to append the box to, based on the minimum difference in y-coordinates. If no suitable row is found, we create a new row. The result of these steps are shown in Figure 12.

Once we have found the rows, we can extend the bounding boxes from the left side of the ledger page to the middle for the left boxes and from the middle to the right side of the page for the right boxes. We are now left with both the column and row bounding boxes.

Once the row and column bounding boxes are determined, we can use the intersection of these bounding boxes to determine the table cells. Figure 6 shows an example of these found table cells.

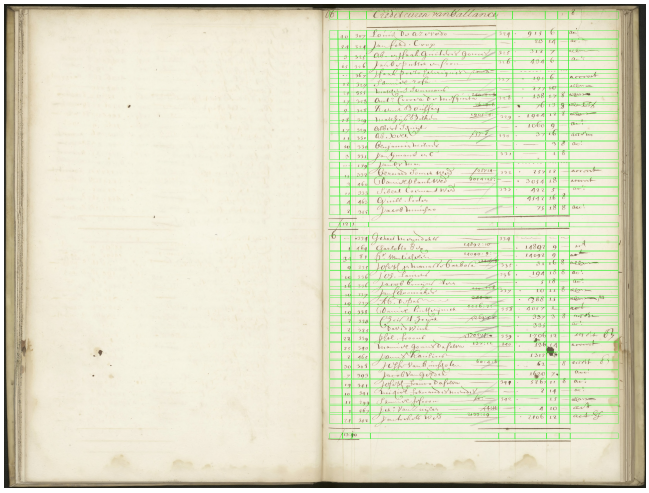


Figure 6: The bounding boxes of the predicted table cells after taking the intersection of the rows and columns.

6.2 Evaluation

For the layout analysis methods, we take the ground truth table cell bounding boxes and compare them to our method's predicted table cell bounding boxes. We do this by matching each ground truth cell to a predicted cell based on the highest IoU score between the two. Only one ground truth cell can be matched to one predicted cell and vice versa. We then determine whether cells are a true positive by setting an IoU threshold. If ground truth and predicted cells are matched with an IoU score higher than this threshold, we consider it a true positive. Ground truth cells that are not matched with an IoU score above the threshold are considered a false negative and predicted cells that are not matched with an IoU score above the threshold are considered a false positive. We will use a threshold of 0.5, 0.25 and >0. We can then determine the recall and precision scores, using the same formula as before. Additionally, we calculate the F1-score:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (5)$$

Furthermore, after matching the cells as described above, we further verify the correctness of the match by comparing the text content of the predicted table cells against the text content in the ground truth table cells. This done by taking the words from the HTR output that match location with the predicted cells. The text comparison is done using a normalized Levenshtein distance with various thresholds: 0 (exact match), 0.25, 0.5, and 0.75. We use these thresholds because we cannot be sure that words from the HTR output match exactly with ground truth words because of the imperfections of the HTR system. For each threshold, we calculate the mean accuracy of the word matches. Additionally, we calculate the CER for all the words in the ground truth cells and the words in the predicted cells. If the cells are indeed matched correctly, we would expect to see good mean accuracy of the word matches (especially for the lower thresholds) and we would expect the CER to be close to the value found in Section 4.

6.3 Results

The mean performances using IoU thresholds of 0.5, 0.25 and >0 are shown in Table 4. The 'IoU Score' indicates the mean score for the true positives found at this IoU threshold. As can be seen, when using a higher IoU threshold, the accuracy scores are low and when using a lower IoU threshold, the accuracy scores are better. This means that when using a lower IoU threshold, a lot of the ground truth table cells are matched with predicted cells while this is not the case when using a higher IoU threshold. One of the main reasons for this behaviour is the fact that the bounding boxes of our predicted cells are much more tight than the ground truth boxes. Furthermore, the bounding boxes for our predicted table cells are often placed slightly lower than the ground truth bounding boxes. This is because we use the word bounding boxes from the HTR model to determine the rows, and these bounding boxes are almost always found to be slightly lower than the actual location of the words on the image. This causes the IoU score of a predicted cell and ground truth cell to be relatively low, even when both capture the same contents of the page. When taking a IoU threshold of 0.25, we see the best ratio of IoU scores and accuracy scores. Here we see that matched cells have a mean IoU score of 0.473, which is quite close to the usual threshold of 0.5. Furthermore, we see a score of 0.804, 0.898 and 0.844 for mean precision, recall, and F1-score respectively. This shows that we have a good rate of false positives and false negatives, as well as having a reasonable area overlap for predicted and ground truth cells. To get a more detailed picture of these results on subdirectory level, we show the results for the IoU tresholds 0.5, 0.25 and >0 in Appendix C in Table 8, Table 9 and Table 10 respectively.

The accuracy of comparison of text contents for ground truth and predicted cells can be seen in Table 5. Here, 'LT 0' until 'LT 0.75' means how much percent of the text in the ground truth and predicted cells differ. So for the first row, 0.5369 means that 53.69% of texts are matched exactly, 0.5996 means that 59.96% of texts are matched with at most 25% of characters changed etc. If we look at the IoU threshold of 0.25, which seemed to perform well in the previous result, we can see that about 85% of words are matched with at most 75% of the characters differing. Furthermore we see a CER of 0.3859. This is higher than the CER found in Section 4. However, this can partly be explained due to the fact that in the first experiment, we calculated CER by matching one HTR word with a ground truth cell text. That means that we discarded some of the false positives (HTR words that were not matched). In this case, we first append all HTR words into a predicted cell and then compare predicted cell content to ground truth cell content, meaning that in this case we do keep these false positives when comparing the texts. Keeping this in my mind, we can determine that most of the predicted cells and ground truth cells that are matched by IoU, are also matched based on text contents, showing the success of our method.

Table 4: Mean table cell performance metrics at different IoU Thresholds. 'IoU Score' indicates the mean score for the true positives found at this IoU threshold.

IoU Threshold	IoU Score	Precision	Recall	F1
>0	0.182	0.845	0.946	0.887
0.25	0.473	0.804	0.898	0.844
0.5	0.627	0.416	0.459	0.434

Table 5: Mean accuracy and CER for matching ground truth table cells with predicted table cells using different IoU thresholds and comparing their text contents using normalized Levenshtein distance thresholds. Values for LT 0, 0.25, 0.50, and 0.75 indicate the mean accuracy at which the normalized Levenshtein distance between the ground truth and predicted texts is within 0 (total match), 25%, 50%, and 75% respectively. CER indicates the average character error rate for the text content in the matched cells.

IoU Thresh.	LT 0	LT 0.25	LT 0.50	LT 0.75	CER
>0	0.5369	0.5996	0.7333	0.8242	0.4290
0.25	0.5555	0.6213	0.7563	0.8469	0.3859
0.50	0.5898	0.6586	0.7984	0.8770	0.3362

7 LAYOUT PARSER

7.1 Method

The final experiment of our research involves the use of a deep learning model to detect the columns on our ledger pages. If we are able to create an accurate model to detect and segment different columns, we can use this to run our HTR model for each column. This could potentially improve the HTR output by eliminating overlapping word bounding boxes on different columns aswell as allowing the model to run in numeric or alphabetic mode depending on the column.

Creating annotations for these pages is a tedious task and since we only have a relatively small dataset, we cannot use this to create a train-test split. Instead, we will use an approach similar to the one

used by Chen et al. [3]. This means will we use our approach to detect columns as described in Section 6 to create pseudo-annotations for columns and use these as the training set for our deep learning model. As shown in the previous section, the accuracy of this rule based method can vary between the different subdirectories so there exist certain outliers where the method does not perform well. By creating pseudo-annotations using our previous method for enough pages, we aim to get a more accurate and robust model, that might perform better on these outliers.

We will test this by training two models, one with a smaller pseudo-annotated train set consisting of 691 pages and one with a larger train set consisting of 1861 pages. We will test our model on the same hand-annotated set as before. Since our pseudo-annotations are not flawless, this experiment can help us determine if the sheer volume of train data can compensate for the noise introduced by inaccuracies in the annotations and create a more generalizable model.

For this experiment, we will make use of the Layout Parser Tool [20], which consists of multiple pre-trained layout detection models. Specifically, we will use the mask_rcnn_X_101_32x8d_FPN_3x model trained on the PubLayNet dataset and use transfer learning for our training. The training and evaluation is performed using a NVIDIA Tesla T4 GPU. For the training and evaluation of the model, we started with the basic configuration and made several adjustments to optimize performance for our specific use case. The changed hyperparameters are shown in Table 6. An example of the output created by this model (trained on the large dataset) is shown in Appendix A in Figure 10.

Table 6: Hyperparameter settings for model training.

Hyperparameter	Value
DATALOADER.NUM_WORKERS	2
SOLVER.IMS_PER_BATCH	2
MODEL.ROI_HEADS.BATCH_SIZE _PER_IMAGE	256
MODEL.ROI_HEADS.NUM_CLASSES	1
SOLVER.WEIGHT_DECAY	0.0005
SOLVER.BASE_LR	0.001
SOLVER.LR_SCHEDULER_NAME	"WarmupMultiStepLR"
SOLVER.STEPS	(333, 666)
SOLVER.WARMUP_ITERS	100
SOLVER.MAX_ITER	1000

7.2 Evaluation

For the evaluation of this experiment, we will use the standard Common Objects in Context (COCO) evaluation metrics. We will use the average precision (AP) calculated over a range of IoU threshold from 0.50 to 0.95 aswell as show the average precision at IoU threshold 0.5 (AP50) and 0.75 (AP75). We will show the results both for the bounding box aswell as the segmentation (mask) of the columns.

7.3 Results

The results are shown in Table 7. We can see that at iteration 1000, the AP stops improving much or even decreases. We see that initially the model trained on the small dataset seems to perform better at lower iterations. After iteration 600, the performance of the model trained on the small dataset decreases whereas the model trained on the large dataset improves further. This suggests that the model trained on the small dataset converges faster initially. The best scores for the model trained on the large dataset are at iteration 1000 with AP 57.750, AP50 88.515 and AP75 62.306. For the model trained on the smaller dataset, the best performance is at iteration 600 with AP 56.366, AP50 84.020 and AP75 61.875.

The large dataset likely contains more diverse examples, enabling the model to generalize better and avoid overfitting, even with extended training. This is reflected in the sustained improvement in metrics up to iteration 1000. The small dataset might lack sufficient diversity, leading to quicker initial convergence but also to overfitting with longer training. This shows that when supplemented with a larger volume of data, the training process is improved. The large dataset might help mitigate some of the noise present in the pseudo-annotations, leading to better overall performance.

Table 7: Comparison of evaluation results for bbox and segmentation metrics of columns between large and small datasets at different iterations.

Iteration	Dataset	Metric	AP	AP50	AP75
200	Large	BBox	41.937	78.181	39.855
		Segm	39.311	76.370	36.690
	Small	BBox	43.818	75.341	47.760
		Segm	43.382	75.232	44.457
400	Large	BBox	54.308	86.919	58.568
		Segm	51.570	86.647	53.775
	Small	BBox	53.752	83.035	58.756
		Segm	51.680	82.202	56.379
600	Large	BBox	56.889	89.015	62.029
		Segm	53.364	87.851	57.041
	Small	BBox	56.366	84.020	61.875
		Segm	53.025	83.152	59.184
800	Large	BBox	57.702	88.816	62.442
		Segm	53.945	88.389	56.905
	Small	BBox	54.212	83.443	58.479
		Segm	51.661	82.009	56.954
1000	Large	BBox	57.750	88.515	62.306
		Segm	53.948	88.186	57.151
	Small	BBox	54.022	83.344	58.046
		Segm	51.926	81.720	57.215

8 DISCUSSION

In this work, we have analyzed a HTR system and found what the main errors of the system are. We have then proposed and implemented various methods for the different set of errors in the system.

8.1 Analysis of Findings

What factors contribute to text recognition errors in handwritten ledgers? The primary factors contributing to errors in text recognition include the variability in handwriting styles, the quality of the original documents and the presence of noise such as stains and faded ink. Our analysis revealed that line spacing and overlapping text further worsen the recognition accuracy.

What are the different types of errors encountered in handwritten text recognition? We identified several categories of main errors in handwritten text recognition:

- Substitution Errors: Where one character is incorrectly recognized as another.
- Insertion Errors: Where extra characters are included in the output.
- Deletion Errors: Where characters are omitted in the output.
- Segmentation Errors: Where the system fails to accurately distinguish between distinct text regions or words or the bounding boxes of found words are inaccurate.
- Layout Error: The layout of the page is not detected which causes the word order to be wrong.

How can different processing methods improve HTR output in handwritten ledgers? Our research proposed various methods to mitigate the identified errors:

Post-processing We implemented a post-processing technique where we created a dictionary which we used to filter out excessive characters. Furthermore, we showed that it is possible to undo substitutions of characters in specific situations. These steps allowed us to get a better recognition output of the HTR model.

Layout Analysis We implemented a rule-based method that is able to extract columns and rows from the ledger pages and thus recognize the table cells with an mean F1-score of 0.844 and mean IoU value of 0.473 when using an IoU threshold of 0.25. This method improves the layout error where the word order was not recognized at all by the HTR system. We show a CER of 0.3859 when matching predicted table cells with ground truth cells based on an IoU threshold of 0.25, which is not far of the CER found in the initial error analysis, showing the effectiveness of our layout analysis.

We have also shown that a deep learning model such as Layout Parser can be effective in recognizing the columns of our ledger pages. We have created pseudo-annotations based on our rule-based method and used these to train our model. We have shown that even though noise is present in these pseudo-annotations, they can still be effective in training a model when using a large quantity. The model showed to have promising accuracy on the test set.

8.2 Limitations

Our rule-based method requires document homogeneity to work effectively. Variations in document structure and quality can significantly impact the performance of the method used. While almost all documents follow the same structure, there can exist outliers.

Furthermore, the evaluation of the table cell layout detection in Section 6 proved to be difficult using a standard IoU threshold of 0.5 due to the fact that our rule-based method did not always accurately match the ground truth table cells. Our method did perform well using a lower threshold, showing a mean IoU value of 0.473 for the matched cells, just below the usual threshold of 0.5. While we tried to support our evaluation by also comparing text contents, this in itself is again based on a different threshold. While this still gives a good idea of our method's performance, it does not completely validate the accuracy of the cell layout detection.

8.3 Future Work

This work aims to detect common errors in a HTR model and lays a foundation to solve some of these errors. Future research could focus on the integration of the layout detection methods into the actual HTR model to create a structured pipeline for text detection. This could also involve testing whether running the HTR model on each column seperately (either in numeric or alphabetic mode) would improve the accuracy of the output of the HTR model. Furthermore, it could be worthwhile to investigate whether models using a different architecture perform better on a layout detection task like this one. Lastly, the deep learning model trained with pseudo-annotations could be fine-tuned with hand annotated data to further improve its performance.

9 CONCLUSION

This work aimed to address the challenges associated with the digitization of handwritten ledgers, with a specific focus on error identification and resolution within HTR systems. Our main research question aimed to explore the extent to which categorizing errors in handwritten ledger analysis can enhance the identification and resolution of the HTR model inaccuracies.

Through a series of experiments, we identified key factors contributing to HTR errors and proposed various methods to improve the output of the model. Our analysis revealed several types of errors, including issues related to layout detection and text recognition. By implementing post-processing techniques and layout analysis methods, we observed notable improvements in the accuracy of the HTR output.

Despite these results, our study faced several limitations. One significant limitation was the difficulty in evaluating the table cell layout detection using a standard IoU threshold of 0.5. Our rule-based method did not consistently match the ground truth table cells, performing better with a lower IoU threshold. This indicates that while our method provides a good approximation of performance, it does not fully validate the accuracy of the cell layout detection.

In conclusion, this work contributes to the ongoing efforts to digitize historical handwritten documents by providing insights into the causes of HTR errors and proposing methods to mitigate them. While our findings show promising results for improving HTR accuracy, further research is necessary to fully overcome the challenges identified. By addressing the limitations and possibly exploring new methodologies, future work can build upon this work to improve the performance of HTR systems in digitizing historical documents.

REFERENCES

[1] 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

[2] 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

[3] Hsiang-Chieh Chen and Zheng-Ting Li. 2021. Automated Ground Truth Generation for Learning-Based Crack Detection on Concrete Surfaces. *Applied Sciences* 11, 22 (2021). https://doi.org/10.3390/app112210966

[4] 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

[5] 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

[6] 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]

[7] 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

[8] 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]

[9] 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

[10] 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

[11] 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]

[12] Siddharth Misra and Yaokun Wu. 2020. Chapter 10 - Machine learning assisted segmentation of scanning electron microscopy images of organic-rich shales with feature extraction and feature ranking. In *Machine Learning for Subsurface Characterization*, Siddharth Misra, Hao Li, and Jiabo He (Eds.). Gulf Professional Publishing, 289–314. https://doi.org/10.1016/B978-0-12-817736-5.00010-7

[13] 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

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

[15] 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

[16] 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

[17] 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

[18] 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.

[19] 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

[20] 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]

[21] 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.

[22] 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

[23] 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 SUPPLEMENTARY FIGURES

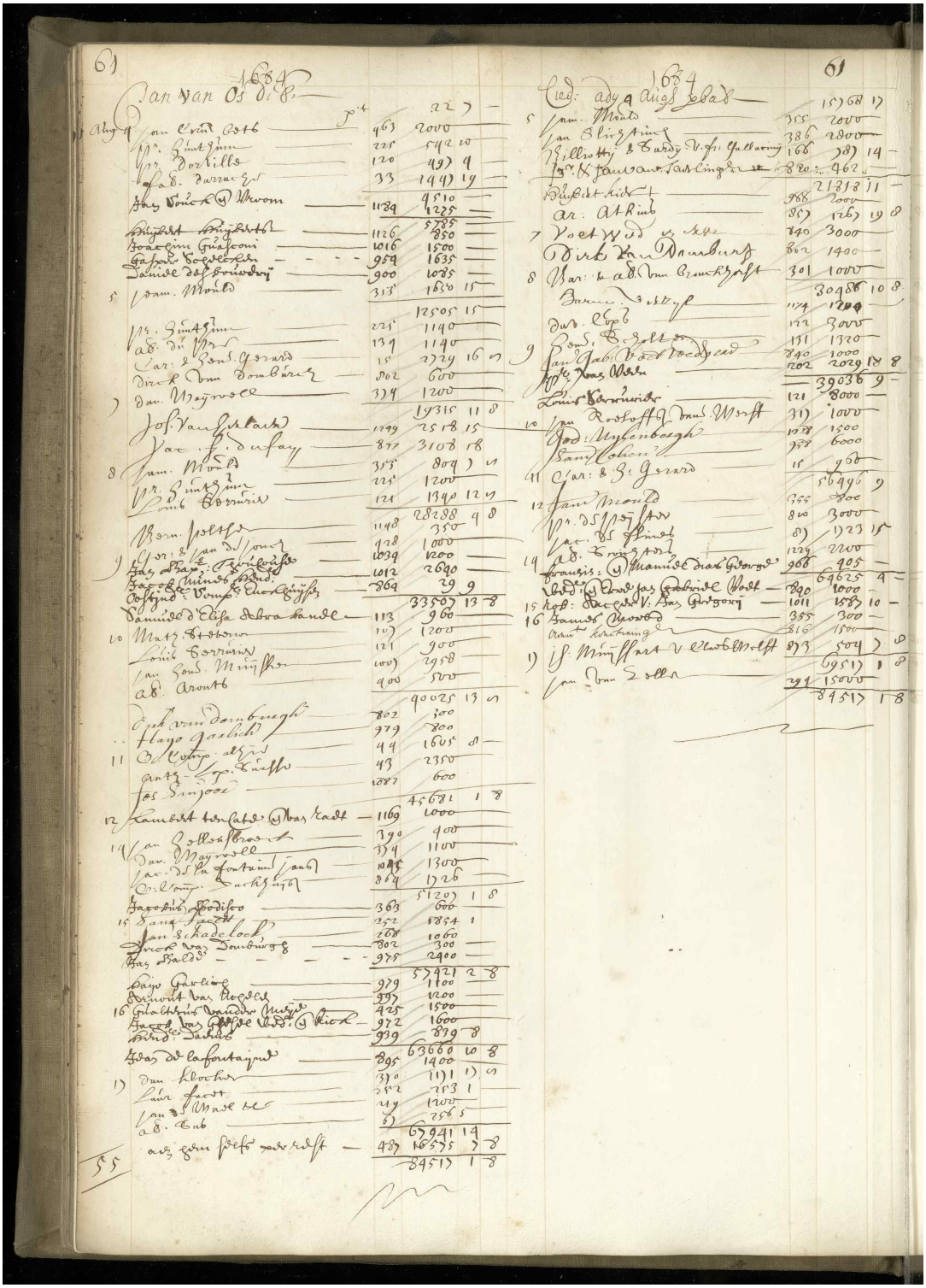
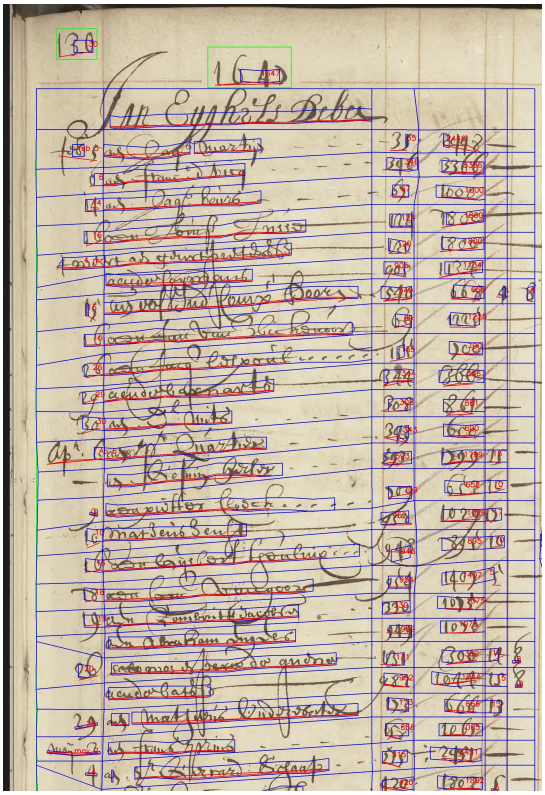
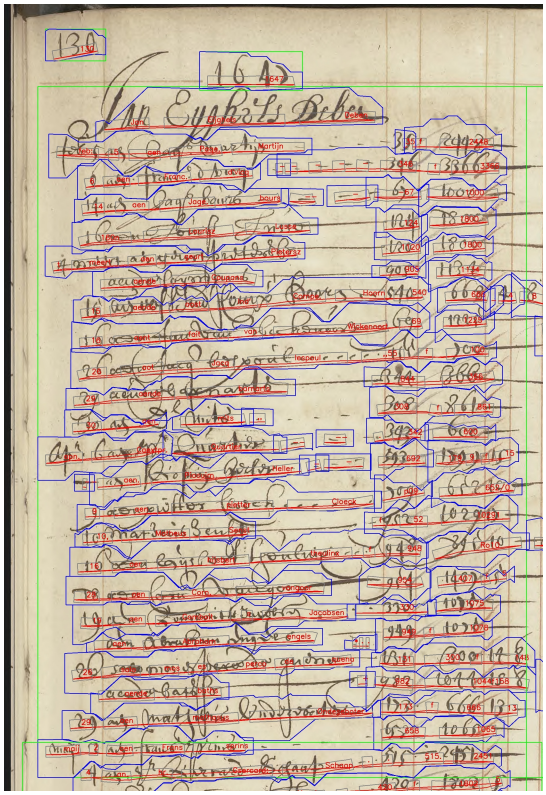


Figure 7: An example page from the ledger collection.



(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 8: 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.

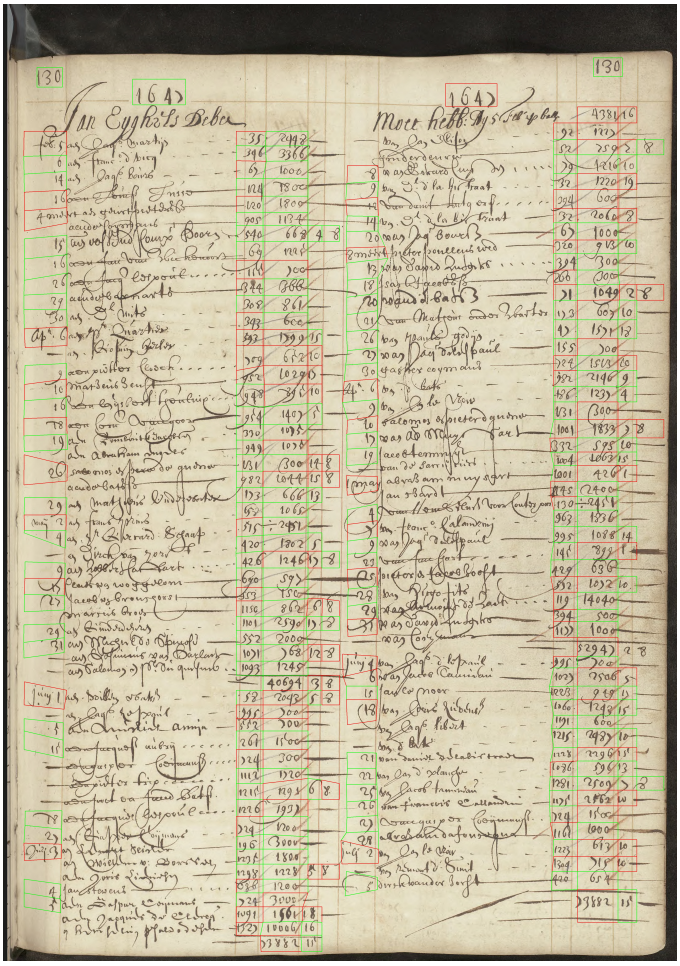


Figure 9: An example page showing the words correctly matched by the HTR system in green and the words wrongly matched in red. This can be helpful in visually inspecting the page to see whether there are common characteristics on parts of the page that could cause mistakes.

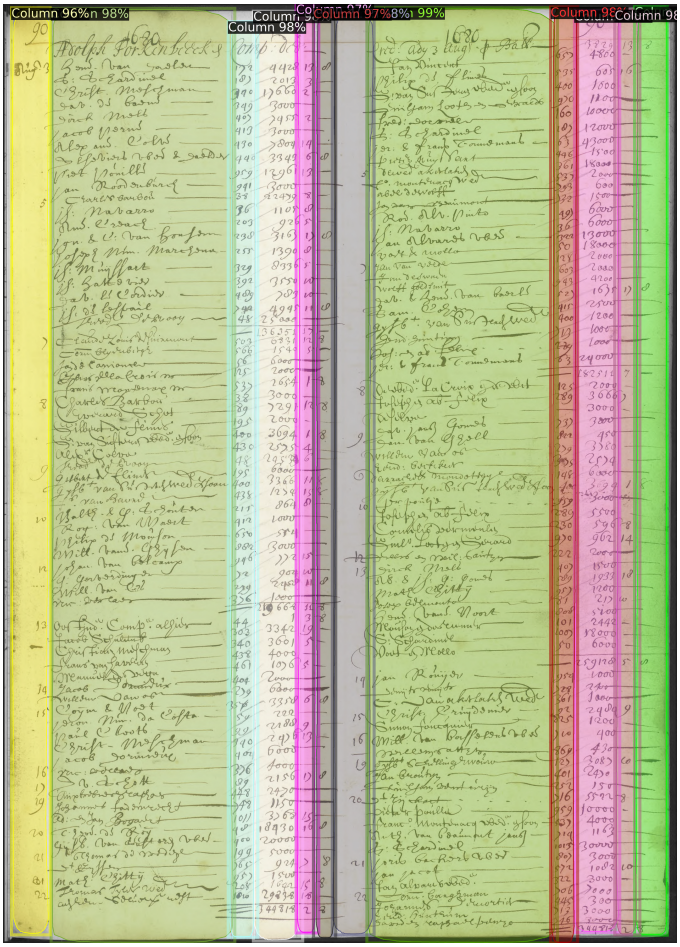


Figure 10: An example page showing a prediction of columns made by the Layout Parser model trained on our large dataset with pseudo-annotations.

Appendix B DETAILED METHODOLOGY

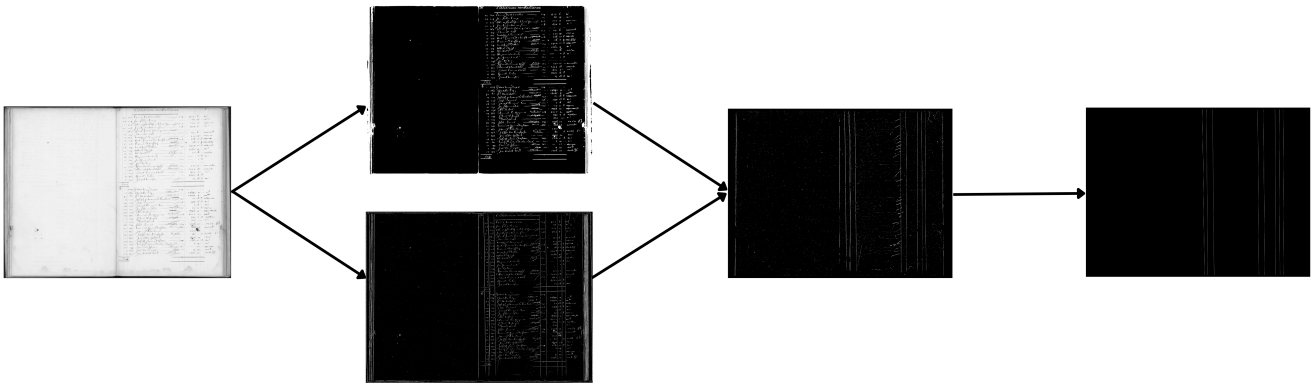


Figure 11: The process of the extraction of column lines. We start with the blurred grayscale image. From there we create two threshold images, one containing the faint column lines and one that does not contain them. We subtract the images from each other, remove noise and use Hough line detector to find the columns lines.

Figure 12: The process of extracting the rows from the ledger page. We use the HTR system to determine bounding boxes of words from the original page. We then determine the row bounding boxes based on the y-coordinates of the word bounding boxes.

Appendix C COMPLEMENTARY RESULT TABLES

Table 8: Table Cell Performance Metrics at IoU Threshold 0.5.

Subdir	IoU Score	Precision	Recall	F1
1045	0.638	0.674	0.674	0.674
1390	0.641	0.303	0.468	0.363
62	0.627	0.437	0.487	0.460
149	0.608	0.310	0.383	0.342
1099	0.621	0.532	0.499	0.515
99	0.638	0.525	0.504	0.514
109	0.645	0.520	0.552	0.534
119	0.617	0.422	0.437	0.429
139	0.626	0.435	0.475	0.453
59	0.664	0.555	0.541	0.548
89	0.632	0.466	0.519	0.490
69	0.626	0.391	0.456	0.420
55	0.631	0.287	0.319	0.302
79	0.614	0.389	0.439	0.412
1016	0.661	0.637	0.635	0.636
199	0.594	0.243	0.339	0.278
129	0.617	0.342	0.404	0.369
Mean	0.627	0.416	0.459	0.434

Table 9: Table Cell Performance Metrics at IoU Threshold 0.25.

Subdir	IoU Score	Precision	Recall	F1
1045	0.546	0.990	0.990	0.990
1390	0.498	0.513	0.785	0.613
62	0.471	0.854	0.959	0.902
149	0.440	0.710	0.879	0.784
1099	0.473	0.939	0.881	0.909
99	0.504	0.916	0.878	0.896
109	0.508	0.848	0.894	0.867
119	0.460	0.869	0.901	0.885
139	0.474	0.819	0.894	0.853
59	0.525	0.910	0.888	0.899
89	0.475	0.862	0.961	0.907
69	0.464	0.821	0.963	0.884
55	0.451	0.726	0.811	0.765
79	0.452	0.829	0.929	0.875
1016	0.553	0.835	0.831	0.832
199	0.432	0.639	0.894	0.731
129	0.453	0.738	0.878	0.801
Mean	0.473	0.804	0.898	0.844

Table 10: Table Cell Performance Metrics at IoU Threshold >0

Subdir	IoU Score	Precision	Recall	F1
1045	0.196	1.000	1.000	1.000
1390	0.152	0.603	0.952	0.729
62	0.190	0.876	0.984	0.925
149	0.169	0.762	0.943	0.841
1099	0.206	0.945	0.886	0.915
99	0.177	0.960	0.920	0.939
109	0.188	0.885	0.933	0.905
119	0.178	0.906	0.939	0.922
139	0.185	0.860	0.938	0.896
59	0.192	0.946	0.923	0.934
89	0.186	0.873	0.974	0.919
69	0.195	0.839	0.984	0.903
55	0.172	0.820	0.911	0.862
79	0.182	0.863	0.967	0.911
1016	0.232	0.845	0.840	0.841
199	0.169	0.690	0.962	0.788
129	0.164	0.802	0.952	0.869
Mean	0.182	0.845	0.946	0.887