Digitizing Historical Balance Sheet Data: A Practitioner's Guide[*]

Sergio Correia[†] Stephan Luck[‡]

December 4, 2021

Abstract

This paper discusses how to successfully digitize large-scale historical micro-data by augmenting optical character recognition (OCR) engines with pre- and post-processing methods. Although OCR software has improved dramatically in recent years due to improvements in machine learning, off-the-shelf OCR applications still present high error rates which limits their applications for accurate extraction of structured information. Complementing OCR with additional methods can however dramatically increase its success rate, making it a powerful and cost-efficient tool for economic historians. This paper showcases these methods and explains why they are useful. We apply them against two large balance sheet datasets and introduce `quipucamayoc`, a Python package containing these methods in a unified framework.

JEL Classification: C81, C88, N80

Keywords: OCR, Data Extraction, Balance Sheets

[*]This paper expresses the views of the authors and not necessarily those of the Board of Governors of the Federal Reserve. We benefited from discussions with Eugene White and Tom Zimmermann, as well as conference participants at the 2021 Methodological Advances in the Extraction and Analysis of Historical Data conference.

[†]Federal Reserve Board, sergio.a.correia@frb.gov

[‡]Federal Reserve Bank of New York, stephan.luck@ny.frb.org

1 Introduction

OCR software has improved dramatically in recent years due to improvements in machine learning techniques. It is a powerful tool to allow researchers to unlock extract data, including previously unavailable historical micro-data. However, off-the-shelf OCR applications still present high error rates which limit their applications for accurate extraction of structured information.[1]

In this paper, we discuss how to successfully digitize large-scale historical micro-data by augmenting optical character recognition (OCR) engines with pre- and post-processing methods. We argue that when applying OCR, *the researcher is the practitioner*. The success of digitizing in most cases does not depend on developing OCR engines themselves. Rather, successful data digitization results from complementing commercial OCR with additional methods which can dramatically increase its success rate, making it a powerful and cost-efficient tool for economic historians. We show how these methods are useful by applying them to two large balance sheet datasets. Further, we introduce `quipucamayoc`, a Python package containing our methods in a unified framework. The methods we developed are collected in our `quipucamayoc` Python package, which we are constantly improving and planning to open source together with the publication with the draft of this paper.

We start out in Section 2 by discussing the general trade-offs a researcher faces when considering using OCR to digitize data. A key takeaway is that OCR is not always the solution for every data digitization project. OCR only becomes useful when the underlying data source is sufficiently large and the structure of the data sufficiently standardized. Thus, a researcher should carefully weigh the cost and benefits of using OCR before embarking on setting up a data digitization workflow.

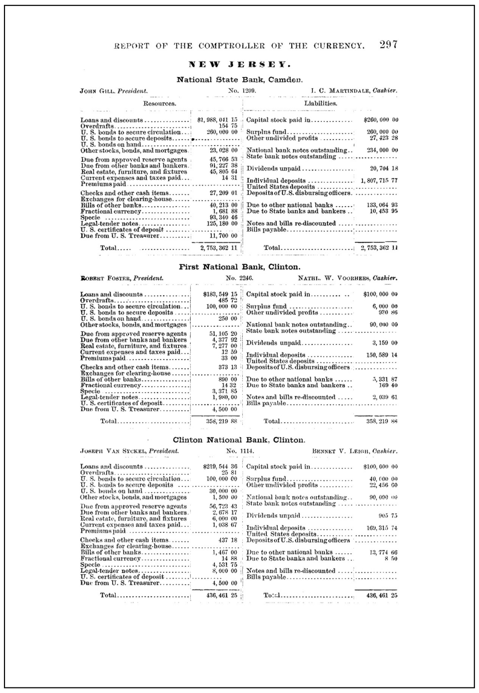
If the researcher concludes that OCR is the most promising route to obtain her data, we provide a classification and description of the general methods we recommend using. We start with the premise that commercial solutions are *not too expensive* compared to manual data entry. Rather, commercial products should be integrated and combined in a way that serves the researcher's purpose. We suggest following a "data extraction pipeline" that has the following steps: In the first step, the original image files are pre-processed (de-warped, contrast adjustments, etc.). In the second step, the commercially available OCR and layout recognition techniques are applied. Third, the data are extracted and validated by leveraging relationships that must hold in the data such as

[1]For instance, applying an OCR engine with a 95% character accuracy rate to a table of ten six-digit numbers will produce an error-free output in only 4% of cases.

accounting identities. A crucial step is a human review in which the researcher herself validates some of the data creating a "ground truth". Such validated data allows researchers to then test and improve the accuracy of the digitization pipeline, by serving as a benchmark against which the digitization output can be compared to construct accuracy metrics. In turn, these metrics allows for more advanced optimization of the parameters used through the digitization process.

We showcase how to apply the data extraction pipeline on two large-scale balance sheet digitization projects:

1. The Office of the Comptroller of the Currency's (OCC) Annual Reports between 1867 and 1904, containing more than 100,000 national bank balance sheets in tabular format (Figure 1a). These data are used, e.g., in Carlson et al. (2022).
2. 30,000 balance sheets of German financial and non-financial firms from 1915 through 1933, published by *Saling's Börsen-Papiere* as text paragraphs (Figure 1b). These data are used in ongoing research, Brunnermeier et al. (2021).



REPORT OF THE COMPTROLLER OF THE CURRENCY. 297

NEW JERSEY.

National State Bank, Camden.

JOHN GILL, *President*. No. 1209. I. C. MARTINDALE, *Cashier*.

Resources.		Liabilities.	
Loans and discounts	$1, 988, 041 15	Capital stock paid in	$260, 000 00
Overdrafts	154 75		
U. S. bonds to secure circulation	260, 000 00	Surplus fund	260, 000 00
U. S. bonds to secure deposits		Other undivided profits	27, 423 28
U. S. bonds on hand			
Other stocks, bonds, and mortgages	23, 028 00	National bank notes outstanding	234, 000 00
Due from approved reserve agents	45, 766 53	State bank notes outstanding	
Due from other banks and bankers	91, 227 38		
Real estate, furniture, and fixtures	45, 805 64	Dividends unpaid	20, 704 18
Current expenses and taxes paid	14 31		
Premiums paid		Individual deposits	1, 807, 715 77
Checks and other cash items	27, 209 01	United States deposits	
Exchanges for clearing-house		Deposits of U.S. disbursing officers	
Bills of other banks	40, 213 00	Due to other national banks	133, 064 93
Fractional currency	1, 681 88	Due to State banks and bankers	10, 453 95
Specie	93, 340 46		
Legal-tender notes	125, 180 00	Notes and bills re-discounted	
U. S. certificates of deposit		Bills payable	
Due from U. S. Treasurer	11, 700 00		
Total	2, 753, 362 11	Total	2, 753, 362 11

First National Bank, Clinton.

ROBERT FOSTER, *President*. No. 2246. NATHL. W. VOORHEES, *Cashier*.

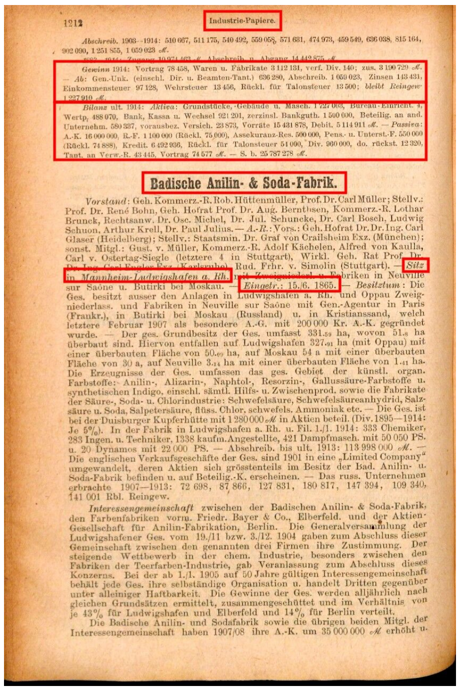
Loans and discounts	$183, 549 15	Capital stock paid in	$100, 000 00
Overdrafts	485 72		
U. S. bonds to secure circulation	100, 000 00	Surplus fund	6, 000 00
U. S. bonds to secure deposits		Other undivided profits	930 86
U. S. bonds on hand	250 00		
Other stocks, bonds, and mortgages		National bank notes outstanding	90, 000 00
Due from approved reserve agents	51, 105 20	State bank notes outstanding	
Due from other banks and bankers	4, 377 92		
Real estate, furniture, and fixtures	7, 277 00	Dividends unpaid	3, 159 00
Current expenses and taxes paid	12 59		
Premiums paid	33 00	Individual deposits	150, 589 14
Checks and other cash items	373 13	United States deposits	
Exchanges for clearing-house		Deposits of U.S. disbursing officers	
Bills of other banks	890 00	Due to other national banks	5, 331 87
Fractional currency	14 32	Due to State banks and bankers	169 40
Specie	3, 371 85		
Legal-tender notes	1, 980, 00	Notes and bills re-discounted	2, 039 61
U. S. certificates of deposit		Bills payable	
Due from U. S. Treasurer	4, 500 00		
Total	358, 219 88	Total	358, 219 88

Clinton National Bank, Clinton.

JOSEPH VAN SYCKEL, *President*. No. 1114. BENNET V. LEIGH, *Cashier*.

Loans and discounts	$219, 544 36	Capital stock paid in	$100, 000 00
Overdrafts	25 81		
U. S. bonds to secure circulation	100, 000 00	Surplus fund	40, 600 00
U. S. bonds to secure deposits		Other undivided profits	22, 456 60
U. S. bonds on hand	30, 000 00		
Other stocks, bonds, and mortgages	1, 500 00	National bank notes outstanding	90, 000 00
Due from approved reserve agents	56, 723 43	State bank notes outstanding	
Due from other banks and bankers	2, 678 17		
Real estate, furniture, and fixtures	6, 000 00	Dividends unpaid	905 75
Current expenses and taxes paid	1, 038 67		
Premiums paid		Individual deposits	169, 315 74
Checks and other cash items	437 18	United States deposits	
Exchanges for clearing-house		Deposits of U.S. disbursing officers	
Bills of other banks	1, 467 00	Due to other national banks	13, 774 66
Fractional currency	14 88	Due to State banks and bankers	8 50
Specie	4, 531 75		
Legal-tender notes	8, 000 00	Notes and bills re-discounted	
U. S. certificates of deposit		Bills payable	
Due from U. S. Treasurer	4, 500 00		
Total	436, 461 25	Total	436, 461 25

(a) *OCC Annual Report*



1212 Industrie-Papiere.

Abschreib. 1903—1914: 510 667, 511 175, 540 492, 559 058, 571 631, 474 973, 459 549, 636 038, 815 164, 902 090, 1 251 855, 1 059 023 ℳ.

1882—1914: Zugang 10 971 463 ℳ. Abschreib. u. Abgang 14 442 875 ℳ.

Gewinn 1914: Vortrag 78 458, Waren u. Fabrikate 3 112 131, verf. Div. 140; zus. 3 190 729 ℳ. — *Ab*: Gen.-Unk. (einschl. Dir. u. Beamten-Tant.) 636 280, Abschreib. 1 059 023, Zinsen 143 431, Einkommensteuer 97 128, Wehrsteuer 13 456, Rückl. für Talonsteuer 13 500; *bleibt Reingew.* 1 227 910 ℳ.

Bilanz ult. 1914: *Aktiva*: Grundstücke, Gebäude u. Masch. 1 727 003, Bureau-Einricht. 4, Wertp. 488 070, Bank, Kassa u. Wechsel 921 201, zerzinsl. Bankguth. 1 500 000, Beteilig. an and. Unternehm. 580 337, vorausbez. Versich. 23 873, Vorräte 15 431 878, Debit. 5 114 911 ℳ. — *Passiva*: A.-K. 16 000 000, R.-F. 1 100 000 (Rückl. 75 000), Assekuranz-Res. 500 000, Pens.- u. Unterst.-F. 550 000 (Rückl. 74 888), Kredit. 6 492 936, Rückl. für Talonsteuer 51 000, Div. 960 000, do. rückst. 12 320, Tant. an Verw.-R. 43 445, Vortrag 74 577 ℳ. — S. b. 25 787 278 ℳ.

Badische Anilin- & Soda-Fabrik.

Vorstand: Geh. Kommerz.-R. Rob. Hüttenmüller, Prof. Dr. Carl Müller; Stellv.: Prof. Dr. René Bohn, Geh. Hofrat Prof. Dr. Aug. Bernthsen, Kommerz.-R. Lothar Brunck, Rechtsanw. Dr. Osc. Michel, Dr. Jul. Schuncke, Dr. Carl Bosch, Ludwig Schuon, Arthur Krell, Dr. Paul Julius. — *A.-R.*: Vors.: Geh. Hofrat Dr. Dr. Ing. Carl Glaser (Heidelberg); Stellv.: Staatsmin. Dr. Graf von Crailsheim Exz. (München); sonst. Mitgl.: Gust. v. Müller, Kommerz.-R. Adolf Kächelen, Alfred von Kaulla, Carl v. Ostertag-Siegle (letztere 4 in Stuttgart), Wirkl. Geh. Rat Prof. Dr. Dr. Ing. Carl Engler Exz. (Karlsruhe), Rud. Frhr. v. Simolin (Stuttgart). — *Sitz* in *Mannheim-Ludwigshafen a. Rh.* mit Zweigniederl. u. Fabriken in Neuville sur Saône u. Butirki bei Moskau. — *Eingetr.*: 15./6. 1865. — *Besitztum*: Die Ges. besitzt ausser den Anlagen in Ludwigshafen a. Rh. und Oppau Zweigniederlass. und Fabriken in Neuville sur Saône mit Gen.-Agentur in Paris (Frankr.), in Butirki bei Moskau (Russland) u. in Kristianssand, welch letztere Februar 1907 als besondere A.-G. mit 200 000 Kr. A.-K. gegründet wurde. — Der ges. Grundbesitz der Ges. umfasst 331.59 ha, wovon 51.8 ha überbaut sind. Hiervon entfallen auf Ludwigshafen 327.91 ha (mit Oppau) mit einer überbauten Fläche von 50.09 ha, auf Moskau 54 a mit einer überbauten Fläche von 30 a, auf Neuville 3.14 ha mit einer überbauten Fläche von 1.41 ha. Die Erzeugnisse der Ges. umfassen das ges. Gebiet der künstl. organ. Farbstoffe: Anilin-, Alizarin-, Naphtol-, Resorzin-, Gallussäure-Farbstoffe u. synthetischen Indigo, einschl. sämtl. Hilfs- u. Zwischenprod. sowie die Fabrikate der Säure-, Soda- u. Chlorindustrie: Schwefelsäure, Schwefelsäureanhydrid, Salzsäure u. Soda, Salpetersäure, flüss. Chlor, schwefels. Ammoniak etc. — Die Ges. ist bei der Duisburger Kupferhütte mit 1 280 000 ℳ in Aktien beteil. (Div. 1895—1914: Je 5%). In der Fabrik in Ludwigshafen a. Rh. u. Fil. 1./1. 1914: 333 Chemiker, 283 Ingen. u. Techniker, 1338 kaufm. Angestellte, 421 Dampfmasch. mit 50 050 PS. u. 20 Dynamos mit 22 000 PS. — Abschreib. bis ult. 1913: 113 998 000 ℳ. — Die englischen Verkaufsgeschäfte der Ges. sind 1901 in eine „Limited Company" umgewandelt, deren Aktien sich grösstenteils im Besitz der Bad. Anilin- u. Soda-Fabrik befinden u. auf Beteilig.-K. erscheinen. — Das russ. Unternehmen erbrachte 1907—1913: 72 698, 87 866, 127 831, 180 817, 147 394, 109 340, 141 001 Rbl. Reingew.

Interessengemeinschaft zwischen der Badischen Anilin- & Soda-Fabrik, den Farbenfabriken vorm. Friedr. Bayer & Co., Elberfeld. und der Aktien-Gesellschaft für Anilin-Fabrikation, Berlin. Die Generalversammlung der Ludwigshafener Ges. vom 19./11 bzw. 3./12. 1904 gaben zum Abschluss dieser Gemeinschaft zwischen den genannten drei Firmen ihre Zustimmung. Der steigende Wettbewerb in der chem. Industrie, besonders zwischen den Fabriken der Teerfarben-Industrie, gab Veranlassung zum Abschluss dieses Konzerns. Bei der ab 1./1. 1905 auf 50 Jahre gültigen Interessengemeinschaft behält jede Ges. ihre selbständige Organisation u. handelt Dritten gegenüber unter alleiniger Haftbarkeit. Die Gewinne der Ges. werden alljährlich nach gleichen Grundsätzen ermittelt, zusammengeschüttet und im Verhältnis von je 43% für Ludwigshafen und Elberfeld und 14% für Berlin verteilt.

Die Badische Anilin- und Sodafabrik sowie die übrigen beiden Mitgl. der Interessengemeinschaft haben 1907/08 ihre A.-K. um 35 000 000 ℳ erhöht u.

(b) Saling's Börsen-Papiere

Figure 1: ***Case Studies.*** *These figures show examples of the two datasets showcased in this paper. Panel (a) shows page 297 of the 1882 OCC Annual Report to Congress. Panel (b) shows page 1212 of* Teil 2 *of the 1915 edition of* Saling's Börsen-Papiere*, with the items to be extracted highlighted in red.*

Although both sets of documents represent firm-level balance-sheet data, they do so in diametrically opposed layouts, with the OCC dataset in a more standard tabular form, and the *Saling's* dataset in free-form paragraphs. Further, these sets of documents differ in their language, in the quality of the scanned documents, the fonts used, and even the century when they were published. This increases the likelihood that the methods discussed here are general enough and not specific to a certain document or type of document.

We discuss each step of the data extraction pipeline in depth in Section 3, showcasing both their Python implementation as well as how they were used to improve the digitization process of the two examples that differ in various dimensions such as layout and language.

Finally, in Section 4, we discuss advanced methods that are especially beneficial for larger-scale digitization projects. First, we discuss ensemble methods that allow to combine multiple OCR engines to create a synthetic engine designed to be able to correctly detect text even when all the OCR engines failed to do so. Second, we discuss parameter tunings, that show in detail how one can leverage ground truth generated by human reviewers to optimize our choice of the parameters used in the digitization pipeline.

2 Extracting balance sheet data at scale

We start by discussing the general choices and trade-offs a researcher faces when considering using OCR to digitize data. Depending on the size of the historical records that will be digitized, practitioners can generally opt for three different approaches. As shown in figure 2, these approaches differ in the initial setup cost, as well as in the variable cost-per-page.

A naive, manual approach would be to manually input every value onto a spreadsheet. This approach involves no programming but inputting each page is a slow process with roughly no returns to scale involved: as each page takes the same amount of work as the last, and we cannot take advantage of patterns or structures common to all the pages to speed up the work. For instance, digitizing a single page of the OCC bank balance sheet dataset took the authors roughly twenty minutes per page. At this rate, digitizing the 37,000 pages comprising the 1867-1904 OCC annual reports would have required *12,300 hours* of work.

A second approach, suitable for small, well-formatted datasets—properly scanned, with standard fonts and simple tables—consists of directly applying off-the-shelf commercial OCR software, and then manually correcting and reshaping the resulting data. For the OCC bank balance sheet

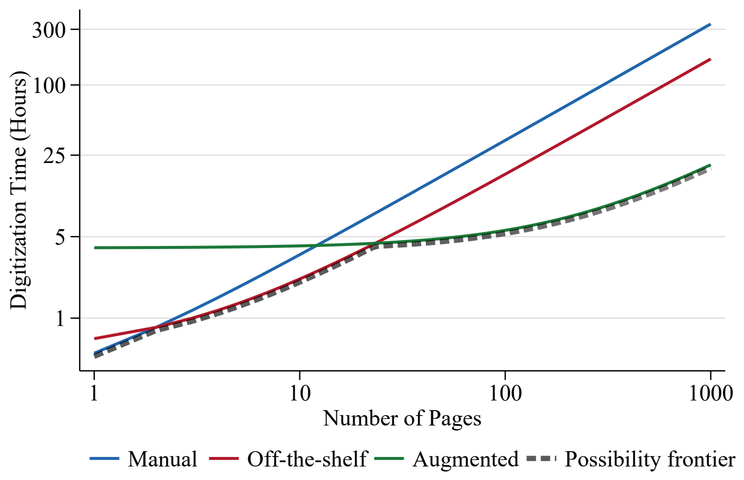


Figure 2: ***Digitization time by type of approach.*** *Note: axes in log-scale.*

dataset, this reduced the manual review time by roughly 60%, to 8 minutes per page. This is still unfeasible for a large-scale digitization, as digitizing all the pages would have required roughly 4,900 hours of manual work.

The third approach, which we recommend for large-scale datasets, involves augmenting the OCR step with pre- and post-processing steps that improve the quality of the resulting data and thus reduce the time required for human review. In particular, this approach reduced the review time of the OCC bank balance sheet to about a minute per page—with some pages requiring extensive corrections but most requiring minimal review. Thus, the total time required to digitize the OCC bank balance sheet dataset was a much more manageable 600 hours or roughly fifteen weeks of full-time work.

Together, these three approaches form the digitization possibility frontier, shown as the gray dashed line of figure 2. Note that because the graph is scaled in logarithms, the vast majority of documents—with 25 pages of data or more—are best suited for the large-scale "augmented" digitization approach seen in the green line.

To explain how this augmented digitization approach works, we have classified its different elements into six broad components, illustrated in figure 3. Depending on the complexity and scale of the input documents, researchers might want to apply only some of these components and omit others.

The starting point of this pipeline is a scanned document, either as a PDF or as a set of images. Its endpoint is a dataset, either in tabular form—CSV files or Excel tables—or in hierarchical

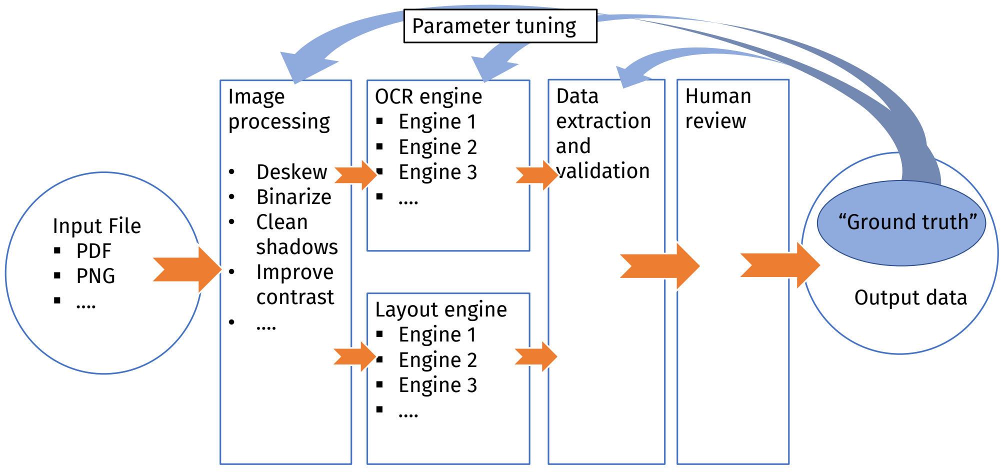


Figure 3: ***Data extraction pipeline.*** *This figure shows the six steps of the augmented digitization approach which is the focus of this paper.*

form—such as a JSON file. Note that the resulting datasets might not just be mere digitization of the documents, but instead require further transformations to make the data useful for researchers. For instance, items in the *Saling's* dataset are listed as free-form labels, so a given balance sheet might contain items such as "Inventory of Portland cement at the Berlin warehouse", which then needs to be aggregated into a broader "Inventory" category in order to be comparable across firms and industries. Similarly, even the more standardized OCC dataset lists items as specific as "Gold dust on hand", which we aggregate into the "Specie" category.

The first step of the pipeline consists of transforming the scanned images to improve the accuracy of the subsequent OCR steps. It consists of a sequence of transformations known as image filters or image kernels, where each filter has a specific purpose, such as fixing the image alignment or increasing its contrast.

The second step consists on applying one *or more* OCR engines to the transformed images. As discussed in section 4, there are two reasons for using more than one engine. First, there might not be an OCR engine uniformly better than another, even within a single document. Rather, an engine might perform better on some pages due to page and scanning idiosyncrasies. Thus, assuming we can compute a page-level measure of data quality—such as the count of numbers recognized—running multiple engines might allow us to choose the engine best suited for a given page. Second, at an even deeper level, we could combine multiple OCR outputs through "ensemble methods" that allow us to recover correct datums—words or numbers—even in cases where all OCR engines outputted incorrect values. This is feasible because most modern engines—such as Amazon's Textract and Google's Vision AI—output not only text but its coordinates, hierarchy—i.e. the line, paragraph, and block where they belong—and even confidence values of each word.

The third step of the pipeline is the layout engine, where we identify the different layout elements of each page, such as tables, columns, paragraphs, and headers. This step—not often required—provides us with two advantages. First, by knowing the section of a page where a datum is located, we can assign it correctly in a given dataset. For instance, in the OCC example the names, locations, and charter numbers of each bank are located in the header of their respective balance sheet table. Similarly, in the *Saling's* example, the different balance sheets and profit and loss statements are separated by paragraph breaks. The second advantage, particularly useful for poorly scanned datasets with one or more tables, is that by knowing where a table is located, we can select the image corresponding to that table and use this selection as an input to the OCR engine, which will likely perform much better than when inputted the entire page.

The fourth, and arguably most important step for the researcher, is the data extraction and validation step. This step transforms the mostly unstructured data generated by the OCR and layout recognition engines into structured data suitable to be loaded and processed by statistical software. Further, once the document is transformed into a structured dataset we can evaluate the quality of its data and validate it against a set of invariants, i.e. conditions that must always hold. For instance, a balance sheet might have a finite and predefined list of labels, and its values must follow predefined patterns, so for instance "123,456.00" might be a valid number but "023,456.00" and "123,4.56" might not.

Step five, performed after the data has already been transcribed and converted into key-value pairs, is the human validation step. This step serves two purposes: first, most obviously, it can be used to fix data incorrectly transcribed. Depending on the situation, researchers might want to either review all data or data that fails sanity checks or invariants and is thus flagged as problematic. The second, more ancillary, purpose of this step is to construct ground truth—data that we have reviewed and *know* is correct.

The sixth and last step, parameter tuning, consists of adjusting the parameters used in all the previous steps to reduce the error rate against the ground truth available. For instance, in many instances it is not clear what thresholds should we select when processing the images, or whether we should apply a spell-checker or not. Ground truth data provides allows us to construct metrics of accuracy which we can in turn use to improve the performance of the parameters.

As we will show in the next section, only two steps—OCR and data extraction and validation—are required, while the others are more context-dependent and might be omitted depending on the quality and scale of the documents at hand.

3 Data extraction methods

This section discusses in detail the different components of the data extraction pipeline outlined in section 2, except for two advanced topics—ensemble methods and parameter tuning—which are treated on section 4. For each component, we first discuss the underlying algorithm or method and then show the role it played in improving our two reference digitization projects, section 1. We accompany this exposition with code samples based on our `quipucamayoc` Python package.

To start, we first load the scanned documents into Python and select a page:

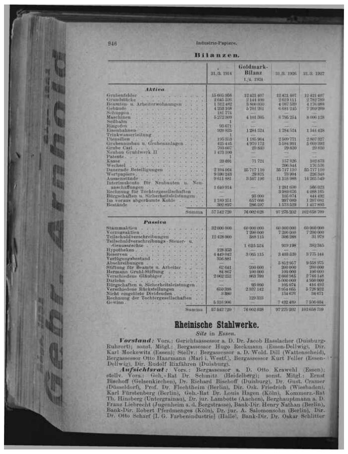
Listing 1: *Starting up*

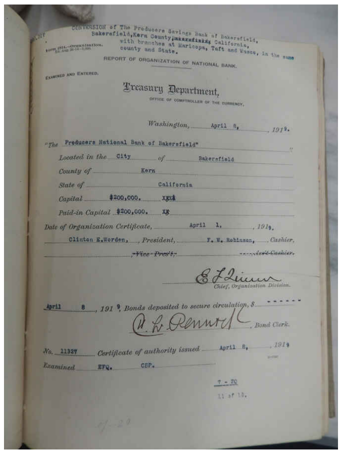
```python
import quipucamayoc
doc = quipucamayoc.Document(cache_path='./temp', clear_cache=False)
doc.read_pdf('occ-1920.pdf')  # Read PDF
doc.describe()

page = doc.pages[0]   # Select a page to process
page.view_image()
page.save_image()    # By default, image saved as './temp/raw/00000.jpg'
```

Note that the Python package is designed from the ground up to allow for saving intermediate results. This facilitates fast iterations when prototyping the pipeline, reduces total computing time, and reduces costs by avoiding running duplicate OCR steps.

3.1 Image processing

The main goal of this step is to undo any distortions in the digitized image created by the scanning process. These distortions can alter the size, shape, color, and brightness of the documents, and thus adversely affecting the performance of the subsequent OCR recognition. They are particularly problematic for historical documents, whose pages are often discolored, and might suffer from artifacts such as foxing and bleed-through (Gupta et al., 2015). Further, sheet-feed scanners—which minimize the curvature and skew of the scanned images—are often unfeasible as they require removing the book spines, permanently damaging them.


(a) *Distorted image size*


(b) *Distorted image shape*

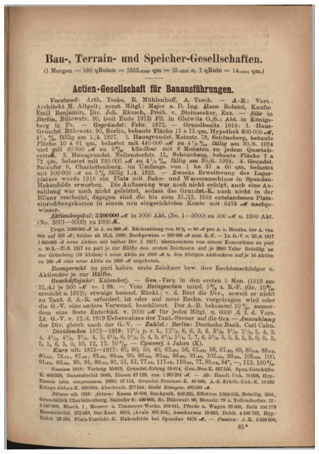

(c) *Distorted image color*

Figure 4: ***Image distortions.*** *This figure shows examples of the three main types of image distortions in scanned documents. Panel (a) shows size distortions in a* Saling's *page due to the fore-edge of the book and the scanner background. Panel (b) shows geometry or shape distortions in a national bank organization report. Panel (c) shows a* Saling's *page with yellowed background and bleed-through of the text on the reverse page.*

Size distortions

Due to the inability to use sheet-feed scanners, most document scanning efforts involve either flatbed scanners or overhead cameras. These often fail to restrict the image to the document itself, and instead produce a scanned image that is too large and that contains part of the scanner background bed or of the book edge, as shown in figure 5a. In the *Saling's* dataset, both of these extraneous image elements significantly reduced the accuracy of all the OCR engines we tested.

To trim the image into what corresponds to the page itself, we follow a three-part approach, illustrated in figure 5, which proved quite robust while only requiring minimal tuning. In essence, this approach exploits the fact that the page itself is in a lighter color than the often-black scanner background and the often-gray book edge. Then, it identifies the largest white rectangle in the image and assumes that the rectangle corresponds to the page itself.

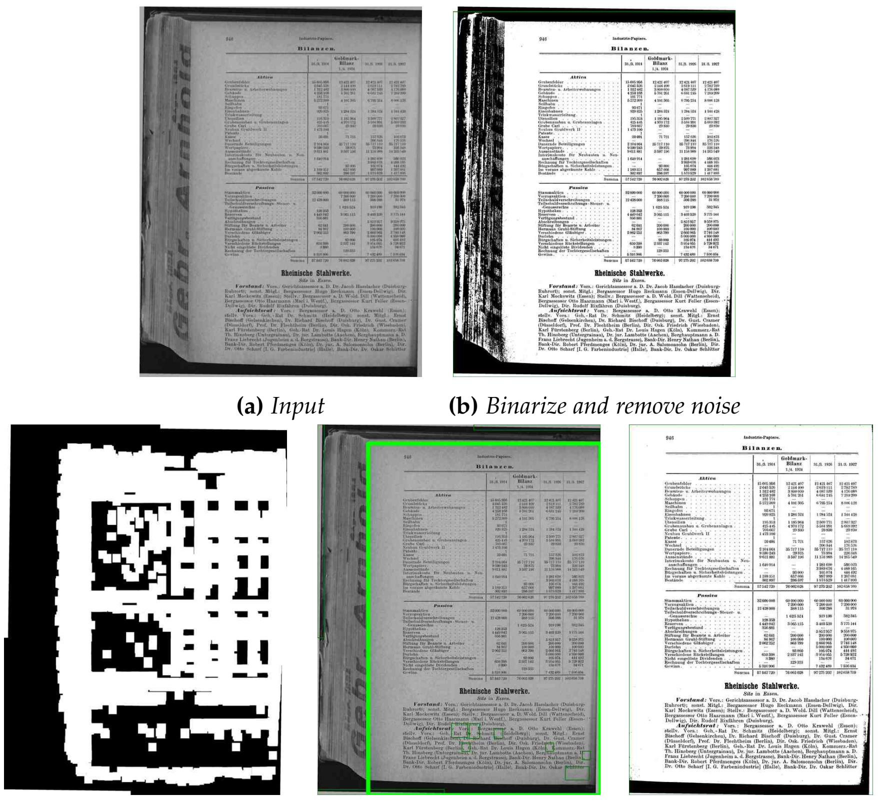
In more detail, the three steps are as follow:

1. We binarize the image, converting it into a black-white version. This is implemented by converting all the pixels lighter than a certain threshold to white color and all others into

black. Then we remove the noise in the image. This is implemented by sequentially applying the `erosion` and `dilation` OpenCV image filters available in Python (Kaehler and Bradski, 2016). The output of this step can be seen in figure 5b.

2. We expand the white area to further remove noise and ensure that all margins of the page are white. This is implemented by using the OpenCV `dilate` filter. Its results are seen in figure 5c.

3. We identify the largest white rectangle in the image. For this, we first detect all rectangles with the `findContours` filter, which implements Suzuki et al. (1985). Then, we combine all large rectangles into a single one that in most cases exactly encompasses the page itself.[2]. The selected area is shown as a green rectangle in figure 5d, and it is this area that we use to crop the original document into figure 5e.



(a) *Input* **(b)** *Binarize and remove noise*

(c) *Dilate white area* **(d)** *Identify white rectangle* **(e)** *Trimmed output*

Figure 5: ***Removing fore-edges on* Saling's *balance sheets.*** *Panel (a) contains the input image. Note the black background and the fore-edge with the words "Gebr. Arnhold". Panel (b) shows the output of the binarization and denoise step. Panel (c) (c) further reduces noise by expanding the white space. Lastly, in panel (d) we select the largest white rectangle in the text, corresponding to the page itself, and crop the image to this rectangle in panel (e).*

[2]Note that if the resulting rectangle has a proportion too different than that of the initial document, we abort and avoid trimming the image at all. This is done because it is better not to crop an image than to crop too much of it

Geometric distortions

Geometric distortions are a particularly pernicious scanning artifact that in many cases OCR engines fail to solve. They can be classified into two types. The first, simplest case are 2D distortions, which occur when the scanner or camera is at an angle relative to the flat document. This is solved by a process known as 2D page dewarping and deskewing, implemented via OpenCV's `getPerspectiveTransform` filter, which corrects the perspective of the camera as if the document was scanned with the camera directly on top.[3]. The second distortions occur when the pages are also curved while being scanned. They are more prominently at the curved edges between two pages, creating what is known as gutter shadows. In this case, a transforming the camera perspective would be insufficient to rectify the distortion, and we would instead require a more complex transformation that can map the 3D curved surface of the paper into a 2D rectangular scanned image and then reverse this mapping to recover the original 2D document.

Note, however, that most modern OCR engines automatically implement some degree of page dewarping, so this step is less important than even a few years ago. Moreover, most scanned documents available in public repositories—such as Google Books—have undergone sophisticated dewarping, so this step is often not required. See Lefevere and Saric (2009) for such an example.

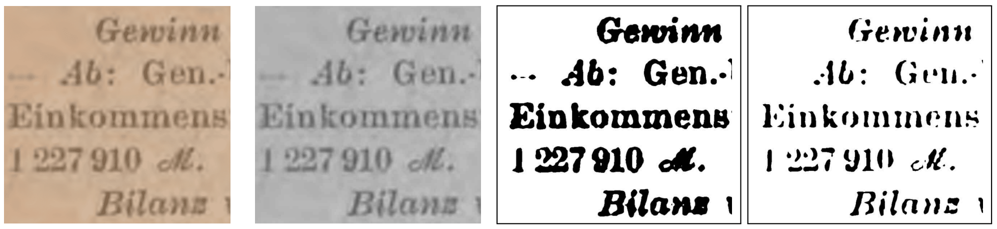
The dewarping step is partly implemented in `quipucamayoc` via the `page.dewarp()` method. Relevant work includes Li et al. (2019) and Das et al. (2019)—based on deep-learning approaches—as well as Fu, Wu, Li, Li, Xu, and Yang (Fu et al.), You et al. (2016), and Tian and Narasimhan (2011), based on more traditional approaches based on the properties of curved surfaces such as papers.

Color distortions

As discussed above, scanned historical documents often have color distortions caused by the scanning process as well as due to the age of the document. Backgrounds might be yellow and some areas might be stained in an oxidation process known as *foxing*. The ink of the text on the reverse page might be *bleeding through* to the current page. There might be shadows caused by poor lighting or by the curvature of the pages, particularly at the *gutter shadows*, the page edges connected to the book spine, which have a more pronounced curvature and might thus have a darker background.

[3] An even simpler case involves image rotation, which can be treated as a special case of the 2D perspective transform

To address this issue, the user must first decide the type of the output image, which in most cases is either a grayscale image or a black-and-white monochrome image. On principle, monochrome images are better because the underlying document is also monochrome—black for text and white for the background. Further, monochrome images occupy a much smaller size and are much faster to both upload and process. However, binarization, the process of converting images to black-and-white, often produces poor results with low-quality scans. This is because forcing each pixel to take binary values eliminates information that both OCR engines and human reviewers might be able to use otherwise to better recognize the images. Thus, the choice of grayscale or monochrome outputs is a practical matter that depends on the characteristics of the documents at hand. Such a phenomenon is illustrated in figure 6, which shows how converting a snippet of the *Saling's* dataset to grayscale maintains the readability of the text, but converting it to black-and-white makes some characters close to unreadable—although most have become much easier to discern.


Gewinn
Ab: Gen.
Einkommens
1 227 910 M.
Bilanz

(a) *Input image* **(b)** *Grayscale* **(c)** *Binarized ($\tau = 160$)* **(d)** *Binarized ($\tau = 140$)*

Figure 6: ***Grayscale and monochrome conversion of color images.*** *Panel (a) shows a zoomed-in snippet of figure 1b. Panel (b) shows its grayscale version, where all text is still easily readable. Panel (c) shows a binarized version where all pixels with values above 160 are converted to white and all below to black (note: 0=black and 255=white). In this panel, most letters are clearer, but some, such as the number "9", or the letter "z" at the end of* Bilanz *have become harder to distinguish as their font is too thick. Lastly, Panel (c) shows how reducing the binarization parameter from 160 to 140 is able to produce thinner fonts, but at the cost of other values becoming harder to distinguish.*

Color corrections with grayscale output The main methods used to create color-corrected grayscale images revolve around *histogram equalization*. The starting point of these methods is a "intensity histogram" characterizing all the pixels in an image according to their intensity, from black (0) to white (255). If an image has poor contrast, then its intensity will have a narrow distribution which will be reflected in an also narrow histogram. This can be seen in panels (a) and (d) of figure 7, which show an image from the *Saling's* dataset exhibiting poor contrast between the page background and the text.

A simple solution to the lack of contrast in this image involves "stretching" the intensity histogram, so it resembles more that of a uniform distribution. This is achieved through a process known as image equalization, showcased in figure 7b and figure 7e. Note that there are two problems with this process. First, because the histogram only takes values from 0 to 255, equalizing the image will create gaps in the support, evidenced by looking at the large gaps between intensity values in the 100-200 range of figure 7e. Second, the equalization is done uniformly across the document, but if the document was not perfectly flat and uniformly lit when scanned, then some parts of the document, such as the page margins, will be darker than others, leading to distortions in the output—starkly evident in the page margins of figure 7b.

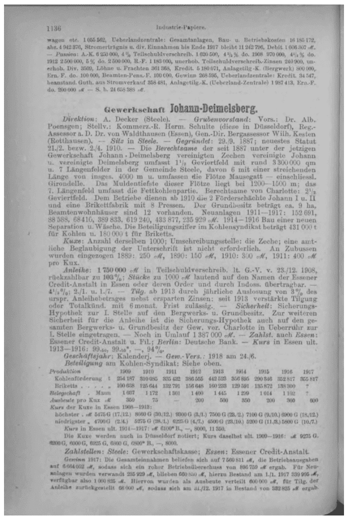
To overcome these issues, we recommend instead implementing an *adaptive* histogram equalization, which instead creates multiple histograms in different regions of the page. These histograms are there stretched locally, so these methods deal better with differences in page lighting across the page. In particular, we recommend using the Contrast Limited Adaptive Image Equalization method, or CLAHE (Pizer et al., 1990), which adds further optimizations. For a review of using CLAHE with historical documents, see Koistinen et al. (2017), which evaluates its performance on Finish historical newspapers.

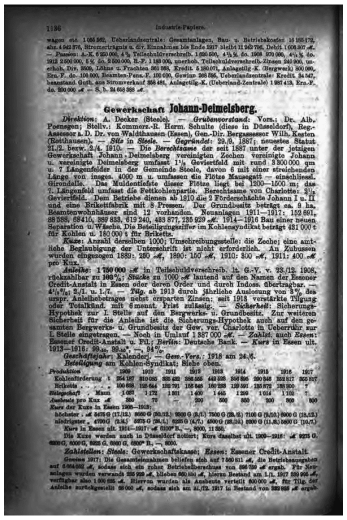
Color corrections with monochrome output Depending on the quality and characteristics of the scanned documents, binarization—converting the document into a black-and-white representation—might achieve better OCR engine performance than grayscale conversion. Further, other methods discussed in this paper, such as line detection, rely on a monochrome image as its input, so achieving high-quality binarization is an essential part of the digitization pipeline.

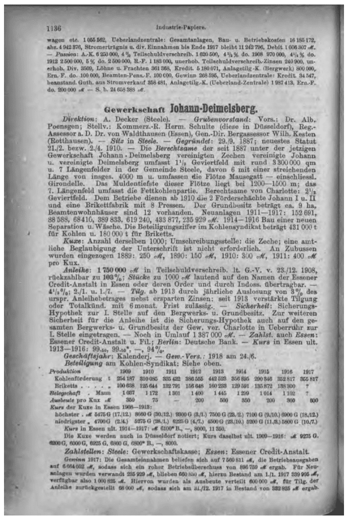
Nonetheless, binarizing historical documents involves several challenges, caused due to the degradation of the paper itself—leading to yellowed backgrounds and page *foxing*—as well as to the ink—leading to ink bleed-through. These and other challenges are discussed in quite extensive detail in Sulaiman et al. (2019), to which we will defer.

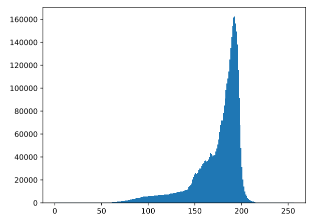
We explore five types of binarization methods, as seen in figure 8. The first method, shown in figure 8b, simply converts all pixels above a predetermined threshold to white and those below to black. However, the choice of this threshold parameter is problematic because it depends on the characteristics of each page, and choosing a wrong threshold will result in an illegible image. For instance, in this example, we chose as the threshold the value 127, the midpoint between 0 and 255. This value appears to be too low, as many font elements have been converted into white regions.

The second method, known as Otsu's binarization, avoids this problem by automatically choosing

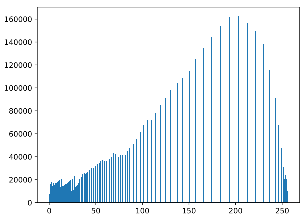

(a) *Input*


(b) *Equalized*

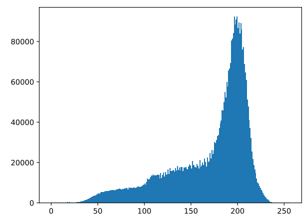

(c) *CLAHE*



(d) *Histogram of input*



(e) *Histogram of equalized*

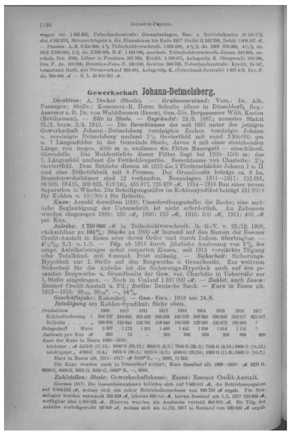


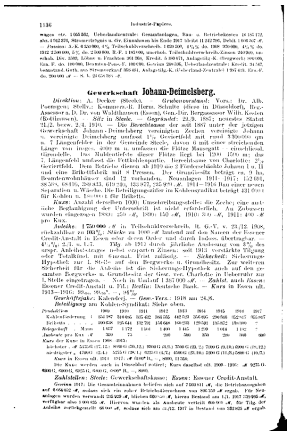
(f) *Histogram of CLAHE*

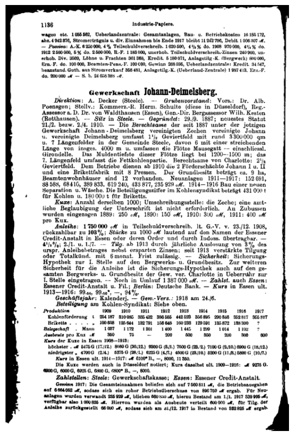
Figure 7: ***Improving contrast of grayscale* Saling's *image.*** *Panel (a) contains the input image. Note the black background and the fore-edge with the words "Gebr. Arnhold". Panel (b) shows the output of the binarization and denoise step. Panel (c) (c) further reduces noise by expanding the white space. Lastly, in panel (d) we select the largest white rectangle in the text, corresponding to the page itself, and crop the image to this rectangle in panel (e).*

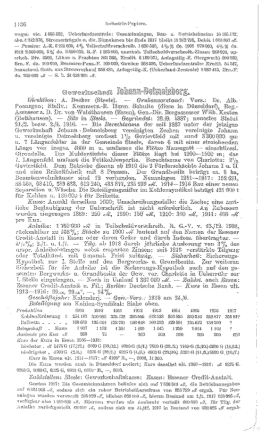
the threshold that minimizes the intra-class variance of the pixel intensity. However, it performs poorly if the document brightness is not uniform. For instance, curved documents might be darker in the margins. That is in fact what we observe in figure 8c, where we see that although most of the text was binarized correctly, there are nonetheless black regions in the margins of the page.

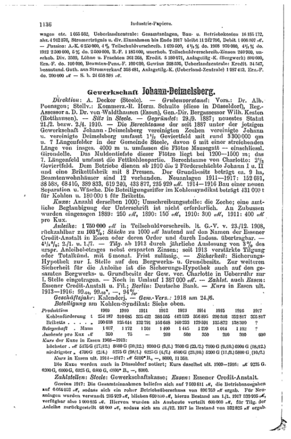
To address brightness differences, the next set of methods are known as local or adaptive methods, and similarly to the adaptive histogram equalization discussed above, compute thresholds at multiple areas of the page. In particular, figure 8d implements a mean-based adaptive binarization, figure 8e implements the method by Sauvola and Pietikäinen (2000), and figure 8f implements Wolf and Jolion (2004). From all these methods, we have found the last two to be the most robust ones across different types of documents. This finding is similar to that of Michalak and Okarma (2019), who compare the performance of these and other binarization methods across documents with different font faces, font sizes, and illumination artifacts.


(a) *Input*


(b) *Threshold binarization*


(c) *Otsu's binarization*


(d) *Adaptive mean binarization*


(e) *Sauvola binarization*

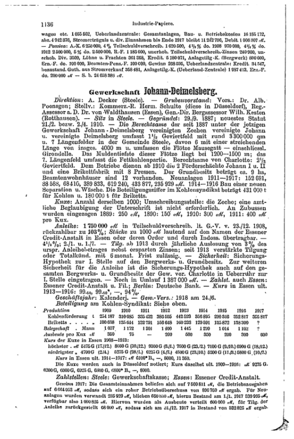

(f) *Wolf binarization*

Figure 8: ***Comparison of binarization algorithms.*** *Panel (a) contains the input image. Panels (b) and (c) show global binarization methods, which fail to account for brightness differences across different page regions. Panels (d)-(f) show the performance of three adaptive binarization methods, with methods (e) and (f) leading to the clearest documents.*

Additional improvements Beyond equalization methods, there are other approaches aimed at solving particular the existence of shadows in a document. In particular, Bako et al. (2016) implemented a novel method based on explicitly identifying shadowed regions and increasing their brightness. Further, other approaches bridge multiple methods, such as Feng et al. (2021), who implements a *transformer* machine learning model that simultaneously deskews scanned documents ("geometric unwarping") and removes their shadows ("illumination correction").[4].

Lastly, note each set of documents is unique and might face different issues. For instance, we have seen that images stored at a resolution different from 300 dots per inch (DPI) perform poorly with most OCR engines. Further, extracting images from PDF maintained in online repositories, such

[4]See this Github repository for a Python implementation

as Google Books and HathiTrust, is often problematic, as these libraries tend to embed multiple images in each page—often watermarks, but sometimes a given page has been split into multiple images internally, which need to be stitched back to be fed to an OCR engine.

`quipucamayoc` implementation In terms of the `quipucamayoc` package, the image processing step is implemented as a set of methods that can be iteratively applied to the images, as shown in listing 2. These methods are, for the most part, based on the OpenCV and scikit-image libraries, and are thus computationally efficient.

Listing 2: *Image processing*

```
page.remove_black_background()          # Crop black background
page.remove_fore_edges(threshold=160)   # Remove book fore-edges
page.dewarp()                           # Dewarp and deskew image
page.convert('grayscale')               # Convert image to grayscale
page.apply_clahe()                      # Improve contrast with CLAHE
page.binarize(method='sauvola')         # Apply Sauvola's binarization
```

3.2 Optical Character Recognition

Due to its nature, this step is at the core of the entire digitization process. However, most OCR engines act as a black box and are poorly customizable, so for practical purposes, this step is the simplest one for the user, with the only key decision being the choice of the OCR engine to use. Although there is a large number of OCR solutions, in this section we are going to focus on four widely used OCR engines. The first three are cloud-based commercial products from Google, Amazon, and Microsoft. The last one, Tesseract, is an open-source engine.

As documented by Hegghammer (2021), the commercial engines vastly outperform the open-source one, particularly in noisy documents such as the ones one might find when conducting historical research. Thus, our suggestion for most users is to select a cloud-based solution, which is likely to be both faster[5] and cheaper, once we factor in the reduced time spent in the human review process.

Table 1 compares four of the most widely used OCR engines. Strikingly, they have mostly converged in terms of their features and of the characteristics of the data they return. For instance,

[5]Cloud providers can parallelize OCR tasks across many servers, while users running their own servers would need to either wait very large amounts of time for large-scale documents, or manage digitization tasks across many of their own servers, a costly and cumbersome task.

for each individual all offerings will return its coordinates, its confidence (likelihood of a correct transcription of the word), and even a hierarchy of blocks to which the word belongs. Moreover, because they are very similar in terms of their output, `quipucamayoc` has built a wrapper around them so they are relatively interchangeable. This should allow users to compare the performance of alternative engines and thus choose the ones performing better for their given documents.

Table 1: *Comparison of OCR engines*

Provider	Product	Coordinates	Confidence	Hierarchy	Lang. hints
Google	Vision AI (Google Cloud Vision)	Yes	Yes	Symbol, word, paragraph, block	Yes
Amazon	Textract	Yes	Yes	Word, line, table, cell	No
Microsoft	Azure Computer Vision	Yes	Yes	Word, line	No
N/A	Tesseract	Yes	Yes	Word, line, paragraph, block	Yes

Lastly, note that beyond plain text recognition, some of the commercial engines have begun to offer more advanced products, such as table and form detection, as well as handwritten recognition. Note, however, that these products are usually trained with modern documents, so they often fail to perform well on historical records.

`quipucamayoc` implementation As shown in listing 3, `quipucamayoc` is designed so different OCR engines are as interchangeable as possible.

However, working with any of these engines involves a certain initial cost, which includes registering an account, and creating and downloading credentials for the cloud services. In particular, within the commercial engines, Amazon's textract stands out for the large setup complexity. To illustrate this, below we list some of the initial steps that must be carefully followed in order to start using Amazon's OCR engine:

1. Set up an AWS account.
2. Create an Identity and Access Management (IAM) user.
3. Install and configure an AWS Software Development Kit (SDK).

4. Create an Amazon Simple Notification Service (SNS) topic and write down its Amazon Resource Name (ARN).
5. Create an Amazon Simple Queue Service (SQS) standard queue and write down its ARN.
6. Subscribe the Amazon SQS to the Amazon SNS by using its ARN.
7. Grant permission to the Amazon SNS topic to message the Amazon SQS queue.
8. Create an IAM service role to grant Textract access to the Amazon SNS topics.
9. Add a JSON inline policy to the IAM user created in step 2.

We believe that for most practitioners, these steps are prohibitively difficult, time-consuming, and error-prone.[6] Thus, for users to avoid knowing the nuances of how to "register the ARN of the SQS in the SNS in order for AWS's IAM to grant Textract access to the SNS", we have implemented this setup process as part of `quipucamayoc`.

Listing 3: *Access multiple OCR engines*

```
# Amazon's OCR engine involves a difficult initial setup
quipucamayoc.setup_textract(bucket='MyBucket')

page.run_ocr(engine='google')     # Synonyms: gcv, visionai
page.run_ocr(engine='amazon')     # Synonyms: aws, textract
page.run_ocr(engine='microsoft')  # Synonyms: azure (not currently implemented)
page.run_ocr(engine='tesseract')  # (not currently implemented)
```

3.3 Layout recognition

The main purpose of the layout recognition step is help the researchers in assigning a given word or number to specific categories or groups. For instance, in a two-column table, identifying the location of the line delimiting the columns would allow users to identify to which column each word belongs.

In the case of our examples, we relied mostly on two types of layout recognition outputs.

First, to digitize the OCC dataset we needed to know the boundaries of the three tables present on each page, so we could then assign key-value pairs to the correct table. We did so by applying the

[6]Note that these instructions are perfectly reasonable for Amazon's target market, which for the most part is not composed of economic historians but by large corporations with dedicated IT *divisions*.

algorithm described in figure 9. There, we first apply a Canny Edge Detector (Canny, 1986) to identify boundaries between regions or objects. Then, we apply a probabilistic Hough transform (Kiryati et al., 1991, Hough (1959)) to the pixels highlighted as boundary regions, and identify possible lines in the text. Lastly, we apply a custom algorithm to reduce the number of lines and avoid false positives, and end up with the lines identified in figure 9a. With this information, and with the coordinates of each word produced by the OCR engines, we are thus able to assign each word into a given column of a table. In terms of the `quipucamayoc` library, this algorithm is available as follows:

Listing 4: *Layout recognition*

```
page.detect_lines(columns=5, save_annotated_image=True)
```

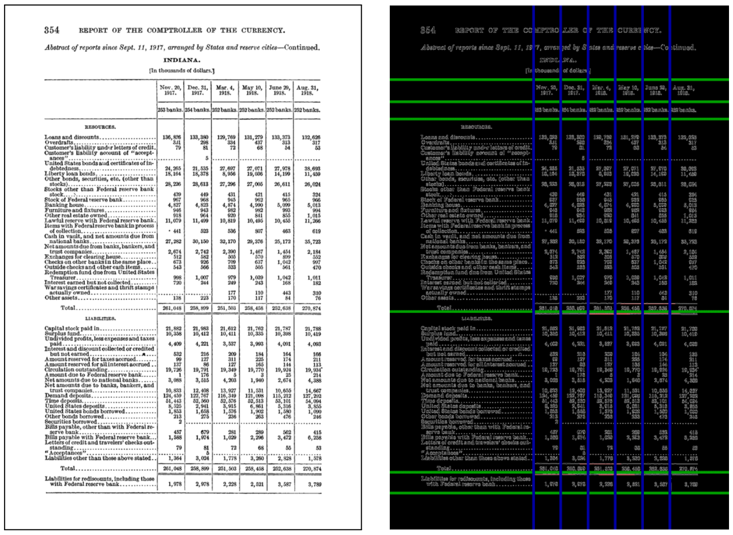
The second application of layout recognition was implemented for the *Saling's* dataset. Here, the goal was to detect individua paragraphs, which in turn might correspond to separate balance sheets and income tables. Further, centered text was used to indicate the start of a new firm, so it had to be identified correctly. Luckily, the different block elements outputted by Google's OCR engine were sufficient for us, as they allowed us to identify whether lines were centered or not, as well as the font size of each word, and the spacing above and below it. Altogether, these three pieces of information allowed us to quite accurately identify the headers with firm information.

3.4 Data extraction and validation

One often overlooked aspect of data digitization is that even if we are perfectly able to represent the scanned data digitally, this representation often does not correspond to the one needed to create a dataset useful for statistical purposes, such as an R Data Frame or a Stata `.dta` file. In particular, we need to be able to assign the words and numbers recognized by the OCR step into key-value pairs, which will then be concatenated and form variables in the final datasets.

Thus, except for very simple cases, it is inevitable that practitioners may need to rely on some programming to assign or reshape the data to their needs. For this, we believe that Python is an ideal language, as its accessible, widely known, and comes with top-of-the-line libraries.

Beyond the relatively simple data reshaping, one might also use this step to reduce error rates, and more generally, correct mistakes caused by the scanning and OCR processes, as otherwise they would have to be corrected manually in the next step, which is particularly costly and slow.



354 REPORT OF THE COMPTROLLER OF THE CURRENCY.

Abstract of reports since Sept. 11, 1917, arranged by States and reserve cities—Continued.

INDIANA.

[In thousands of dollars.]

	Nov. 20, 1917.	Dec. 31, 1917.	Mar. 4, 1918.	May 10, 1918.	June 29, 1918.	Aug. 31, 1918.
	253 banks.	254 banks.	252 banks.	252 banks.	252 banks.	252 banks.
RESOURCES.						
Loans and discounts	136,826	133,380	129,769	131,279	133,373	132,626
Overdrafts	351	298	334	437	313	317
Customer's liability under letters of credit	79	81	72	68	54	53
Customer's liability account of "acceptances"		5				
United States bonds and certificates of indebtedness	24,265	21,535	27,697	27,071	27,978	38,693
Liberty loan bonds	18,164	18,378	8,956	19,036	14,199	11,459
Other bonds, securities, etc. (other than stocks)	28,226	28,613	27,296	27,005	26,611	26,024
Stocks other than Federal reserve bank stock	439	449	431	421	415	324
Stock of Federal reserve bank	967	968	945	962	965	966
Banking house	4,827	4,823	4,874	4,990	5,099	5,013
Furniture and fixtures	946	943	982	982	993	994
Other real estate owned	918	964	920	841	855	1,015
Lawful reserve with Federal reserve bank	11,079	11,499	10,819	10,495	10,455	11,266
Items with Federal reserve bank in process of collection	441	523	536	807	463	619
Cash in vault, and net amounts due from national banks	27,282	30,150	32,170	29,376	25,172	35,723
Net amounts due from banks, bankers, and trust companies	2,674	2,742	2,390	1,467	1,454	2,184
Exchanges for clearing house	512	582	505	570	899	552
Checks on other banks in the same place	673	926	709	637	1,042	997
Outside checks and other cash items	543	566	523	505	561	470
Redemption fund due from United States Treasurer	998	1,007	979	1,039	1,042	1,011
Interest earned but not collected	720	244	249	243	168	182
War savings certificates and thrift stamps actually owned			177	110	443	310
Other assets	138	223	170	117	84	76
Total	261,048	258,809	251,503	258,458	252,638	270,874
LIABILITIES.						
Capital stock paid in	21,882	21,983	21,612	21,762	21,787	21,788
Surplus fund	10,358	10,412	10,411	10,335	10,398	10,419
Undivided profits, less expenses and taxes paid	4,409	4,221	3,537	3,993	4,091	4,093
Interest and discount collected or credited but not earned	532	216	209	184	164	166
Amount reserved for taxes accrued	99	127	311	225	174	211
Amount reserved for all interest accrued	137	86	127	165	144	113
Circulation outstanding	19,726	19,791	19,349	19,770	19,924	19,934
Amount due to Federal reserve bank	1	176	5	3	25	214
Net amounts due to national banks	3,088	3,515	4,203	1,940	2,674	4,388
Net amounts due to banks, bankers, and trust companies	10,833	12,408	13,927	11,531	10,655	14,667
Demand deposits	124,459	122,767	116,349	121,098	115,212	127,292
Time deposits	51,443	52,560	52,576	52,513	53,101	54,094
United States deposits	8,525	2,941	3,915	6,861	5,316	3,855
United States bonds borrowed	1,853	1,658	1,576	1,902	1,580	1,090
Other bonds borrowed	213	275	236	263	476	246
Securities borrowed	2					
Bills payable, other than with Federal reserve bank	457	679	281	289	562	415
Bills payable with Federal reserve bank	1,588	1,974	1,029	2,296	3,472	6,258
Letters of credit and travelers' checks outstanding	79	81	72	68	55	53
"Acceptances"		5				
Liabilities other than those above stated	1,364	3,024	1,778	3,260	2,828	1,578
Total	261,048	258,809	251,503	258,458	252,638	270,874
Liabilities for rediscounts, including those with Federal reserve bank	1,978	2,978	2,228	2,521	3,587	3,789

(a) *Preprocessed image* **(b)** *Image edges and detected lines*

Figure 9: *Detecting table delimiters on OCC balance sheets.* *We combine two computer vision techniques to identify the lines that separate the different sections of a table. First, we apply a Canny edge detector to identify discontinuities in the brightness of the page. Then, we apply a probabilistic Hough transform to identify likely lines within the text. From this set, we extract horizontal lines (green) which detect table sections and vertical lines (blue) which detect column delimiters.*

In this sense, we show two types of improvements that we have used in the OCC and *Saling's* digitization efforts, and which we believe might be generally useful.

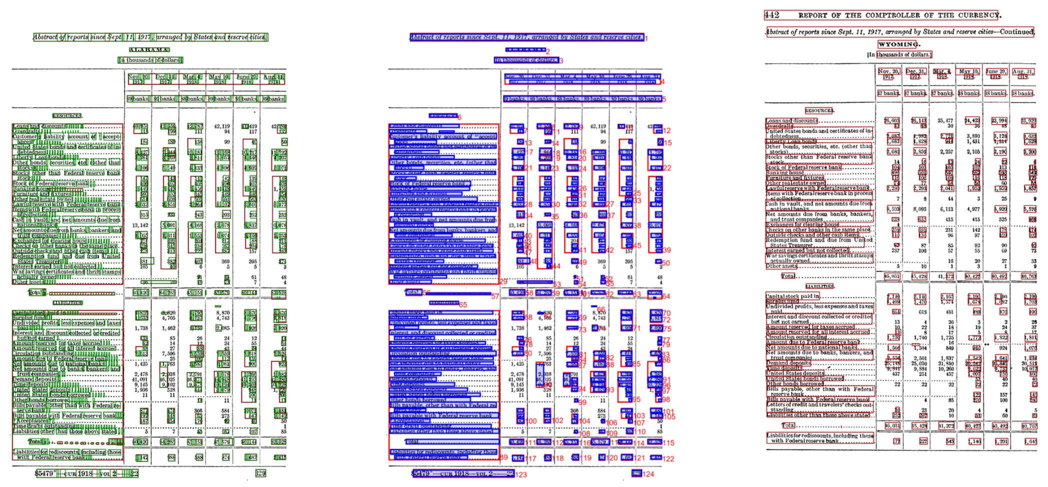
3.4.0.1 Correction of words and numbers Often, there are restrictions on the possible values that each key or value can have. For instance, numbers representing dollar values cannot contain letters. Thus, if the letter "O" was found in between two digits ("109"), it would be prudent to replace it with the letter zero. Similar patterns apply for other letters, so "1GB" could be replaced into "168". Note, however, that there is a cost to these replacements in that they might introduce false positives, so users must avoid being too aggressive with these replacements, and always try to benchmark how each replacement affects the quality of the overall dataset.

In terms of labels, they can often take only a finite set of possible values. For instance, in most years the OCC balance sheets contained only a few dozen possible labels. Similarly, the city where each bank was located—written at the top of each table—could be validated against the list of all existing and ghost US towns maintained by the U.S. Geological Survey. As a last resort, one could validate a free-form word against the set of valid words in a given language—its dictionary—to assert whether the word is valid or not.

Once a word has been diagnosed as invalid, it can be fixed by applying a spellchecker to it. For instance, Peter Norvig's famous spellchecker (see Kemighan et al., 1990; Jurafsky and Martin, 2009) can be easily implemented in a few lines of code as long as there is a dictionary with the list of all valid values.

3.4.0.2 Correction of document hierarchy As discussed in the previous subsection, most OCR engines provide incredibly useful information on the hierarchy to which each word belongs—its line and paragraph. However, in the same way as words might be misrecognized, this hierarchy can also be detected incorrectly. This is illustrated in figure 10. There, figure 10a shows all the words and paragraphs identified by Google's Vision AI. Because the page represents a table, the paragraphs are not recognized correctly although the words are.

To solve this, the first step, shown in figure 10b, involves grouping words together into lines, which will represent the balance sheet labels. For this, we simply pair up words in the same column as long as they have enough overlap in the y-axis. However, several labels are long enough to overflow the column width and thus occupy multiple labels. To solve this, we exploit the fact that the second line in each label is always indented, which allows us to arrive at the corrected output in figure 10c.



(a) *OCR-recognized "words"* **(b)** *Lines identified by* `quipucamayoc` **(c)** *Combined lines (correct)*

Figure 10: ***Identifying balance sheet labels.*** *To correctly identify rows—and thus balance sheet items—of each table, we apply a three-part process. First, we identify the* words *recognized by an OCR engine such as Google Vision AI. Second, we concatenate words based on their relative horizontal distance. Lastly, we concatenate lines based on their indenting and vertical distance and are thus able to correctly identify row labels.}*

3.4.0.3 Data invariants and sanity checks The OCR process is inherently error-prone, so it is crucial from the beginning to plan for ways to flag potential errors, which can be then reviewed and corrected by hand. Otherwise, even a human reviewer that goes through every output page might overlook potentially problematic mistakes in the data.

This is potentially important for panel balance sheet dataset, which has two particular properties. First, errors are very costly in these datasets. For instance, suppose a bank has stable total assets of \$100, but in one year there is a typo that adds one zero at the end, so the reported value of total assets becomes \$1000 for that year. This mistake would lead to an incorrect 1000% increase in total assets in the year where it occurs, plus another incorrect value of the subsequent year, where total assets would be reported to fall by 90%. Second, balance sheets have several constraints or invariants that we could exploit to validate the quality of the data. For instance, the balance sheet identity must hold, so the sum of total assets must equal the "total assets" label, which in turn must equal the "total liabilities and equity" label.

To increase the confidence in the quality of the digitized data, and to help guide human reviewers, the practitioner must identify as many constraints as possible, and incorporate them into the code. For instance, the following were some of the constraints implemented in the OCC digitization project:

1. Accounting identity: the sum of all assets must equal the label "total assets"; the sum of all liability and equity labels must equal the label "total liabilities and equity"; the two total fields must be equal to each other.

2. Reserve requirements, bond-holding requirements, and minimum-capital requirements. Whenever such a constraint was broken, the page was automatically flagged for human review.

3. Constraints across time: it was difficult to change the amount of certified bank capital, so if we noted a change across time then the two pages involved were flagged.

4. Label and numeric constraints: balance sheet labels that were not part of the valid label list, as well as invalid numeric fields were automatically flagged. For instance, the number "0123" was always flagged because dollar values do not have a leading zero digit.

5. Bank charter numbers that were either duplicated or not present in a given year led to a manual review of the scanned documents.

Altogether, these constraints allowed us to flag the most prominent errors in the dataset, which freed time for a more careful inspection of the documents afterward.

3.5 Human review

In large-scale digitization projects, the human review step needs special consideration, as this is often the most expensive and time-consuming part. Crucially, we have found that the most important thing is to ensure human reviewers are *on task* for as much time as possible. Any micro interruptions, such as having to save corrected files, load and scroll through PDFs, or manually alt-tab to review the list of pages pending review, is enough to distract and slow down the reviewers.

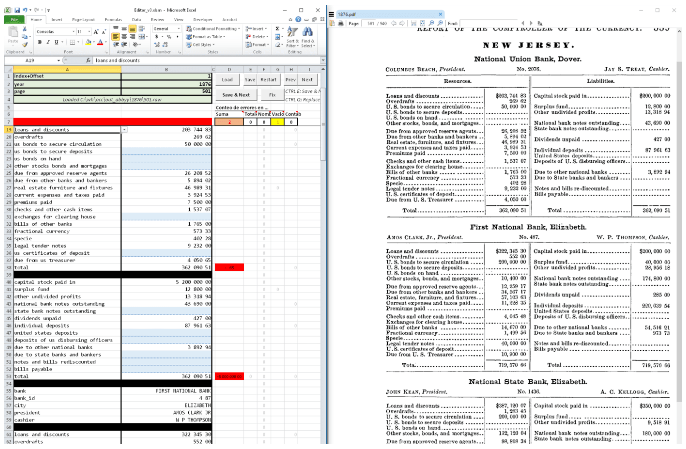
Instead, we have found that automating the human review process as much as possible increases the attention available to reviewers, leading to faster reviews with fewer errors. In terms of software, for both the OCC and *Saling's* project, we have relied on an Excel workbook containing VBA functions automating steps previously done manually by users. More recently, in an ongoing project digitizing data from St. Louis Fed's FRASER archives, we have used a browser-based solution, which we now recommend.[7] Figure 11 shows screenshots of both of these tools.

Based on our experiences, below we list some of the steps where we found automating was invaluable:

1. Show data and scanned images side-by-side. Whenever the reviewer loads a new page or table, the corresponding image should be automatically loaded as well.
2. Add shortcuts as much as possible, for loading and saving CSV files, navigating the list of pages that need to be reviewed, etc.
3. Flagging likely errors in red or yellow colors, so the reviewer can monitor those items more closely.
4. Automatically keeping track of the balance sheet identity, so the reviewer can know when there are errors still remaining.

Once we automated these steps, we found a substantial increase in review speed, leading to a larger

[7] Although the Excel approach worked quite well for the authors, we found that, when sharing the review tasks with others, the browser solution worked better, as it reduced setup issues and helped to allocate pending pages more easily across reviewers.



A19 | loans and discounts

	A	B	D	E	F	G	H	I
1	index+Offset	1	Load	Save	Restart		Prev	Next
2	year	1876						
3	page	501	Save & Next		Fix		CTRL E: Save & N	
4	Loaded C:\wh\occ\out_abbyy\1876\501.raw						CTRL Q: Replace	
5			Conteo de errores en ...					
6			Suma	Total	Nomb	Vacio	Contab	
7			2	0	0	1	0	
19	loans and discounts	203 744 83						
20	overdrafts	269 62						
21	us bonds to secure circulation	50 000 00						
22	us bonds to secure deposits							
23	us bonds on hand							
24	other stocks bonds and mortgages							
25	due from approved reserve agents	26 208 52						
26	due from other banks and bankers	5 894 02						
27	real estate furniture and fixtures	46 989 31						
28	current expenses and taxes paid	3 924 53						
29	premiums paid	7 500 00						
30	checks and other cash items	1 537 07						
31	exchanges for clearing house							
32	bills of other banks	1 765 00						
33	fractional currency	573 33						
34	specie	402 28						
35	legal tender notes	9 232 00						
36	us certificates of deposit							
37	due from us treasurer	4 050 65						
38	total	362 090 51	- 65					
39								
40	capital stock paid in	5 200 000 00						
41	surplus fund	12 800 00						
42	other undivided profits	13 318 94						
43	national bank notes outstanding	43 690 00						
44	state bank notes outstanding							
45	dividends unpaid	427 00						
46	individual deposits	87 961 63						
47	united states deposits							
48	deposits of us disbursing officers							
49	due to other national banks	3 892 94						
50	due to state banks and bankers							
51	notes and bills rediscounted							
52	bills payable							
53	total	362 090 51	-5 000 000 00					
54								
55	bank	FIRST NATIONAL BANK						
56	bank_id	4 87						
57	city	ELIZABETH						
58	president	AMOS CLARK JR						
59	cashier	W P THOMPSON						
60								
61	loans and discounts	322 345 30						
62	overdrafts	552 00						

REPORT OF THE COMPTROLLER OF THE CURRENCY. 555

NEW JERSEY.

National Union Bank, Dover.

COLUMBUS BEACH, *President*. No. 2076. JAY S. TREAT, *Cashier*.

Resources.		Liabilities.	
Loans and discounts	$203, 744 83	Capital stock paid in	$200, 000 00
Overdrafts	269 62		
U. S. bonds to secure circulation	50, 000 00	Surplus fund	12, 800 00
U. S. bonds to secure deposits		Other undivided profits	13, 318 94
U. S. bonds on hand			
Other stocks, bonds, and mortgages		National bank notes outstanding	43, 690 00
		State bank notes outstanding	
Due from approved reserve agents	26, 208 52		
Due from other banks and bankers	5, 894 02	Dividends unpaid	427 00
Real estate, furniture, and fixtures	46, 989 31		
Current expenses and taxes paid	3, 924 53	Individual deposits	87 961 63
Premiums paid	7, 500 00	United States deposits	
Checks and other cash items	1, 537 07	Deposits of U. S. disbursing officers	
Exchanges for clearing house			
Bills of other banks	1, 765 00	Due to other national banks	3, 892 94
Fractional currency	573 33	Due to State banks and bankers	
Specie	402 28		
Legal tender notes	9, 232 00	Notes and bills re-discounted	
U. S. certificates of deposit		Bills payable	
Due from U. S. Treasurer	4, 050 00		
Total	362, 090 51	Total	362, 090 51

First National Bank, Elizabeth.

AMOS CLARK, Jr., *President*. No. 487. W. P. THOMPSON, *Cashier*.

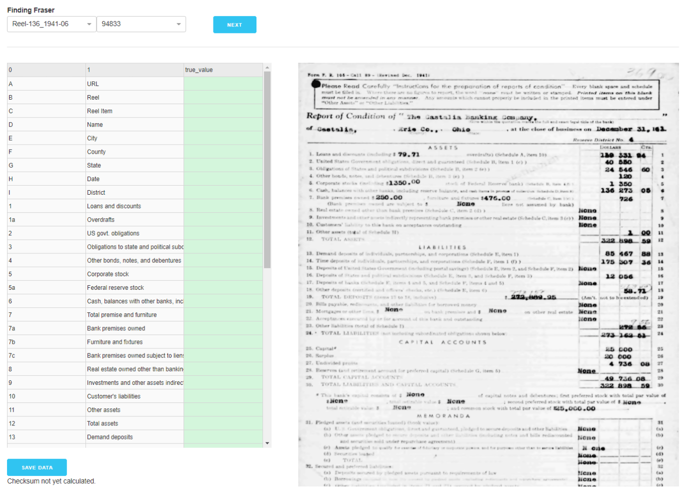
Loans and discounts	$322, 345 30	Capital stock paid in	$200, 000 00
Overdrafts	552 00		
U. S. bonds to secure circulation	200, 000 00	Surplus fund	40, 000 00
U. S. bonds to secure deposits		Other undivided profits	28, 956 18
U. S. bonds on hand			
Other stocks, bonds, and mortgages	10, 400 00	National bank notes outstanding	174, 800 00
		State bank notes outstanding	
Due from approved reserve agents	12, 259 17		
Due from other banks and bankers	34, 567 17	Dividends unpaid	285 00
Real estate, furniture, and fixtures	57, 103 63		
Current expenses and taxes paid	11, 228 35	Individual deposits	220, 039 54
Premiums paid		United States deposits	
Checks and other cash items	4, 045 48	Deposits of U. S. disbursing officers	
Exchanges for clearing house			
Bills of other banks	14, 670 00	Due to other national banks	54, 516 21
Fractional currency	1, 499 56	Due to State banks and bankers	973 73
Specie			
Legal tender notes	40, 000 00	Notes and bills re-discounted	
U. S. certificates of deposit		Bills payable	
Due from U. S. Treasurer	10, 900 00		
Total	719, 570 66	Total	719, 570 66

National State Bank, Elizabeth.

JOHN KEAN, *President*. No. 1436. A. C. KELLOGG, *Cashier*.

Loans and discounts	$387, 120 07	Capital stock paid in	$350, 000 00
Overdrafts	1, 283 45		
U. S. bonds to secure circulation	200, 000 00	Surplus fund	
U. S. bonds to secure deposits		Other undivided profits	9, 518 91
U. S. bonds on hand			
Other stocks, bonds, and mortgages	112, 120 04	National bank notes outstanding	180, 000 00
Due from approved reserve agents	98, 808 34	State bank notes outstanding	

(a) *OCC Dataset*


Finding Fraser

Reel-136_1941-06 | 94833 | NEXT

0	1	true_value
A	URL	
B	Reel	
C	Reel Item	
D	Name	
E	City	
F	County	
G	State	
H	Date	
I	District	
1	Loans and discounts	
1a	Overdrafts	
2	US govt. obligations	
3	Obligations to state and political subc	
4	Other bonds, notes, and debentures	
5	Corporate stock	
5a	Federal reserve stock	
6	Cash, balances with other banks, inc	
7	Total premise and furniture	
7a	Bank premises owned	
7b	Furniture and fixtures	
7c	Bank premises owned subject to lien:	
8	Real estate owned other than bankin	
9	Investments and other assets indirec	
10	Customer's liabilities	
11	Other assets	
12	Total assets	
13	Demand deposits	

SAVE DATA

Checksum not yet calculated.

Form F. R. 105 - Call 89 - (Revised Dec. 1941)

Please Read Carefully "Instructions for the preparation of reports of condition" Every blank space and schedule must be filled in. Where there are no figures to report, the word "none" must be written or stamped. *Printed items on this blank must not be amended in any manner.* Any amounts which cannot properly be included in the printed items must be entered under "Other Assets" or "Other Liabilities."

Report of Condition of " The Castalia Banking Company, "

of Castalia, , Erie Co., Ohio , at the close of business on December 31, 1941

Reserve District No. 4

ASSETS

			Dollars		Cts.	
1.	Loans and discounts (including $ 79.71 overdrafts) (Schedule A, item 10)		139	331	94	1
2.	United States Government obligations, direct and guaranteed (Schedule B, item 1 (c))		40	550		2
3.	Obligations of States and political subdivisions (Schedule B, item 2 (e))		24	546	60	3
4.	Other bonds, notes, and debentures (Schedule B, item 3 (e))			120		4
5.	Corporate stocks (including $1350.00 stock of Federal Reserve bank)		1	350		5
6.	Cash, balances with other banks, including reserve balance, and cash items in process of collection (Schedule D, item 8)		136	273	05	6
7.	Bank premises owned $ 250.00 , furniture and fixtures $476.00 (Bank premises owned are subject to $ None liens not assumed by bank)			726		7
8.	Real estate owned other than bank premises (Schedule C, item 2 (d))	None				8
9.	Investments and other assets indirectly representing bank premises or other real estate (Schedule C, item 3 (c))	None				9
10.	Customers' liability to this bank on acceptances outstanding	None				10
11.	Other assets (total of Schedule H)			1	00	11
12.	TOTAL ASSETS		322	898	59	12

LIABILITIES

13.	Demand deposits of individuals, partnerships, and corporations (Schedule E, item 1)		85	467	88	13
14.	Time deposits of individuals, partnerships, and corporations (Schedule F, item 1 (f))		175	307	36	14
15.	Deposits of United States Government (including postal savings) (Schedule E, item 2, and Schedule F, item 2)	None				15
16.	Deposits of States and political subdivisions (Schedule E, item 3, and Schedule F, item 3)		12	056		16
17.	Deposits of banks (Schedule E, items 4 and 5, and Schedule F, items 4 and 5)	None				17
18.	Other deposits (certified and officers' checks, etc.) (Schedule E, item 6)			58.71		18
19.	TOTAL DEPOSITS (items 13 to 18, inclusive) $ 272,889.95	(Am't. not to be extended)				19
20.	Bills payable, rediscounts, and other liabilities for borrowed money	None				20
21.	Mortgages or other liens, $ None on bank premises and $ None on other real estate	None				21
22.	Acceptances executed by or for account of this bank and outstanding	None				22
23.	Other liabilities (total of Schedule I)			272	86	23
24.	TOTAL LIABILITIES (not including subordinated obligations shown below)		273	162	81	24

CAPITAL ACCOUNTS

25.	Capital*		25	000		25
26.	Surplus		20	000		26
27.	Undivided profits		4	736	08	27
28.	Reserves (and retirement account for preferred capital) (Schedule G, item 5)	None				28
29.	TOTAL CAPITAL ACCOUNTS		49	736	08	29
30.	TOTAL LIABILITIES AND CAPITAL ACCOUNTS		322	898	59	30

* This bank's capital consists of $ None of capital notes and debentures; first preferred stock with total par value of $ None , total retirable value $ None ; second preferred stock with total par value of $ None , total retirable value $ None ; and common stock with total par value of $25,000.00

MEMORANDA

31.	Pledged assets (and securities loaned) (book value):		31
(a)	U. S. Government obligations, direct and guaranteed, pledged to secure deposits and other liabilities	None	(a)
(b)	Other assets pledged to secure deposits and other liabilities (including notes and bills rediscounted and securities sold under repurchase agreement)	None	(b)
(c)	Assets pledged to qualify for exercise of fiduciary or corporate powers, and for purposes other than to secure liabilities	None	(c)
(d)	Securities loaned	None	(d)
(e)	TOTAL	None	(e)
32.	Secured and preferred liabilities:		32
(a)	Deposits secured by pledged assets pursuant to requirements of law	None	(a)
(b)	Borrowings [illegible]	None	(b)

(b) *FRASER Dataset*

Figure 11: ***Interfaces for human review of transcription output.*** *Panel (a) shows a screenshot of the program used to validate the OCC and* Saling's *dataset. It consists of an Excel workbook powered by a set of VBA functions that load and save the data, and display the corresponding images. Panel (b) shows a screenshot of an ongoing program used to validate St. Louis Fed's FRASER archives. It is a self-contained website powered by Python Dash.*

amount of ground-truth data, which in turn can be utilized to improve the overall digitization pipeline, as we will see in section 4.

4 Advanced methods

This section discusses two advanced methods that are usually not required for moderately-sized projects, but which might be beneficial for larger-scale ones. First, we will discuss ensemble methods, where we combine the output of multiple OCR engines in order to create a synthetic engine designed to be able to correctly detect text even when all the OCR engines failed to do so. Then, we will discuss parameter tunings, where we show we can utilize the ground truth generated by human reviewers in order to optimize our choice of the parameters used in the previous stages.

Ensemble methods When discussing the OCR step, we argued that the main decision of the researcher was in choosing which OCR engine to select. Here, we argue that there is no need to select only a single OCR engine and that we can instead combine their output into an *ensemble* that performs better than any engine by itself.

Ensemble methods go beyond looking at each page and choosing which OCR engine performed better for the page. Instead, they go deeper into the word level, and compare the words and numbers produced by the different OCR engines. For instance, suppose we use three engines, which identify a given number as "123", "120", and "123" respectively. Here, the OCR engines can "vote" amongst the two candidates and thus select "123" as the chosen value. Moreover, it could also happen that while the true value is "123", the engines identify instead "23", "120", and "153". Here, no single value wins the vote, so we can instead right-align the values and have the engines vote at the character level, voting between "(blank)", "1", and "1" for the first digit, between "2", "2" and "5" for the second digit, and between "3", "0", and "3" for the third digit. In this case, even though no single OCR engine correctly identified the number 123, the ensemble of the three engines did vote for the correct number.

There are two practical assumptions behind the use of ensemble methods, without which the method would likely provide no advantage to the user:

1. The OCR engines are good enough to correctly identify the words, even though they fail to identify all their characters. If some engines completely fail to identify a piece of text, then the accuracy of the method becomes limited as fewer engines would be voting on the results.
2. The errors made by the OCR engines are idiosyncratic. Otherwise, there would be no advantage to using multiple engines if their errors are systematic and common across all of them.

A more in-depth discussion of ensemble methods is contained in Lund (2014). Further, an experimental Stata implementation is available online.

Parameter tuning Parameter tuning—or more appropriately, hyper-parameter tuning, as it's known in the machine learning community—consists in using a set of ground truth data to select to create a measure of the overall digitization process, and then tune the parameters used in the different OCR steps to improve this measure.

At its simplest, this tuning could be done manually. That is what we do in figure 12, where we implemented a grid search in order to select the optimal binarization threshold of the fore-edge removal step (figure 5).

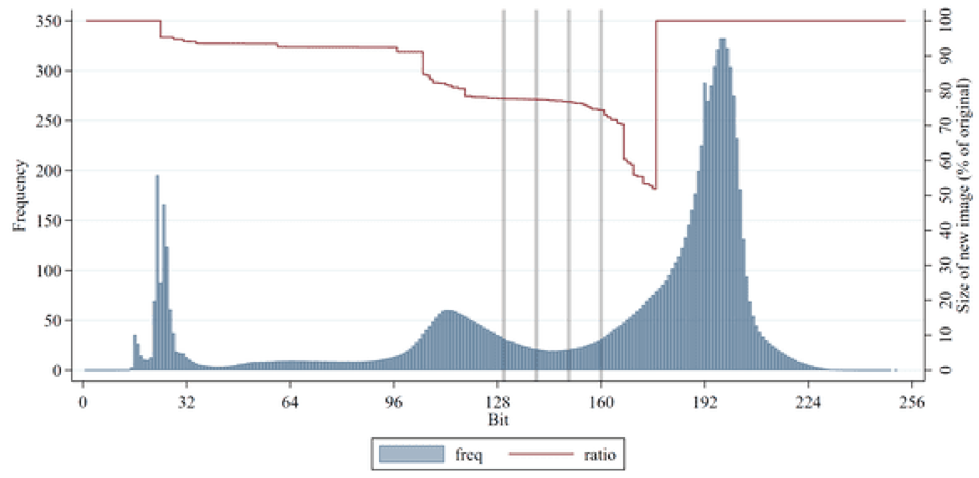


Figure 12: ***Fine-tuning pre-processing parameters.*** *This figures how the binarization parameter used in figure 5b affects the performance of the trimming step shown in figure 5. The histogram in blue shows the distribution of all points in a scanned page, from 0 (black) to 255 (white). The three modes of the histogram correspond to the black background and fonts, the dark gray book edge, and the light gray page background. The red line shows how changing the binarization parameter affects how much of the page is trimmed. Choosing a number between 128 and 160 will correctly separate the page background from the book edges, so the edges can be removed, without trimming down the actual page contents.*

In more advanced uses, grid search or manual tuning could be avoided via a descent method. Moreover, if we expect to perform large amounts of automatic parameter tuning, then we would advise users to leave aside part of the ground truth data so it can serve for cross-validation purposes, to avoid overfitting the algorithms to work on the ground truth data.

5 Summary

Altogether, we suggest that when using OCR to digitize data, the researcher becomes the practitioner. Although there is no one-size-fits-all solution, we argue for leveraging well-established and battle-tested tools, such as OpenCV and cloud-based OCR software, in order to construct digitization pipelines that can be tailored to the data sources at hand while requiring the least amount of customized, ad-hoc, code.

In particular, for large-scale datasets, we encourage researchers to apply a large share of their coding efforts in developing metrics for identifying errors in the data, either by constructing ground truths via human reviews or by exploiting characteristics of the data at hand. For instance, balance sheet records are ideally suited for large-scale digitization, as we can exploit balance sheets identities to allow for straightforward error detection. In contrast, security price data may be less well suited, as they contain fewer constraints that we can validate against. Moreover, accuracy metrics allow researchers to easily test and tune the different components of the digitization pipeline, so instead of being relegated to the latter stages of the digitization, they can be used from the beginning to build a more accurate pipeline.

Lastly, although the pipeline discussed in this paper can be quite complex depending on the scale and difficulty of the digitization task, we believe that for most use cases simple pipelines would still perform quite accurately while maintaining only light programming requirements. Instead, we recommend starting with simpler pipelines and only adding steps as needed, which maximize the likelihood of successful digitization efforts and avoid premature optimization problems.

References

Bako, S., S. Darabi, E. Shechtman, J. Wang, K. Sunkavalli, and P. Sen (2016). Removing shadows from images of documents. *Asian Conference on Computer Vision (ACCV 2016).*

Brunnermeier, M., S. Correia, S. Luck, and T. Zimmermann (2021). The Real Effects of Price Instability: Evidence from Hyperinflation and Deflation. *mimeo.*

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*(6), 679–698.

Carlson, M., S. Correia, and S. Luck (forthcoming). The Effects of Banking Competition on Growth and Financial Stability: Evidence from the National Banking Era. *Journal of Political Economy*.

Das, S., K. Ma, Z. Shu, D. Samaras, and R. Shilkrot (2019). Dewarpnet: Single-image document unwarping with stacked 3d and 2d regression networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 131–140.

Feng, H., Y. Wang, W. Zhou, J. Deng, and H. Li (2021). Doctr: Document image transformer for geometric unwarping and illumination correction.

Fu, B., M. Wu, R. Li, W. Li, Z. Xu, and C. Yang. A model-based book dewarping method using text line detection. In *Proc. CBDAR 2007*, pp. 63–70.

Gupta, A., R. Gutierrez-Osuna, M. Christy, B. Capitanu, L. Auvil, L. Grumbach, R. Furuta, and L. Mandell (2015). Automatic assessment of OCR quality in historical documents. In B. Bonet and S. Koenig (Eds.), *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pp. 1735–1741. AAAI Press.

Hegghammer, T. (2021, Nov). OCR with Tesseract, Amazon Textract, and Google Document AI: a benchmarking experiment. *Journal of Computational Social Science*.

Hough, P. V. (1959). Machine analysis of bubble chamber pictures. In *Proc. of the International Conference on High Energy Accelerators and Instrumentation, Sept. 1959*, pp. 554–556.

Jurafsky, D. and J. H. Martin (2009). *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, N.J.: Pearson Prentice Hall.

Kaehler, A. and G. Bradski (2016). *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library* (1st ed.). O'Reilly Media, Inc.

Kemighan, M. D., K. Church, and W. A. Gale (1990). A spelling correction program based on a noisy channel model. In *COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*.

Kiryati, N., Y. Eldar, and A. M. Bruckstein (1991). A probabilistic hough transform. *Pattern recognition 24*(4), 303–316.

Koistinen, M., K. Kettunen, and T. Pääkkönen (2017, May). Improving optical character recognition of Finnish historical newspapers with a combination of fraktur & antiqua models and image preprocessing. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, Gothenburg, Sweden, pp. 277–283. Association for Computational Linguistics.

Lefevere, F.-M. and M. Saric (2009). Detection of grooves in scanned images. US Patent 7,508,978.

Li, X., B. Zhang, J. Liao, and P. V. Sander (2019). Document rectification and illumination correction using a patch-based CNN. *CoRR abs/1909.09470*.

Lund, W. B. (2014). *Ensemble Methods for Historical Machine-Printed Document Recognition*. Brigham Young University.

Michalak, H. and K. Okarma (2019). Improvement of image binarization methods using image preprocessing with local entropy filtering for alphanumerical character recognition purposes. *Entropy 21*(6).

Pizer, S. M., J. R. Eugene, J. P. Ericksen, B. C. Yankaskas, and K. E. Muller (1990). Contrast-limited adaptive histogram equalization: Speed and effectiveness. In *Proceedings of the First Conference on Visualization in Biomedical Computing, Atlanta, Georgia*, Volume 337.

Sauvola, J. and M. Pietikäinen (2000). Adaptive document image binarization. *Pattern Recognition 33*(2), 225–236.

Sulaiman, A., K. Omar, and M. F. Nasrudin (2019). Degraded historical document binarization: A review on issues, challenges, techniques, and future directions. *Journal of Imaging 5*(4), 48.

Suzuki, S. et al. (1985). Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing 30*(1), 32–46.

Tian, Y. and S. G. Narasimhan (2011). Rectification and 3d reconstruction of curved document images. In *CVPR 2011*, pp. 377–384. IEEE.

Wolf, C. and J.-M. Jolion (2004). Extraction and recognition of artificial text in multimedia documents. *Pattern Analysis & Applications 6*(4), 309–326.

You, S., Y. Matsushita, S. N. Sinha, Y. Bou, and K. Ikeuchi (2016). Multiview rectification of folded documents. *CoRR abs/1606.00166*.