OCR with Tesseract, Amazon Textract, and Google Document AI: A Benchmarking Experiment*

Thomas Hegghammer†

24/06/2021

Abstract

Optical Character Recognition (OCR) can open up understudied historical documents to computational analysis, but the accuracy of OCR software varies. This article reports a benchmarking experiment comparing the performance of Tesseract, Amazon Textract, and Google Document AI on images of English and Arabic text. English-language book scans (n=322) and Arabic-language article scans (n=100) were replicated 43 times with different types of artificial noise for a corpus of 18,568 documents, generating 51,304 process requests. Document AI delivered the best results, and the server-based processors (Textract and Document AI) were substantially more accurate than Tesseract, especially on noisy documents. Accuracy for English was considerably better than for Arabic. Specifying the relative performance of three leading OCR products and the differential effects of commonly found noise types can help scholars identify better OCR solutions for their research needs. The test materials have been preserved in the openly available "Noisy OCR Dataset" (NOD).

Keywords: OCR, cloud computing, benchmarking

Word count: 4,042

*I thank Neil Ketchley and the participants in the University of Oslo Political Data Science seminar on 17 June 2021 for inputs and suggestions. Supplementary information and replication materials are available at https://github.com/Hegghammer/noisy-ocr-benchmark.

†Norwegian Defence Research Establishment (FFI) - thomas.hegghammer@ffi.no

1 Introduction

Few technologies hold as much promise for the social sciences and humanities as optical character recognition (OCR). Automated text extraction from digital images can open up large quantities of understudied historical documents to computational analysis, potentially generating deep new insights on the human past.

But OCR is a technology still in the making, and available software provides varying levels of accuracy. The best results are usually obtained with a tailored solution involving corpus-specific pre-processing (Bieniecki, Grabowski, and Rozenberg 2007; Dengel et al. 1997; Holley 2009; Lat and Jawahar 2018; Volk, Furrer, and Sennrich 2011; Wemhoener, Yalniz, and Manmatha 2013), model training (Boiangiu et al. 2016; Reul et al. 2018; Springmann et al. 2014; Wick, Reul, and Puppe 2018), or postprocessing (Kissos and Dershowitz 2016; Strohmaier et al. 2003; Thompson, McNaught, and Ananiadou 2015), but such procedures can be labour-intensive. Pre-trained, general OCR processors have a much higher potential for wide adoption in the scholarly community, and hence their out-of-the box performance is of scientific interest.

For long, general OCR processors such as *Tesseract* (tesseract-ocr 2019; Patel, Patel, and Patel 2012) only delivered perfect results under what we may call laboratory conditions, i.e., on noise-free, single-column text in a clear printed font. This limited their utility for real-life historical documents, which often contain shading, blur, shine-through, stains, skewness, complex layouts, and other things that produce OCR error.

Historically, general OCR processors have also struggled with non-Western languages (Kanungo, Marton, and Bulbul 1999), rendering them less useful for the many scholars working on documents in such languages.

In the past decade, advances in machine learning have led to substantial improvements in standalone OCR processor performance. Moreover, the past two years have seen the arrival of server-based processors such as Amazon Textract and Google Document AI, which offer document processing via an application processing interface (API) (Walker, Fujii, and Popat 2018). Media and blog coverage indicate that these processors deliver strong out-of-the-box performance[1], but those tests usually involve a small number of documents. Meanwhile, most rigorous benchmarking studies (Tafti et al. 2016; Vijayarani and Sakila 2015) predate the server-based processors, so we do not know how well they perform.

To find out, I conducted a benchmarking experiment comparing the performance of *Tesseract*, *Textract*, and *Document AI* on English and Arabic page scans. The objective was to generate statistically meaningful measurements of the accuracy of a selection of general OCR processors on document types commonly encountered in social scientific and humanities research.

[1]See, for example, Ted Han and Amanda Hickman, "Our Search for the Best OCR Tool, and What We Found," *OpenNews*, February 19, 2019 (https://source.opennews.org/articles/so-many-ocr-options/); Fabian Gringel, "Comparison of OCR tools: how to choose the best tool for your project," *Medium.com*, January 20, 2020 (https://medium.com/dida-machine-learning/comparison-of-ocr-tools-how-to-choose-the-best-tool-for-your-project-bd21fb9dce6b); Manoj Kukreja, "Compare Amazon Textract with Tesseract OCR — OCR & NLP Use Case," *TowardDataScience.com*, September 17, 2020 (https://towardsdatascience.com/compare-amazon-textract-with-tesseract-ocr-ocr-nlp-use-case-43ad7cd48748); Cem Dilmegani, "Best OCR by Text Extraction Accuracy in 2021," *AIMultiple.com*, June 6, 2021 (https://research.aimultiple.com/ocr-accuracy/).

The exercise yielded specifications for the relative performance of three leading OCR products as well as the differential effects of commonly found noise types. The findings can help scholars identify better OCR solutions for their research needs. The test materials, which have been preserved in the openly available "Noisy OCR Dataset" (NOD), can be used in future research.

2 Design

The experiment involved taking two document collections of 322 (English) and 100 (Arabic) page scans, replicating each over 40 times with different types of artificially generated noise, processing the full corpus of ~18,500 documents in each engine, and measuring the accuracy against ground truth using the Information Science Research Institute (ISRI) tool.

2.1 Processors

Tesseract, *Textract*, and *Document AI* were selected on the basis of their wide use, reputation for accuracy, and availability for programmatic use. Budget constraints prevented the inclusion of additional reputable processors such as *Adobe PDF Tools*, *ABBYY FineReader*, and *Microsoft Azure Computer Vision*, but these can be tested in the future using the same procedure and test materials.

A full description of these processors is beyond the scope of this article, but

Table 1: Features of Tesseract, Textract, and Document AI

Name	Maintainer	Installation	Architecture	Languages	Cost
Tesseract	Tesseract OCR Project	Local	LSTM	116	Free
Textract	Amazon Web Services	Server-based	Undisclosed	6	$1.50 per 1000 pages
Document AI	Google Cloud Services	Server-based	Undisclosed	60+	$1.50 per 1000 pages

Table 1 summarizes their main user-related features.[2] All processors are accessible programmatically from the main three operating systems and in multiple programming languages, including R and Python. The main difference is that *Tesseract* is open source and installed locally, whereas *Textract* and *Document* are paid services accessed remotely via a REST API.

2.2 Data

Test data were chosen for their similarity with archival materials commonly studied in the social sciences and humanities. This is in contrast to forms, receipts, and other business documents, which commercial OCR engines are primarily designed for, and which tend to get the most attention in media and blog reviews. Since many scholars work on materials in languages other than English, I also sought to include test materials in a non-Western language. Arabic was selected for its size as a world language and for its alphabetic structure, which lends itself to accuracy measurement with the ISRI tool.

The English test corpus consisted of the "Old Books Dataset" (Barcha 2017), a collection of 322 colour page scans from ten books printed between 1853 and 1920 (see

[2]For documentation, see the product websites: https://github.com/tesseract-ocr/tesseract, https://aws.amazon.com/textract/, and https://cloud.google.com/document-ai.

figures 1a and 1b) and subsequently extracted from the *Project Gutenberg* website. The dataset comes as 300 DPI and 500 DPI TIFF image files accompanied by ground truth in TXT files. I used the 300 DPI files in the experiment.

The Arabic test materials were drawn from the "Yarmouk Arabic OCR Dataset" (Doush, AlKhateeb, and Gharibeh 2018), a collection of 4,587 *Wikipedia* articles printed out to paper and colour scanned to PDF (see figures 1c and 1d). The dataset contains ground truth in HTML and TXT files. Due to the homogeneity of the collection, a randomly selected subset of 100 pages was deemed sufficient for the experiment.

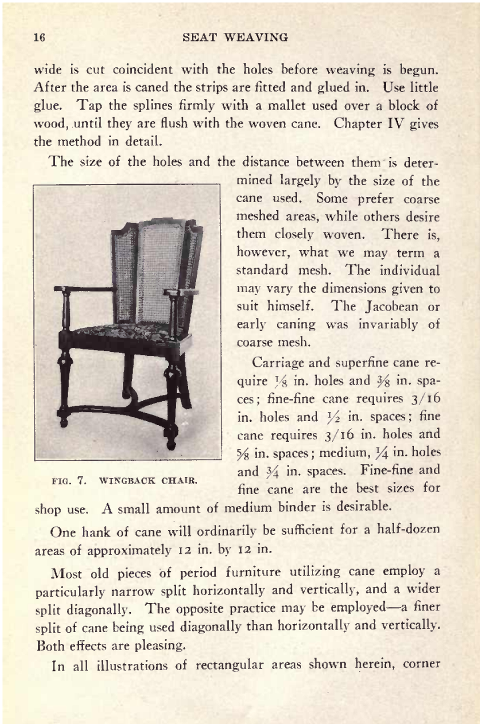
The Yarmouk dataset is suboptimal because it does not come from historical printed documents, but it is one of very few Arabic language datasets of some size with accompanying ground truth data. The Arabic and English test materials are thus not directly analogous, and in principle the former poses a lighter OCR challenge than the latter. Another limitation of the experiment is that the test materials only includes single-column text due to the complexities involved in measuring layout parsing accuracy.

2.3 Noise application

Real-life historical document scans rarely come in a form optimized for OCR processing. A key objective of the experiment was therefore to gauge the effect of different types of visual noise on performance. To achieve this, I programmatically applied different

Figure 1: Sample test documents in their original state

(a) Old Books j020



16 SEAT WEAVING

wide is cut coincident with the holes before weaving is begun. After the area is caned the strips are fitted and glued in. Use little glue. Tap the splines firmly with a mallet used over a block of wood, until they are flush with the woven cane. Chapter IV gives the method in detail.

The size of the holes and the distance between them is determined largely by the size of the cane used. Some prefer coarse meshed areas, while others desire them closely woven. There is, however, what we may term a standard mesh. The individual may vary the dimensions given to suit himself. The Jacobean or early caning was invariably of coarse mesh.

Carriage and superfine cane require 1/8 in. holes and 3/8 in. spaces; fine-fine cane requires 3/16 in. holes and 1/2 in. spaces; fine cane requires 3/16 in. holes and 5/8 in. spaces; medium, 1/4 in. holes and 3/4 in. spaces. Fine-fine and fine cane are the best sizes for shop use. A small amount of medium binder is desirable.

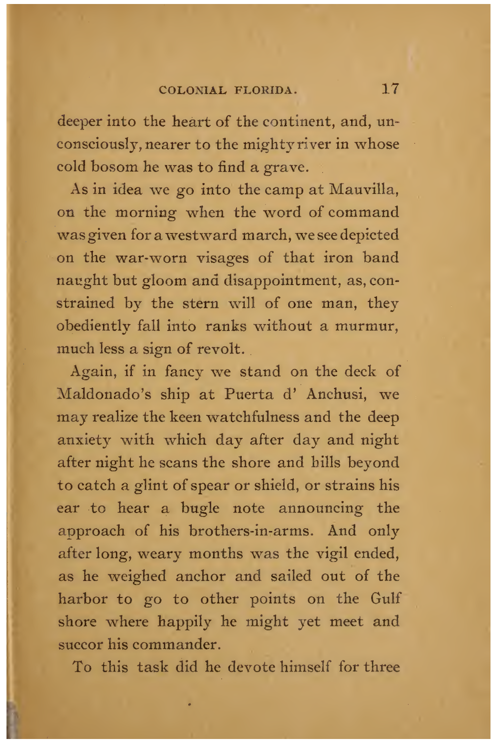
FIG. 7. WINGBACK CHAIR.

One hank of cane will ordinarily be sufficient for a half-dozen areas of approximately 12 in. by 12 in.

Most old pieces of period furniture utilizing cane employ a particularly narrow split horizontally and vertically, and a wider split diagonally. The opposite practice may be employed—a finer split of cane being used diagonally than horizontally and vertically. Both effects are pleasing.

In all illustrations of rectangular areas shown herein, corner

(b) Old Books g023



COLONIAL FLORIDA. 17

deeper into the heart of the continent, and, unconsciously, nearer to the mighty river in whose cold bosom he was to find a grave.

As in idea we go into the camp at Mauvilla, on the morning when the word of command was given for a westward march, we see depicted on the war-worn visages of that iron band naught but gloom and disappointment, as, constrained by the stern will of one man, they obediently fall into ranks without a murmur, much less a sign of revolt.

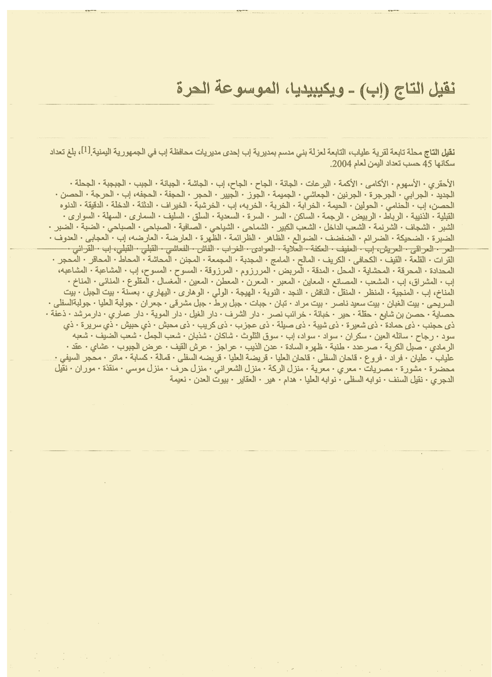
Again, if in fancy we stand on the deck of Maldonado's ship at Puerta d' Anchusi, we may realize the keen watchfulness and the deep anxiety with which day after day and night after night he scans the shore and bills beyond to catch a glint of spear or shield, or strains his ear to hear a bugle note announcing the approach of his brothers-in-arms. And only after long, weary months was the vigil ended, as he weighed anchor and sailed out of the harbor to go to other points on the Gulf shore where happily he might yet meet and succor his commander.

To this task did he devote himself for three

(c) Yarmouk 25223-1

إحيائية - ويكيبيديا، الموسوعة الحرة

(d) Yarmouk 4155-1



نقيل التاج (إب) - ويكيبيديا، الموسوعة الحرة

نقيل التاج محلة تابعة لقرية علياب، التابعة لعزلة بني منسم بمديرية إب إحدى مديريات محافظة إب في الجمهورية اليمنية[1]، بلغ تعداد سكانها 45 حسب تعداد اليمن لعام 2004.

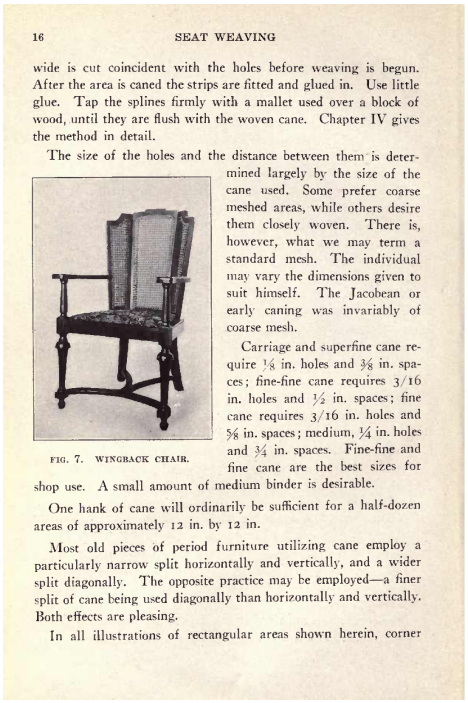
types of artifical noise to the test materials, so as to allow isolation of noise effects at the measurement stage. Specifically, the two dataset were duplicated 43 times, each with a different type of modification. The R code used for noise generation is included in the supplementary information.

I began by creating a binary version of each image, so that there were two versions — colour and greyscale — with no added noise (see figure 2a and 2b). I then wrote functions to generate six ideal types of image noise: "blur," "weak ink," "salt and pepper," "watermark," "scribbles," and "ink stains" (see figures 2c-d and 3a-d). While not an exhaustive list of possible noise types, they represent several of the most common ones found in historical document scans. I applied each of the six filters to both the colour version and the binary version of the images, thus creating 12 additional versions of each image. Lastly I applied all available combinations of two noise filters to the colour and binary images, for an additional 30 versions.

This generated a total of 44 image versions divided into three categories of noise intensity: 2 versions with no added noise, 12 versions with one layer of noise, and 30 versions with two layers of noise. This amounted to an English test corpus of 14,168 documents and an Arabic test corpus of 4,400 documents. The dataset is preserved on Zenodo as the "Noisy OCR Dataset" (Hegghammer 2021).

Figure 2: Sample test document ("Old Books j020") with noise applied

(a) Original



16 SEAT WEAVING

wide is cut coincident with the holes before weaving is begun. After the area is caned the strips are fitted and glued in. Use little glue. Tap the splines firmly with a mallet used over a block of wood, until they are flush with the woven cane. Chapter IV gives the method in detail.

The size of the holes and the distance between them is determined largely by the size of the cane used. Some prefer coarse meshed areas, while others desire them closely woven. There is, however, what we may term a standard mesh. The individual may vary the dimensions given to suit himself. The Jacobean or early caning was invariably of coarse mesh.

FIG. 7. WINGBACK CHAIR.

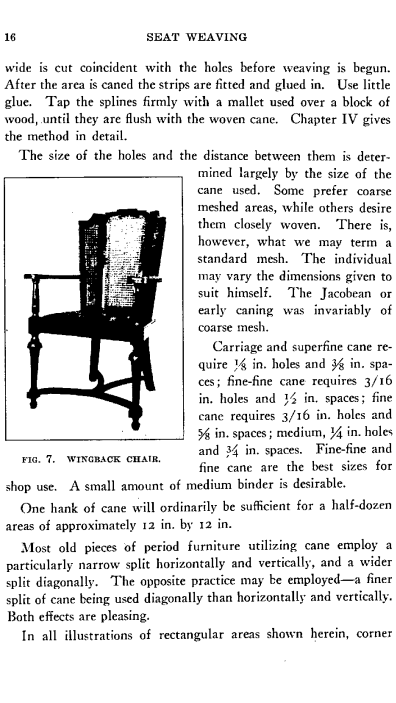
Carriage and superfine cane require ⅛ in. holes and ⅜ in. spaces; fine-fine cane requires 3/16 in. holes and ½ in. spaces; fine cane requires 3/16 in. holes and ⅝ in. spaces; medium, ¼ in. holes and ¾ in. spaces. Fine-fine and fine cane are the best sizes for shop use. A small amount of medium binder is desirable.

One hank of cane will ordinarily be sufficient for a half-dozen areas of approximately 12 in. by 12 in.

Most old pieces of period furniture utilizing cane employ a particularly narrow split horizontally and vertically, and a wider split diagonally. The opposite practice may be employed—a finer split of cane being used diagonally than horizontally and vertically. Both effects are pleasing.

In all illustrations of rectangular areas shown herein, corner

(b) Binary



16 SEAT WEAVING

wide is cut coincident with the holes before weaving is begun. After the area is caned the strips are fitted and glued in. Use little glue. Tap the splines firmly with a mallet used over a block of wood, until they are flush with the woven cane. Chapter IV gives the method in detail.

The size of the holes and the distance between them is determined largely by the size of the cane used. Some prefer coarse meshed areas, while others desire them closely woven. There is, however, what we may term a standard mesh. The individual may vary the dimensions given to suit himself. The Jacobean or early caning was invariably of coarse mesh.

FIG. 7. WINGBACK CHAIR.

Carriage and superfine cane require ⅛ in. holes and ⅜ in. spaces; fine-fine cane requires 3/16 in. holes and ½ in. spaces; fine cane requires 3/16 in. holes and ⅝ in. spaces; medium, ¼ in. holes and ¾ in. spaces. Fine-fine and fine cane are the best sizes for shop use. A small amount of medium binder is desirable.

One hank of cane will ordinarily be sufficient for a half-dozen areas of approximately 12 in. by 12 in.

Most old pieces of period furniture utilizing cane employ a particularly narrow split horizontally and vertically, and a wider split diagonally. The opposite practice may be employed—a finer split of cane being used diagonally than horizontally and vertically. Both effects are pleasing.

In all illustrations of rectangular areas shown herein, corner

(c) Blur

16 SEAT WEAVING

wide is cut coincident with the holes before weaving is begun. After the area is caned the strips are fitted and glued in. Use little glue. Tap the splines firmly with a mallet used over a block of wood, until they are flush with the woven cane. Chapter IV gives the method in detail.

The size of the holes and the distance between them is determined largely by the size of the cane used. Some prefer coarse meshed areas, while others desire them closely woven. There is, however, what we may term a standard mesh. The individual may vary the dimensions given to suit himself. The Jacobean or early caning was invariably of coarse mesh.

FIG. 7. WINGBACK CHAIR.

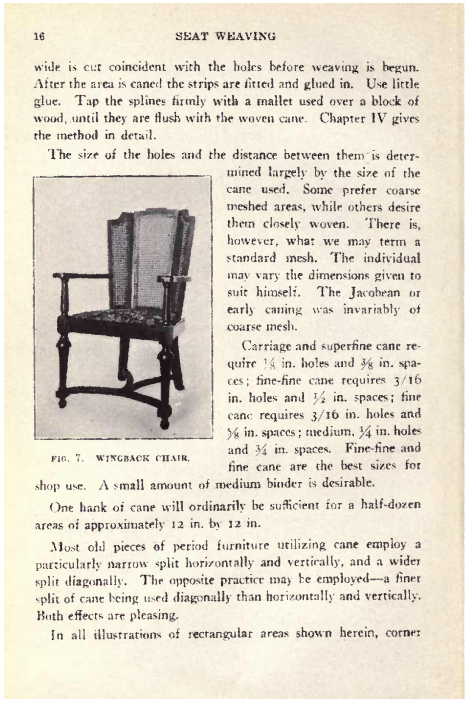
Carriage and superfine cane require ⅛ in. holes and ⅜ in. spaces; fine-fine cane requires 3/16 in. holes and ½ in. spaces; fine cane requires 3/16 in. holes and ⅝ in. spaces; medium, ¼ in. holes and ¾ in. spaces. Fine-fine and fine cane are the best sizes for shop use. A small amount of medium binder is desirable.

One hank of cane will ordinarily be sufficient for a half-dozen areas of approximately 12 in. by 12 in.

Most old pieces of period furniture utilizing cane employ a particularly narrow split horizontally and vertically, and a wider split diagonally. The opposite practice may be employed—a finer split of cane being used diagonally than horizontally and vertically. Both effects are pleasing.

In all illustrations of rectangular areas shown herein, corner

(d) Weak ink



16 SEAT WEAVING

wide is cut coincident with the holes before weaving is begun. After the area is caned the strips are fitted and glued in. Use little glue. Tap the splines firmly with a mallet used over a block of wood, until they are flush with the woven cane. Chapter IV gives the method in detail.

The size of the holes and the distance between them is determined largely by the size of the cane used. Some prefer coarse meshed areas, while others desire them closely woven. There is, however, what we may term a standard mesh. The individual may vary the dimensions given to suit himself. The Jacobean or early caning was invariably of coarse mesh.

FIG. 7. WINGBACK CHAIR.

Carriage and superfine cane require ⅛ in. holes and ⅜ in. spaces; fine-fine cane requires 3/16 in. holes and ½ in. spaces; fine cane requires 3/16 in. holes and ⅝ in. spaces; medium, ¼ in. holes and ¾ in. spaces. Fine-fine and fine cane are the best sizes for shop use. A small amount of medium binder is desirable.

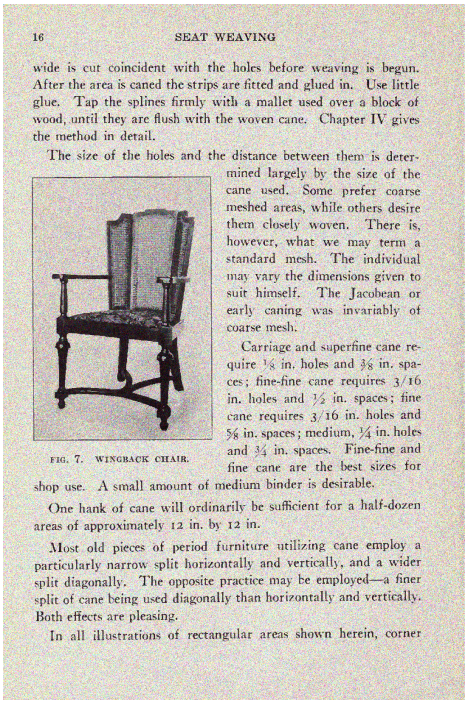
One hank of cane will ordinarily be sufficient for a half-dozen areas of approximately 12 in. by 12 in.

Most old pieces of period furniture utilizing cane employ a particularly narrow split horizontally and vertically, and a wider split diagonally. The opposite practice may be employed—a finer split of cane being used diagonally than horizontally and vertically. Both effects are pleasing.

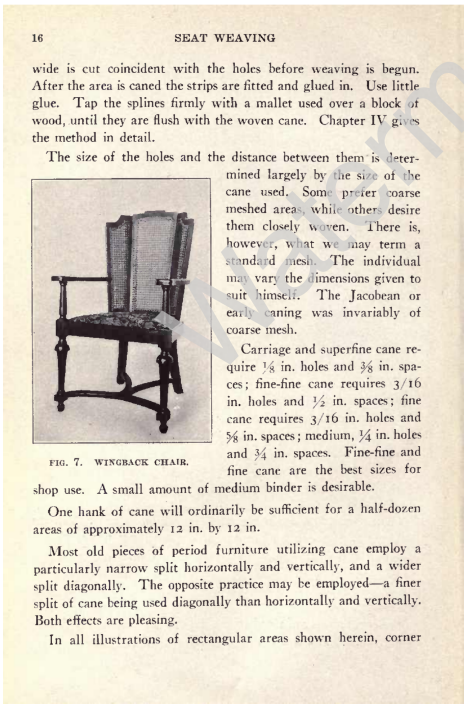
In all illustrations of rectangular areas shown herein, corner

Figure 3: Sample test document ("Old Books j020") with noise applied (cont.)"

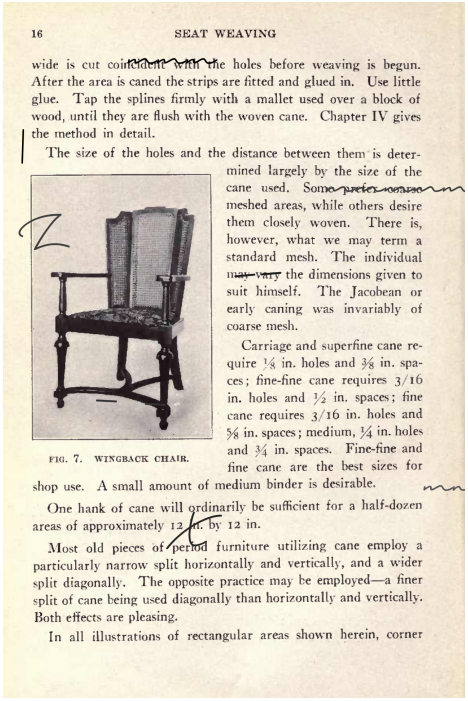
(a) Salt and pepper



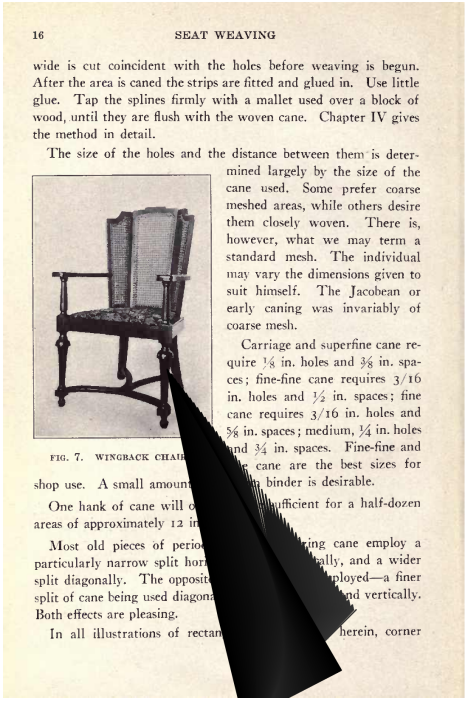
(b) Watermark



(c) Scribbles



(d) Ink stains



2.4 Processing

The experiment sought to measure out-of-the-box performance, so documents were submitted without further preprocessing using the OCR engines' default settings.[3] While this is admittedly an uncommon use of *Tesseract*, it treats the engines equally and helps highlight the degree to which Tesseract is dependent on image preprocessing.

The English corpus was submitted to all three OCR engines in a total of 42,504 document processing requests. The Arabic corpus was only submitted to *Tesseract* and *Document AI* — since *Textract* does not support Arabic — for a total of 8,800 processing requests.

The *Tesseract* processing was done in R with the package `tesseract` (v4.1.1) on a desktop computer. For *Textract*, the processing was carried out via the R package `paws` (v0.1.11), which provides a wrapper for the Amazon Web Services API. For *Document AI*, I used the R package `daiR` (v0.8.0) to access the *Document AI* API v1 endpoint. The processing was carried out in April and May of 2021.

2.5 Measurement

Accuracy was measured using the ISRI tool (Rice and Nartker 1996) in Eddie Santos (2019) updated version with UTF-8 support.[4] ISRI compares two texts — in this case OCR output to ground truth — and returns a range of measures for divergence,

[3] The only exception was the setting of the relevant language libraries in *Tesseract*.

[4] Alternatives exist (Carrasco 2014; Alghamdi, Alkhazi, and Teahan 2016), but ISRI was deemed sufficient, notably because the Yarmouk texts do not contain diacritics.

notably a document's overall character accuracy and word accuracy expressed in percent. Character accuracy was not used, because the character accuracy rates returned by the ISRI tool contained a substantial proportion (~20%) of irregular values, notably percentages below zero and over 100. In the reporting below I therefore use word accuracy rates, transformed to word error rates by subtracting them from 100.[5]

3 Results

The main results are shown in Figure 4 and reveal clear patterns. *Document AI* had consistently lower error rates, with *Textract* coming in a close second, and *Tesseract* last. More noise yielded higher error rates in all engines, but *Tesseract* was significantly more sensitive to noise than the two others. Overall, there was a significant performance gap between the server-based processors (*Document AI* and *Textract*) on one side and the local installation (*Tesseract*) on the other. Only on noise-free documents in English could *Tesseract* compete with the two other processors.

We also see a significant performance difference across languages. Both *Document AI* and *Tesseract* delivered markedly lower accuracy for Arabic than they did for English. This was despite the Arabic corpus consisting of Internet articles in a single, very common font, while the English corpus contained old book scans in several

[5]The word accuracy measurements contained a certain proportion (REF) of null values, especially in the output produced by *Tesseract*. Visual inspection of a sample of the concerned documents suggests they contained largely garbled text, so they were treated as zeroes.

different fonts. An analogous Arabic corpus would likely have produced an even larger performance gap. This said, *Document AI* represents a significant improvement on *Tesseract* as far as out-of-the-box Arabic OCR is concerned.

Disaggregating the results by noise type shows a more detailed picture (see Figures 5 and 6). Beyond the patterns already described, we see, for example, that both *Textract* and *Tesseract* performed somewhat better on greyscale versions of the test images than on the colour version. We also note that all engines struggled with blur, while *Tesseract* was much more sensitive to salt & pepper noise than the two other engines. Incidentally, it is not surprising that the ink stain filter yielded lower accuracy throughout since it completely concealed part of the text.

4 Conclusion

This article described a systematic test of three general OCR processors on a large new dataset of English and Arabic documents. Its results suggests that the server-based engines *Document AI* and *Textract* deliver markedly higher out-of-the-box accuracy than the standalone *Tesseract* library, especially on noisy documents. It also indicates that certain types of "integrated" noise, such as blur and salt and pepper, generate more error than "superimposed" noise such as watermarks, scribbles, and even ink stains. Furthermore, it suggests that the "OCR language gap" still persists, although *Document AI* seems to have partially closed it, at least for Arabic.

The findings can help scholars develop OCR solutions to suit their use cases.

Figure 4: Word error rates by engine and noise level for English and Arabic documents

Mean error rates in coloured boxes. X axes cropped for visibility, leaving out the tails of the distributions.

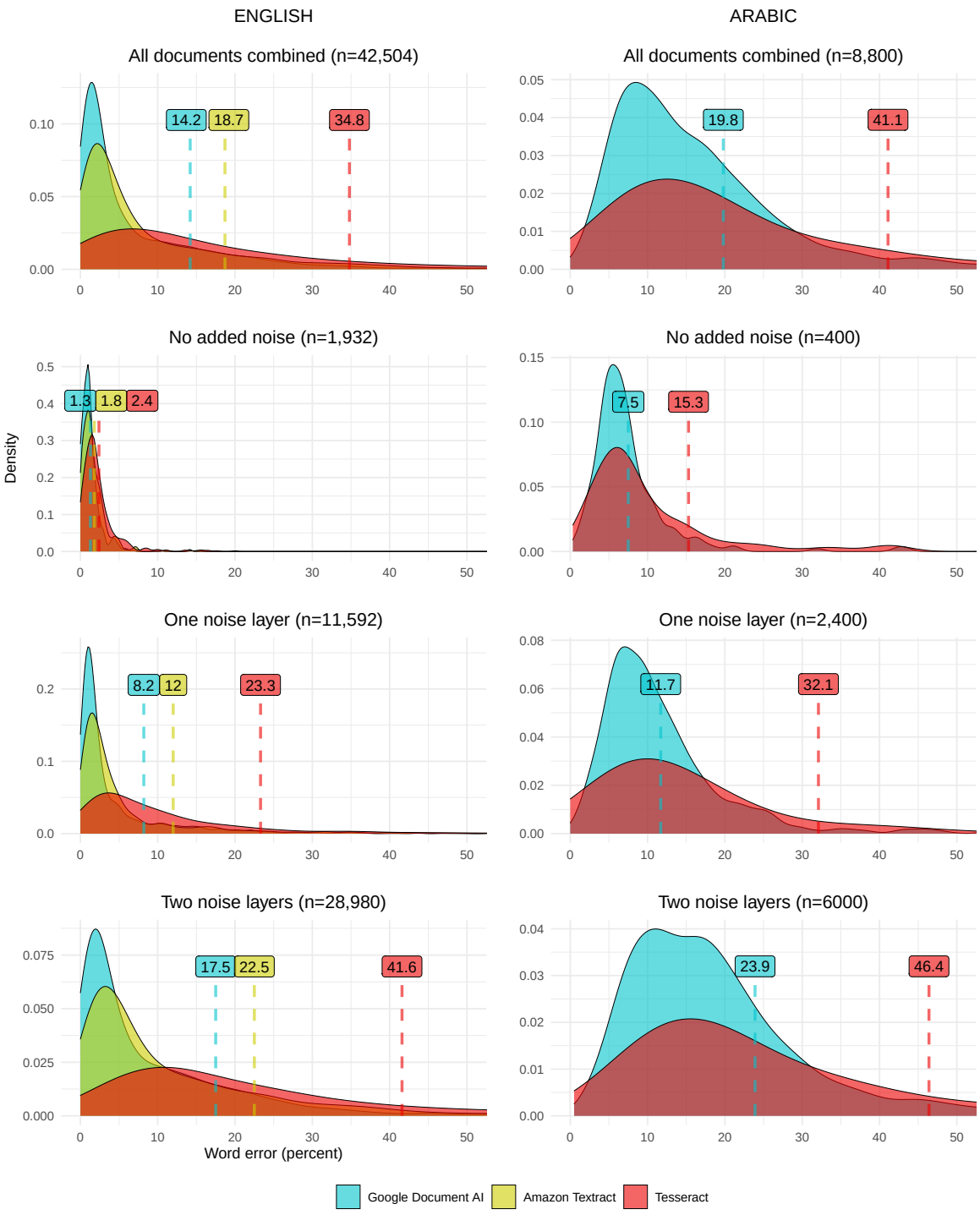


Figure 5: Word error rates by engine and noise type for English-language documents

Data: Single−column text in historical book scans with noise added artificially (n=42,504; 322 per engine and noise type).
Noise codes: 'col'=colour, 'bin'=binary, 'blur'=blur, 'weak'=weak ink, 'snp'=salt&pepper, 'wm'=watermark, 'scrib'=scribbles, 'ink'=ink stains.

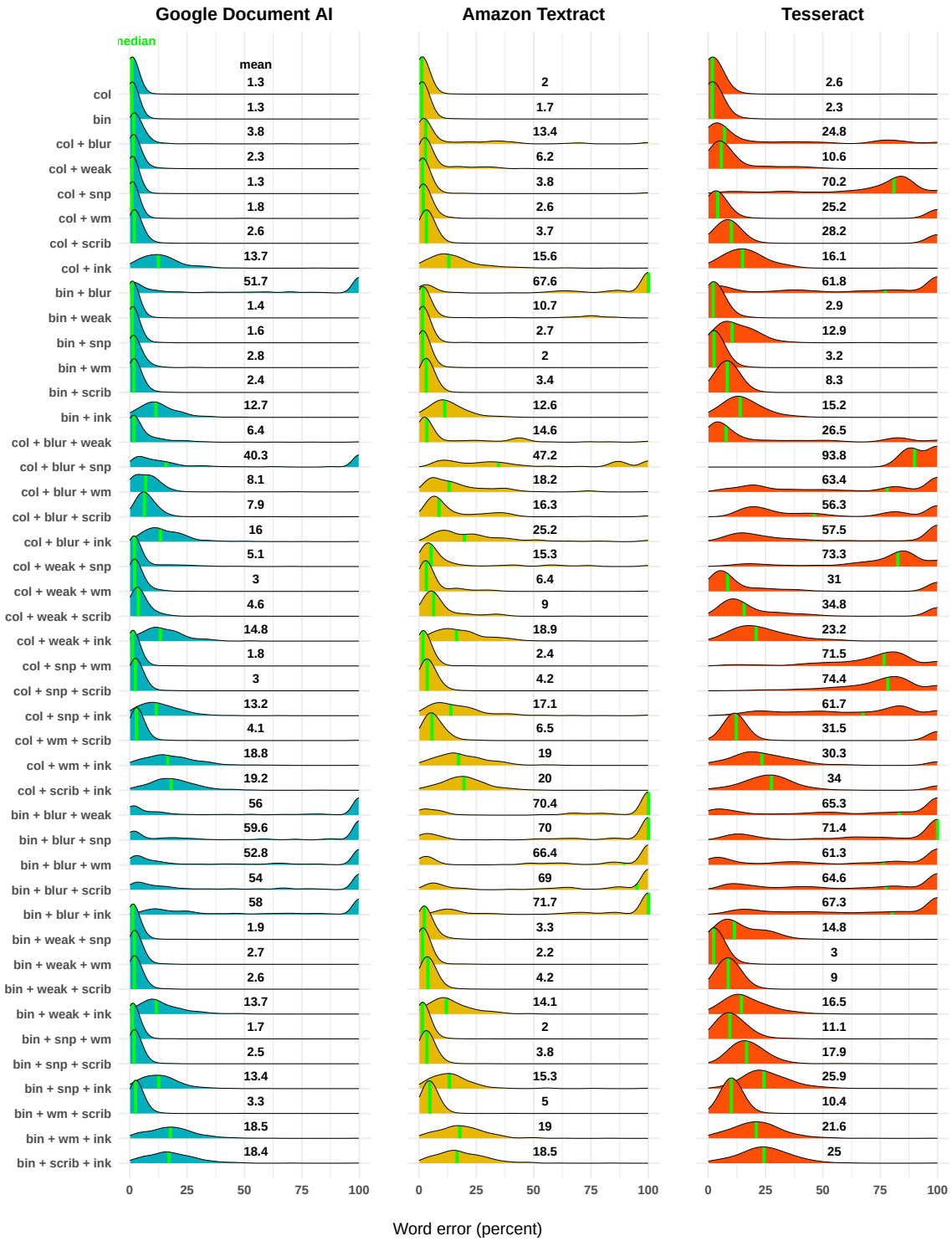
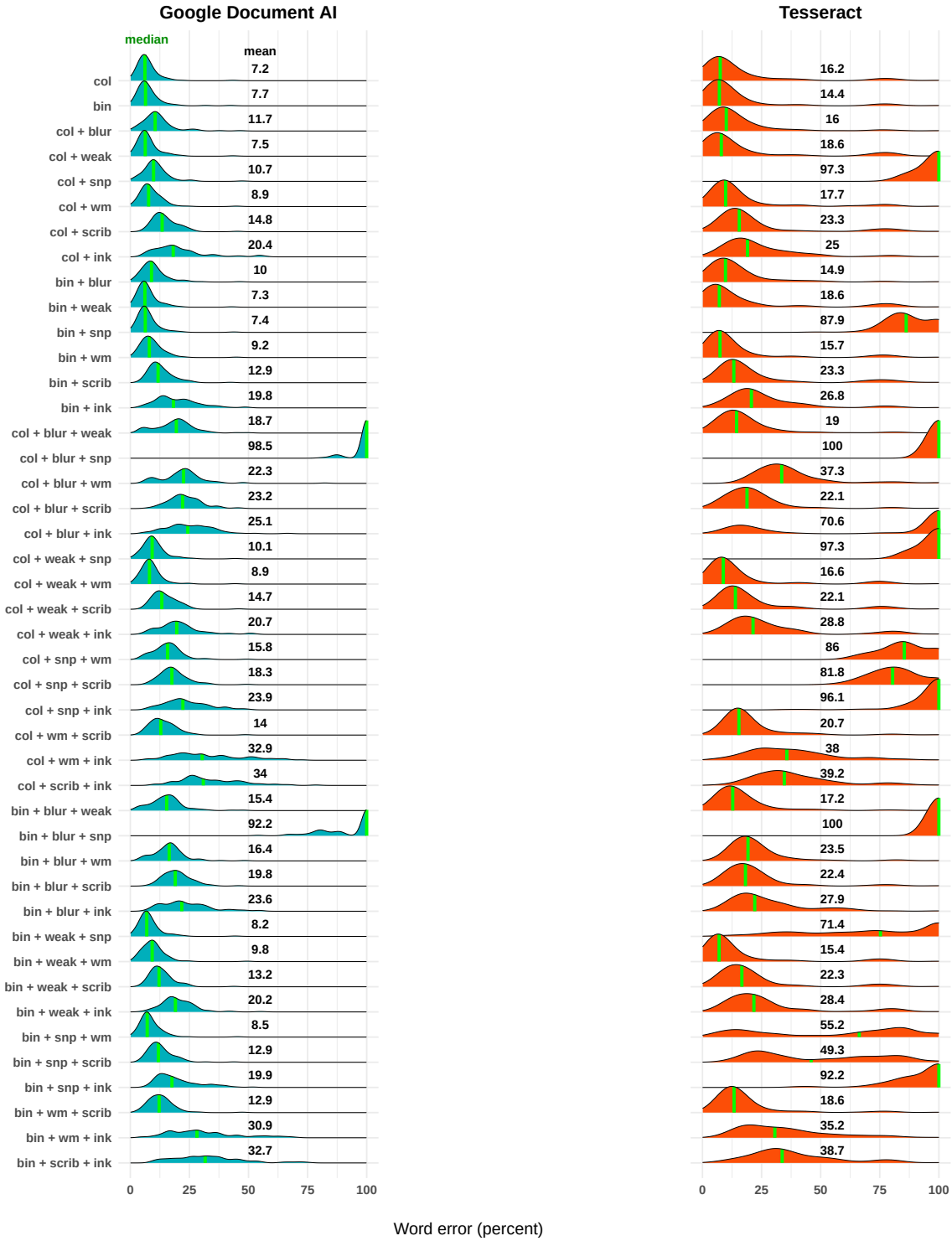


Figure 6: Word error rates by engine and noise type for Arabic-language documents

Data: Single−column text in image scans of Arabic Wikipedia pages with noise added artificially (n = 8800; 100 per engine and noise type).
Noise codes: 'col'=colour, 'bin'=binary, 'blur'=blur, 'weak'=weak ink, 'snp'=salt&pepper, 'wm'=watermark, 'scrib'=scribbles, 'ink'=ink stains.



Out-of-the box accuracy rates are not the only relevant metric, since Tesseract and other engines can be trained, and since server-based processors have drawbacks such as financial cost and data privacy concerns. But having baseline data on relative processor performance and differential effects of noise types can inform the choice of processor and help design preprocessing strategies.

The article also contributes to specialist OCR research in three ways: by presenting new benchmarking data, by developing a new systematic approach to studying noise effects, and by introducing a large new OCR dataset for use in future OCR research. The study has several limitations, notably the narrow range of processors and the use of single-column test materials, which does not capture layout parsing capabilities. This author's anecdotal experience with *Document AI* and *Textract* on multi-column text suggest that these engines have excellent character recognition but still struggle with layout parsing. This, in other words, appears to be the principal remaining frontier in OCR development.

References

Alghamdi, Mansoor A, Ibrahim S Alkhazi, and William J Teahan. 2016. "Arabic OCR Evaluation Tool." In *2016 7th International Conference on Computer Science and Information Technology (CSIT)*, 1–6. IEEE.

Barcha, Pedro. 2017. "Old Books Dataset." *GitHub Repository*. GitHub. https://github.com/PedroBarcha/old-books-dataset.

Bieniecki, Wojciech, Szymon Grabowski, and Wojciech Rozenberg. 2007. "Image Preprocessing for Improving Ocr Accuracy." In *2007 International Conference on Perspective Technologies and Methods in MEMS Design*, 75–80. IEEE.

Boiangiu, Costin-Anton, Radu Ioanitescu, Razvan-Costin Dragomir, and others. 2016. "Voting-Based OCR System." *The Proceedings of Journal ISOM* 10: 470–86.

Carrasco, Rafael C. 2014. "An Open-Source OCR Evaluation Tool." In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, 179–84.

Dengel, Andreas, Rainer Hoch, Frank Hönes, Thorsten Jäger, Michael Malburg, and Achim Weigel. 1997. "Techniques for Improving OCR Results." In *Handbook of Character Recognition and Document Image Analysis*, 227–58. World Scientific.

Doush, Iyad Abu, Faisal AlKhateeb, and Anwaar Hamdi Gharibeh. 2018. "Yarmouk Arabic OCR Dataset." In *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, 150–54. IEEE.

Hegghammer, Thomas. 2021. "Noisy OCR Dataset." *Zenodo Repository*. Zenodo.

https://zenodo/TBC.

Holley, Rose. 2009. "How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs." *D-Lib Magazine* 15 (3/4).

Kanungo, Tapas, Gregory A Marton, and Osama Bulbul. 1999. "Performance Evaluation of Two Arabic OCR Products." In *27th AIPR Workshop: Advances in Computer-Assisted Recognition*, 3584:76–83. International Society for Optics; Photonics.

Kissos, Ido, and Nachum Dershowitz. 2016. "OCR Error Correction Using Character Correction and Feature-Based Word Classification." In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, 198–203. IEEE.

Lat, Ankit, and CV Jawahar. 2018. "Enhancing Ocr Accuracy with Super Resolution." In *2018 24th International Conference on Pattern Recognition (ICPR)*, 3162–67. IEEE.

Patel, Chirag, Atul Patel, and Dharmendra Patel. 2012. "Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study." *International Journal of Computer Applications* 55 (10): 50–56.

Reul, Christian, Uwe Springmann, Christoph Wick, and Frank Puppe. 2018. "Improving OCR Accuracy on Early Printed Books by Utilizing Cross Fold Training and Voting." In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, 423–28. IEEE.

Rice, Stephen V, and Thomas A Nartker. 1996. "The ISRI Analytic Tools for OCR Evaluation." *UNLV/Information Science Research Institute, TR-96* 2.

Santos, Eddie Antonio. 2019. "OCR Evaluation Tools for the 21st Century." In *Proceedings of the 3rd Workshop on the Use of Computational Methods in the Study of Endangered Languages Volume 1 (Papers)*, 23–27. Honolulu: Association for Computational Linguistics. https://www.aclweb.org/anthology/W19-6004.

Springmann, Uwe, Dietmar Najock, Hermann Morgenroth, Helmut Schmid, Annette Gotscharek, and Florian Fink. 2014. "OCR of Historical Printings of Latin Texts: Problems, Prospects, Progress." In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, 71–75.

Strohmaier, Christian M, Christoph Ringlstetter, Klaus U Schulz, and Stoyan Mihov. 2003. "Lexical Postcorrection of OCR-Results: The Web as a Dynamic Secondary Dictionary?" In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, 3:1133–33. Citeseer.

Tafti, Ahmad P, Ahmadreza Baghaie, Mehdi Assefi, Hamid R Arabnia, Zeyun Yu, and Peggy Peissig. 2016. "OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym." In *International Symposium on Visual Computing*, 735–46. Springer.

tesseract-ocr. 2019. "Tesseract OCR 4.1.1." *GitHub Repository*. GitHub. https://github.com/tesseract-ocr/tesseract.

Thompson, Paul, John McNaught, and Sophia Ananiadou. 2015. "Customised OCR

Correction for Historical Medical Text." In *2015 Digital Heritage*, 1:35–42. IEEE.

Vijayarani, S, and A Sakila. 2015. "Performance Comparison of OCR Tools." *International Journal of UbiComp (IJU)* 6 (3): 19–30.

Volk, Martin, Lenz Furrer, and Rico Sennrich. 2011. "Strategies for Reducing and Correcting OCR Errors." In *Language Technology for Cultural Heritage*, 3–22. Springer.

Walker, Jake, Yasuhisa Fujii, and Ashok C Popat. 2018. "A Web-Based Ocr Service for Documents." In *Proceedings of the 13th IAPR International Workshop on Document Analysis Systems (DAS), Vienna, Austria*. Vol. 1.

Wemhoener, David, Ismet Zeki Yalniz, and R Manmatha. 2013. "Creating an Improved Version Using Noisy OCR from Multiple Editions." In *2013 12th International Conference on Document Analysis and Recognition*, 160–64. IEEE.

Wick, Christoph, Christian Reul, and Frank Puppe. 2018. "Comparison of OCR Accuracy on Early Printed Books Using the Open Source Engines Calamari and OCRopus." *J. Lang. Technol. Comput. Linguistics* 33 (1): 79–96.