





# Making PDFs Accessible for Visually Impaired Users (and Findable for Everybody Else)

Ruben van Heusden<sup>(✉)</sup> , Hazel Ling, Lars Nelissen, and Maarten Marx<sup>(✉)</sup> 

Information Retrieval Lab, Informatics Institute, University of Amsterdam,  
Amsterdam, Netherlands

{r.j.vanheusden,maartenmarx}@uva.nl  
<https://irlab.science.uva.nl>

**Abstract.** We treat documents released under the Dutch Freedom of Information Act as FAIR scientific data and find that they are not findable nor accessible, due to text malformations caused by redaction software. Our aim is to repair these documents. We propose a simple but strong heuristic for detecting wrongly OCRed text segments, and we then repair only these OCR mistakes by prompting a large language model. This makes the documents better findable through full text search, but the repaired PDFs do still not adhere to accessibility standards. Converting them into HTML documents, keeping all essential layout and markup, makes them not only accessible to the visually impaired, but also reduces their size by up to two orders of magnitude. The costs of this way of repairing are roughly one dollar for the 17K pages in our corpus, which is very little compared to the large gains in information quality.

**Keywords:** Optical Character Recognition · Corpus Curation · Quality Control · Digital Libraries

## 1 Introduction

The guidelines of the European Union on the re-usability of data stipulate that data released by governments should be reusable [8]. In fact, the guidelines prescribed bear a strong resemblance to the FAIR data principles [17]: released data should be findable, accessible, interoperable and indeed reusable. Findability and accessibility, in particular for the visually impaired, are greatly hampered by the application of redaction software to documents that the government is obliged to release under the Freedom of Information Act. This redaction software is used to black out text for reasons of privacy, national security, competition, etc., using named entity recognition techniques [5].

---

**GitHub:** <https://github.com/irlabamsterdam/accessibilifier>.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023  
O. Alonso et al. (Eds.): TPD 2023, LNCS 14241, pp. 239–245, 2023.  
[https://doi.org/10.1007/978-3-031-43849-3\\_21](https://doi.org/10.1007/978-3-031-43849-3_21)

The definition of visual impairment, as specified by the World Wide Web Consortium (W3C), states that someone who is visually impaired is not blind, but whose vision is severely limited, even with the use of glasses or other implements.<sup>1</sup> Visually impaired individuals often use additional software, (e.g. for magnification or speak aloud) which uses the machine readable text in a PDF.

We found that 30% of over 1 million text pages released by the Dutch government does not contain a single machine readable character, and almost all other pages contain non existing words, most likely caused by OCR errors. Believe it or not, but treating brand new digital born documents as if they originate from the 19th century is common practice in redaction software: scan the document, black out the detected text, and, if one is lucky, make the text on the scan also machine interpretable using optical character recognition (OCR). Of course, this is effective in ensuring that redacted text cannot be brought back [3], but is detrimental from the perspective of information quality. Full text search in these documents will not work well [15], and listening to automatically read aloud documents without machine readable text or text containing sequences of malformed words is not informative nor pleasant.

The research reported in this paper originates in this problem and our wish to repair the damage done to these documents. We do this in three consecutive steps: discover segments of mangled OCRed text using a simple heuristic, repair these segments using a Large Language Model (ChatGPT), and convert the inaccessible PDF into XML containing the repaired text. The converted text then does apply to accessibility standards<sup>2</sup> and as a bonus is typically two orders of magnitude smaller in size [10]. Although it does not need training data, the mangled OCR segment heuristic works well. Only repairing those segments is more efficient, cheaper and helps avoiding false positives. We compare repairing by Tesseract and by ChatGPT, with the latter producing less non-existing words, but having a slightly higher Word Error Rate due to hallucination, besides from being more expensive both in processing time and costs. Both methods are effective in reducing the average length of mangled segments, thus improving accessibility, as shorter segments of mangled words are easier to still understand when hearing them spoken through Text-to-Speech software.

If anything, this work highlights the need for organizations and governments to take accessibility seriously, and to prevent the problems pointed out in this paper by changing their workflow at the source, where mistakes are easier to fix than the reconstruction and correction steps that have to be taken when post-correcting mistakes.

## 2 Related Work

Techniques for automatic detection of OCR error vary from dictionary look-ups and ngram methods to more recent sequence-to-sequence models, as well as unsupervised methods that rely on known-good background corpora [1, 2, 4,

<sup>1</sup> <https://www.w3.org/TR/low-vision-needs/>.

<sup>2</sup> <https://www.iso.org/standard/58625.html>.

14]. These researchers typically not only detect bad segments of text, but also propose methods for repairing it, such as using ngram probabilities to replace low-probability ngrams with higher probability ones, or by using a sequence-to-sequence models to 'translate' the incorrect words [1,2]. Recently, a pipeline that combines much of the aforementioned techniques has been presented for OCR error detection in the Dutch language [6]. The system, named *QuPipe*, combines dictionary lookup, trigrams, garbage detection, language detection and statistics on word and document level to detect errors in historical Dutch news articles.

A work that is conceptually close to ours is that of Schaefer & Neudecker [13], who also employ a two-step detection and correction pipeline for OCR post-correction on historical documents. In their work they train a character-level LSTM to detect OCR mistakes in the input text and repair them using a trained sequence-to-sequence model. However, since we do not have ground truth data, we cannot train an LSTM model to detect bad segments, and instead use an unsupervised method. We do however follow their reasoning in opting for a high precision detector model. Turró [16] mentions that the most accessible form of a PDF contains tags that define the structural elements of the PDF. A way to create these structural tags from a PDF is to use *pdfhtml*, a tool that is included in the *poppler* package and that outputs information regarding the fonts and positions of the text.

## 3 Method

### 3.1 Data

For the evaluation of our approach we use the cleanest part of the 1 million page corpus of Dutch Freedom of Information Act (Woo in Dutch) [12] documents [9]. This part consists of 4K decision letters (17K pages) coming from Dutch ministries written in 2020–2022 by legally trained civil servants. These documents are digital born, carefully drafted and edited and hence virtually error free, and with a simple layout and markup. The machine readable text from the PDFs is extracted with *pdftotext*, part of the Xpdf suite of PDF tools.

### 3.2 Mangled OCR Detection and Repair

Detection of segments of mangled text goes as follows. We use a word list consisting of the OpenTaal list<sup>3</sup>, combined with the vocabulary extracted from the Dutch subcorpus of the ParlaMint project [7]. This corpus contains the manual transcriptions of parliamentary debates, and is of very high quality, thus also virtually error free and contains words that are roughly in the same genre as the decision letters. The combined word list contains 650K words, with 410K in the intersection of both lists, 300K exclusive to the TaalUnie list, and 130K exclusive to the parliamentary vocabulary.

<sup>3</sup> <https://github.com/OpenTaal/opentaal-wordlist>.

We define *N-mangled segments* as maximal sequences of Out-Of-Vocabulary (OOV) tokens, which may contain subsequences of In-Vocabulary words of at most length  $N$ . Mangled segments must start and end with an OOV token. We call tokens in mangled segments MTTs (short for Mangled Text Tokens). Note that an MTT can be both an Out-Of- or an In-Vocabulary token. We experimented with the value of  $N$  and found  $N = 3$  to yield the most natural “mangled segments”. In the rest of the paper this  $N = 3$  is fixed. In the sentence, “H1erb1j w1l ik u graag leten wetn dat uw verzoek is geweigerd.” the mangled segment is underlined and it contains a bold triple of 3 In-Vocabulary Dutch words. In the second step, we repair the OCR mistakes in the detected segments. We compare three strategies. As a strong baseline we OCR the complete text again with Tesseract 5.0 configured for Dutch. We detect mangled segments in both the original text and in the output of Tesseract and send these segments to ChatGPT to be corrected. We use the *gpt-3.5-turbo* model instance through the OpenAI API. We have experimented with different prompts, with the best performing one being the following:

Correct the spelling mistakes in the following Dutch text delimited by triple backticks and remove the triple backticks afterwards. Leave the correct words untouched.

We then insert the corrected segments back into the original text, and perform mangled segment detection again.

### 3.3 Document Transformation

In order to make the Woo documents more accessible, the PDFs are converted to the more accessible XML format. In fact we convert to Markdown, which is trivially convertible back and forth to XML. Using the *pdftohtml* tool<sup>4</sup>, the text, along with layout information, is transformed from PDF into XML, preserving headings, paragraphs and reading order. In turn the XML is then converted into Markdown format. The converter primarily focuses on the position, font size, and font style of a specific piece of text. Based on these characteristics, it determines whether it is a heading, paragraph, or emphasis. For example, by analyzing the differences in font sizes of the headings, the order of nested headings, and thus the reading order, is determined. Hyperlinks and emphasis from the PDF are effectively captured using the 'pdftohtml' library and are directly transferred. The alt text for images is generated using the LAVIS image captioning library from Salesforce<sup>5</sup>.

### 3.4 Evaluation

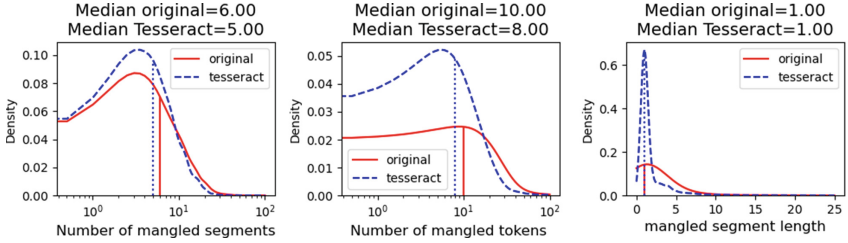
To evaluate the performance of the correction methods for the mangled segments, the mangled segments of 10 documents (totalling 227 segments and 672 words)

<sup>4</sup> <https://linux.die.net/man/1/pdftohtml>.

<sup>5</sup> <https://github.com/salesforce/LAVIS>.

were extracted and corrected with all three methods, and the Word Error Rate (WER) [11] was calculated by visually comparing the original text in the PDF with the corrected segment. We also noted whether a mistake was the result of a hallucination, or an OCR error.

To evaluate the *pdftomarkdown* conversion tool, we randomly selected 20 PDF documents from the corpus. The converted documents were then compared to the original documents based on information retention. For each document, the presence of errors in the converted file was examined for each tag.



**Fig. 1.** Comparison between the original text and Tesseract for the number and length of mangled segments, and the number of MTTs (N=17,613 pages).

## 4 Results

### 4.1 OCR Correction

Figure 1 displays the effect on the mangled segments when correcting OCR mistakes using Tesseract. The distribution of the number of mangled segments remains roughly the same, but both the number of MTTs and the length of the mangled segments decreases significantly. Roughly 70% percent of the mangled segments after correction by Tesseract were contained within mangled segments in the original text. This indicates that most of the time Tesseract shortens mangled segments, and often drastically.

Table 1 shows that all three correction strategies greatly reduce the number of MTTs in the original text as well as the number and the length of the mangled segments per page. Although Tesseract removes many

**Table 1.** Comparison of the Tesseract, ChatGPT and Tesseract+ChatGPT correction strategies, with the statistics averaged over pages.(N=17,613) ChatGPT corrections were performed on a random sample of 100 pages due to processing times.

Method	Number of MTTs	Number of Mangled Segments	Mangled Segment Length
original	41.98	9.54	4.4
original+ChatGPT	15.42	4.17	3.7
Tesseract	20.15	7.87	2.5
Tesseract+ChatGPT	14.42	5.27	2.7

OCR errors, applying ChatGPT on mangled segments remaining in Tesseract’s output further reduces the mean number of MTTs and the number of mangled segments.

The WER of the text corrected by ChatGPT, Tesseract, and first Tesseract and then ChatGPT was 11, 7 and 3%, respectively ( $N = 672$  words). Mistakes made by Tesseract are always out-of-vocabulary terms. With ChatGPT however, half of its mistakes are vocabulary terms, thus harder to spot, and sometimes leading to confusing text.

*Costs.* For our corpus of 17K pages, extracting the machine readable text from the PDF using `pdftotext` and detecting the mangled segments takes less than 2 min on a 2019 Macbook Pro with 16 GB of RAM and an 8th generation i5 CPU. Running Tesseract takes 5 h and running ChatGPT takes 50 h. The monetary expenses are very reasonable given the large increase in information quality. With an hour price of 2.5 dollar cent at Amazon<sup>6</sup>, running Tesseract costs 12.5 cents. The costs for ChatGPT based on the pricing from OpenAI<sup>7</sup> of 0.2 dollar cents per 1000 tokens then comes to 80 cents (based on sending mangled segments with in total 400K tokens).

## 4.2 Accessibility Improvement

The 20 documents used for testing the conversion to HTML contained 184 heading and 593 paragraph elements, which were correctly converted with an accuracy of 0.84 and 0.67 respectively. In the final HTML versions, the documents were roughly 156 times smaller than the original PDF file when compressed with gzip.

## 5 Conclusion

Findability and accessibility of PDF documents which have been severely damaged by redaction software can be greatly improved using simple out-of-the-box technology like Tesseract and ChatGPT. Converting the PDFs to markdown or HTML, and thereby making layout elements like headings, paragraphs and lists explicit can be done with good accuracy and as a bonus drastically reduces the size of documents. A point of attention should be given to the manner in which ChatGPT repairs OCR mistakes. Whereas mistakes made by Tesseract are easy to spot because they are most often non-existing words, ChatGPT’s mistakes are often In-Vocabulary words which on careless reading could be mistaken as correct. As we deal with official governmental documents such mistakes could be worse than “normal OCR mangling”. As an example, in our manual evaluation we found that the original term “verkregen” (obtained) was OCRred as “verkragan” which ChatGPT replaced by “verkrachten” (to rape).

<sup>6</sup> <https://aws.amazon.com/ec2/pricing/on-demand/>.

<sup>7</sup> <https://openai.com/pricing>.

**Acknowledgements.** This research was supported in part by the Netherlands Organization for Scientific Research (NWO) through the ACCESS project grant CISC.CC.016.

## References

1. Ahmed, F., Luca, E.W.D., Nürnberger, A.: Revised N-gram based automatic spelling correction tool to improve retrieval effectiveness. *Polibits* **40**, 39–48 (2009)
2. Amrhein, C., Clematide, S.: Supervised OCR error detection and correction using statistical and neural machine translation methods. *J. Lang. Technol. Comput. Linguist. (JLCL)* **33**(1), 49–76 (2018)
3. Bland, M., Iyer, A., Levchenko, K.: Story beyond the eye: glyph positions break PDF text redaction. arXiv preprint [arXiv:2206.02285](https://arxiv.org/abs/2206.02285) (2022)
4. Booth, C., Shoemaker, R., Gaizauskas, R.: A language modelling approach to quality assessment of OCR’ed historical text. In: Proceedings of the Thirteenth Language Resources and Evaluation Conference (LREC), pp. 5859–5864 (2022)
5. Data protection commission: redacting documents and records (2021). <https://www.dataprotection.ie/sites/default/files/uploads/2021-08/Redacting/%20Documents/%20and/%20Records.pdf>
6. Cuper, M.: Examining a multi layered approach for classification of OCR quality without ground truth. *Digit. Humanit. Benelux J.* **43** (2022)
7. Erjavec, T., et al.: The ParlaminT corpora of parliamentary proceedings. *Lang. Resour. Eval.* 415–448 (2022)
8. European commission: a European strategy for data. Technical Report (2020). <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020DC0066>
9. Marx, M.: Woogle dump. Technical Report, DANS (2023), <https://doi.org/10.17026/dans-zau-e3rk>
10. Marx, M., Gielissen, T.: Digital weight watching: reconstruction of scanned documents. *Int. J. Doc. Anal. Recognit. (IJ DAR)* **14**, 229–239 (2011)
11. McCowan, I.A., et al.: On the Use of Information Retrieval Measures for Speech Recognition Evaluation. Technical Report, IDIAP (2004)
12. Rijksoverheid: wet open Overheid (woo) (2023). <https://www.rijksoverheid.nl/onderwerpen/wet-open-overheid-woo>
13. Schaefer, R., Neudecker, C.: A two-step approach for automatic OCR post-correction. In: Proceedings of the 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature, pp. 52–57 (2020)
14. Strange, C., McNamara, D., Wodak, J., Wood, I.: Mining for the meanings of a murder: the impact of OCR quality on the use of digitized historical newspapers. *Digit. Hum. Q.* **8**, 16 p. (2014)
15. Traub, M.C., Samar, T., Van Ossenbruggen, J., Hardman, L.: Impact of crowd-sourcing OCR improvements on retrievability bias. In: Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, pp. 29–36 (2018)
16. Turró, M.R.: Are pdf documents accessible? *Inf. Technol. Librar.* **27**(3), 25–43 (2008). <https://doi.org/10.6017/ital.v27i3.3246>, <https://ejournals.bc.edu/index.php/ital/article/view/3246>
17. Wilkinson, M.D., et al.: The FAIR guiding principles for scientific data management and stewardship. *Sci. Data* **3**(1), 1–9 (2016)