





Detection of Redacted Text in Legal Documents

Ruben van Heusden^(✉), Aron de Ruijter, Roderick Majoor,
and Maarten Marx^(✉)

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

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

Abstract. We present a technique for automatically detecting redacted text in legal documents, using a combination of Optical Character Recognition (OCR) and morphological operations from the Computer Vision domain, allowing us to detect a wide variety of different types of redaction blocks with little to no training data. As this is a segmentation task, we evaluate our technique using the Panoptic Quality methodology, with the algorithm obtaining F1 scores of 0.79, 0.86 and 0.76 on black, colored and outlined redaction blocks respectively, and an F1 score of 0.62 for gray blocks. The total running time of the algorithm is two seconds on average measured on a thousand pages from a government supplier, with roughly 98% of this time being used by Tesseract and the conversion from PDF to PNG, and 2% by the detection algorithm. Detecting text redaction at scale thus is feasible, allowing a more or less objective measurement of this practice. The redacted text detection code and the manually labelled dataset created for evaluation is released via Github.

Keywords: Text Redaction · Image Segmentation · Panoptic Quality

1 Introduction

Redacted text is text that has been made unreadable or has been covered up. This can be due to privacy and legal reasons, or because the text reflects the opinion of an employee, or because of commercial conflicts that might arise from the publication of the data [5]. Multiple countries have *Freedom of Information* acts that require governmental bodies to release documents upon the request of civilians [7, 13]. This has resulted in multiple commercial text redaction tools in use by governments to speed up the very time-consuming manual redaction process. The form of redacted text varies, from (traditional) completely black filling to gray bars to completely white pieces, and even manual crossing out with a pen, see Fig. 1. A tool as Zylab [14] has a white with black border option, as several scandals have given the blacked out version a bad connotation and the

Github: <https://github.com/irlabamsterdam/TPDLTextRedaction>

Demo: <https://lakdetector.wooverheid.nl>.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Alonso et al. (Eds.): TPD 2023, LNCS 14241, pp. 310–316, 2023.
https://doi.org/10.1007/978-3-031-43849-3_28

white fill with black borders is more 'social-media friendly' and less aggressive-looking.

The task of automatic recognition of redacted text has not yet been described as such in the literature. It is briefly mentioned by Bland et al. [2], as part of their algorithm to de-redact text from legal documents. Their detection step is based on the location of characters in a PDF document, but this approach does not work on scanned-in documents, where this information is not present.

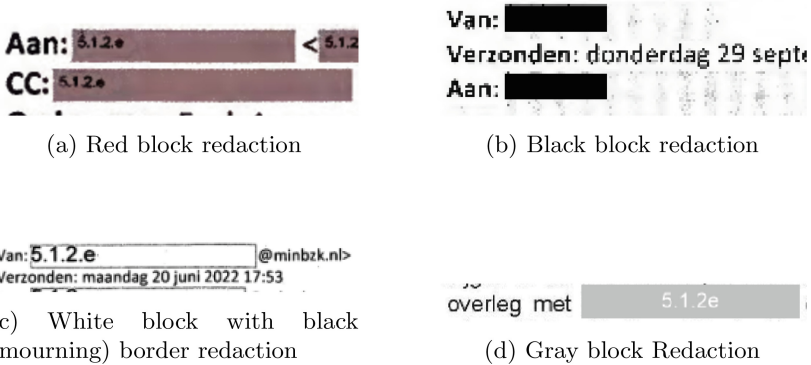


Fig. 1. Examples of the four most common types of text redaction blocks. Codes like 5.1.2.e inserted in the redacted regions indicate the legal article used to redact the particular piece of text.

The main problem of redacted text detection is to ensure high recall, while avoiding picking up logo's, images, layout structures and other noise from the page [2]. This problem is complicated by the fact that there is a large variety of different types of text redaction, as described above. The task can be seen as the complement of document-image segmentation, in which the goal is to detect the text present in an image of a document [3].

We present an unsupervised approach for detecting redacted text blocks by using Tesseract to detect and remove regular/unredacted text from the image, after which we use morphological operations to remove text missed by Tesseract, similar to the approach used by Bloomberg [3]. Finally we filter out shapes that are too big or too small, or that are taller than they are wide, reducing the amount of false positives. This approach needs no external training data (except for Tesseract, which has been trained on large amounts of text), and is not limited to detecting black or gray blocks, but can also detect blocks with outlines, and colored blocks.

By being able to automatically detect redacted text with a high confidence we can gather statistics on how much text has been redacted in a corpus of released documents, e.g., the distribution of the redacted character ratio per page. These then can be compared across different governmental bodies.

Our algorithm performs well on black, colored and outlined redaction blocks, with F1 (PQ’s recognition quality) scores of 0.79 and 0.86 and 0.76 respectively. Recall for gray blocks is hard, resulting in a somewhat lower F1 of 0.69. The overall F1 measured on 1.530 items is 0.77.

2 Related Work

Bloomberg [3] presents a method for segmenting an image into text and non-text pieces, using the morphological operations *erosion* and *dilation*, which respectively add and remove pixels from the boundaries of objects. These operations are useful, as they can be used to remove noise from an image, by first eroding the image to remove noise pixels, and then dilating the image to re-add the edges. Improvements to the technique of Bloomberg were made by Bukhari et al. to allow for the detection of drawings and graphs, instead of only halftone images [4]. The current state-of-the-art uses Transformer based models in combination with CNNs to segment images [1]. The most recent techniques make use of large quantities of training data, whereas our method is rule-based and requires no training data (except for Tesseract, as the most recent model is an LSTM trained on large amounts of textual data). This task is somewhat the complement of our task: the detection of visible text and non text versus the detection of hidden text, while also having to filter out other non-text elements such as figures and logos.

Redacted text detection is used by Bland et al. [2] as part of their method for breaking text-redaction schemes, where they develop the X-Ray Tool¹ for detecting improper text redaction. Their detection method relies on information on the location of text within a document, and detecting the existence of multiple spaces between characters. If these spaces are coloured then it is assumed that a block of redacted text is present. One of the major downsides of this approach is that it relies on knowing the position of characters on the page, something that is not present in scanned documents. At least for the Dutch redacted text landscape, scanning and then again OCR-ing documents is the predominant technique used by text-redaction tools. We thus need a method that works on scanned documents.

3 Method

3.1 Dataset

Our manually labelled dataset consists of 170 pages with 1.530 redacted text blocks from decision letters originating from Dutch ministries written in 2020–2022, originally published at <https://open.overheid.nl> and now available as a curated dataset at the Dutch scientific Data repository DANS [11]. The set is split into the 4 redaction types from Fig. 1. The support column in Table 1 specifies the number of examples per type. Redacted regions were annotated using

¹ <https://free.law/projects/x-ray>.

VGG Image Annotator [6] by two annotaters, each specializing in a redacting type, with no data being annotated twice.

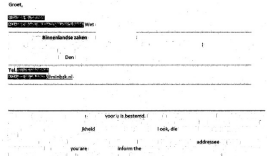
If there were humanly visible gaps between redaction blocks these were annotated as separate blocks. Horizontally touching blocks on separate lines were annotated as one block when the touching region was longer than the non touching one.

3.2 Algorithm for Detecting Redacted Text

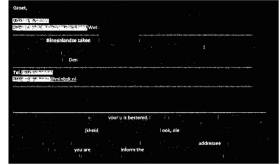
Figure 2 shows the main steps of the algorithm with the output for each step. The algorithm consists of 5 main steps, namely text detection, text removal, image thresholding, contour detection, and a final contour filtering step to remove False Positives usually coming from images and logos. For the precise details of the algorithm together with examples of output for each step, we refer to the Github repository. We now briefly describe the five steps.



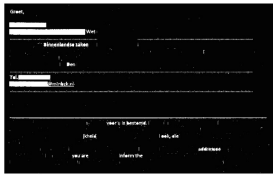
(a) Text Detection with Tesseract



(b) Text Removal



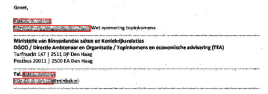
(c) Image Thresholding



(d) Contour Detection



(e) Contour Filtering



(f) Output of the algorithm

Fig. 2. The steps in the redacted block detection algorithm shown on an example with a gray redacted text containing the valediction of a letter (name, function, phone and email have been redacted).

Preprocessing. In preparation for the text detection by Tesseract, three preprocessing steps are applied to increase the quality of the image for Optical Character Recognition (OCR), following Patil et al. [12]. First, the image is converted to grayscale, after which dilation and erosion are applied to remove noise. Finally, bilateral blur with a 5 by 5 kernel is used to further remove noise while maintaining sharp edges, as proposed by Kumar [10].

For the preparation of the image used for the contour detection, erosion and dilation are applied to connect text redaction areas that are only separated by a few pixels, but that should be considered one redacted block. We apply erosion and dilation with a horizontal 1×3 and a vertical 3×1 kernel. We opted for this small kernel size as it allows us to connect lines of borders of redacted blocks while keeping the rest of the page mostly un-distorted. A larger kernel would connect edges of more bounding boxes, but at the price of more false positives. After the dilation and erosion, a bilateral blur with a kernel size of 5 by 5 is used to remove noise from the image.

Text Detection and Removal. Tesseract (Version 5) [8] is run to obtain bounding boxes of the text on a page. Because the documents are primarily in Dutch, both the Dutch and English language files are used with Tesseract. Using the confidence scores returned by Tesseract, all text with a confidence score of 65 or higher is removed, to avoid accidentally removing redaction blocks. The text is removed by filling the detected text contours with white, as can be seen in Fig. 2b. A downside of this approach is that there can still be words left in the text after this, which is why image thresholding is performed to remove this text.

Image Thresholding. To remove the text missed by Tesseract, we use the approach from Bukhari et al. [4] based on morphological operations. We threshold the image, to only keep parts that are filled significantly, where we opted for Otsu binarization in favor of a fixed threshold. We then apply another pass of erosion and dilation with a 5×5 kernel, which removes text not detected by Tesseract.

Contour Detection and Filtering. Here we use OpenCV to detect the coordinates of the remaining blocks in the image. We remove rectangles that are taller than they are wide (such as images, logos etc.), and also put a minimum on the size of the rectangle, (0.025% of the page size). The output is thus a list of bounding box coordinates that contain redacted text. Using these bounding boxes we estimate the total number of characters that have been redacted by using heuristics based on the used font size, and the total portion of the page that has been redacted in terms of characters.

3.3 Evaluation Metrics

As text redaction is a segmentation task, we use the *Panoptic Quality (PQ)* metric [9] for evaluation. In this approach, a pair of gold standard and predicted redaction block is considered a True Positive if their IoU, measured in pixels, is strictly larger than .5 (i.e., the overlapping region is strictly larger than the concatenation of the (in our case usually two) non overlapping regions). The Segmentation Quality (SQ) then is the mean IoU over the True Positives; the

Recognition Quality (RQ) is the F1 score and PQ is simply RQ weighted by SQ. We aggregated the scores over all pages, in essence viewing the entire dataset as one large image. We compute precision and recall using the same set of true positives.

4 Results

Table 1 contains the evaluation of our algorithm, grouped by type of redaction. There is little variation in the segmentation quality SQ and it is high. Thus if the overlap is large enough ($IoU > .5$), segmentation goes very well. The recognition quality RQ or F1 score on the other hand does vary a lot, from .86 for colored blocks to .69 for (light)gray blocks. The precision and recall columns somewhat explain these scores: the high F1 score for colored blocks is due to a strong recall with a good precision; for the other 3 types, recall is much lower, and also lower than precision. The algorithm has the most difficulty picking up the (light)gray blocks.

Looking into these errors we found that the False Negatives are mostly due because the legal codes (like 5.1.2.e) within the boundary boxes get recognized as text, which causes part of the redaction block to be removed from the image. The thickness of the border also plays a part, as in some cases the line is too thin and gets removed by the contour filtering step. A similar explanation holds for the gray colored

Table 1. Panoptic quality metrics for our redacted text detection method grouped by the type of redaction. The support column contains the number of redacted text segments used in the evaluation.

	SQ	RQ/F1	PQ	Precision	Recall	Support
color	0.89	0.86	0.77	0.84	0.89	247
black	0.92	0.79	0.73	0.85	0.75	371
border	0.93	0.76	0.71	0.85	0.69	264
gray	0.90	0.69	0.62	0.77	0.63	468
All	0.91	0.77	0.70	0.82	0.72	1.530

redacted text blocks: often the text within the blocks gets recognized, and the entire block gets removed by the text removal step. If we changed the thresholding step after the text removal to a fixed threshold instead of the Otsu variation, the results improved for the gray type, but decreased slightly for the other types. On a 2019 Macbook Pro with 16 GB of RAM and an 8th generation i5 CPU our algorithm takes just over 2s per page on average. Of this, 88% is used by Tesseract, 10% by the PDF to PNG conversion, and just 2% by our detection and pre- and post-processing.

5 Conclusion

We presented an algorithm for automatically detecting a wide range of different redaction types using Tesseract and simple morphological operations. We evaluated the algorithm using the Panoptic Quality method and found that the algorithm performs best on redaction blocks that are black, colored or have a mourning border, and that it does not perform well on blocks that are (light)gray.

As a possible improvement of the algorithm, a pre-classification can be done on the type of redaction block (or this information might already be present, given that some suppliers use one type exclusively), after which the algorithm can be fine-tuned for a specific class, by changing for example the threshold parameters in the contour detection step.

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. Biswas, S., Banerjee, A., Lladós, J., Pal, U.: DocSegTr: an instance-level end-to-end document image segmentation transformer. arXiv preprint [arXiv:2201.11438](https://arxiv.org/abs/2201.11438) (2022)
2. 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)
3. Bloomberg, D.S.: Multiresolution morphological approach to document image analysis. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Saint-Malo, France (1991)
4. Bukhari, S.S., Shafait, F., Breuel, T.M.: Improved document image segmentation algorithm using multiresolution morphology. In: Document Recognition and Retrieval XVIII, vol. 7874, pp. 109–116. SPIE (2011)
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. Dutta, A., Zisserman, A.: The via annotation software for images, audio and video. In: Proceedings of the 27th ACM International Conference on Multimedia (ICM), pp. 2276–2279 (2019)
7. United States Government: Freedom of information act (2023). <https://www.foia.gov>
8. Kay, A.: Tesseract: an open-source optical character recognition engine. *Linux J.* **2007**(159), 2 (2007)
9. Kirillov, A., He, K., Girshick, R., Rother, C., Dollár, P.: Panoptic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9404–9413 (2019)
10. Kumar, B.S.: Image denoising based on Gaussian/Bilateral filter and its method noise thresholding. *Sig. Image Video Process.* **7**(6), 1159–1172 (2013)
11. Marx, M.: Woogle dump. Technical report, DANS (2023). <https://doi.org/10.17026/dans-zau-e3rk>
12. Patil, S., et al.: Enhancing optical character recognition on images with mixed text using semantic segmentation. *J. Sens. Actuator Netw.* **11**(4), 63 (2022)
13. Rijksoverheid: Wet Open Overheid (woo) (2023). <https://www.rijksoverheid.nl/onderwerpen/wet-open-overheid-woo>
14. Zylab: The Zylab ediscovery Platform (2023). <https://www.zylab.com>