International Journal on Document Analysis and Recognition (IJDAR) (2020) 23:279–295
https://doi.org/10.1007/s10032-020-00359-9

ORIGINAL PAPER

Optical character recognition with neural networks and post-correction with finite state methods

Senka Drobac[1] · Krister Lindén[1]

Received: 12 December 2019 / Revised: 9 June 2020 / Accepted: 4 August 2020 / Published online: 20 August 2020

Abstract
The optical character recognition (OCR) quality of the historical part of the Finnish newspaper and journal corpus is rather low for reliable search and scientific research on the OCRed data. The estimated character error rate (CER) of the corpus, achieved with commercial software, is between 8 and 13%. There have been earlier attempts to train high-quality OCR models with open-source software, like Ocropy (https://github.com/tmbdev/ocropy) and Tesseract (https://github.com/tesseract-ocr/tesseract), but so far, none of the methods have managed to successfully train a mixed model that recognizes all of the data in the corpus, which would be essential for an efficient re-OCRing of the corpus. The difficulty lies in the fact that the corpus is printed in the two main languages of Finland (Finnish and Swedish) and in two font families (Blackletter and Antiqua). In this paper, we explore the training of a variety of OCR models with deep neural networks (DNN). First, we find an optimal DNN for our data and, with additional training data, successfully train high-quality mixed-language models. Furthermore, we revisit the effect of confidence voting on the OCR results with different model combinations. Finally, we perform post-correction on the new OCR results and perform error analysis. The results show a significant boost in accuracy, resulting in 1.7% CER on the Finnish and 2.7% CER on the Swedish test set. The greatest accomplishment of the study is the successful training of one mixed language model for the entire corpus and finding a voting setup that further improves the results.

Keywords OCR · Historical periodicals · Finnish · Swedish

1 Introduction

The OCR of historical newspapers published in Finland 1771–1929 is of unsatisfactory quality. The entire corpus[1] has been recognized with ABBYY FineReader 11 and presents a character error rate between 8 and 13%. This error rate is rather high for meaningful and reliable scientific research on this data set, so there is a need to re-OCR the entire corpus.

OCRing the corpus is difficult because it contains very diverse data written in a non-standard language. Newspapers in Finland from the eighteenth to the early twentieth century were printed in the two main languages of Finland (Finnish and Swedish) using two font families (Blackletter and Antiqua) with a large variety of fonts. Also, the data are not evenly distributed. In earlier data, there is more material printed in Swedish with Blackletter fonts, whereas the later data is mostly printed in Finnish with Antiqua fonts. However, there are periods when both languages and both font families were used extensively, sometimes even on the same pages. The standardization of the Finnish literary language began in the early nineteenth century [16]; hence, a large part of the corpus that we are working on contains spellings from different Finnish dialects.

Due to this diversity of fonts and languages, it is an interesting research problem to try to train a model that is capable of adequately recognizing all segments of the data set. It is also important to note that the corpus is very large, with almost 5 billion tokens, so it is important to find a time-efficient method for the re-OCRing. Potential multiple processing steps like separate language and font recognition are time-consuming and less desirable. In addition, each separate step tends to introduce errors on its own, so a single-step process is preferred if feasible.

[1] https://digi.kansalliskirjasto.fi.

✉ Senka Drobac
senka.drobac@helsinki.fi

Krister Lindén
krister.linden@helsinki.fi

[1] University of Helsinki, Helsinki, Finland

In previous work on this data set, Koistinen et al. [17,20,21] trained models to recognize Finnish Blackletter pages with the open-source tool Tesseract, but they focused only on the material printed in Finnish Blackletter. On the other hand, in [5,6], we used the open-source tool Ocropy to train models on both Finnish and Swedish data and both the Blackletter and Antiqua font families. However, probably due to the limitations of the shallow one-dimensional Long short-term memory (LSTM) neural network used by Ocropy, we did not manage to train a single high-quality mixed model, but obtain better accuracy for Swedish with a monolingual language model.

In this paper, we use the Calamari[2] training tool, which supports training models with custom-made deep neural network (DNN) specifications. Using a DNN improves the accuracy compared with a shallow network. Also, the DNN is able to benefit from additional training data. With a tailored DNN and additional training data for Swedish and Finnish in Antiqua, we managed to train a mixed model for the entire corpus that does better than or as well as monolingual language models.

It is an important milestone to have trained a single mixed model for the entire corpus, as it drastically simplifies the re-OCRing process. Using DNNs and additional training data also results in a significant performance improvement. In comparison to the results from [6], the character error rates go down 3.3% for Finnish, and 4.4% for Swedish (or over 60% of relative improvement), which are statistically significant gains. Furthermore, using the Calamari tool brings some additional benefits, such as the ability to run the recognition on a graphics processing unit (GPU), which is faster than the central processing unit (CPU) recognition with Ocropy. We also run experiments with voting between several OCR models to assess which model combination is most beneficial. Finally, we test the post-processing method used in [5] to find whether it is still successful on the new, improved OCR results.

Contributions As the main contribution of the paper, we demonstrate that it is possible to create a single model that recognizes two non-related languages and two different font families, where each font family consists of many fonts. Furthermore, we demonstrate an improvement in the results using a mixed model, which seems to indicate that for OCR the language model on the word level and above is less important, whereas having sufficient representation of individual characters and their variants is crucial, with a small benefit to be gained from *n*-grams of characters in the immediate context.

[2] https://github.com/Calamari-OCR/calamari.

The new data sets and the trained Calamari models (mixed models, Finnish model and Swedish model) are publicly available at: https://github.com/sdrobac/ijdar-2020.

2 Related work

The demand for OCR of historical documents has initiated large-scale projects whose aim is to make OCR easily accessible to non-technical scholars. They usually create an easy-to-use graphical interface with a semi-automatic complete OCR workflow, from image optimization and page layout analysis to automatic post-correction. For example, READ (Recognition and Enrichment of Archival Documents) is an e-Infrastructure project funded by the European Commission with a focus on making archival material more accessible. They have developed Transkribus,[3] a service platform for OCR of both printed and handwritten material.

In Germany, there is the OCR-D[4] coordination project, with 8 project modules focused on various stages of OCR. Also in Germany, OCR4all [28] has recently published an open-source tool providing a (semi-)automatic OCR Workflow for historical prints. The workflow was created using different open-source tools.

However, due to specific fonts used in different areas, it is sometimes difficult to use pre-trained models offered by other initiatives, but there are existing open-source software packages that allow custom training and optimization, as well as recognition. The most popular open-source tools are Ocropy, Kraken,[5] Tesseract and Calamari. While Ocropy and Kraken train a one-level LSTM, the new versions of Tesseract and Calamari train OCR models using Deep Neural Networks.

Ocropy has been widely used for years. Springmann et al. [36] apply different OCR methods to historical prints of Latin text and obtain high accuracy with Ocropy. Some work on Blackletter fonts has been reported in [3], where models were trained on artificial training data and achieved high accuracy on scanned books with Blackletter text. Shafait [33], along with an overview of different OCR methods, presents the architecture of Ocropy and explains the different steps of a typical OCR process. Springmann and Lüdeling [35] use Ocropy to recognize scanned images of books printed between 1487 and 1870 and report character error rates lower than 10%.

Calamari was created in 2018 and first published in [39], which presented the Calamari-OCR software for training and recognition that allows users to set up their own DNN structure including convolutional neural networks (CNNs) and LSTMs. Convolutional neural networks have also been used

[3] https://read.transkribus.eu.

[4] http://www.ocr-d.de/eng.

[5] http://kraken.re/.

in recognition of handwritten texts. For example, Guha et al. [13] reports success in training deep CNNs to extract features in recognition of handwritten Devanagari characters.

The Calamari tool can be used as a replacement for Ocropy training and recognition tools, and besides DNNs, it offers other important features in comparison to Ocropy. For example, models can be trained and text recognized on a GPU, which dramatically improves performance. Additional features such as early stopping, cross-fold voting, and pre-training have been implemented. All these features lead to lower error rates.

In [40], Calamari performance is tested in comparison to Ocropy on historical books, demonstrating that a combination of a convolutional and an LSTM network performs better than a single layer LSTM. Additionally, they experiment with a voting mechanism and data augmentation and find data augmentation to be beneficial in cases when there is only a small amount of training data available (< 3000 lines on average). It is found that, for recognizing a book, a combination of DNN, voting, data augmentation and pre-trained models requires only 60 lines of ground truth (GT) data to achieve an error rate below 2%.

The same paper also checks the performance of training and recognition. Training Calamari's deep network is faster than training Ocropy when several CPU cores are used(>4). However, training on a GPU is at least 4 times faster. In the prediction phase, Calamari is faster than Ocropy by a factor of 3 even with a single CPU core, and about 30 times faster on a GPU. However, when using voting, the prediction time for a single line needs to be multiplied by the number of prediction models.

Reul et al. [29] train mixed Blackletter models with various data sources and then test the models on nineteenth century unseen data from books, journals, and a dictionary. They compare results from Tesseract, Ocropy and ABBYY Finereader and Calamari to find that Calamari outperforms the other tools in most cases. They also find that training on real data gives better results than training on artificial data.

There have been several attempts to recognize historical prints published in Finland. In our earlier work [5], we used Ocropy to recognize historical newspapers and journals published in Finland and we achieved character error rates between 4.79 and 7%. In further work [6], we experimented on an optimal training data set size for the same data set. We also trained on Swedish lines and unsuccessfully tried to get a mixed model to recognize everything, achieving better accuracy for Swedish with a monolingual model. However, we found that adding Swedish data to the model reduced the Ocropy character error rate for Finnish (-0.5%).

Another group has also been trying to OCR data from the same corpus using Tesseract. In [20,21], they describe their OCR process. In [17], they create the ground truth data on Finnish Blackletter text (from the time period 1771–1910), perform recognition with Tesseract and report word error rates (WER) between 13 and 14.6%. In [18], they report improvements in both accuracy and efficiency. The new method consists of a combination of five different image pre-processing techniques, a new Finnish Blackletter model for Tesseract 3.04.01 with word candidate selection (voting). The voting is based on character-level rules information as well as the use of morphological analyzers and a language model. They report CER 2.36% and WER 5.51%.

2.1 Post-correction

There are many different approaches to OCR post-processing. Some of the basic methods look at post-correction as a sequence to sequence spelling correction task (e.g., [8,24,34]). In this work, we use a sequence to sequence method that can be described as an unstructured classifier and is proposed in [34]. This is a relatively simple method from both a theoretical and a computational point of view without a language model or a segmentation model. It can use a lexicon to determine OCR errors, but can also work without a lexicon.

However, there are also more specialized methods that use properties of the OCR results as features. All of the methods have at least two steps:

1. Generation of correction candidates
2. Decision making to accept proposed corrections

The implementation and methodology behind each step vary from approach to approach.

Certain methods use a lexicon to check if the suggested words are valid words of the language, but historical lexicons are not always available or are of poor coverage. An example of this approach is [19], in which Arabic data with poor OCR (WER 30% on the document level) is corrected and a 9% WER improvement reported. They use a lexicon to select incorrect words, for which they create correction candidates with a regression model trained on a confusion matrix created from a transcribed set of images. In the second stage, the right correction is selected again using a regression model, but this time based on word features obtained from a language model built from a large, publicly available text data set.

Similarly, but with historical data, Généreux et al. [11] perform post-correction of newspapers printed in German Fraktur between 1910 and 1920. They use a lexicon to find correction candidates and an external corpus together with edit distance and n-gram frequencies to score the candidates and find the winner. They conclude that their model works well when the correct result is not more than two edits from the candidate and when they have "a perfect dictionary." Furthermore, Evershed and Fitch [10] post-correct Australian historical newspapers with a high error rate (over 20% WER).

They also use an external representative text corpus to create a language model by extracting and using *n*-gram frequency lists. Their error model uses statistically weighted multiple character edits with the addition of a "reverse OCR" procedure. The "reverse OCR" is a procedure for getting the confusion cost by generating low-resolution glyph bitmaps for each word pair, with a generic font. The bitmaps are then overlaid in memory and adjusted to find the best alignment, from which they calculate a bit correlation measure. This measure helps improve the final confusion cost slightly.

Manual post-correction of OCR text is perceived to achieve high-quality results, but it is time-consuming, expensive and hard work. Human correctors need to have a good knowledge of the text domain and be trained to use correction tools. Also, humans are bad at recognizing similar character confusions. There are semiautomatic approaches that make manual correction easier and more effective. In the approach presented in [32], human annotators use an interactive interface to correct Handwritten Text Recognition (HTR) results. The user's input is considered in real time and applied to provide better automated suggestions. Additionally, an ergonomic and user-friendly touchscreen interface allows observation of the annotator's pointer actions, which is used to improve annotation performance.

Another tool PoCoTo [38] is a state-of-the-art web-based tool for semiautomated correction of OCR results on historical texts. The tool was first developed as a desktop application in the EU project IMPACT and was later adapted for web use. It uses an automated profiling mechanism that calculates which words in an OCR document are probably errors using a combination of statistical and lexical methods. The user interface shows page snippets together with OCR text containing possible errors and also offers possible corrections. Thus it makes it easier for users to correct errors manually.

Recently, a new, improved and fully automated version A-PoToCo [9] has been published. It uses a multi-input OCR approach together with a logistic regression model to rank post-correction candidates. To obtain multiple OCR inputs, different engines are used, or different models created by the same engine, to OCR the same historical document multiple times. To decide whether to accept the highest-ranked correction candidate, a logistic regression model is trained, taking the confidence of the highest-ranked candidate and the difference to the confidence of the second-highest ranked candidates as features.

The methods are tested on two data sets, one from 1557, and the other from 1841, both with quite low starting accuracy (35% and 23% WER, respectively, on tokens with more than 3 characters). The authors conclude that automated post-correction on very old data with large spelling variation is difficult, although they obtain better results on the older data set than the modern one (−4,8% vs. −3,06% WER). They are planning to implement an interactive interface for humans to make the decisions on correction candidates in the next version of the tool, called A-I-PoToCo.

In another multi-input OCR approach by [25], five binarized versions of the same grayscale image are created using different binarization thresholds (a value between 0 and 255 that determines whether each pixel will be assigned to a white or a black pixel group depending on whether it is below or above the threshold value). Once all the candidates are obtained, a hypothesis lattice is created and a single word selected using a supervised discriminative machine learning tool [26]. Similarly, Dong and Smith [4] create an unsupervised framework for OCR error correction for both single-input and multi-input correction tasks. They focus specifically on the data with several OCR versions which they align to create parallel OCR data. Then they train a uniform error model on parallel OCR data with sequence-to-sequence methods with attention. The optimal post-correction is then chosen via bootstrapping.

A completely different approach is presented in [14], where they use the property that OCR errors of the same word due to semantic similarity are grouped in vector space. They train a Word2Vec [27] model on the entire newspaper collection to obtain clusters of the OCR errors together with the real synonyms. Using a dictionary, they check which of the words are correctly spelled and then group the correct words with all misspelled variants using the Levenshtein edit distance. They use this parallel data set to train a character-level neural machine translation (NMT) model with which they perform a character-level translation of the erroneous words.

A similar clustering approach was presented in [31] and further in [30]. They created a post-correction Text-Induced Corpus Clean-up tool TICCL that also clusters all similar words in the corpus, but instead of being grouped semantically, they are grouped by character distances in Euclidean space, calculated and scored by an anagram hashing method. They use external data (lexicon, OCR confusions, and morphological rules) to distinguish between correct words and correction candidates and to adjust the scores. Finally, they perform the correction by selecting the correction candidate with the highest score.

3 OCR process

The OCR process (see Fig. 1) usually begins with pre-processing of the image files to make the images more uniform. Commonly, pre-processing includes image de-skewing, normalization, and binarization, which transforms each image pixel into a black or white pixel, resulting in a black and white image.

Binarization reduces noise in the images, usually making the text clear (Fig. 2). However, with images of poor quality,

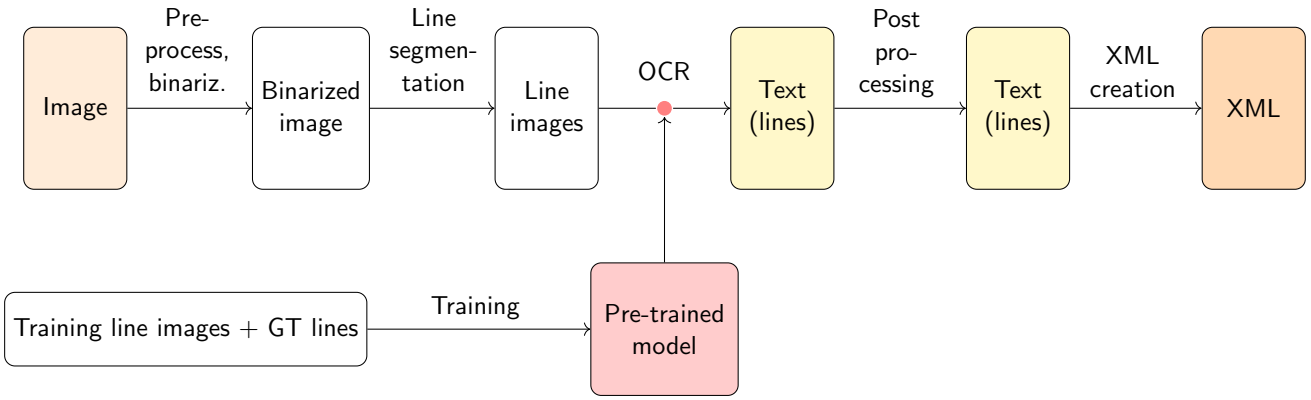


Fig. 1 OCR process

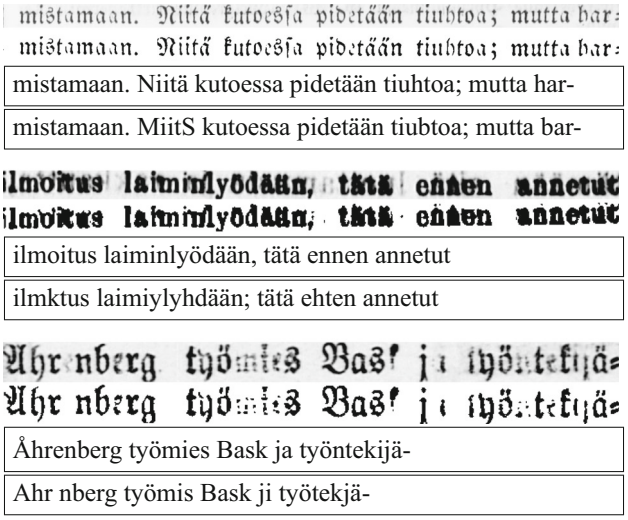


Fig. 2 Three examples of original images versus binarized images, GT and OCR texts

binarization may sometimes remove useful information. Furthermore, a binarized OCR model requires binarized images in the recognition phase, which has a performance impact on mass recognition. However, binarization is usually performed together with normalization, so the added cost is not prohibitive.

As we have deep neural networks that can handle and remember large quantities of information, it might be possible to train networks with the original images or grayscale ones, as is customary with handwritten manuscripts. However, that begs the question of whether such models also require more training data. At this point, such experiments are left for future work.

The second step in the process is image segmentation. When we think about recognizing a document, we often think of processing one page at a time. However, most of today's OCR models are trained on and can recognize only one line of text at a time. In the past, there were approaches that trained models on a glyph level, like Tesseract 3. However, it is difficult to correctly segment each glyph, so those models produce many segmentation mistakes. Additionally, creating glyph-level training data for many different fonts, especially for historical data, is very time-consuming. With those problems in mind and the ability to feed entire lines of text into neural networks, line-level segmentation is currently the state-of-the-art.

There are existing segmentation tools. For example, Ocropy has a segmentation tool in its toolset. However, all of our data had already been recognized once with ABBYY Fine reader, so we reused their segmentation. We do not have a good measure of the quality of the segmenting tool or a comparison with other tools, but from manual inspection, it seems that all the currently available tools perform approximately similarly. It is important to note that the segmentation of newspapers is challenging and it is an open research problem, so until there is significant progress in the field we will use the already existing segmentation.

When we have line images, we feed them into a recognizer for which we have trained a recognition model. As output, we obtain lines of text that can be further processed and corrected using language technology. The final result is usually written into XML files.

In this paper, we focus on training the most accurate recognition models we can, considering our data. Furthermore, we experiment with post-correction to see if we can achieve additional gains.

4 Data and tools

4.1 Data

In this paper, we work on data from a corpus of historical newspapers and journals published in Finland between 1771 and 1939. The data have been written mainly in Finnish and

Swedish. From our random sample, we can conclude that about the same amount of the Swedish text is printed in Blackletter and Antiqua, while three-quarters of the Finnish text is printed in Blackletter and only one quarter in Antiqua. We have also found that the material contains Cyrillic and Greek letters to a lesser degree.

4.2 Data preparation

In this paper, we use already available data sets as well as data sets that we created on our own. We describe the process of selecting and preparing the data for training and testing. Part of the data that we use in this paper was taken from [6]. However, due to the larger memory capacity of DNNs, we created additional training data for both Finnish and Swedish.

The existing data sets are from [6] and consist of 9500 lines of Finnish text and 6500 lines of Swedish text (418 of each set are used solely for testing). Both data sets were randomly harvested from the corpus of historical newspapers and journals. The Finnish data set is extracted from the time period 1820–1939 and the Swedish data set from 1771 to 1874.

In addition to the existing data sets, we added 5000 Swedish and 4000 Finnish Antiqua lines that were also randomly picked from the corpus from the same periods as the previous data and manually transcribed. To obtain lines from the corpus, we used ABBY's segmentation information (line coordinates), stored in METS-ALTO files. The XML files also contain some information about the languages and fonts found on each page, but we decided not to use that information for two reasons. First, we do not know how reliable the information is. Second, we do not have font and language labels on a line level. Therefore, to harvest the new data we used the language detection tool described in [15] to check if a line is written in Finnish or Swedish. The tool works very well on OCR data. We used the font-family recognition tool described in [7] to check if a line is written in Blackletter or Antiqua. This was especially useful for the automatic selection of additional Finnish Antiqua lines.

In the end, we had around 11,500 Swedish lines and 13,500 Finnish lines, both consisting of a similar percentage of Blackletter and Antiqua lines. All line images were cut from binarized, normalized and de-skewed page images.

To test our results, we perform fivefold cross-validation. The entire data set (minus 418 lines from each language set which are used just for testing purposes) is randomly divided into 5 equal-sized subsets. We use this test set for fivefold cross-validation. Since, in this case testing, is performed with a balanced data set (the font family distribution is equal in this set), we call this the *Balanced test set*.

For testing how the models behave on a specific language and font-family, we use the 418 testing lines per language from [6] that are never trained on. These lines have been manually checked, so for each line we know both the language and font-family it was written in. This data set was randomly picked from the corpus for each language, so it represents the real font-family distribution of the original corpus. We call this test set the *Fine-grained test set*. Testing on this test set allows us to directly compare our new results with the previous results reported in [6].

In Table 1, we show the statistics of the training and test sets. We present the number of lines, words, and characters, as well as the Blackletter and Antiqua ratio in each set.

Table 1 Details of Swedish and Finnish training and test sets

Dataset	# lines	# words	# chars	B:A
Training				
Swe	11,094	63,561	478,054	50%:50%
Fin	13,037	54,941	513,987	50%:50%
Testing				
Fine-grained Swe	418	2357	17,621	50%:50%
Fine-grained Fin	418	1756	16,622	75%:25%
Balanced Fin (1)	1303	5575	52,472	50%:50%
Balanced Swe (1)	1109	6368	47,791	50%:50%

B:A Blackletter:Antiqua ratio

4.3 Evaluation

For the evaluation of the results, we use two error measurements: character error rate (CER) and word error rate (WER).

The character error rate is the percentage of erroneous characters in the system output and is a common metric in OCR-related tasks. It is the number of erroneous characters divided by the sum of correct characters and errors in the system output. Similarly, the word error rate is the percentage of erroneous words in the system output. We calculate it as the number of erroneous words divided by the sum of correct characters and errors in the system output.

$$\text{CER, WER} = \frac{\text{errors}}{\text{correct} + \text{errors}} \tag{1}$$

To obtain the number of errors, we first aligned the ground truth and the OCR lines on the character level (for both CER and WER). Then we calculated the overall Levenshtein distance [22] between the system output and the ground truth including deletions and insertions.

While calculating the CER is pretty straightforward, different evaluation systems use different alignment approaches when calculating WER. For example, Fig. 3 shows two different alignments of a misspelled word *example*. The first alignment is on a character level, so it will pair the missing

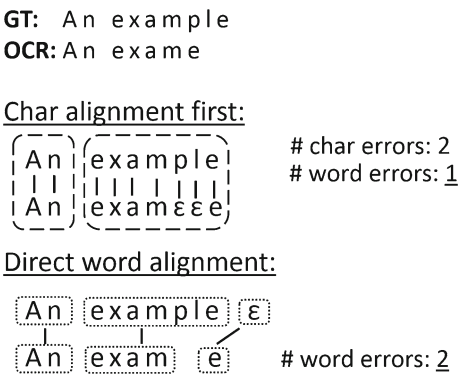


Fig. 3 An example of how the number of word errors can vary depending on the alignment. When the lines are first aligned on a character level, aligning them on the word level afterward gives a word error count of 1 (the word *example* is aligned with *exam*$\varepsilon\varepsilon e$, ε being the empty string). If we try to align words directly, we get a word error count of 2 because we recognize *exam* as one word and *e* as another

letters *p* and *l* with the empty string ε. If this alignment is then used to calculate WER, the word *example* will be paired with the whole word *exam* $\varepsilon\varepsilon$ *e* and will result in one error. On the other hand, if we immediately pair on a word-level, we will get the whole word *example* paired with *exam*, and *e* gets paired with ε, giving us two errors in total.

Since both the Finnish and the Swedish *Fine-grained test sets* (Sect. 4.2) have around 16,000 characters, we used only one decimal place for CER. Going further than that would not express any significant difference in our test sets as a 0.01% CER improvement means 1.6 characters. For similar reasons, we have left out all decimal points in the WER measure.

When comparing OCR systems on different languages with different word lengths such as, e.g., Finnish and English, with Finnish having relatively long words, character-based measures are better indicators for comparing the overall quality of the recognition results between languages. As longer words are statistically more likely to contain errors, word-based measures will seem artificially low for Finnish compared with English. The legibility of Finnish text may improve considerably with increasing character-based results without any notable change in the word-based measure. Furthermore, character-based measures make it easier to compare different OCR systems, because different alignment and evaluation calculations can make a big difference on the word level. However, one should note that a high CER and WER means that the errors are distributed among many words and this makes the text difficult to read, while a high CER and low WER makes the text easier to read. So using both character-based and word-based measures is useful.

For the fivefold testing, we calculate the mean and the sample standard deviation over all 5 test results and estimate a confidence interval of 95% (statistical significance level of 5%). In the fivefold cross-validation on the *Balanced test set*, we test each model on an unseen distinct test set and obtain 5 test results. In the fine-grained testing, we test each model on the same test set every time, also obtaining 5 test results.

4.4 Baseline

The corpus of newspapers and journals has previously been recognized with ABBYY FineReader 11. The commercial tool performed a line segmentation of each page and stored segmentation information together with OCR results in the METS-ALTO file format. On Finnish lines, the OCR character error rate is about 9–10%, while on Swedish lines it is about 8–10%.

4.5 Tools

4.5.1 Ocropy

Ocropy (previously known as OCRopus [1–3]) was for many years a leading open-source software toolkit for training OCR models. However, several of the limitations in its training and recognition tools (only a one-dimensional LSTM, no ability to use GPUs, a slow NumPy implementation) have led to the development of other training systems. However, Ocropy tools for page segmentation, pre-processing and evaluation are still used today. In this paper, we use the Ocropy binarization and normalization tool for preparing the data. We also use the Ocropy evaluation tool to calculate confusion matrices.

4.5.2 Calamari-OCR

Calamari-OCR [39] is a toolkit for training and recognition of image lines. It was created as a remedy to the previously mentioned Ocropy deficiencies, so it does not have tools for pre-processing and segmentation. Calamari supports a user-defined deep CNN-LSTM-Hybrid architecture, which has proven to increase model accuracy. They use Tensorflow as the backend, which reportedly [40] increases computation performance in comparison to Ocropy, especially while training and recognizing on a GPU. In addition to the performance improvement, Calamari offers some additional tools that are useful in certain instances (like voting and data augmentation).

4.5.3 FST post-correction

We chose to perform OCR post-correction using the unstructured classifier described by [34] and used by [5] because we wanted to see if there was any difference in the performance of the post-correction method with different OCR quality in the data set.

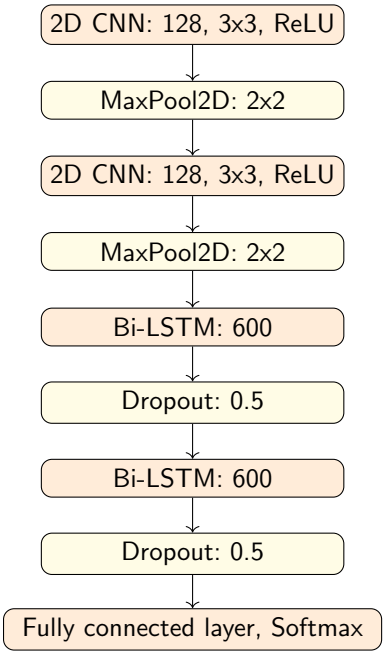


Fig. 4 The neural network consists of two pairs of convolution and pooling layers with a ReLU activation function, followed by two bidirectional LSTM layers with dropout layers after each one. Finally, the model ends with a fully connected layer and a Softmax output layer

The method corrects input tokens and creates an error model, which is a set of weighted context-sensitive parallel replacement rules implemented as a weighted finite-state transducer. After we have an error model, we use a lexical lookup to validate or discard suggestions generated by the error model. We run all finite-state operations using HFST[6] [23] tools.

5 Method

5.1 Initial experiments

Before adding new data, we performed some initial experiments from which we concluded that the Calamari results, even with the default network, were definitely better than Ocropy (with a mixed model we got 4.6% on the Swedish test set and 3.1% on the Finnish test set, compared with the best Ocropy monolingual models with only 7.1% on Swedish and 5.0% on Finnish the test set). As we added extra training data to Calamari, we needed to expand the network dimensions from the default settings in order to take full advantage of the new data.

[6] https://hfst.github.io/.

5.2 OCR: training models

To train OCR models, we chose to use Calamari because of its reported best performance results in [40]. Calamari has a default neural network:

$$\begin{aligned}&\text{cnn} = 40 : 3 \times 3, \text{pool} = 2 \times 2, \text{cnn}\\&= 60 : 3 \times 3, \text{pool} = 2 \times 2, \text{lstm} = 200, \text{dropout} = 0.5\end{aligned}$$

As initially tested in Sect. 5.1, there is a significant improvement in the results even when switching from Ocropy to the default Calamari model. However, as we added more data, we wanted to see if increasing the network size would improve the results further. Thus, through fivefold cross-validation testing with different network configurations, we found that we achieved the best results with the following neural network:

$$\begin{aligned}&\text{cnn} = 128{:}\, 3 \times 3, \text{pool} = 2 \times 2, \text{cnn}\\&= 128{:}\, 3 \times 3, \text{pool} = 2 \times 2, \text{lstm}\\&= 600, \text{dropout} = 0.5, \text{lstm} = 600, \text{dropout} = 0.5\end{aligned}$$

The same neural network with more detail is shown in Fig. 4. We used two CNN layers, the same as in the default network, but we increased the depth of the layers from 40 to 128. This was the maximum considering the GPU hardware restrictions. After each CNN layer, we have a MaxPooling layer to reduce dimensions. Therefore, in the end we have the original image reduced 4 times in size because of the two pooling operations. This dimension reduction limits our number of MaxPooling layers to prevent short lines from getting too reduced, which would result in poorer results.

After the CNN layers, we have two bidirectional LSTM layers of dimension 600, each followed by a dropout [37] with a rate of 0.5 to prevent overfitting. We tried to train models with smaller and larger LSTM dimensions, as well as different combinations of the number of layers and dimensions per layer. Empirically, we seem to get statistically similar results as long as the total number of LSTM dimensions is greater than 1200. It seems that it does not matter how the dimensions are distributed through layers. For example, the network with two LSTM layers of size 600 gives similar results as a network with three LSTM layers of size 400, with the same dropout layer in between the LSTM layers. Due to computational restrictions, we could not train with only one LSTM layer of size 1200. Therefore, this is the minimum network configuration that does not increase the error rate.

Finally, in the end there is a fully connected layer and a Softmax output layer. The loss is computed using the Connectionist Temporal Classification (CTC) algorithm described in [12].

The models were trained with early stopping after the results on the validation set had not improved after 5 steps in batches of 32 lines.

5.3 Recognition phase and voting

In the recognition phase, we use the Calamari predict tool. The main advantage of this tool is that it can be run on a GPU, which makes the recognition phase very fast.

Another feature of the tool is the ability to recognize an image with several models at the same time and then choose the optimal result with voting. Each model predicts several candidates with their accompanying probabilities and then the voting mechanism decides which candidate wins. By default, the tool uses confidence voting that adds all probabilities for each candidate. The candidate with the highest sum of probabilities wins.

Wick et al. [39] showed that voting always reduces errors, so we also tested it on our data. The downside of the method is that it requires recognition with several models, which slows down the recognition process.

5.4 Post-correction

The post-correction training process was the same as described in [5]. Error models of varying sizes were trained on the Finnish and Swedish training sets, having been OCRed by the best OCR model. We picked the optimal error model on the validation set and tested it on the *Fine-grained test set*. Since this post-correction method works only on words, the algorithm includes splitting the lines into tokens at blanks, post-correcting and then joining the strings back into lines. Thus, the algorithm cannot affect spaces.

For Finnish, we performed experiments both with and without a historical Finnish lexicon.[7] We further modified this lexicon to accept strings with leading and trailing punctuation, as punctuation often provides important clues for finding the correct substitution, which could be lost if the data was tokenized and the punctuation removed.

As we do not have a convenient Swedish historical lexicon for the time period, we only ran experiments without a lexicon on the Swedish test set.

6 Experiments and results

To evaluate the performance, we ran experiments with:

1. Mixed models (models trained on both Finnish and Swedish data);
2. monolingual models (models trained on Finnish or Swedish data);
3. Voting on different model combinations;
4. Post-correction.

6.1 Mixed models

With the first experiment, we wanted to see if we could train mixed models that would be able to recognize all the different parts of the data set.

To test the performance of mixed models, we perform five-fold cross-validation on the *Balanced test set*, which contains both Finnish and Swedish training data. Additionally, we test the same models on the *Fine-grained test set* to get a better understanding of how models behave on each language and font family. The results are shown in Table 2 and they are expressed as a five-model mean error rate with 95% confidence intervals. For every test result, we calculate both the character and the word error rates.

The mixed models achieve an average error of 2.6% CER and 10% WER on the *Balanced test set*. On the Swedish *Fine-grained test set*, they achieve 3.4% CER and 13% WER in total, with 3.5% CER and 14% WER on Blackletter lines and 3.1% CER and 13% WER on Antiqua lines. On the Finnish *Fine-grained test set* the total performance is 2.2% CER and 10% WER, with 1.9% CER and 8% WER on Blackletter lines and 3.6% CER and 14% WER on Antiqua lines.

The 95% confidence interval is very narrow for CER. On the *Fine-grained test set* it is under 0.08% in all cases except for Finnish Antiqua, where it is 0.24%. On the *Balanced test set*, it is 0.22%. For WER, the 95% confidence interval is quite uniform, between 0.2 and 0.4%.

Table 2 Test results with mixed models

Test set	CER (%)	WER (%)
Balanced	2.6 ± 0.22	10 ± 0.4
Fine-grained		
swe-test	3.4 ± 0.05	14 ± 0.3
swe-blackletter	3.5 ± 0.05	14 ± 0.2
swe-antiqua	3.1 ± 0.07	13 ± 0.4
fin-test	2.2 ± 0.08	10 ± 0.3
fin-blackletter	1.9 ± 0.08	8 ± 0.3
fin-antiqua	3.6 ± 0.24	14 ± 0.4

Results show the mean character (CER) and word error rates (WER) with 95% confidence interval calculated with fivefold cross-validation models. The first part of the table shows the results on the *Balanced test set* and the second part on the *Fine grained test set*

[7] https://github.com/jiemakel/omorfi.

6.2 Monolingual models

In the second experiment, we trained Finnish and Swedish monolingual models, so as to be able to compare the results with the results achieved with mixed models.

For monolingual models, we performed the same fivefold cross-validation on the *Balanced test set* and additional testing on the *Fine-grained test set*, with the only difference being that we performed training and testing separately for each language. Therefore, the Finnish models were trained and tested on Finnish data only, and the Swedish models on Swedish data only. The results are shown in Table 3 and also expressed as a five-model mean error rate with a 95% confidence interval for both CER and WER.

The first part of the table shows results on the *Balanced test set* for each model, as well as the combined error rate for both balanced test sets, `fin+swe`, which results in 2.8% CER and 10% WER. We calculated and included this line for easier comparison with the mixed model results, and we can see that the result is in the same range as the mixed model result, albeit with a larger error margin (0.71%). The second part of the table shows how the Swedish models performed on the Swedish part of the *Fine-grained test set* (3.1% CER and 13% WER in total) and the third part of the table shows the results of Finnish models on the Finnish part of the *Fine-grained test set* (4.1% CER and 11% WER). While the results on the Swedish test set are the same as with the mixed model, the monolingual models did significantly worse on the Finnish test set.

Table 3 Test results on monolingual models

	CER (%)	WER (%)
Test set		
swe-balanced	3.8 ± 0.37	13 ± 0.7
fin-balanced	1.7 ± 0.09	8 ± 0.5
fin + swe	2.8 ± 0.71	10 ± 1.9
Swedish models		
Fine-grained		
swe-test	3.4 ± 0.09	13 ± 0.5
swe-blackletter	3.6 ± 0.15	15 ± 0.8
swe-antiqua	3.1 ± 0.08	13 ± 0.7
Finnish models		
Fine-grained		
fin-test	4.1 ± 0.16	11 ± 0.4
fin-blackletter	3.5 ± 0.15	9 ± 0.6
fin-antiqua	5.8 ± 0.24	17 ± 1.0

The first table shows errors (CER and WER) for balanced models. Five Swedish and five Finnish models were cross-trained and tested. The first row shows mean errors with 95% confidence interval on Swedish fivefold testing, the second one on Finnish fivefold testing and the third row mean errors on their combined results (5 Finnish and 5 Swedish test results). The second and third tables shows errors on fine-grained test sets. In the second table, we tested Swedish models on Swedish test sets, and in the third table Finnish models on Finnish test sets

6.3 Voting

To test how the voting mechanism affects the results, we performed voting experiments with six different model combinations. All models used for voting were trained and used in the fivefold cross-validation experiments with mixed models, or with monolingual models. Initial results showed that voting performed worse with only three models, so we tested combinations with 5 models and one additional 7-model setup. We used the following model combinations:

1. 5 mixed models
2. 1 mixed model, 2 Finnish monolingual models, 2 Swedish monolingual models
3. 3 mixed models, 1 Finnish monolingual model, 1 Swedish monolingual model
4. 5 mixed models, 1 Finnish monolingual model, 1 Swedish monolingual model
5. 5 monolingual Finnish models
6. 5 monolingual Swedish models.

Table 4 shows the results after voting with the model combinations. The upper table shows the character error rates and

Table 4 Recognition results with voting on *Fine-grained test set*, CER (upper table) and WER (below)

Test/models	CER (%) (1)	(2)	(3)	(4)	(5)	(6)
swe-test	2.9	4.0	3.0	2.9	6.6	**2.8**
swe-blackletter	3.1	4.2	3.3	3.2	7.1	**2.9**
swe-antiqua	**2.7**	3.9	**2.7**	**2.7**	6.1	2.8
fin-test	**1.9**	2.0	2.1	**1.9**	2.2	3.1
fin-blackletter	**1.6**	1.6	1.7	**1.6**	1.7	2.8
fin-antiqua	**2.9**	3.2	3.3	3.1	3.8	4.3
	WER (%)					
swe-test	12	14	12	12	23	**11**
swe-blackletter	**12**	14	13	**12**	25	**12**
swe-antiqua	**11**	13	**11**	**11**	20	**11**
fin-test	**8**	**8**	9	**8**	9	14
fin-blackletter	**7**	**7**	8	**7**	8	13
fin-antiqua	**11**	12	12	**11**	14	17

Bold denotes the best results in a row
All models used for voting were created and used with fivefold cross-validation. Mixed models are trained on both Finnish and Swedish data. When there are less than 5 models, we chose the ones with the best results on the balanced tests. We used the following models for voting: (1) 5 mixed models (2) 1 mixed model, 2 Finnish, 2 Swedish models (3) 3 mixed models, 1 Finnish, 1 Swedish model (4) 5 mixed models, 1 Finnish, 1 Swedish model (5) 5 monolingual Finnish models (6) 5 monolingual Swedish models

Table 5 Post-correction results (CER/WER) on *Fine-grained test set*

Test set	OCR	Lexicon	No lexicon
swe-test	2.9/12	–	2.7/11
swe-blackletter	3.1/12	–	2.9/12
swe-antiqua	2.7/11	–	2.4/10
fin-test	1.9/8	1.9/8	1.7/7
fin-blackletter	1.6/7	1.6/7	1.4/6
fin-antiqua	2.9/11	2.9/11	2.9/11

Swedish error model was trained and tested on Swedish data, Finnish on Finnish data. The first column shows a test set, the second OCR accuracy, the third post-correction results with the use of the lexicon and the last column post-correction accuracy without a lexicon

the lower table the word error rates. This time we only performed one test per model combination, and the testing was performed only on the *Fine-grained test set*, because that set was not used for training any of the models. We could not use the *Balanced test set* for this experiment to avoid testing on data that had been trained on.

On the Swedish test set, we obtained the lowest error rates with setup (6), consisting only of monolingual Swedish models (2.8% CER and 11% WER). The Finnish test set obtained the best results with 5 mixed models (1.9% CER and 8% WER). The best results with voting between 5 models were better than the results with one model only.

6.4 Post-correction

The last experiment was post-correction. We wanted to see if there is something left for a separate post-processing unit to pick up on when using a multilingual OCR model.

As input for the post-correction training, we took re-OCRed text produced by voting with 5 mixed models. The post-correction error models were trained on the same training set as the OCR models with the optimal threshold chosen on the same validation set. For Swedish, the optimal threshold was 16. For Finnish with a lexicon it was 9 and without a lexicon 11.

We show the recognition results on the *Fine-grained test set*, as well as the post-correction results in Table 5. The OCR error rates with voting is 2.9% CER and 12% WER on the Swedish test set and 1.9% CER and 8% WER on the Finnish test set. However, the best results for both test sets were achieved with post-correction without lexicon (2.7% CER and 11% WER for Swedish, 1.7% CER and 7% WER for Finnish). Post-correction without lexicon managed to improve all the results except for the Finnish Antiqua, which stayed the same. Post-correction with lexicon did not make any changes.

Table 6 A confusion matrix for the *Balanced test set* and the Swedish and Finnish *Fine-grained test sets* with the best mixed model from the fivefold cross-validation

Balanced			Swe Fine-grained			Fin Fine-grained		
#	ocr	gt	#	ocr	gt	#	ocr	gt
64	__		16	ä	å	12	__	
42	u	n	10	.	,	7	ä	a
37	.	,	9	å	ä	7	t	i
32	t	l	8	s	f	6	i	t
31	å	ä	8	u	n	6	__	i
26	__	.	7	,	.	5	.	__
24	ä	å	6	t	l	5	n	u
24	n	u	6	__	,	5	l	i
22	e	c	6	.	__	5	i	l
22	s	f	5	a	ä	4	k	t

For each test set, the first column shows the frequency of the mistakes, the second column the recognition result and the third column the ground truth. Deletions are marked with "__" in the OCR column and insertions with "__" in the ground truth column. The *Balanced test set* has a total of 95,886 characters, the Swedish *Fine-grained test set* 15,850 and the Finnish 16,890 characters

7 Error analysis

In this section, we perform OCR error analysis on the character level using confusion matrices, as well as a word-level and line-level analysis. Furthermore, we analyze post-correction results on the line level.

7.1 Confusion matrices of OCR results

In Table 6, we show the confusion matrices with the 10 most frequent confusions for 3 different test sets: the *Balanced test set* from fivefold cross-validation (with CER 2.2%) and the Swedish and Finnish *Fine-grained tests sets* recognized with the same model.

The first 3 columns in the tables show confusions of the *Balanced test set*, the next 3 columns of the Swedish set and the final 3 columns of the Finnish *Fine-grained test set*. On each test set, the first column is the frequency of the mistakes, the second column the recognition result and the third column the ground truth. Deletions are marked with "__" in the OCR column and insertions with "__" in the ground truth column.

The top mistake in the *Balanced test set* is the deletion of a space, with 64 occurrences, followed by "u" and "n", "." and "," and "t" and "l" confusions. In the *Fine-grained test set*, the most common confusion for Swedish is "ä" and "å", followed by "." and ",", "å" and "ä" and "s" and "f". For Finnish, the most common error was a wrongly deleted space, followed by "ä" and "a", "t" and "i" and "i" and "t" confusions.

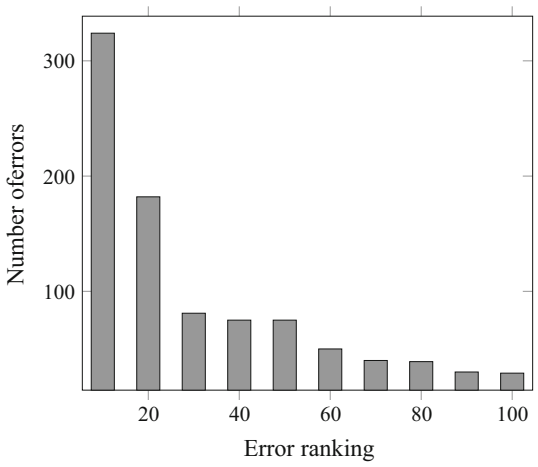


Fig. 5 The graph shows the number of errors per every 10 error-rank in the *Balanced test set*, starting with the most frequent ones. The first bar shows that the top ten mistakes make up slightly more than 300 errors, the sum of the next ten most frequent errors is shown in the second bar, etc

Confusion matrices are a good way to see which letters are confused the most. However, in our case, the most frequent ten mistakes make up only a small portion of the entire data set and correcting those few would bring only a small overall improvement. The *Balanced test set* has a total of 95,886 characters, and in the *Fine-grained test set* the Swedish part has 15,850 and Finnish 16,890 characters. Together, the top ten mistakes in the *Balanced test set* make up only 0.34% of the characters, in the Swedish *Fine-grained test set* 0.48% and in the Finnish test set 0.39%. We also calculated percentages for the next 10 mistakes (not shown in the table) and, together, they account for 0.19% of the *Balanced test set*, 0.23% of the Swedish and 0.20% of the Finnish *Fine-grained test set*.

To visualize the numbers, we created a graph shown in Fig. 5 that illustrates the cumulative number of errors for every 10 positions in the *Balanced test set* confusion table. The top ten mistakes account for slightly more than 300 errors, the next ten mistakes only half of that, then the next 10 mistakes again drop by half, etc.

The confusion table and the graph show us that although some characters are confused more often than others, most of the mistakes are widely distributed across a large and spurious variety of characters.

7.2 Word level error analysis

In Table 7, we show how many words from the *Fine-grained test set* are found also in the training sets. The Swedish test set has more words in total, but almost the same number of unique words as the Finnish one. However, words from the Swedish test set are better represented in the training set. This is not surprising since Finnish is morphologically richer than Swedish and we searched for the words in their exact form as they appear in the test set.

Table 7 Word representation of Finnish and Swedish Fine-grained test sets in the respective training sets

	FIN test	SWE test
Total words	2169	2774
Unique words	1741	1754
Found in training set	57.6%	64.6%
Incorrect words	169	323
Unique incorrect words	160	289
Found in training set	41.9%	49.8%

The upper part shows the number of total and unique words, and the percentage of the unique words found in the training set. The bottom table shows the number of incorrect words (total and unique) after recognition with a mixed model, as well as the percentage of unique incorrect words, whose correct version is present in the training set

We analyze the incorrect words in the bottom part of the table. The number of unique erroneous words is just slightly lower than the total count and the percentage of the correct version of those words found in the training set is lower than in the upper table. However, this percentage is still quite high, meaning that not only rare words have been misspelled.

7.3 Line level error analysis

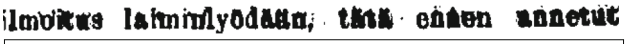
In order to further understand the nature of the mistakes, we performed an error analysis on the line level. We checked how many lines were recognized completely correctly and found that most of the lines did not have any errors (on average over 72% of the Finnish lines, and over 51% of the Swedish lines, while [5] report 37% correct lines on the Finnish test set recognized with Ocropy). In the set of incorrect lines, most of them only have some errors here and there, and those are usually common OCR confusions (as shown in Table 6). However, some lines have very high error rates so we focus on characterizing what type of lines they represent. We manually went through the lines with CER over 25% (see examples in Fig. 6) and found that the mistakes mostly belong to one of three groups:

1. *Wrong segmentation* When there is visible text from the previous or the next line, the recognizer gets completely confused and cannot recognize anything (Fig. 6a). It behaves similarly if the text is surrounded with graphics as in Fig. 6f.
2. *Poor image quality* Although most of the images are of good quality, some of the data suffers from poor quality, as shown in Fig. 6b. When preparing training and testing data, we deleted line images that we could not transcribe. However, there is a certain portion of lines

ne Fiskals-Enkan Brita Lindroths Arfwingars, under N:o 21 och 25 i Klöster-Qwarte

ffnkt sr. K2e Aersed, sl fe öt td ir t iöftö mfn

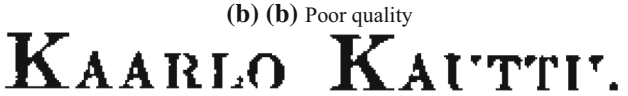
(a) (a) Wrong segmentation



ilmoitus laiminlyödään; tätä ennen annetut

ilmktus laimiylyhdään; tätä ehten annetut

(b) (b) Poor quality



KAARLO KAUTTU.

KAAnL0 KArTrr.

(c) (c) All capital letters

Suomalainen Wirallinen Lehti.

Snomalainen Dirailimen Lehl.

(d) (d) Special font

Rahaksi muutto

Kaladiusta

(e) (e) Tall narrow font

Pirilän ja Kaulan

JHin ja käsl

(f) (f) Surrounded text

Fig. 6 Example lines with lots of errors. In every subfigure, the first line shows the binarized image line, the second line the ground truth text, and the last one the recognized text. The examples show OCR models performing poorly on lines: **a** with wrong segmentation, there is visible text from the previous line, **b** of poor quality, **c** containing words with all capital letters, **d** printed in specialized font, **e** printed in tall and narrow font, **f** graphics surrounding the text

with poor image quality that we could transcribe based on the context, but the OCR models seem to struggle with those lines.

3. *Specialized fonts* Since our training data was harvested randomly from the corpus, we have a good representation of frequent fonts, but poor representation of specialized fonts that rarely occur. Therefore, it is not surprising that the model has trouble recognizing lines printed in specialized fonts as those shown in Fig. 6d, e. Furthermore, the model struggles with recognizing lines in all capital letters (Fig. 6c). Some capital letters occur more frequently than others, so they are better represented in the training data. Also, the model rarely sees all capital letter examples in the training data, so it might not recognize some capital letters surrounded by other capital letters because it has not seen them before in that context.

Table 8 The results of recognition with mixed models on the trimmed *Fine-grained test sets* after removing header and advertisement lines

Fine-grained	CER (%)	Δ	WER (%)	Δ
swe-test	3.2 ± 0.07	-0.2	13 ± 0.0	-1
swe-blackletter	3.5 ± 0.06	0.0	14 ± 0.0	0
swe-antiqua	2.9 ± 0.10	-0.2	12 ± 0.5	-1
fin-test	1.7 ± 0.06	-0.5	9 ± 0.5	-1
fin-blackletter	1.6 ± 0.06	-0.3	8 ± 0.0	0
fin-antiqua	2.2 ± 0.08	-1.4	11 ± 0.5	-3

On the Swedish data set, we removed 25 lines, 10 Blackletter (5%) and 15 Antiqua (7%). On the Finnish set, we removed 31 lines in total, 10 Blackletter (3%) and 21 Antiqua (22%). The first column in the table describes the test sets, the second the CER values, the third the difference in the CER compared with results on the original *Fine-grained test set*. The last two columns show the WER results on the altered test set and the difference from the original WER results

Going through the lines with high error rates, we realized that most of the lines printed in specialized fonts are either headings or parts of advertisements. In order to check how much of the error is caused by those lines, we prepared a trimmed version of the *Fine-grained test set*, which did not have any headings or advertisements. We identified and removed 25 lines in total as headings from the Swedish test set, of which 10 were Blackletter (5%) and 15 Antiqua (7%). From the Finnish test set, we removed 31 lines, of which 10 were Blackletter (3%) and 21 Antiqua (22%) lines.

The recognition results without those lines are shown in Table 8, as well as the difference from the results on the original *Fine-grained test set*. The results on the "trimmed" test set were better than the original results. The largest improvement comes from the Antiqua lines. On the Swedish set, the CER results improve 0.2% and the entire improvement comes from the Antiqua part. On the Finnish set, the results improve 0.5%, with the largest improvement also on the Antiqua part (1.4%). Word error rates also improve (1% on both sets).

7.4 Post-correction error analysis

As seen in Table 5, the post-correction managed to improve both the Finnish and the Swedish test data.

Table 9 The number of lines of the *Fine-grained test set* (of a total of 418 lines in each Finnish and Swedish part) that became better or worse after post-correction

	Finnish		Swedish
	Lexicon	No lexicon	No lexicon
Better	0	11	17
Worse	0	0	2

The first row shows the number of lines that improved after post-correction and the second row shows the number of lines that got worse after post-correction. The first two columns show the tests on the Finnish data set, with and without a lexicon, and the last column shows the results for the Swedish test set

```
OCR: muualla): ,,siwistyneet'' eiwät wiitsi
 PC: muualla): „siwistyneet" eiwät wiitsi
 GT: muualla): „siwistyneet" eiwät wiitsi
```

Fig. 7 A post-correction example. The first line shows the OCR output, the second one the post-correction result and the third one the ground truth. The error of the OCR line is 10.5% before correction and 0% after correction

In order to analyze the post-correction results, we checked how many lines improved and how many deteriorated with post-correction. The tests were performed on the *Fine-grained test set* and the results are shown in Table 9. We can see that the Finnish model without a lexicon corrected 11 lines and did not make any bad decisions, versus the one with a lexicon, which did not make any changes. The Swedish error model managed to correct 17 lines and made two lines worse.

Since we are dealing with such a small number of lines, we manually checked all the results. Surprisingly, almost all corrections were changes to quotation marks: two consecutive commas were changed into a left lower double quotation mark and two apostrophes were changed into one upper double quotation mark. Figure 7 shows an example of an OCR line with CER 10.5% that was post-corrected into a 100% correct line by correcting the quotation marks in two places.

The other changes were the insertions and deletions of commas. For example, both of the Swedish lines got worse after correction because one comma was wrongly deleted. In one line, that was a real mistake and the error went up from 0 to 2.2%. However, the other Swedish line that got worse had an OCR character error rate of 68.8%; therefore, deletion of one comma did not make a big difference because the OCR text was completely incomprehensible anyway.

We speculated that the reason why only punctuation gets corrected could be the fact that the error models were trained on the same data set that the OCR models were trained on. The error rates of the training set are lower than of the regular OCR test sets, so maybe the model does not have enough examples to learn which mistakes to correct. To test this, we trained the post-correction error models on the real OCR test data. For this purpose, we took test sets from the fivefold cross-validation, recognized with corresponding models, and used for post-correction training. With this approach, the post-correction performed more corrections, but also made more incorrect decisions. In the end, the error rates with this method were higher than with the original approach, because the starting OCR error rates were higher than with voting.

8 Discussion

In this section, we analyze and discuss the OCR results presented in the paper as well as the error analysis and post-correction results.

The important accomplishment is that we managed to train a mixed model that is able to recognize both Finnish and Swedish data, as well as Blackletter and Antiqua fonts. While the mixed model performs equally well as a Swedish monolingual model on Swedish data, it is interesting that on the Finnish data, we obtained significantly better results (around −2% CER) with a mixed model than with a monolingual one. In [6], we report improvements on Finnish Antiqua with a mixed model, but in our case, we achieve the same improvement on both the Blackletter and Antiqua sets.

Furthermore, our results are a lot better than earlier Ocropy results reported in [6] on the same test data set. With the new mixed model, we get −3.7% CER and −12% WER on the Swedish test set and −2.8% CER and −11% WER on the Finnish test data. With voting, the CER falls an additional 0.6% for Swedish and 0.3% for Finnish, and WER goes down 2% for both languages.

We can also compare our Finnish Blackletter results with voting (1.6% CER and 7% WER) with the Tesseract results published in [18]. They report CER 2.36% and WER 5.51% after voting enhanced with an external language model, which after rounding equals to 2.4% CER and 6% WER. In comparison, we get 0.8% better CER on OCR results, but our WER is 1% worse than what they report. After post-correction, the difference is 1% CER in our favor and WER is the same. It is interesting that with significantly higher CER, they get a lower or the same WER as we do. Since we do not test on the same data, maybe the test data they use has more short words than our test data. The other possibilities for this result could be that voting with a language model enables better decisions on a word level or that the difference in the results is a consequence of different alignment methods when calculating WER. However, to get an accurate comparison, it would be necessary to test both methods on the same test data. Despite the lower WER, the obvious drawback of their method is that their model focuses only on one language and one font-family. Even if they acquire training data for Swedish and Antiqua, they would have the challenge of incorporating voting with two language mod-

els. We should keep in mind that they also used additional advanced image processing before the recognition.

All our mixed model results show that, despite the similar sizes of the Finnish and Swedish training sets, the results on the Swedish test set are always lower than on the Finnish test set (about 1% CER and 3% WER). Furthermore, the Swedish Blackletter results are always worse than the Swedish Antiqua, and conversely, the Finnish Antiqua results are always worse than the Finnish Blackletter. In [6], the poor Finnish Antiqua results were speculated to be due to a small number of Finnish Antiqua training lines. Now we have the same number of training lines for both Blackletter and Antiqua, but Finnish Antiqua results are still lagging behind. A new hypothesis is that there is more variety in the Swedish Blackletter fonts as they were initially hand-crafted, whereas there is more variety in the Finnish Antiqua fonts as technology progressed and advertising increased with the advent of industrialization, advertising and fancy fonts. Both of these should be remedied with additional training data.

When comparing mixed and monolingual model results, it is interesting that the Swedish test set results are the same in both cases, but the Finnish monolingual model performs a lot worse than the mixed model on the Finnish test set. In [6], Finish Antiqua benefited from being combined with Swedish in the mixed model, but now both Blackletter and Antiqua get a large performance boost from the mixed model. It seems that the added variety is still more important than the language context, indicating that the model could still benefit from more Finnish training data.

It is known that voting can improve OCR results (as reported in [39]). It is not surprising that the results on our models also improved with voting. However, it is interesting that having similar models voting (5 mixed models or 5 monolingual models) outperformed models trained on different segments of the data set (combinations of monolingual and mixed models). This contradicts what has been reported in [39]: "The benefit of voting depends highly on the variance of the individual voters. If the voters predict very similar results, errors are less probable of being removed, than if more diverse models are used." That is not the case in our test, where we achieve the best voting results with similar models. Perhaps there is a limit to how diverse the models should be or how variety is accumulated in the models. On the other hand, the obvious downside of voting is the recognition speed. Voting with 5 models means slowing down the recognition by a factor of 5. However, the recognition is performed on a GPU, which is very fast, so this might not be a real issue for the re-OCR process.

The error analysis on the character level showed that errors are widely distributed. The top 10 errors make up only 0.3–0.5% of the total character count, and 13–17% of the remaining error rate. Correcting them would require a substantial effort, but would yield only a small overall improvement.

The analysis on the line level showed that most of the lines are correct. However, there are lines with a high error rate, and to improve general accuracy, it would be useful to focus on identifying what kind of lines they are and find a way to correct them. For example, we identified that a certain number of lines with high error rates are printed in specialized fonts, and these are usually headings or advertisements.

While random sampling of lines across the corpus proved to be effective for training a general model, to further increase the recognition rate of specialized fonts, we need a more focused approach. We could still randomly sample the lines from the corpus, but it would be useful to use an automatic font classifier that would recognize lines printed in specialized fonts. The font-family classifier described in [7] could possibly be trained for this purpose. This would allow us to collect Cyrillic and Greek data, which we have so far ignored.

Finally, we discuss post-correction. Interestingly, despite the large error reduction in the OCR results, the same basic sequence to sequence post-correction method manages (without a lexicon) to consistently reduce error rates. While we do not know what kind of error corrections the post-correction performed in [5], it is interesting that in this case it focused on correcting punctuation. We could argue that this is mainly because the error model was trained on the same set as was OCRed, and because of the initial low error rates, it does not frequently learn to correct other character confusions. However, training on the real test set did not improve overall accuracy and did introduce more incorrect "corrections." It is probably better to go with the safe option and obtain even a small post-correction improvement.

9 Future work

Although we are happy with the current OCR results, we believe that there is still room for improvement. We believe that adding more training data for specialized fonts could result in further improvement.

Binarization has been a standard procedure in the OCR workflow for a long time. However, with DNNs available, it would be interesting to see if it would be possible to train successful models on grey or even colored images. Would more information in the image require more training data and a larger neural network?

Finally, voting has shown to be a successful mechanism for improving OCR accuracy. We achieved the best results with 5 relatively similar mixed models, but it would be interesting to see if a different voting algorithm would be more suited to voting successfully on more diverse models (for example, combinations of mixed and monolingual ones).

10 Conclusions

With deep neural networks and randomly sampled training data, it is possible to train one mixed model for the entire data set that performs better than the monolingual models. Voting with five relatively similar models further reduces error rates and post-correction models correct punctuation, resulting in 2.73% CER and 11% WER on the Swedish, and 1.7% CER and 73% WER on the Finnish test set.

The results are far better than any previous results on this data set, including commercial and Ocropy results. Furthermore, with CER on the Finnish Blackletter test set being 1% lower than with Tesseract, this approach results in the best recognition results on this corpus currently available.

Acknowledgements Open access funding provided by University of Helsinki including Helsinki University Central Hospital.

References

1. Breuel, T.: Recent progress on the OCRopus OCR system. In: Proceedings of the International Workshop on Multilingual OCR, p. 2. ACM (2009)
2. Breuel, T.M.: The OCRopus open source OCR system. In: Electronic Imaging 2008, pp. 68150F–68150F. International Society for Optics and Photonics (2008)
3. Breuel, T.M., Ul-Hasan, A., Al-Azawi, M.A., Shafait, F.: High-performance OCR for printed English and Fraktur using LSTM networks. In: 2013 12th International Conference on Document Analysis and Recognition, pp. 683–687. IEEE (2013)
4. Dong, R., Smith, D.A.: Multi-input attention for unsupervised OCR correction. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2363–2372 (2018)
5. Drobac, S., Kauppinen, P., Lindén, K.: OCR and post-correction of historical Finnish texts. In: Proceedings of the 21st Nordic Conference on Computational Linguistics, pp. 70–76 (2017)
6. Drobac, S., Kauppinen, P., Lindén, K.: Improving OCR of historical newspapers and journals published in Finland. In: Proceedings of the 3nd International Conference on Digital Access to Textual Cultural Heritage, pp. 97–102. ACM International (2019)
7. Drobac, S., Lindén, K.: Optical font family recognition using a neural network. In: Proceedings of the Research Data and Humanities (RDHUM) 2019 Conference: Data, Methods and Tools, p. 115. Studia Humaniora Ouluensia, Finland (2019)
8. Eger, S., vor der Brück, T., Mehler, A.: A comparison of four character-level string-to-string translation models for (OCR) spelling error correction. Prague Bull. Math. Ling. **105**, 77–99 (2016). https://doi.org/10.1515/pralin-2016-0001
9. Englmeier, T., Fink, F., Schulz, K.U.: AI-PoCoTo—combining automated and interactive OCR postcorrection. In: Proceedings of the Third International Conference on Digital Access to Textual Cultural Heritage. ACM (2019)
10. Evershed, J., Fitch, K.: Correcting noisy OCR: context beats confusion. In: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, pp. 45–51. ACM (2014)
11. Généreux, M., Stemle, E.W., Lyding, V., Nicolas, L.: Correcting OCR errors for German in Fraktur font. In: Proceedings of the First Italian Conference on Computational Linguistics (CLiC-It 2014) (2014)
12. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 369–376. ACM (2006)
13. Guha, R., Das, N., Kundu, M., Nasipuri, M., Santosh, K., senior member, I.: Devnet: an efficient cnn architecture for handwritten Devanagari character recognition. Int. J. Pattern Recogn. Artif. Intell. (2019)
14. Hämäläinen, M., Hengchen, S.: From the Paft to the Fiiture: a fully automatic NMT and word embeddings method for OCR post-correction. In: Recent Advances in Natural Language Processing, pp. 432–437. INCOMA (2019)
15. Jauhiainen, T.S., Linden, B.K.J., Jauhiainen, H.A., et al.: Heli, a word-based backoff method for language identification. In: Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects VarDial3, Osaka, Japan, December 12 2016 (2016)
16. Kauppinen, P.: OCR post-processing by parallel replace rules implemented as weighted finite-state transducers (2016)
17. Kettunen, K., Kervinen, J., Koistinen, M.: Creating and using ground truth OCR sample data for Finnish historical newspapers and journals (2018)
18. Kettunen, K., Koistinen, M.: Open source Tesseract in re-OCR of Finnish Fraktur from 19th and early 20th century newspapers and journals-collected notes on quality improvement. In: DHN, pp. 270–282 (2019)
19. Kissos, I., Dershowitz, N.: OCR error correction using character correction and feature-based word classification. In: 2016 12th IAPR Workshop on Document Analysis Systems (DAS), pp. 198–203. IEEE (2016)
20. Koistinen, M., Kettunen, K., Kervinen, J.: How to improve optical character recognition of historical Finnish newspapers using open source Tesseract OCR engine. In: Proceedings of the LTC, pp. 279–283 (2017)
21. Koistinen, M., Kettunen, K., Pääkkönen, T.: 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, pp. 277–283 (2017)
22. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Sov. Phys. Dokl. **10**, 707 (1966)
23. Lindén, K., Silfverberg, M., Pirinen, T., Hardwick, S., Drobac, S., Axelson, E.: HFST—An Environment for Creating Language Technology Applications. Studies in Computational Intelligence. Springer, Berlin (2012)
24. Llobet, R., Cerdan-Navarro, J.R., Perez-Cortes, J.C., Arlandis, J.: OCR post-processing using weighted finite-state transducers. In: 2010 20th International Conference on Pattern Recognition, pp. 2021–2024 (2010)

25. Lund, W.B., Kennard, D.J., Ringger, E.K.: Combining multiple thresholding binarization values to improve OCR output. In: Document Recognition and Retrieval XX, vol. 8658, p. 86580R. International Society for Optics and Photonics (2013)
26. Lund, W.B., Walker, D.D., Ringger, E.K.: Progressive alignment and discriminative error correction for multiple OCR engines. In: 2011 International Conference on Document Analysis and Recognition, pp. 764–768. IEEE (2011)
27. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
28. Reul, C., Christ, D., Hartelt, A., Balbach, N., Wehner, M., Springmann, U., Wick, C., Grundig, C., Büttner, A., Puppe, F.: Ocr4all—an open-source tool providing a (semi-) automatic OCR workflow for historical printings. arXiv preprint arXiv:1909.04032 (2019)
29. Reul, C., Springmann, U., Wick, C., Puppe, F.: State of the art optical character recognition of 19th century Fraktur scripts using open source engines. arXiv preprint arXiv:1810.03436 (2018)
30. Reynaert, M.: Ocr post-correction evaluation of early Dutch books online-revisited (2016)
31. Reynaert, M.W.: Character confusion versus focus word-based correction of spelling and OCR variants in corpora. Int. J. Doc. Anal. Recogn. (IJDAR) **14**(2), 173–187 (2010)
32. Romero, V., Toselli, A.H., Vidal, E.: Multimodal Interactive Handwritten Text Transcription, vol. 80. World Scientific, Singapore (2012)
33. Shafait, F.: Document image analysis with OCRopus. In: Multitopic Conference, 2009. INMIC 2009. IEEE 13th International, pp. 1–6. IEEE (2009)
34. Silfverberg, M., Kauppinen, P., Lindén, K.: Data-driven spelling correction using weighted finite-state methods. In: Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata, pp. 51–59. Association for Computational Linguistics, Berlin (2016). http://anthology.aclweb.org/W16-2406
35. Springmann, U., Lüdeling, A.: OCR of historical printings with an application to building diachronic corpora: a case study using the RIDGES herbal corpus. arXiv preprint arXiv:1608.02153 (2016)
36. Springmann, U., Najock, D., Morgenroth, H., Schmid, H., Gotscharek, A., Fink, F.: OCR of historical printings of latin texts: problems, prospects, progress. In: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, pp. 71–75. ACM (2014)
37. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
38. Vobl, T., Gotscharek, A., Reffle, U., Ringlstetter, C., Schulz, K.U.: Pocoto—an open source system for efficient interactive postcorrection of OCRed historical texts. In: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, pp. 57–61. ACM (2014)
39. Wick, C., Reul, C., Puppe, F.: Calamari—a high-performance tensorflow-based deep learning package for optical character recognition. arXiv preprint arXiv:1807.02004 (2018)
40. Wick, C., Reul, C., Puppe, F.: Comparison of OCR accuracy on early printed books using the open source engines Calamari and OCRopus. JLCL **33**, 79–96 (2018)