
Volume 33

—

Number 1

—

2018

—

ISSN 2190-6858

JLCL

Journal for Language Technology
and Computational Linguistics

Special issue on automatic text and layout recognition

Herausgegeben von

Edited by

Kay-Michael Würzner, Alexander Geyken & Günter Mühlberger

GSCL

Gesellschaft für Sprachtechnologie & Computerlinguistik

Contents

Editorial	
<i>Kay-Michael Würzner, Alexander Geyken, Günter Mühlberger</i>	i
Reviewer index	iii
Research articles	1
Improving OCR Accuracy on Early Printed Books by combining Pretraining, Voting, and Active Learning	
<i>Christian Reul, Uwe Springmann, Christoph Wick, Frank Puppe</i>	3
Crowdsourcing the OCR Ground Truth of a German and French Cultural Heritage Corpus	
<i>Simon Clematide, Lenz Furrer, Martin Volk</i> . . .	25
Supervised OCR Error Detection and Correction Using Statistical and Neural Machine Translation Methods	
<i>Chantal Amrhein, Simon Clematide</i>	49
Resource and tool descriptions	77
Comparison of OCR Accuracy on Early Printed Books using the Open Source Engines Calamari and OCRopus	
<i>Christoph Wick, Christian Reul, Frank Puppe</i> . .	79

Ground Truth for training OCR engines on historical documents in German Fraktur and Early Modern Latin
Uwe Springmann, Christian Reul, Stefanie Dipper, Johannes Baiter 97

Author index 115

Impressum

Herausgeber	Gesellschaft für Sprachtechnologie und Computerlinguistik (GSCL)
Aktuelle Ausgabe	Band 33 – 2018 – Heft 1
Gastherausgeber	Kay-Michael Würzner, Alexander Geyken, Günter Mühlberger
Anschrift der Redaktion	Adrien Barbaresi Berlin-Brandenburgische Akademie der Wissenschaften Jägerstr. 22/23 10117 Berlin barbaresi@bbaw.de
ISSN	2190-6858
Erscheinungsweise	2 Hefte im Jahr, Publikation nur elektronisch
Online-Präsenz	www.jlcl.org

The automatic recognition of text in images and its transformation into machine-readable formats is a long-standing promise of research in computer science to the text-based humanities and natural sciences but also to libraries. But while users can expect virtually error-free results for modern sources, the situation is different for more challenging materials: For example, newspapers from the first half of the 20th century already show many difficulties in recognition. The prospects for documents from the 15th-18th century are even worse.

Libraries expect no more and no less than reliable and efficient methods for the text digitization of their own collections. This is associated with the hope of greater outreach, especially via the Internet. With the help of virtual collections, libraries may reach substantially larger user groups which would hardly find their way into the local reading rooms. Unfortunately, text digitization is currently all too often implemented as a by-product of image digitization. A thorough analysis of the results is only carried out to a very limited extent.

On the part of the text-based humanities and natural sciences, there is a desire for large amounts of text data, which do not have to be collected by means of time-consuming, detailed transcription work but can be conveniently retrieved, searched and analyzed. Scientific investigations could thus concentrate on the actual research instead of spending a great deal of time and effort on data collection and processing. However, this requires easy-accessible research data of high quality.

For computer sciences, automatic text recognition is primarily a scientific problem. The aim is to develop algorithmic solutions the quality of which can be verified in a comparable manner using standardized, more or less representative data sets. The results which are presented, practically domain-independent error rates of below 1 %, always read promisingly, but often have only a very limited validity for their application in mass digitization scenarios and consequently for the creation of a reliable basis for text-based research. It took a critical examination of the results of industrial projects such as Google Books or research projects such as IMPACT (Improving Access to Text) to bring the task of text recognition back into the focus of scientific efforts.

However, there are legitimate reasons to hope that error rates which can be observed for modern sources may also be in range for historical documents: The use of statistical learning methods based on (deep) neural networks has led to an enormous leap in quality also in the field of text and layout recognition. This is reflected in a significant reduction of error rates due to a substantially higher tolerance to variances in the material which is to be processed. In addition, the task of handwriting recognition and the recognition of printings are regarded as instances of the same scientific problem.

A prerequisite for the successful use of machine learning methods, however, are great amounts of texts and structural annotations which are sufficiently accurate and representative of the subject (and therefore called ground truth). They serve as a reference point for training and evaluation. The creation of such materials is therefore at least as important as the development and adaptation of algorithms which make use of ground truth data.

This volume of the *Journal for Language Technology and Computational Linguistics* aims to meet both basic requirements for successful automatic text recognition. It contains methodological contributions concerning the actual recognition process as well as contributions dealing with the creation and optimization of necessary training and evaluation data. It is completed by the presentation of concrete resources and tools intended to contribute to the quality of automatic text recognition results with the perspective to qualify them as genuine research data in the not too distant future.

We would like to thank all authors for their contributions. Next, we would also like to thank the reviewers, who have contributed to the quality of the published articles in an excellent way. Last but not least we want to express our gratitude to the German Society for Computational Linguistics and Language Technology and especially Adrien Barbaresi and Lothar Lemnitzer for providing a forum for the topic of automatic text recognition.

The guest editors, Kay-Michael Würzner, Alexander Geyken, Günter Mülberger.

Reviewer index

Konstantin Baierer
Berlin State Library

Gregory Ralph Crane
Leipzig University

Susanne Haaf
Berlin-Brandenburg Academy of Sciences and Humanities

Bryan Jurish
Berlin-Brandenburg Academy of Sciences and Humanities

Roger Labahn
University of Rostock

Marcus Liwicki
Luleå University of Technology

Clemens Neudecker
Berlin State Library

Frank Puppe
University of Würzburg

Joan Andreu Sánchez Peiró
Technical University of Valencia

Klaus Schulz
Ludwig Maximilian University of Munich

Thomas Stäcker
University and State Library Darmstadt

Philipp Zumstein
Mannheim University Library

Research articles

Improving OCR Accuracy on Early Printed Books by combining Pretraining, Voting, and Active Learning

Abstract

We combine three methods which significantly improve the OCR accuracy of OCR models trained on early printed books: (1) The pretraining method utilizes the information stored in already existing models trained on a variety of typesets (*mixed models*) instead of starting the training from scratch. (2) Performing cross fold training on a single set of ground truth data (line images and their transcriptions) with a single OCR engine (OCRopus) produces a committee whose members then vote for the best outcome by also taking the top-N alternatives and their intrinsic confidence values into account. (3) Following the principle of maximal disagreement we select additional training lines which the voters disagree most on, expecting them to offer the highest information gain for a subsequent training (active learning). Evaluations on six early printed books yielded the following results: On average the combination of pretraining and voting improved the character accuracy by 46% when training five folds starting from the same mixed model. This number rose to 53% when using different models for pretraining, underlining the importance of diverse voters. Incorporating active learning improved the obtained results by another 16% on average (evaluated on three of the six books). Overall, the proposed methods lead to an average error rate of 2.5% when training on only 60 lines. Using a substantial ground truth pool of 1,000 lines brought the error rate down even further to less than 1% on average.

1 Introduction

Recent progress on OCR methods using recurrent neural networks with LSTM architecture (Hochreiter and Schmidhuber, 1997) enabled effective training of recognition models for both modern (20th century and later) and historical (19th century and earlier) manuscripts and printings (Fischer et al., 2009; Breuel et al., 2013; Springmann et al., 2014; Springmann and Lüdelling, 2017). Individually trained models regularly reached character recognition rates of about 98% for even the earliest printed books. The need to train individual models in order to reach this level of recognition accuracy for early printings sets the field of historical OCR apart from readily available (commercial and open-source) *general* (also called *polyfont* or *omnifont*) models trained on thousands of modern fonts which yield better than 98% recognition rates on 19th century printings and better than 99% on modern documents. Training historical recognition models on a variety of typesets results in *mixed models* which may be seen as a first approximation

to modern polyfont models, but their predictive power is considerably lower than that of individual models.

In view of the mass of available scans of historical printings we clearly need automatic methods of OCR which in turn require good historical polyfont models. As long as these models are not available and at present cannot be easily constructed (we lack the necessary historical fonts to be able to synthesize large amounts of training material automatically), our next best approach is to maximize the recognition rate of a small amount of manually prepared ground truth (GT). This is the subject of the present paper which applies the methods of pretraining, voting, and active learning (AL) to the field of historical OCR. Using an already trained model as a starting point for subsequent training with additional individual material requires the capability to add specific characters not previously included in the symbol set (the *codec*) and the dynamic expansion (and reduction) of the output layer of the neural network. In the context of recurrent neural networks this was recently made possible by Christoph Wick¹ as reported in Reul et al. (2017c). Voting is a well known method of classifier combination resulting in fewer errors than the best single classifier output (Rice et al., 1992). Active learning ensures that lines showing maximal disagreement among classifiers are included in the training set to enable the maximal learning effect. While more training data is always better, combining these three methods results in a level of recognition accuracy that could otherwise only be reached by a much larger amount of GT and therefore a much larger manual effort to generate it.

Section 2 summarizes the extensive corpus of related work for each of the three methods. In Section 3 we describe the printing material which the experiments of Section 4 are based on. Section 5 contains the discussion of our results and we conclude the paper with Section 6.

2 Related Work

In this section we first sum up a selection of important contributions concerning OCR relevant to our task and introduce findings with regard to training and applying mixed models using OCRopus. Next, a brief summary of the history of voting techniques and applications in the field of OCR is provided. After a short section on transfer learning we give an overview over some basic AL concepts.

2.1 OCR and Mixed Models

Breuel et al. (2013) used their own open source tool OCRopus² to recognize modern English text and German Fraktur from the 19th century by training mixed models, i.e. models trained on a variety of fonts, typesets, and interword distances from different books. The English model was trained on the UW-III data set³ consisting of modern

¹https://github.com/ChWick/ocropy/tree/codec_resize

²<https://github.com/tmbdev/ocropy>

³<http://isis-data.science.uva.nl/events/dlia/datasets/uwash3.html>

English prints. Applying the model to previously unseen lines from the same dataset yielded a character error rate (CER) of 0.6%. The training set for the Fraktur model consisted of mostly synthetically generated and artificially degraded text lines. The resulting model was evaluated on two books of different scan qualities yielding CERs of 0.15% and 1.37%, respectively.

Ul-Hasan and Breuel (2013) promoted an approach not only mixing different types but also various languages by generating synthetic data for English, German, and French. Apart from three language specific models they also trained a mixed one. While the language specific models unsurprisingly performed best when applied to test data of the same language yielding CERs of 0.5% (English), 0.85% (German) and 1.1% (French) the mixed model also achieved a very low CER of 1.1% on a mixed dataset. These experiments indicate a certain robustness or even independence of the OCRopus LSTM architecture regarding different languages in mixed models.

After proving that OCR on even the earliest printed books is not only possible but can be very precise (down to 1% error rate, Springmann, 2015), Springmann et al. adapted the idea of training mixed models to early prints in different application scenarios. In Springmann et al. (2016) their corpus consisted of twelve Latin books printed with Antiqua types between 1471 and 1686. Training on one half of the books and evaluating on the other half mostly yielded CERs of under 10%. Admittedly, these results are far off the numbers reported above which can be explained by the vastly increased variety of the types. Still, the trained models provide a valid starting point for further model improvements through individual training. Additionally, a clear correlation between the intrinsic confidence values of OCRopus and the resulting CER was demonstrated.

In Springmann and Lüdeling (2017) a similar experiment was conducted on the 20 German books of the RIDGES Fraktur corpus⁴. Again, by training mixed models on half of the books and evaluating on the held-out data impressive recognition results of around 5% CER on average were achieved. As expected, the individually trained models performed even better, reaching an average CER of around 2%.

2.2 Alignment and Voting

Handley (1998) gives an overview regarding topics concerning the improvement of OCR accuracy through the combination of classifier results and discusses different methods to combine classifiers and string alignment approaches.

Rice and Nartker (1996) released a collection of command line scripts for the evaluation of OCR results called the ISRI Analytic Tools. Their voting procedure first aligns several outputs using the Longest Common Substring (LCS) algorithm (Rice et al., 1994) and then performs a majority vote. In several competitions they applied their tools to evaluate the results of various commercial OCR engines on modern prints (see, e.g., Rice et al., 1992, 1996). By voting on the output of five engines on English business letters the character accuracy rate (CAR = 1 – CER) increased from between 90.10% and 98.83% to 99.15%.

⁴<http://korpling.org/ridges>

A simple but effective way to achieve variance between the voting inputs was proposed by Lopresti and Zhou (1997) by simply scanning each page three times. While using only a single OCR engine they still achieved a reduction of error rates between 20% and 50% on modern prints resulting in a CAR of up to 99.8%.

Boschetti et al. (2009) improved the output of the best single engine (ABBYY, up to 97% CAR) by an absolute value of 2.59 percentage points by applying a Naive Bayes classifier on the aligned output of three different engines on Ancient Greek editions from the 19th and 20th century. Beforehand, they performed a progressive alignment which starts with the two most similar sequences and extends the alignment by adding additional sequences.

Lund et al. (2011) used voting and dictionary features as well as maximum entropy models trained on synthetic data. Applied to a collection of typewritten documents from Word War II they recorded a relative gain of 24.6% over the word error rate of the best of the five employed OCR engines.

An approach for aligning and combining different OCR outputs applicable to entire books was introduced by Wemhoener et al. (2013). First, a pivot is chosen among the outputs. Then, all other outputs are aligned pairwise with the pivot by first finding unique matching words in the text pairs to align them using an LCS algorithm. By repeating this procedure recursively, two texts can be matched in an efficient way. Finally, all pairs are aligned along the pivot and a majority vote determines the final result.

Liwicki et al. (2011) tackled the task of handwritten text recognition acquired from a whiteboard by combining several individual classifiers of diverse nature. They used two base recognizers which incorporated hidden Markov models and bidirectional LSTM networks and trained them on different feature sets. Moreover, two commercial recognition systems were added to the voting. The multiple classifier system reached an accuracy of 86.16% on word level and therefore outperformed even the best individual system (81.26%) significantly.

Al Azawi et al. (2015) trained neural LSTM networks on two OCR outputs aligned by weighted finite-state transducers based on edit rules in order to return a best voting. After training the network on a vast amount of data very similar to the test set, it was able to predict even characters which were not correctly recognized by either of the two engines. During tests on printings with German Fraktur and the UW-III data set the LSTM approach led to CERs of around 0.40%, considerably outperforming the ISRI voting tool and the method presented in Wemhoener et al. (2013) (between 1.26% and 2.31%). However, applying this method to historical spellings has a principal drawback as it relies on fixed input-output relationships. Since historical spelling patterns are much more variable than modern ones and the same word is often spelled and printed in more than one form even in the same document, it is not possible or at least may not be desired to map each OCR token to a single ‘correct’ token.

In Reul et al. (2018) we implemented a cross-fold training procedure with subsequent confidence voting in order to reduce the CER on early printed books. This method shows considerable differences compared to the work presented above. Not only is it

applicable to some of the earliest printed books, but it also works with only a single open source OCR engine. Furthermore, it can be easily adapted to practically any given book using even a small amount of GT without the need for excessive data to train on (60 to 150 lines of GT corresponding to just a few pages will suffice for most cases).

By dividing the GT into N different folds and aligning them in a certain way we were able to train N strong but also diverse models. Then, these models acted as voters both in the default sequence voting (see ISRI tools above) and a newly created confidence voting scheme which also takes the intrinsic confidence information of the top- n (not just top-1) predictions of OCRopus into consideration. Experiments on seven books printed between 1476 and 1675 led to the following observations:

1. For all experiments the cross fold training and voting approach led to significantly lower CERs compared to performing only a single training. Gains between 19% and 53% were reported for several books and different number of lines of GT.
2. OCR texts with a lower CER benefitted even more than more erroneous results.
3. The amount of available GT did not show a notable influence on the improvements achievable by confidence voting. Yet, a very high number of lines leads to a drop in voting gains for most books. This has to be expected for models that get closer to perfection as most of the remaining errors are unavoidable ones such as characters with defects or untrained glyphs missing in the training set.
4. Increasing the number of folds can bring down the CER even further, especially when training on a large set of lines. However, considering the range of available GT lines and the required computational effort, five folds appeared to be a sensible default choice until further testing regarding parameter optimization has been performed.
5. The confidence voting always outperformed the standard sequence voting approach by reducing the amount of errors by another 5% to 10%.

2.3 Transfer Learning and OCR Pretraining

While to the best of our knowledge there is no suitable related work regarding transfer learning in the field of OCR, it was applied successfully to a variety of other tasks (e.g. Yosinski et al. (2014) for labeling arbitrary images and Wick and Puppe (2017) for leaf classification) by deploying deep convolutional neural networks after performing a pretraining on data from a different but somewhat similar recognition task.

Admittedly, these examples of transfer learning used far deeper networks than OCRopus with only a single hidden layer, resulting in a dramatically increased number of parameters and consequently more opportunities to learn and remember useful low-level features. Nonetheless, since scripts in general should show a high degree of similarity we still expected a noteworthy impact of pretraining and studied the effect of building from an already available mixed model instead of starting training from scratch

(see Reul et al., 2017c). As starting points we used the models for modern English, German Fraktur from the 19th century, and the Latin Antiqua model described above. From our experiments we arrived at the following conclusions:

1. Building from a pretrained model significantly reduced the obtainable CER compared to starting the training from scratch.
2. Improvement rates decrease with an increasing amount of GT lines available for training. While models trained on only 60 lines of GT gained over 40% on average over starting from scratch, this number went down to around 15% for 250 lines.
3. The incorporation of a whitelist for standard letters and digits which cannot be deleted from the codec even if they do not occur in the training GT showed an additional average gain of 5%.
4. Even the mixed models for modern English and 19th century Fraktur which were completely unrelated to the individual books in terms of printing type and age of the training material led to significant improvements compared to training from scratch.

2.4 Active Learning

Settles (2012) gives a very comprehensive overview over the literature dealing with Active Learning (AL). Apart from introducing different usage scenarios and discussing theoretical and empirical evidence for the application of AL techniques they define a typical AL scenario as follows: A learner starts out with access to a (possibly very small) pool of labeled examples to learn from. In order to improve performance it is possible to send queries consisting of one or several carefully selected unlabeled examples to a so-called oracle (teacher/human annotator) who then returns a label for each example in the query. Afterwards, the learner can utilize the obtained additional data. Obviously, the progress of the learner heavily depends on the examples selected to be labeled. Furthermore, the goal is to learn as much as possible from as few as possible queried examples, keeping the oracle's effort to a minimum.

One of the most successful query techniques is called *query by committee* and was introduced by Seung et al. (1992). The basic idea is that a committee of learners/models/voters is trained on the current labeled set. Each member of the committee is allowed to cast a vote on a set of query candidates (unlabeled examples). The assumption is that the candidate the voters disagree most on is also the one which offers the biggest information gain when being added to the training set. This is called the *principle of maximal disagreement*.

Among others, the effect of this approach was demonstrated by Krogh and Vedelsby (1995) who trained five neural networks to approximate the square-wave-function. They performed 2x40 independent test runs starting from a single example and using passive and active learning. While the next example was chosen randomly during the passive tests the networks always got handed the example with the largest ambiguity among

Table 1: Books used during the experiments as well as the amount of GT lines set aside for Training, Evaluation, and Active Learning.

ID/Year	Language	Training	Evaluation	Active Learning
1476	German	1,000	1,000	750
1488	German	1,000	1,000	1,928
1505	Latin	1,000	1,000	1,039
1495	German	1,000	1,114	-
1500	Dutch	1,000	1,250	-
1572	Latin	1,000	1,098	-

the five voters out of 800 random ones. Evaluation showed that AL led to a significantly better generalization error and that the individual additional training examples on average contributed much more to the training process when chosen according to the principle of maximal disagreement.

As for OCR, Springmann et al. (2016) performed some initial experiments on selecting additional training lines in an active way. After recognizing lines with a mixed model they tested several strategies according to which they chose lines for further transcription. The best result was obtained when using a mixture of randomly selected lines combined with lines with low confidence values. It is worth mentioning that after transcribing these lines they started their training from scratch since the pretraining approach introduced above had not been developed, yet.

3 Materials and Methods

In this section we first introduce the early printed books and mixed models we used for our experiments. Then our previous approaches for separate voting and pretraining are briefly described on a technical level. Finally, we show how the principle of maximal disagreement can be utilized in order to choose additional training lines in an informed way within an iterative AL approach.

3.1 Books

The experiments were performed on six early printed books (see Table 1). The AL experiments were carried out on the three books above the horizontal line. We focused on these books as they provided a large amount of GT which is needed to perform the procedure. In a real world application scenario it would be sensible to choose the additional training lines by recognizing all lines without GT and choose the worst ones. Therefore, as many lines as possible are required to be able to evaluate this scenario.

To avoid unwanted side effects resulting from different types or varying line length only lines from running text were used and headings, marginalia, page numbers, etc. were excluded. 1505 represents an exception to that rule as we chose the extensive

Dem hoeghen marschalck sent qurijn
 dienen. Da was der keyser gar fro dz sein syn
 en vnd materien erlengert vnd sch
 Sonder beghefel volmarct in vruughden.
 Stultoz ꝛc. Qm̄ inquā stultoz vt
 complexus fit: ut sub æquatore est

Figure 1: Different example lines from the six books used for evaluation. From top to bottom: excerpts from books 1476, 1488, 1495, 1500, 1505, and 1572.

commentary lines instead, as they presented a bigger challenge due to very small inter character distances and a higher degree of degradation. Figure 1 shows some example lines.

The books published in 1495, 1500, and 1505 are editions of the Ship of Fools *Narrenschiff* and were digitized as part of an effort to support the Narragonien project at the University of Würzburg⁵. Despite their similar content these books differ considerably from an OCR point of view since they were printed in different print shops using different typefaces and languages (Latin, German, and Dutch, see Figure 1). Ground truth for the 1488 book was gathered during a case study of highly automated layout analysis (see Reul et al., 2017a). 1476 is part of the Early New High German Reference Corpus⁶ and 1572 was digitized in order to be added to the AL-Corpus⁷ of Arabic-Latin translations.

3.2 Mixed Models

During our experiments we made use of three mixed models. Our first model LH (abbreviated for Latin Historical) was trained on all twelve historical books introduced in Springmann et al. (2016). The books are printed in Latin and with Antiqua types. The training was performed on 8,684 lines and was stopped after 109,000 iterations. We evaluated all resulting models on 2,432 previously unseen test lines in order to determine the best model which occurred after 98,000 training steps achieving a CER of 2.92% .

⁵<http://kallimachos.de/kallimachos/index.php/Narragonien>

⁶<http://www.ruhr-uni-bochum.de/wegera/ref/index.htm>

⁷<http://arabic-latin-corpus.philosophie.uni-wuerzburg.de>

Additionally, we used the freely available OCRopus standard models for English (ENG)⁸ and German Fraktur (FRK)⁹ introduced in Breuel et al. (2013) and described above.

It is worth mentioning that all the books, which were introduced in the last section and will be used for evaluation, were disjoint with the training materials of the mixed models.

3.3 Cross Fold Training and Confidence Voting

In the absence of viable alternatives to OCRopus we introduced variations in the training data in order to obtain highly performant individual yet diverse voters. This was done by applying a technique called cross fold training: The available GT is divided into N folds with N being the number of models which will later participate in the vote. Then, N training processes take place by using one fold for testing, i.e. choosing the best model, and the rest for training. While the training data shows a significant overlap for each training the folds used for testing are distinct.

After determining the best model of each training process each one of the resulting best models recognizes the same set of previously unknown lines. As an output not only the OCR text is stored but also so-called *extended lloc* (**L**STM **l**ocation **o**f **c**haracters) files which store the probability of the actually recognized character as well as the probabilities of its alternatives.

During the confidence voting process the different result texts are first aligned by applying the ISRI sync tool which identifies the positions of OCR differences as well as the output of each voter at this position. The confidence voting is performed by identifying the corresponding extended llocs and adding up the confidence values for each of the character alternatives. The character with the highest confidence sum gets voted into the final output (Reul et al., 2018).

3.4 Pretraining utilizing Transfer Learning

The character set of pretrained mixed models often comprises more or fewer characters than the data the new model is supposed to be trained on. In order to assure full compatibility we had to make some enhancements on the code level regarding the OCRopus training process which are available at GitHub¹⁰. These changes allow us to comfortably extend or reduce the codec depending on the available training GT and the characters it contains. When starting the training process the character sets of the existing model and the GT are matched. For each character which occurs in the GT but is not part of the model an additional row is attached to the weight matrix and the corresponding weights are randomly initialized. A character which is known to the model but is not part of the GT leads to the deletion of the corresponding row in the

⁸<http://www.tmbdev.net/en-default.pyrnn.gz>

⁹<http://tmbdev.net/ocropy/fraktur.pyrnn.gz>

¹⁰https://github.com/ChWick/ocropy/tree/codec_resize

matrix. In order to avoid blind spots especially when dealing with small amounts of GT and important but less frequent characters like numbers or capital letters it is possible to define a so-called *whitelist*. Characters on the whitelist will not get deleted from the matrix regardless of whether they occur within the GT or not (Reul et al., 2017c).

3.5 Active Learning

As explained above the basic idea behind Active Learning is to allow the learners, i.e. the different voters (see 3.5), to decide which training examples they benefit the most from instead of selecting additional lines randomly. Since in a real world application scenario there usually is no GT available for potential new training lines, we cannot just use the ones for which our current models give the worst results, i.e. the ones with the highest CER. However, we can still identify the lines we expect to be most suitable for further training by following the principle of maximal disagreement.

After recognizing a line with each model of an ensemble consisting of n voters, all outputs are compared to each other in pairs. As a measure for the difference between two OCR output strings a , b , we define the Levenshtein Distance Ratio $LDR(a, b)$ as the Levenshtein distance between a and b , divided by the maximum string length of a and b . These ratios are then summed up between all pairs of different OCR output strings and divided by the number of pairs $n(n-1)/2$ to yield the average LDR_{\emptyset} . The pseudo-code in Algorithm 1 details our calculations.

Data: *line* image without GT, ensemble of n voters ($n > 1$)

Result: the LDR_{\emptyset} of the *line*

outputs \leftarrow *recognizeWithAllVoters(line)*

sum = 0

foreach a_i, a_j with $(i < j) \in$ *outputs* **do**

$LDR(a, b) \leftarrow \frac{Lev(a, b)}{\max\{|a|, |b|\}}$
 sum \leftarrow *sum* + $LDR(a, b)$

end

$LDR_{\emptyset} = \frac{sum}{n(n-1)/2}$

Algorithm 1: The calculation of a line’s average Levenshtein Distance Ratio (LDR_{\emptyset}). $Lev(a, b)$ denotes the Levenshtein distance between two outputs a and b . The length (= the number of characters) of an output a is given by $|a|$.

Finally, after processing various lines, they are sorted in descending order according to their LDR_{\emptyset} s. In a real world application scenario the lines are handed to a human annotator who then produces GT by transcribing them or by correcting one of the individual OCR results or the output of the confidence voting, if available. While the idea of the whole process is to give the committee the lines it requested, it is still up to the human annotator to decide whether a line is suitable for further training or not.

For example, a line might have been badly recognized due to a severe segmentation error or due to an unusually high degree of degradation making recognition pretty much impossible. These lines cannot be expected to make a noteworthy contribution to the training process and are therefore discarded.

4 Experiments

In order to evaluate the effectiveness of the methods described above we performed two main experiments. First, the voting and pretraining approaches were combined¹¹ by performing the voting procedure with models which were not trained from scratch but started from one or several pretrained mixed models. Second, the voters resulting from the first experiment served as a committee during an AL approach following the principle of maximal disagreement.

Since the train/test/evaluation distribution of the GT lines has changed, the results can differ from the ones obtained from earlier experiments. Based on the previous results reported in sections 2.2 and 2.3 we chose to implement the following guidelines for all of the upcoming experiments.

1. The number of folds during cross-fold training and consequently the number of voters is set to 5.
2. OCR results are always combined by enforcing the confidence voting approach.
3. Whenever pretraining is used a generic minimal whitelist consisting of the letters a-z, A-Z and the digits 0-9 is added to the codec.
4. Each model training is carried out until no further improvement is expected (e.g. 30,000 iterations for 1,000 lines of training GT).

4.1 Combining Pretraining and Voting

Naturally, the combination of voting and pretraining seems attractive and should be evaluated. The number of lines used for training was varied in six steps from 60 to 1,000. Each set of lines was divided into five folds and the allocation was kept fixed for all experiments. We used two different approaches for pretraining. First, we trained the five voters by always building from the LH model since it yielded the best results during previous experiments. Second, we varied the models used for pretraining. We kept voters 1 and 2 from the first setup (trained from LH). For voters 3 and 4 FRK was utilized as a starting point since it was trained on German Fraktur fonts which are somewhat similar to the broken script of the books at hand. Only one voter (5) was built from ENG as it was the least similar one out of the available mixed models regarding both age and printing type of the training data. This setup was slightly

¹¹The corresponding code is available at GitHub: <https://github.com/chreul/mptv>

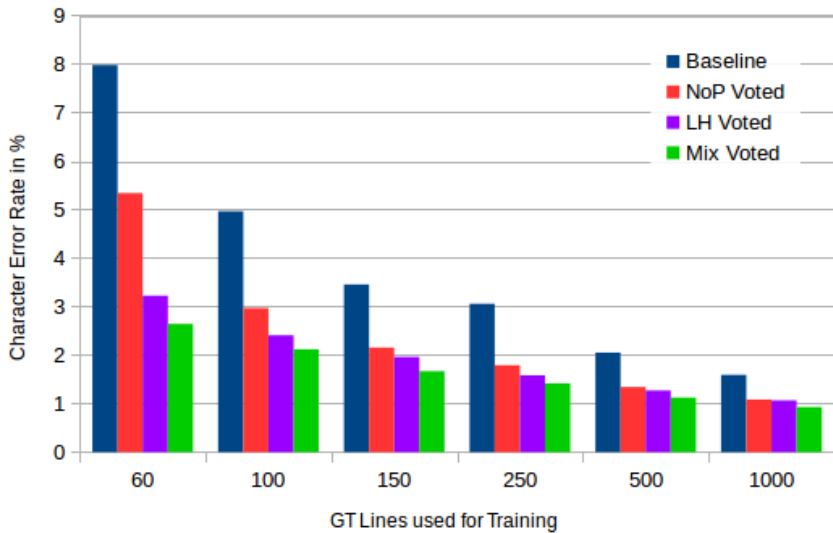


Figure 2: Comparison of the CERs (averaged over all books for each set of lines) of four different approaches: *Baseline* (no pretraining, no voting), *NoP* (no pretraining, voted), *LH* (all five folds trained from the LH model, voted), and *Mix* (mixed pretraining, voted).

adapted for book 1572 as it was printed using Antiqua types. Therefore, in this case one of the two FRK folds was pretrained with ENG instead.

The idea was to still train strong individual models while increasing diversity among them, hoping for a positive effect on the final voting output. The results are summed up in Table 2. For reasons of clarity, detailed numbers are only provided for three books, i.e. the ones which will be used for further experiments. The general behaviour averaged over all books can be seen in Figure 2 and an overview over the progress made by adding more training lines is presented in Figure 3.

In the majority of cases the combination of pretraining and voting considerably outperforms both the default voting approach showing gains of 14% (LH) and 26% (Mix) as well as the default pretraining approach showing gains of 29% (LH) and 39% (Mix) when averaging over all six books and lines. As expected, the best improvements can be achieved when using a small number of GT lines, resulting in gains ranging from 40% (LH) and 51% (Mix) for 60 lines to 2% (LH) and 14% (Mix) for 1,000 lines.

Overall, the average CER on individual folds pretrained with LH (2.70%) is effectively identical to the one achieved by building from a variety of mixed models (2.69%). However, in the case of applying the voting procedure over all five folds, the Mix approach yields considerably better results than just using LH as a pretrained model,

Table 2: CER in % of combining pretraining (*Single Folds*) and voting (*Voting Result*). *Single Folds* contain the baseline without pretraining (*Base*), pretraining with LH model (*LH*), and with a mixture of models (LH, LH, FRK, FRK/ENG, ENG) (*Mix*). *Voting Result* shows the results of different voters based on no pretraining (*NoP*), pretraining with LH model (*LH*), and with mixed model (*Mix*). The *Improvement* columns show the voting gains of LH over NoP (*NL*), Mix over NoP (*NM*), Mix over LH (*LM*), and Mix over the base (*BM*). The underlined CERs represent the starting points for the upcoming AL experiment.

<u>1476</u> Lines	Single Folds			Voting Result			Improvement			
	Base	LH	Mix	NoP	LH	Mix	NL	NM	LM	BM
60	8.12	5.58	4.96	4.72	4.10	2.79	13%	41%	32%	66%
100	6.82	3.99	3.92	3.49	2.67	<u>2.23</u>	23%	36%	16%	67%
150	4.10	3.15	3.03	2.47	2.14	1.66	13%	33%	22%	60%
250	3.24	2.40	2.37	1.70	1.63	<u>1.47</u>	4%	14%	10%	55%
500	2.11	1.73	1.75	1.17	1.13	1.03	3%	12%	9%	51%
1000	1.55	1.30	1.22	0.97	0.88	0.75	9%	23%	15%	52%
<u>1488</u> Lines	Single Folds			Voting Result			Improvement			
	Base	LH	Mix	NoP	LH	Mix	NL	NM	LM	BM
60	7.28	3.97	4.28	4.38	3.05	2.50	30%	43%	18%	66%
100	4.19	2.85	3.20	2.73	2.06	<u>1.84</u>	25%	33%	11%	56%
150	2.96	2.26	2.33	1.81	1.51	1.24	17%	31%	18%	58%
250	2.59	1.82	1.89	1.29	1.21	<u>1.07</u>	6%	17%	12%	59%
500	1.50	1.40	1.38	0.91	0.95	0.79	-4%	13%	17%	47%
1000	1.17	1.06	1.13	0.71	0.72	0.61	-1%	14%	15%	48%
<u>1505</u> Lines	Single Folds			Voting Result			Improvement			
	Base	LH	Mix	NoP	LH	Mix	NL	NM	LM	BM
60	6.54	5.00	5.27	4.58	3.70	3.45	19%	25%	7%	47%
100	4.54	3.96	4.15	3.16	2.82	<u>2.68</u>	11%	15%	5%	41%
150	3.54	3.16	3.16	2.34	2.27	<u>2.02</u>	3%	14%	11%	43%
250	2.85	2.66	2.18	1.98	1.77	<u>1.60</u>	11%	19%	10%	44%
500	2.24	2.18	2.11	1.59	1.60	1.43	-1%	10%	11%	36%
1000	1.84	1.85	1.82	1.35	1.40	1.26	-4%	7%	10%	32%
<u>All</u> Lines	Single Folds			Voting Result			Improvement			
	Base	LH	Mix	NoP	LH	Mix	NL	NM	LM	BM
60	7.98	4.42	4.49	5.34	3.22	2.64	40%	51%	18%	67%
100	4.97	3.38	3.49	2.97	2.41	2.12	19%	29%	12%	57%
150	3.46	2.78	2.87	2.15	1.96	1.67	9%	23%	15%	52%
250	3.06	2.28	2.20	1.79	1.59	1.42	11%	21%	11%	54%
500	2.05	1.86	1.77	1.34	1.27	1.12	5%	16%	12%	45%
1000	1.59	1.46	1.42	1.08	1.07	0.93	2%	14%	13%	42%
Avg.	3.85	2.70	2.69	2.45	1.92	1.65	14%	26%	13%	53%

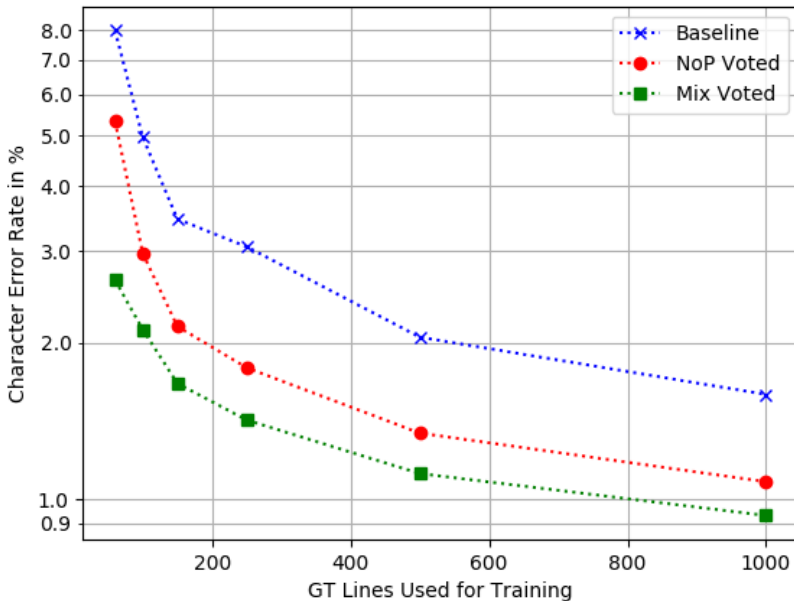


Figure 3: Influence of the number of GT training lines compared for the approaches *Baseline* (no pretraining, no voting), *NoP Voted* (no pretraining, voted), and *Mix Voted* (pretrained with different mixed models, voted) on a logarithmic scale for CER.

leading to an additional reduction of recognition errors by over 13% without an apparent correlation regarding the amount of training GT.

Comparing the best method (Mix + Voting) with the baseline, i.e. the default OCRopus approach (training a single model without any pretraining or voting), shows the superiority of the proposed approach yet more clearly. Even the error rates of strong individual models trained on a 1,000 lines of GT are reduced by more than 40% on average. In general, a substantial amount of GT (>250 lines) is required in the standard OCRopus training (see ‘baseline’ in Figure 2) to match the result achieved by a mere 60 lines when incorporating mixed pretraining and voting, indicating a GT saving factor of 4 or more. A similar factor can be derived when considering the average baseline CER for 1,000 lines of GT.

Table 3: Results of comparing active to passive learning. *Base Lines* is the number of lines used to train the voters of the previous iteration. The lines added (*Add. Lines*) randomly (*RDM*) or by maximizing the LDR_{\emptyset} (*AL*) correspond to 50% of the base lines. Compared are the resulting error rates (*CER*) after performing a confidence vote and (in case of *RDM*) an averaging calculation. Finally, (*Avg. Gain*) shows the average improvement of the voters trained by *AL*.

Book	Base		Add. Lines Rdm/AL	CER		Average Gain
	Lines	CER		Rdm	AL	
1476	100	2.23	50	1.80	1.31	26%
	250	1.47	125	1.18	0.90	24%
1488	100	1.84	50	1.54	1.05	32%
	250	1.07	125	0.86	0.65	24%
1505	100	2.68	50	2.17	2.21	-2%
	250	1.81	125	1.60	1.57	2%

4.2 Incorporating Active Learning

To select additional training lines we utilized the models (voters) obtained from the previous experiment. Each model recognized the GT lines set aside for *AL* and the best candidates were determined by choosing the ones with the highest LDR_{\emptyset} as explained in section 3.5. Next, the candidates underwent a quick visual inspection in order to sort out lines where a positive impact on the training was considered highly unlikely due to very rare segmentation errors or extreme degradations. Indeed, this adds a bit of subjectivity to the task but we expected the decision whether to keep or drop a line to be trivial in most cases. However, in our experiments we never skipped a line proposed by the *AL* approach despite coming across several borderline cases which will be discussed below.

We performed two experiments starting with different numbers of (base) GT lines (100/250) which we kept from the previous experiment. For the passive learning approach we added an additional 50% (50/125) of randomly selected lines. This was performed five times and the results were averaged. As for *AL* we chose the lines by following the principle of maximal disagreement incorporating the LDR_{\emptyset} . Since the number of characters per line may vary we made an effort to select only as many lines as necessary to match the average amount of characters in the passive learning approach.

After selecting the lines the base fold setup was kept and we distributed the additional GT evenly over the five folds to ensure an effect on the training itself but also on the selection of the best model. Afterwards, the training was started from scratch/from the default mixed models while the voters were discarded.

Since the previous experiment showed the superiority of the mixed pretraining approach we decided to omit the pretraining using *LH* during the upcoming experiments. Table 3 shows the results.

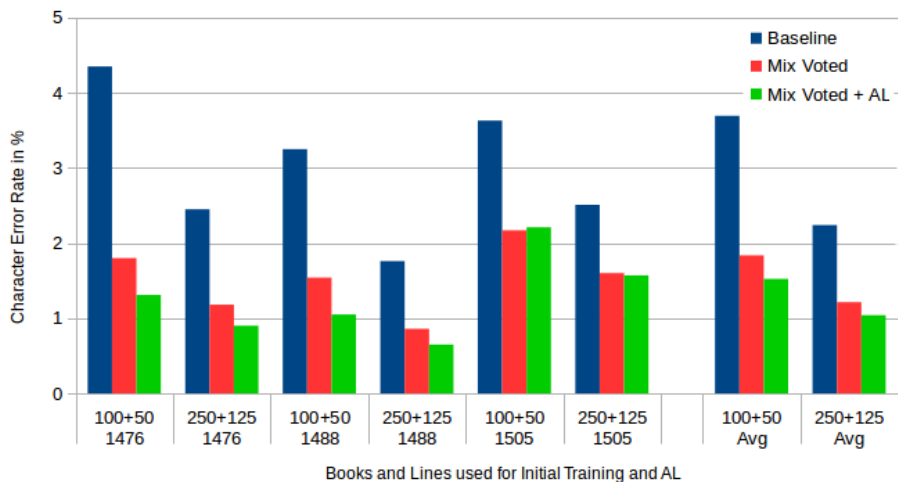


Figure 4: Results of the AL experiments for three books at two different sets of lines comparing the *Baseline* (no pretraining, no voting), *Mix Voted* (pretrained with different mixed models, voted) and *Mix Voted + AL* (Mix with additional lines chosen by AL).

Incorporating AL leads to lower CERs for four out of six tested scenarios. While important improvements with an average gain of almost 27% can be reported for 1476 and 1488, 1505 does not improve at all (see the discussion in the next section).

Moreover, it is worth mentioning that no clear influence of the number of the GT lines available for training can be inferred on the basis of these results. Even when starting from an already quite comprehensive GT pool of 250 lines AL yielded an average gain of 16% compared to randomly chosen lines.

Finally, Figure 4 sums up the results by comparing the baseline to the best pretraining (Mix) approach combined with confidence voting with and without AL.

5 Discussion

The experiments show that the combination of pretrained models and confidence voting is an effective way to further improve the achievable CER on early printed books. While the obtainable gain is highest when the number of available GT lines is small, a substantial reduction of OCR errors can still be expected even when training with several hundreds of lines.

An interesting result of our experiments is that the variability of the voters has a clear influence on the voting result and can even outweigh a superior individual quality of the single voters. This explains why using a variety of models for pretraining considerably

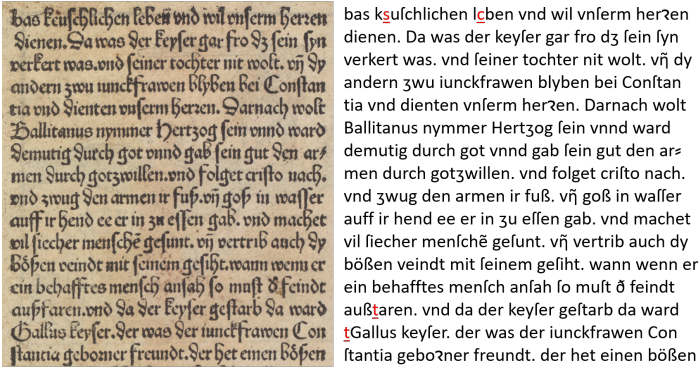


Figure 5: Snippet of a scanned page for 1488 and its best OCR output by combining mixed pretraining and voting. The resulting CER of 0.60% was achieved by training on 1,000 GT lines. The remaining four recognition errors are marked in red and underlined.

outperformed N -fold training from the LH model even though it represented the best fitting one of the available mixed models. Since training from an available model skips the random initialization of values in the weight matrix for pre-existing characters an important chance of introducing diversity to the training process is skipped, resulting in quite similar models even when trained on different but still heavily overlapping folds of training GT.

Following the proposed approach of combining a mixed pretraining with confidence voting allows for a substantially more efficient use of the available GT. On average we were able to achieve the same results as the standard OCRopus approach requiring less than one fourth of the number of GT lines to do so. Moreover, a tiny amount of GT – only 60 lines – was enough to reach an average CAR of close to 97.5%. Only two out of six books showed a CER greater than 3% but comfortably surpassed this value when adding another 40 lines of GT, raising the average to almost 98% character accuracy, which is already considered good enough for many areas of application.

Despite these improvements, there are opportunities for optimizing the achieved results even further, e.g. in applications where the goal is to manually check and correct an entire book in order to obtain a CER of close to 0%. The experiments showed that our method also significantly outperforms the standard approach when training on a very comprehensive GT pool of 1,000 lines, resulting in an average CER of less than 1%. As an example, Figure 5 gives an impression of input (scanned page image) and output (best possible OCR result) for the 1488 printing with a CER of 0.60% reached by training on 1,000 lines combining pretraining using a variety of mixed models and confidence voting.

Even if transcription of an entire book is intended, the goal still should be to minimize the CER by investing the least possible manual correction effort. Therefore, an iterative training approach makes sense and a efficient selection of further training lines is important. Our experiments on AL showed that choosing additional lines in an informed manner can offer an even more efficient way to use the available GT. Despite one of the books not responding at all to the proposed method due to heavily and inconsistently degraded glyphs the three evaluated books still showed an improvement of 17% compared to the mixed voting approach when selecting additional training lines randomly for transcription. It is worth mentioning that during our experiments the number of lines presented to the committee of voters was considerably smaller than in a real world application scenario where it might be sensible to take all yet unknown lines into consideration in order to choose the most promising ones in terms of information gain. This might have a positive effect on the achievable results of the AL approach.

A likely explanation for the lack of improvements of the 1505 book by AL is the degree and type of degradations of the lines queried for training by the voting committee which is illustrated by some example lines shown in Figure 6. The lines on the left are examples the voters fully agreed on and, as expected, got recognized correctly by the base models trained with 100 lines of GT. On the right some of the lines for which the committee disagreed the most are shown, i.e. the ones with highest LDR_{\emptyset} . For 1476 (top) the lines shown had a ratio of 0.45 and CER of 9.44%. There are some signs of degradation, mostly moderately faded glyphs. The worst lines selected from 1488 (middle, 0.72 LDR_{\emptyset} , 33.68% CER) mostly suffered from noise while the glyphs of 1505 (bottom, 0.64 LDR_{\emptyset} , 30.77% CER) frequently show severe deformations. This might be a sensible explanation why AL works very well for 1476 and 1488 but not at all for 1505. Despite the fading and the noise the glyphs of 1476 and 1488 look much more regular than the deformed ones of 1505, at least to the human eye. Therefore, the models trained for 1476 and 1488 using AL learned to see through the effects of fading and noise and earned additional robustness, resulting in a considerable gain in CER. In the case of 1505 the AL models were fed many lines showing severe but very irregular degradations which may have led to an increased robustness but probably did not improve the recognition capability of regular lines as much as the passive learning lines did.

6 Conclusion and Future Work

In this paper we proposed a combination of pretraining, confidence voting, and AL in order to significantly improve the achievable CER on early printed books. The methods were shown to be very effective for different amounts of GT lines typically available from several hours' transcription work of early printings.

In the future we aim to utilize the positive effect of more diverse voters even further. In general, a higher degree of diversity can be achieved in two ways:

Dat dye nuyffer tſamen moedy ch	W guedt ind oick v lijff
Der nuyffer ſtarcke moelen torn	freuwoy ch rieffē ſy ſtaic vaſt yr ſromē ritter ind
Garmhartzich gnedige frauw	Zo my nrebroederē ind do con
fagen ir ſolt ſye abgötter anbeten Des will er	zu nam.
vnd bereyter ſy ſchar wol nach weyſheyrt. vij	gen freuden.
vmb ſeinen dienſt Den du mir gethan haſt. vij	Sein boten gar ſer. vnd emboet ſant Antoni hin
te aut inopia: z in ſua verſutia z opibus	med s tenebris. Prouer. xx.
fenſorē fugiet colon? .i. agricola ſine vi. i.	vet mel. quinto.
mutabile ac tranſitorioiū ſe cōuertunt:	curare mortalia aut nullā vindictā facturū

Figure 6: Example lines of the three books 1476 (top), 1488 (middle), and 1505 (bottom) which were presented to the committee. The left column shows perfectly recognized lines (LDR_{\emptyset} of 0.0). On the right some of the most erroneous lines are presented (LDR_{\emptyset} of 0.45 (1476), 0.72 (1488), and 0.64 (1505)).

1. By the inclusion of more mixed models for pretraining. Since there are only a few good mixed models freely available to date, there is a great need and sharing is key. To lead by example we made part of our GT data and models available online¹².
2. By varying the network structure used for training representing a viable leverage point to increase diversity even further. First steps in this direction have recently been made by Breuel (2017) and Wick et al. (*Comparison of OCR Accuracy on Early Printed Books using the Open Source Engines Calamari and OCRopus*; this issue).

Clearly, the combination of both intended improvements also represents a very promising approach.

To optimize the achievable results, extensive experiments regarding parameter optimization are required. This includes the number of folds/voters, the kind of network they have been trained on, and the method of combination of mixed models used for pretraining, as well as the number of lines the training is performed on.

As for AL an important additional approach is to not only utilize it in order to choose new training lines before the actual training process but also to get involved during the training itself. The standard OCRopus approach is to randomly select lines and feed them to the network. A more efficient method might be to decrease the chance to get picked for lines which already got perfectly recognized and consequently increase it for lines which still cause the current model a lot of problems.

¹²https://github.com/chreul/OCR_Testdata_EarlyPrintedBooks

Concerning the maximal disagreement approach for line selection, it would be interesting to experiment with other ways to determine those lines that offer a maximal information gain. Utilizing the intrinsic OCRopus confidence values that we already use during our voting approach comes to mind but also measures like the Kullback-Leibler-Divergence.

Finally, despite our focus on early printed books the proposed methods are applicable to newer works as well. Especially 19th century Fraktur presents an interesting area of application. Since the typically used Fraktur typesets are more regular than those of the books used during our experiments the goal is to produce a mixed model with excellent predictive power and to avoid book specific training at all.

Nevertheless, the present paper is just a first step to the larger goal of creating an effective, open-source, computer-assisted OCR workflow that is automated to the largest extent possible. In the context of the OCR-D project this also encompasses methods of page segmentation (Reul et al., 2017b) in the preprocessing phase as well as postcorrection (Fink et al., 2017) in the postprocessing stage which are just as important for the quality of the end result as the recognition process itself. To enable further progress in the direction of better recognition results and better polyfont models, we made an extensive set of historical GT data comprising over 300,000 lines available in the GT4HistOCR (*Ground Truth for Historical OCR*) corpus under a CC-BY-SA-4.0 license at Zenodo¹³ (see the article of Springmann, Reul, Dipper, Baiter in this issue).

References

- Al Azawi, M., Liwicki, M., and Breuel, T. M. (2015). Combination of multiple aligned recognition outputs using WFST and LSTM. In *Document Analysis and Recognition (ICDAR), 2015 13th Int. Conf. on*, pages 31–35. IEEE.
- Boschetti, F., Romanello, M., Babeu, A., Bamman, D., and Crane, G. (2009). Improving OCR accuracy for classical critical editions. *Research and Advanced Technology for Digital Libraries*, pages 156–167.
- Breuel, T. M. (2017). High performance text recognition using a hybrid convolutional-LSTM implementation. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 11–16. IEEE.
- Breuel, T. M., Ul-Hasan, A., Al-Azawi, M. A., and Shafait, F. (2013). High-performance OCR for printed English and Fraktur using LSTM networks. *12th International Conference on Document Analysis and Recognition*, pages 683–687.
- Fink, F., Schulz, K. U., and Springmann, U. (2017). Profiling of OCR’ed historical texts revisited. In *Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage*, DATeCH2017, pages 61–66, New York, NY, USA. ACM.
- Fischer, A., Wüthrich, M., Liwicki, M., Frinken, V., Bunke, H., Viehhauser, G., and Stolz, M. (2009). Automatic transcription of handwritten medieval documents. In *15th International Conference on Virtual Systems and Multimedia, 2009 (VSM’09)*, pages 137–142. IEEE.

¹³<https://zenodo.org/record/1344131>

- Handley, J. C. (1998). Improving OCR accuracy through combination: A survey. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE Int. Conf. on*, volume 5, pages 4330–4333. IEEE.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Kirchner, F., Dittrich, M., Beckenbauer, P., and Nöth, M. (2016). OCR bei Inkunabeln – Offizinspezifischer Ansatz der UB Würzburg. *ABI Technik*, 36 (3):178–188.
- Krogh, A. and Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. In *Advances in neural information processing systems*, pages 231–238.
- Liwicki, M., Bunke, H., Pittman, J. A., and Knerr, S. (2011). Combining diverse systems for handwritten text line recognition. *Machine Vision and Applications*, 22(1):39–51.
- Lopresti, D. and Zhou, J. (1997). Using consensus sequence voting to correct OCR errors. *Computer Vision and Image Understanding*, 67(1):39–47.
- Lund, W. B., Walker, D. D., and Ringger, E. K. (2011). Progressive alignment and discriminative error correction for multiple OCR engines. In *Document Analysis and Recognition (ICDAR), 2011 Int. Conf. on*, pages 764–768. IEEE.
- Nartker, T. A., Rice, S. V., and Lumos, S. E. (2005). Software tools and test data for research and testing of page-reading OCR systems. In *Electronic Imaging 2005*, pages 37–47. International Society for Optics and Photonics.
- Reul, C., Dittrich, M., and Gruner, M. (2017a). Case study of a highly automated layout analysis and OCR of an incunabulum: 'Der Heiligen Leben' (1488). In *Proceedings of the 2Nd Int. Conf. on Digital Access to Textual Cultural Heritage, DATeCH2017*, pages 155–160, New York, NY, USA. ACM.
- Reul, C., Springmann, U., and Puppe, F. (2017b). Larex: A semi-automatic open-source tool for layout analysis and region extraction on early printed books. In *Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage, DATeCH2017*, pages 137–142, New York, NY, USA. ACM.
- Reul, C., Springmann, U., Wick, C., and Puppe, F. (2018). Improving OCR accuracy on early printed books by utilizing cross fold training and voting. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 423–428. IEEE.
- Reul, C., Wick, C., Springmann, U., and Puppe, F. (2017c). Transfer learning for OCRopus model training on early printed books. *027.7 Journal for Library Culture*, 5(1):38–51.
- Rice, S. V., Jenkins, F. R., and Nartker, T. A. (1996). *The fifth annual test of OCR accuracy*. Information Science Research Institute.
- Rice, S. V., Kanai, J., and Nartker, T. A. (1992). *A report on the accuracy of OCR devices*. University of Nevada, Information Science Research Institute.
- Rice, S. V., Kanai, J., and Nartker, T. A. (1994). An algorithm for matching OCR-generated text strings. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(05):1259–1268.
- Rice, S. V. and Nartker, T. A. (1996). The ISRI analytic tools for OCR evaluation. *UNLV/Information Science Research Institute, TR-96-02*.

- Rydberg-Cox, J. A. (2009). Digitizing latin incunabula: Challenges, methods, and possibilities. *Digital Humanities Quarterly*, 3(1).
- Settles, B. (2012). Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.
- Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM.
- Springmann, U. (2015). Ocrocis: A high accuracy OCR method to convert early printings into digital text – A Tutorial. <http://cistern.cis.lmu.de/ocrocis/tutorial.pdf>.
- Springmann, U. and Fink, F. (2016). CIS OCR Workshop v1.0: OCR and postcorrection of early printings for digital humanities.
- Springmann, U., Fink, F., and Schulz, K. U. (2016). Automatic quality evaluation and (semi-) automatic improvement of mixed models for OCR on historical documents. *CoRR*.
- Springmann, U. and Lüdeling, A. (2017). OCR of historical printings with an application to building diachronic corpora: A case study using the RIDGES herbal corpus. *Digital Humanities Quarterly*, 11(2).
- Springmann, U., Najock, D., Morgenroth, H., Schmid, H., Gotscharek, A., and Fink, F. (2014). OCR of historical printings of Latin texts: problems, prospects, progress. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, DATECH '14*, pages 57–61, New York, NY, USA. ACM.
- Stäcker, T. (2014). Konversion des kulturellen Erbes für die Forschung: Volltextbeschaffung und -bereitstellung als Aufgabe der Bibliotheken. *o-bib. Das offene Bibliotheksjournal*, 1(1):220–237.
- Ul-Hasan, A. and Breuel, T. M. (2013). Can we build language-independent OCR using LSTM networks? *Proceedings of the 4th International Workshop on Multilingual OCR*, page 9.
- Wemhoener, D., Yalniz, I. Z., and Manmatha, R. (2013). Creating an improved version using noisy OCR from multiple editions. In *Document Analysis and Recognition (ICDAR), 2013 12th Int. Conf. on*, pages 160–164. IEEE.
- Wick, C. and Puppe, F. (2017). Leaf identification using a deep convolutional neural network. *arXiv preprint arXiv:1712.00967*.
- Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Crowdsourcing the OCR Ground Truth of a German and French Cultural Heritage Corpus

Abstract

Crowdsourcing approaches for post-correction of OCR output (Optical Character Recognition) have been successfully applied to several historical text collections. We report on our crowd-correction platform Kokos, which we built to improve the OCR quality of the digitized yearbooks of the Swiss Alpine Club (SAC) from the 19th century. This multilingual heritage corpus consists of Alpine texts mainly written in German and French, all typeset in Antiqua font. Finding and engaging volunteers for correcting large amounts of pages into high quality text requires a carefully designed user interface, an easy-to-use workflow, and continuous efforts for keeping the participants motivated. More than 180,000 characters on about 21,000 pages were corrected by volunteers in about 7 months, achieving an OCR ground truth with a systematically evaluated accuracy of 99.7% on the word level. The crowdsourced OCR ground truth and the corresponding original OCR recognition results from Abby FineReader for each page are available as a resource for machine learning and evaluation. Additionally, the scanned images (300 dpi) of all pages are included to enable tests with other OCR software.

1 Introduction

Crowdsourcing approaches for post-correction of Optical Character Recognition (OCR) output have been successfully applied to several historical text collections (Holley, 2009b; DTAQ, 2016). We report on our crowd-correction platform Kokos,¹ which we built to improve the text quality of the digitized yearbooks of the Swiss Alpine Club (SAC)² from the 19th century. This multilingual heritage corpus consists of Alpine texts mainly written in German and French, all typeset in Antiqua font.

Finding and engaging volunteers for correcting large amounts of automatically OCRed pages into high quality text requires a carefully designed user interface, an easy-to-use workflow, and continuous efforts for keeping the participants motivated.

The scanned images, the uncorrected output of a standard OCR software and the high-quality text corrected by our crowd are a valuable resource.³ It can be used for

¹<http://kokos.cl.uzh.ch>

²<http://www.sac-cas.ch>

³<http://pub.cl.uzh.ch/purl/OCR19thSAC>

extracting heritage lexicons covering 19th century German in particular, or for training as well as testing automatic OCR error correction systems.

In the following section, we introduce our multilingual corpus and describe the process of its digitization. We report on our efforts in building and maintaining a crowd-correction platform and compare them to other work in the field. In Section 3, we analyze and evaluate the corrections performed by the volunteer collaborators. The released resource is described in the last subsection.

2 Materials and Methods

2.1 Corpus Data

In the Text+Berg project⁴ we digitized the yearbooks of the Swiss Alpine Club (SAC) from 1864 until today (henceforth SAC corpus) for building a multilingual heritage corpus of Alpine texts (Göhring & Volk, 2011).

In this paper we focus on the yearbooks from the 19th century. Including tables of content and index pages, the books from 1864 to 1899 amount to 21,246 pages with around 304,000 sentences and 6.3 million tokens (before correction). This is about 16% of our complete SAC corpus.

Thematically, the corpus contains detailed mountaineering and travel reports (mostly from Switzerland, but also from abroad), historical and biological articles (flora and fauna of the Alps), geological and geographical studies (including frequent glacier observations), linguistic articles (e. g. on language boundaries in the Alps), and protocols of the annual club meetings. The text contains a huge number of proper names, geographical names, and Latin botanical names.

Our statistical sentence-based language identification (Dunning, 1994)⁵ assigns 5.5 million tokens to German and 0.74 million tokens to French. See Figure 3 for the distribution of these languages across yearbooks. Additionally, there are a few thousand tokens in English (mostly book and article titles), Italian, Swiss German, and Romansh,⁶ but note that these numbers do not reflect code-switching within sentences (Volk & Clematide, 2014).

2.2 OCR

All the yearbooks from 1864 until 2000 have been collected in printed form. From 2001 until 2009 the SAC has provided us with PDF files, and since 2011 the SAC generates structured XML files directly out of their authoring system.

We obtained the first 10 yearbooks as leather-bound copies. Through collaboration with the Austrian Academy Corpus (AAC) group in Vienna, we scanned them without destroying them. All yearbooks from 1874 until 2000 were cut open so that we were

⁴<http://textberg.ch>

⁵We use M. Piotrowski's PERL reimplementation `Lingua::Ident`.

⁶Most of the 384 sentences (the vast majority) of the 19th century that were automatically classified as Romansh were in fact Latin, French, toponyms, or OCR errors.

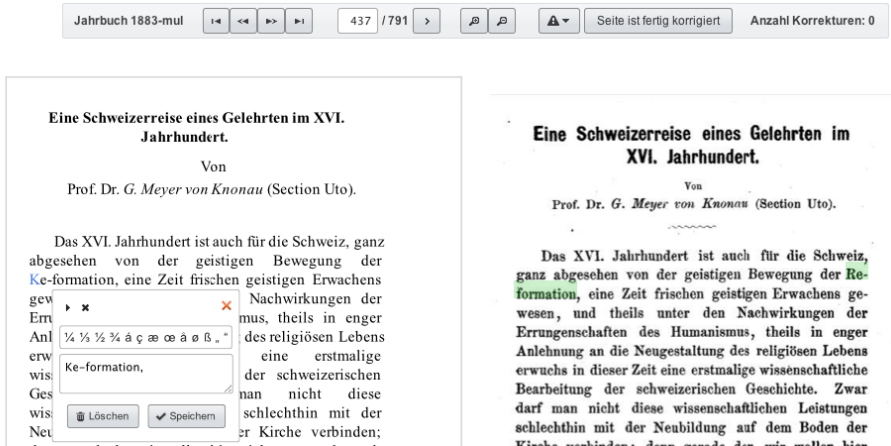


Figure 1: A book page in Kokos: synoptic view of the editable text on the left and the facsimile image on the right. Note the small edit window within the text and the corresponding highlighted word in the facsimile.

able to use a scanner with paper feed. From 1957 onwards, the SAC has published parallel French and German versions of the yearbooks, both of which we processed in the same manner.

After scanning all book pages with 300 dpi, we used the OCR software Abbyy FineReader Pro 7 to convert the images to text (selecting the recognition languages German, French and Italian). This led to mixed text recognition results. The text on some pages was recognized excellently whereas other pages contained a multitude of OCR errors.

Our initial idea was to manually correct these errors in the OCR system since it preserves the mapping between words recognized in the text and the corresponding position on the page. But we soon realized that manual correction is very time-consuming even when working on well-recognized yearbooks of the 20th century. It is prohibitively time-consuming for the yearbooks of the 19th century, where recognition accuracy is inferior because of (a) words that are unknown to the OCR system lexicon (foreign words, old German spellings, toponyms, special terms used in mountaineering, person names, dialect words), (b) special characters (fraction glyphs, Greek letters, old symbols), (c) stains on the paper and curved pages. Generally, the corpus contains many challenges for OCR and Optical Layout Recognition, such as tables, mathematical formulae, spaced type, or words in images.

We investigated various means of improving the OCR quality and correcting OCR errors automatically (Volk, Marek, & Sennrich, 2010). There are only few ways in which a commercial OCR system can be tuned. The most obvious way is to add

“unknown” words to its lexicon. In order to extend the coverage of the built-in lexicon, we collected words with old German spelling patterns (e. g. *acceptiren*, *acceptieren*, *Mittheilung*) and also added the names of 4000 Swiss mountains and cities. This led to some improvements of the OCR quality but a multitude of seemingly random OCR errors persisted.⁷

Then we experimented with two ways of automatic error correction. First we employed a second OCR system (OmniPage) and compared the output of the two systems (Volk et al., 2010). Wherever they disagreed we checked with a German morphological system (Gertwol, see Koskeniemi & Haapalainen, 1996) whether both words were known German words. If so, then we chose the word that occurred more frequently in our corpus. If only one of the words was known, then this was the obvious choice. If none of the words was known, then we trusted Abby FineReader as the more reliable system. This method also led to a small reduction of errors.

Finally we experimented with automatic error correction based on character similarities of words. If an unknown word deviates only in one or two characters from another known word which frequently occurs in our corpus, then we automatically substitute the unknown word with the known word. This method is similar to grammar checking as used in popular text processing software, but needs to work with high precision since human intervention (i. e. manual choice of the correct option) is not possible given the large amounts of text. Therefore we applied this method only for words with a length of more than 15 characters. After all these efforts many spurious OCR errors persisted.

2.3 Crowd Correction

It became obvious that we can only achieve a clean corpus if we organize a large distributed effort for correcting OCR errors via a crowd of volunteers. Therefore, we built the collaborative web-based correction system *Kokos*.⁸ Kokos is based on the wiki idea and is technically built on top of *PmWiki*.⁹ The initial OCR content in Kokos consists of the original Abby FineReader output of all yearbooks of the 19th century, as one of our goals was the assessment of quality of crowdcorrection.

2.3.1 User Interface

We modified the wiki such that it displays the OCRed text of a page and the scan image side by side (see Figure 1). The text is an HTML export from the OCR software, and the layout, paragraphs and font sizes resemble the facsimile.

In the recognized text, each word is a clickable and editable unit. While reading through the text, Kokos correctors can simply click on faulty words in order to open a small editing window (Figure 1). In this window they can modify the word and save the correction. Quick access buttons help to insert frequent incorrectly recognized special

⁷Holley (2009a) comes to similar conclusions.

⁸<http://kokos.cl.uzh.ch>

⁹<http://www.pmwiki.org>

characters, e. g. æ, ß, ¼, or Greek letters. The corrected word immediately becomes visible in the text.

In addition to the correction of characters within a word, three generic operations on the level of one or more adjacent words are frequently needed. First, a delete button removes spurious tokens typically caused by dirt or stains on the page. A second button joins incorrectly split tokens into the edit window, for instance, in the case of spaced type, which was often used to highlight certain words in the 19th century. Third, inadvertently connected words can be split by inserting a blank character.

When the editing window is open or when the user hovers over a word, the corresponding rectangle in the facsimile is highlighted. This is an important and motivating feature that allows the user to quickly spot and doublecheck a suspicious word in the image. The positions of each word were computed by the OCR system during recognition. These coordinates provide the alignment between each word in the text and the corresponding area in the image.

In order to draw the reader's attention to words where the OCR software had low recognition confidence (that is, potential OCR errors), a blue font color was used. Unfortunately, the confidence values of the software were not as reliable as we had hoped, and therefore not as helpful for guiding the human correctors.

In addition to correcting OCR errors, we asked the users to perform *dehyphenation*, i. e. recomposing words that were hyphenated at a line break.

2.3.2 Workflow

In order to attract correctors to work on the task it is important to make initial access as easy as possible. In Kokos we allowed all interested persons to read through the text by browsing and searching. It was then an easy step to register with user name, password and email address in order to sign up as a volunteer corrector. The downside of this is that we know very little about our correctors.

In order to achieve consistent corrections, we provided a set of concise guidelines with typical examples. Additionally, we curated a list of frequently asked questions (FAQ), which was updated according to the problems which our correctors reported. It probably would have been a good idea to introduce the task and the correction guidelines with a short tutorial video.

Users can access the text through a table of contents sorted by yearbook, or a text search, or a "Quick Start" button that leads to a page without or only a few corrections, or an overview of finished and unfinished pages. Especially in the final phase of correction, this view guided our volunteers to correct yearbooks completely. Our basic workflow is "correct errors while reading a text of interest". That crowd correction is driven by curiosity became obvious to us when we noticed that all reports about accidents in the mountains were corrected early on. They are exciting and thrilling. On the other hand, articles about the geology of the Alps with many technical terms (like e.g. *Quarzporphyr*, *Kreideprotogine*, *paläozoische Granite* in 1889) were left until the end.

By clicking on a button, users can mark a page as finished when they consider the text carefully corrected. This button will also automatically advance the view to the next page. Other users can still apply corrections to “finished” pages if need be. We had pondered over whether to lock a page after a user has marked it as finished. The advantage would have been that the page then cannot be affected any more by vandalism. We decided against this automatic locking in order to allow for post-corrections and to send a signal of trust to our contributors. This worked fine.

Kokos also supports an orthogonal workflow via global search and replace, which includes a keyword-in-context view of the search results with facsimile image snippets of the search word (see Figure 2). This speeds up the correction of repeated recognition errors. In order to prevent users from introducing damage by accidental mass replacements, we limited the amount of global replacements to 15 hits per user interaction.

On each page, the correctors were reminded to preserve the spelling of the original text,¹⁰ even if it deviated from modern orthography, or even if they encountered one of the very rare printing errors in our carefully typeset books.

2.3.3 Crowd Management

In January 2014, the SAC monthly magazine LES ALPES, DIE ALPEN, LE ALPI (in all three language versions: French, German and Italian) published a call for volunteer helpers to correct our SAC heritage yearbooks. Dozens of users registered in Kokos and started to contribute within days. After 7 months our active crowd had finished correcting all of the 21,000 pages. We observed a performance pattern which seems to be typical for crowd correction (Holley, 2009b, 15): there were not thousands of volunteers doing tiny bits of work (typical for paid micro-work crowdsourcing), but there was a small crowd of dedicated correctors doing most of the work.

Our correctors were cooperative and reliable, for instance, regarding marking pages as corrected, and we never had to deal with vandalism. Our initial fears that we needed to invest a lot of time to monitor the correction quality, or that a double correction of all pages would be necessary in order to achieve the envisaged quality turned out to be unsubstantiated. This is very much in line with Holley’s tip 14 “Assume volunteers will do it right rather than wrong” (Holley, 2010).

In order to keep the top performers motivated and to give them feedback, a user ranking based on the number of corrections proved to be useful. In our opinion, this kind of gamification is sufficient for volunteers who are inherently interested in a task. For community building, we regularly sent emails to the correctors once a month, informing them about progress and system improvements.

Via social media buttons which we had integrated into each Kokos web page, the users could promote interesting pages to common social media channels. However, this

¹⁰The most important deviations from modern orthography are “c” instead of “z” or “k”, “i” instead of “ie” (*acceptiren*, modern form: *akzeptieren*), “th” instead of “t” (*Thal*).

1868-mul.0526 Pater Pl. a Spescha, mit Anhang von G. Theobald: <i>Das Klima der Alpen am Ende des vorigen und im Anfang des jetzigen Jahrhunderts</i>	Bild davon, was nach der Eiszeit im	Grossen Grössen	geschah, so wie diess auch die Schwankungen
1868-mul.0588 J. Goldschmid: <i>Barometrische Höhenmessungen mit einem neu construirten Aneroidbarometer</i>	geringsten Nachtheil, da diese Differenzen als constante	Grossen Grössen	bei vergleichenden Beobachtungen in Abzug gebracht werden
1868-mul.0593 J. Goldschmid: <i>Barometrische Höhenmessungen mit einem neu construirten Aneroidbarometer</i>	Preisangabe meiner Aneroidbarometer, die ich in verschiedenen	Grossen, Grössen,	von 70 M.M. bis 40 M.M. Durchmesser,
1869-mul.0178 E. v. Fellenberg: 2. <i>Die Ersteigung des Bietschorns</i>	nicht zu verwechseln mit dem eigentlichen oder	Grossen Grössen	Nesthorn (3820 m) östlich vom Lötschthaler
1869-mul.0257 Dr. A. Baltzer: <i>Erste Besteigung der Surettahörner</i>	liegen sie nebeneinander, die von "Weilenmann bezwungenen	Grossen, Grössen,	der spitze Vogelberg, das Rheinwaldhorn mit der

Figure 2: Search result in KWIC view with facsimile snippets for each hit for quick verification.

feature was not used a lot by our correctors, and therefore did not help to attract more volunteer workers.

Even though our crowd correction initiative was advertised in all Swiss language regions, the French texts in our collection were corrected late. We suspect that one of the reasons was that we only offered a German user interface which made the Kokos system foreign to French speakers. It is clearly important that the user interface including the guidelines and the FAQ must be provided in the languages of all targeted contributors.

2.4 Related Work

In order to achieve high quality in the retrodigitization of printed historical text material, there are two viable options (DFG, 2009): (a) manual transcription, or (b) OCR and post-correction.

In the case of manual transcription, independent double-keying by non-native speakers achieves the highest quality (typically, the contracted accuracy on character level is higher than 99.95%), but is most expensive.¹¹ For historical German, Haaf, Wiegand, and Geyken (2013) confirmed these transcription accuracy numbers in their systematic and representative evaluation on texts from 1780 to 1899 taken from the *Deutsches Textarchiv (DTA)*.¹² A sample of 7,208 pages with 9.9 million characters in total was proofread in DTA’s quality assurance platform DTAQ (2016) and 830 transcription errors were revealed. This translates into an overall character-level accuracy of 99.99%. Surprisingly, the accuracy of Antiqua and Gothic typeface was roughly the same.

In the case of OCR, post-correction can be done automatically or manually, for instance, by crowd correction. In the remainder of this section, we discuss relevant manual approaches and initiatives related to our work.

¹¹ Offshore double-keying costs between 0.4 and 0.8 euros per 1,000 keystrokes, depending on structural markup and typeface (Piotrowski, 2012). An accuracy higher than 99% is standard (Long, 1993).

¹² <http://www.deutschestextarchiv.de>

2.4.1 Crowd Correction

The *Distributed Proofreaders* web site,¹³ founded in 2000 by Charles Frank in order to assist the Project Gutenberg in the digitization of Public Domain books, is probably the first crowd-correction initiative, and still active with several thousands of volunteers. The users proofread the OCREd raw text in a simple textual input form while reading through the facsimile. No visual synchronization between the transcribed words and their image location is available. Proofreading is done page per page in two separate rounds by two different proofreaders, optionally, a third round can be applied. In contrast to independent double-keying, these rounds follow each other. The site provides an interesting spell-checking correction mode that presents a view of the text where only words unknown to the spellchecker are editable and, in this way, guides the proofreading process. Book-specific white lists of known and verified word forms can be updated in this mode during the correction in order to adapt the spellchecker to the vocabulary of a text. A qualification system based on the amount of accomplished corrections and successfully passed quizzes concerning the guidelines regulates the type of work a volunteer is allowed to perform. Different statistics monitor the progress of the projects and the individual contributors. The user interface of the website is complex, which partly is due to the fact that the site also includes functionality for formatting the proofread e-books.

Wikisource,¹⁴ another long-term volunteer crowd correction infrastructure, was founded in 2003 and has hundreds of active members. Its German and French instances contain several hundred thousand public domain German and French pages (books and single-leaf prints), many from the 19th century. The technical backbone of every wikisource site is a mediawiki plugin that displays scans and OCREd text side-by-side. No visual synchronization between the transcribed words and their image location is available. Wikisource aims at producing corrected material that satisfies scientific citability and needs. Wiki markup can be used directly in the proofreading phase in order to render some of the typographic layout, for instance spaced type. A page must be proofread in sequence by two different correctors in order to be considered as validated. The correction workflow is openly managed by wiki tags that are set by the users, however, validated pages are protected against further edits, and further corrections must be requested by the wiki discussion pages. On the *Wikisource* as well as on the *Distributed Proofreaders* web site, anyone can import new scanned text material for correction.

The *reCAPTCHA* system (von Ahn, Maurer, McMillen, Abraham, & Blum, 2008) has earned early fame for hiding crowdsourcing effort in OCR correction behind an access system to websites. Users are shown two artificially distorted image snippets where one is known to the system and used for preventing automated abusive access to a website. The other is unknown¹⁵ and its text content will be determined by a

¹³<http://www.pgdp.net>

¹⁴<https://wikisource.org>

¹⁵Actually, the selection criterion for these words is that two different OCR systems suggested two different words.

majority vote of many contributors. Of course, users do not know which word is known and which is unknown. An evaluation on a sample of 50 articles (24,080 words) of the New York Times archive from 1860 to 1970 revealed an accuracy of 99.1% on the word level. This is a large improvement over the standard OCR software word accuracy of 83.5%, and very close to the 99.2% accuracy of the double-keyed transcription in its initial state. The final ground truth was produced by carefully comparing every difference between the manual transcription and the reCAPTCHA output.

The National Library of Australia has set up *trove*,¹⁶ a system for crowd correction of OCRed historical newspapers (Holley, 2009a). The main goal of this initiative is to have the corrected text content accessible for fulltext search, therefore, typographical formatting information is not preserved. The guidelines also explicitly allow for corrections of obvious typesetting errors in the facsimile, however, they encourage the users to add corresponding comments. The user interface has to deal with the complex newspaper column layout. Corrections are applied line by line to segmented articles and the user interface dynamically highlights the corresponding line in the facsimile. There is no proofreader workflow defined on the level of articles or pages, correctors can change any text anytime.

An important technical measure of *trove* for avoiding vandalism is the transparency of user edits: recent edit operations are streamed “live” in the user interface and any user can inspect the recent corrections of any other user. If users detect large amounts of spam or malicious corrections, they can request a roll back. Small fonts in combination with low paper and print quality of historic newspapers often produce bad OCR output, even with the best software available. From 2008 to the end of 2016, almost 220 million lines have been corrected manually. At the end of 2016, there were about 46,000 registered users, but many of them contributed only few corrections, but few correctors contributed a lot. The hall-of-fame reveals that the top ten volunteers have corrected 4.2, 2.9, 2.7, ..., 1.4 million lines by the end of 2016, which amounts to 21.4 millions in total and almost 10% of all corrections.

Commercial platforms for digitization and document collection management such as *Veridian*¹⁷ have also successfully integrated user correction of digitized historical newspapers. *Veridian* is in use by several large libraries around the world. For instance, the *California Digital Newspaper Collection*¹⁸ counted 7.45 million lines corrected by 2,582 users at the end of 2016. Again, the distribution of corrections per user is extreme: the top ten volunteers produced 4.06 millions (54.5%) of all corrections. The user interface and workflow is similar to *trove*; some functionality is missing though, such as the addition of lines that were not recognized at all by the OCR engine. Rose Holley’s blog entry (Holley, 2013) lists five US historical newspapers that employ crowdsourcing for OCR corrections, as well as an Australian, Finnish, Russian, and Vietnamese one. The amount of newspaper pages in combination with the quality of the OCR output

¹⁶<http://trove.nla.gov.au>

¹⁷<http://www.veridiansoftware.com>

¹⁸<http://cdnc.ucr.edu/cgi-bin/cdnc>

make volunteer crowd correction a cost-effective instrument for improving access to these heritage data.

The *Deutsches Textarchiv (DTA)* (Geyken & Gloning, 2015), a large philological archive of 15th–19th-century German texts, includes a web-based quality assurance platform which is open to anyone (Haaf et al., 2013). Although many texts have been transcribed manually by double-keying and are almost free of errors, recently more OCRed texts have been added to the archive.¹⁹ The document representation in the DTA is a highly structured XML format (Haaf, Wiegand, & Geyken, 2014), which requires specific expertise. Therefore, the correction workflow for volunteers is more restricted and adopts the paradigm of issue tickets known from software development. If a user detects a transcription error, he opens a ticket linked to the faulty words, describes the type of error in a form (which covers other issues as well, for instance, formatting or structural problems), and inserts the corrected words in a text field of the form. Each ticket is then resolved by a DTA staff member. This procedure for correcting text errors is more cumbersome for the user, slower than the aforementioned workflows, and probably not suited for the initial correction of raw OCR output. However, it guarantees the preservation of the high philological quality standards of the project. The correction workflow is based on pages and the user has to explicitly mark a page as corrected; the platform distinguishes two types of transcription validation, (a) a confirmation that the extracted text has been read carefully and no text problems have been found, and (b) a confirmation that the extracted text corresponds to the shown facsimile. No visual synchronization between the transcribed words and their image location is available.

The *PoCoTo* system for postcorrection of OCRed historical text comprises several tools and a web-based interface with an interactive workflow whose efficiency has been attested by a small user study (Vobl, Gotscharek, Reffle, Ringlstetter, & Schulz, 2014).²⁰ An interesting feature is the interlinear-like view where facsimile snippets of individual tokens and their recognized text are presented in reading order. The system suggests correction candidates computed in a corpus-based unsupervised manner (Reffle, 2011). The interface also offers batch correction of precomputed error series (e. g. $u \rightarrow n$) in a concordance view.

Citizen science web sites such as *crowdcrafting*²¹ offer a PDF transcription task template which can be used for hosting small-scale projects.

Our review shows that crowd correction for books typically works on the level of pages. For newspapers, corrections are typically applied on the level of individual lines in the context of an article. Yet another approach for involving the crowd into OCR correction is reported by Wang, Wang, and Chen (2013) for ancient Chinese books. They first extract graphically similar Chinese characters and present them to the users in a row for quick verification. This reduces the correction task to the question whether

¹⁹Some of them were taken from the German wikisource site in their corrected form.

²⁰<https://github.com/cisocrgroup/PoCoTo>

²¹See <http://www.crowdcrafting.org>, which is based on the popular *pybossa* crowdsourcing framework.

all logograms in a row are the same. A prototype with a similar approach for old Venice manuscripts has been explored by Simeoni, Mazzei, and Kaplan (2014).

Chronis and Sundell (2011) present *Digitalkoot*,²² a gamification-based system for correcting OCR errors in old Finnish newspapers typeset in Gothic font. The words are taken out of context and inserted into simple games. The authors monitored the activities for the initial two months, in which 4,800 persons played the games and completed 2.5 million microtasks. This was the result of heavy media coverage with more than 30 newspaper articles and some TV programs reporting on the project. The authors remark that a small percentage of users provided one third of the work, therefore showing a similar user behavior compared to volunteers. The quality of the crowd corrections was very high and improved the text from 85 % word accuracy to over 99 %. At the end of the project after 22 months, 8 million microtasks had been solved by the gamers, however, this did not result in 8 million corrected tokens. Several gamers had to agree on a transcription before it will be accepted. Additionally, many known items had to be presented in order to identify “cheaters”. Therefore, Kettunen (2016) concludes that “this approach is clearly not feasible” for the correction of the 837 million words in this corpus and advocates improved automatic OCR post-correction.

Seidman, Flanagan, Rose-Sandler, and Lichtenberg (2016) describe another OCR correction initiative set up as purposeful gaming.²³ The basic idea is similar to reCAPTCHA: words which have been recognized differently by two independent OCR systems are presented to the crowd. The key challenge in the gamification of OCR corrections is the question of how to decide whether a user contribution for an “unknown” word should be rewarded as correct or not. Seidman et al solve the problem by decoupling the reward to the player from the decision on the ground truth. The player is rewarded if his/her contribution exactly matches one of the OCR suggestions, or if it exactly matches an accepted contribution created earlier by another player.²⁴ New contributions by players are only accepted in the system if they match the common substrings of the OCR systems. The ground truth for an unknown word is determined by a threshold of how many times an exact match was found for a contribution. Although the professionally designed game worked technically and even received an award in the field of purposeful gaming, Seidman et al. (2016) had to concede that this approach does not deliver the needed amount of corrections.

The main differences between our platform and the ones mentioned above are the following. First, our system is not practical for documents with complex layout such as newspapers. Second, from a technical point of view our platform is simple, however, it relies on specific HTML output formats produced by the OCR software. Third, instead of trying to create an artificial gaming setup where volunteers would only see single words in isolation, it is important for us to let them read historical documents while correcting OCR errors.

²²<http://www.digitalkoot.fi>

²³A working version of the game is online under <http://smorballgame.org>.

²⁴The game logic tries to minimize false negatives at the cost of allowing the gamer to be rewarded for false positives. Seidman et al. (2016) estimate the false negative rate to be 3%.

In summary, it is safe to state that crowd correction by motivated volunteers is the best strategy to correct annoying OCR errors if your budget does not allow for paid work. Gamification cannot help to correct large amounts of OCR data, even when the input for the correction is reduced to the material where OCR systems disagree. Crowd correction of a large amount of text in a language with a large group of speakers will generally work better; however, even then typically only very few volunteers are doing most of the work. The example of Trove newspaper correction shows that volunteers are also helpful for correcting OCR output with a much lower quality than ours. As a positive side effect, involving a crowd of users is also a good way to disseminate the digitized cultural heritage material.

3 Results

We investigated our crowdsourced corrections in two ways: with a quantitative analysis of the modifications, which is detailed in Section 3.1, and by evaluating the quality of the corrected texts in a representative sample that was checked separately by two persons (Section 3.2).

3.1 Amount of Corrections

For assessing the amount of corrections, we compared two snapshots of the texts, taken at the start and at the end of the correction phase. We determined the amount of corrections by means of the modification rate between the two versions, which is the character edit distance (Levenshtein, 1966) divided by the length of the corrected text. We computed the modification rate with the ISRI frontiers toolkit (Rice, 1996), which is meant for analyzing modifications in OCR text. Across the entire corpus, the edit distance sums up to a total of almost 300,000 edit operations (see Table 1). This means that 0.79 % of the text was modified in the correction process (micro-average). The mean modification rate per page (macro-average) is more than ten times higher, which means that a significant portion of the modifications originates from pages with a small amount of text. Both average figures show an exceptionally high standard deviation of 430 %.²⁵

A major source for this high variability are errors in Layout Recognition, an early stage of OCR responsible for detecting and ordering blocks of text and other page elements, such as figures and tables. When addressing these errors, the correctors often had to rearrange contiguous spans of text (up to multiple paragraphs) in order to establish the correct reading order. However, corrections that involve moving text regions are not appropriately reflected by tools that focus on local changes on the word or character level: While the actual edit action requires only a few clicks and keystrokes independent of the size of the moved text, the tool records a sequence of

²⁵Deleting major portions of a page may lead to an edit distance greater than the length of the corrected text, which results in a modification rate of more than 100 %.

Table 1: Effects of filtering on corpus size and modification rates.

filtering	pages	para- graphs	tokens	characters	cumul. LD	modification rate			
						macro (SD) %	%	micro (SD) %	%
none	21,246	137,395	6,451,906	37,158,538	293,366	8.51	(430.0)	0.79	(430.1)
page-wise	19,029	110,726	5,857,982	33,886,068	180,987	2.83	(18.9)	0.53	(19.1)
./ . DE	17,190	93,575	5,236,748	30,720,939	160,625	2.55	(18.0)	0.52	(18.1)
./ . FR	1839	17,151	621,234	3,165,129	22,969	4.36	(23.3)	0.72	(23.6)
para-wise	19,024	107,043	5,838,302	33,773,261	141,592	1.28	(4.5)	0.42	(4.6)
./ . DE	17,186	90,855	5,221,461	30,632,199	127,316	1.26	(4.3)	0.42	(4.4)
./ . FR	1838	16,188	616,841	3,141,062	14,276	1.39	(5.9)	0.45	(5.9)

deleted characters in one spot and a corresponding insertion elsewhere, measuring a value proportional to the number of characters shifted.

In order to avoid this distorting effect and to measure the amount of typical corrections in running text more reliably, we removed pages that were prone to artificially enlarge the number of edit operations for the evaluation. In particular, we removed table-of-content pages (which had been manually corrected in the initial digitization phase already) and pages with large tables or page-size images. Furthermore, we discarded pages written in one of the sparsely represented languages, i. e. languages other than French or German, and pages containing more than one language,²⁶ which enabled us to analyze the modifications separately for the two major languages.

Identifiers embedded in the HTML markup allowed us to easily align both versions at the paragraph level. We removed paragraphs that were missing in one of the snapshots (which means that they were either completely deleted or inserted in the correction phase). After this filtering, we were left with a set of around 19,000 pages with a total of 111,000 paragraphs and 33.9 million characters, resulting in a reduction of approximately 10 % (see the rows concerning page-wise filtering in Table 1).

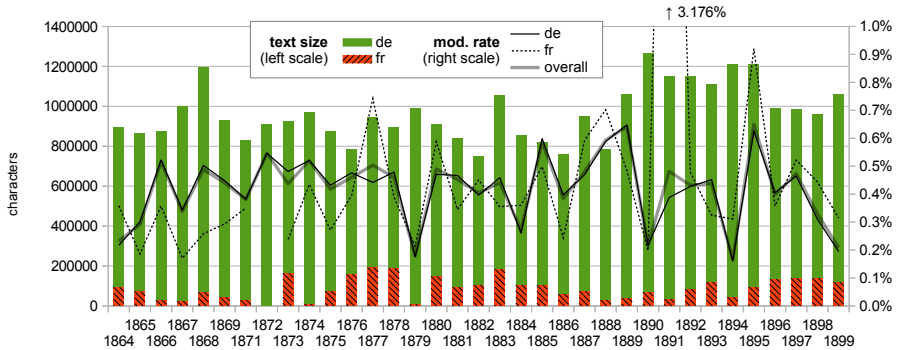
In this filtered corpus, the modifications in the French and German sentences affect 180,000 characters, which equals to an overall modification rate of 0.54 % (micro-average). For French the modification rate is 0.72 %, which is considerably higher than for German (0.52 %). The mean modification rate of all paragraphs (macro-average) shows an even more substantial difference between German and French.

The difference between micro- and macro-average as well as the large variance indicate that, still, a small number of paragraphs have a high modification rate. Inspecting such cases revealed that many occurrences of text reorganisation had remained, e. g. in tables that had not been removed in the first filtering method. Since our ID-based paragraph alignment does not capture text regions moved across paragraphs, we decided to exclude these cases in an additional filtering step. We removed all paragraphs which showed a

²⁶As identified by our downstream processing pipeline.

Table 2: Word error rate of the paragraph-wise filtered corpus.

	macro (SD)		micro (SD)	
./ . DE	4.23 %	(13.25 %)	1.58 %	(13.51 %)
./ . FR	4.39 %	(13.50 %)	1.59 %	(13.78 %)
	3.09 %	(11.59 %)	1.54 %	(11.69 %)

**Figure 3:** Text size and modification rate (micro-average) in the paragraph-wise filtered data.

change in length of 10 % or more between the two snapshots. This reduced the corpus size by only around 0.3 % (see the rows concerning paragraph-wise filtering in Table 1). The overall modification rate decreased by a fifth to 0.42 %, and the gap between German and French became smaller (0.42/0.46 % respectively in micro-average). As a side effect, this filtering also removed spurious paragraphs caused by spots or dirt.

The modification rate is computed on the level of characters, rather than words. This has the advantage that it can be easily derived from the edit distance and that it reflects directly the amount of edit operations (keystrokes) which the correctors performed. Also, it avoids the complexity of different tokenization rules for different languages, and the rate includes modifications that affect non-word characters (punctuation). However, the amount of words that were changed in the correction phase is a meaningful figure too. Therefore, Table 2 shows the proportion of word tokens in the paragraph-wise filtered corpus that were modified by the correctors. We used the *ocrevAlUAtion* tool (Carrasco, 2014) to compute the *word error rate* of the original text, as compared to the crowd-corrected version. It is interesting to see that the word error rate is lower for French than for German (clearly for the macro-average), whereas the inverse distribution is found for the character-level modification rate. This suggests that more misrecognized characters are concentrated in fewer words in the French texts.

Figure 3 shows the modification rates across all yearbooks, plotted against the text size. We found no correlation between the size of a yearbook and its modification

rate, nor did we observe a clear tendency over time (correlation age–modification rate). Often, the modification rates for French and German develop in parallel, which seems intuitive given that paper and printing quality as well as the condition of preservation is the same within one multilingual yearbook. French tends to have a stronger amplitude, showing low values for volumes with a low total rate and even much higher values for highly modified volumes. At least partially, the increased variability might be due to the relatively small amount of French texts, which gives more weight to individual outliers. Some of the slumps in the modification rate (e. g. 1890, 1899) can be attributed to correction efforts early in the digitization process, which were carried out using the user interface of the OCR software. This means that, occasionally, the text quality was already considerably improved before exporting into the online correction system, leaving less work to do for the crowd correctors.

A selection of frequent corrections is given in Table 3. All examples are misrecognized tokens that were corrected multiple times in different places. In many cases, the affected word posed increased challenges to the OCR system, in that it is not expected to be found in a dictionary that covers the general vocabulary of contemporary German or French. Often this is due to orthographic and linguistic variation, such as regional (examples 10–14) and historical spelling (13–16, 34–37) as well as outdated morphology (17–19), or because the word belongs to an open class, such as toponyms (20–23, 44–52), and person names (24–27, 38–40). Many errors are related to spurious or missing diacritic marks (11–14, 17–19, 32–37, 43–44, 48–50). Also, non-alphanumeric characters (55–57), superscripts (51–54), and certain letters (e. g. upper-case *R*, see 7–9, 47) are generally badly recognized. Occasionally, French words appear as if the background dictionary of another language was in place during recognition (41–45). Some place names show a spelling alternation adapted to French orthography, whereas they had been recognized in the spelling of their original language (48–50).

From a natural language processing point of view, it is worthwhile to look at cases that are particularly hard to tackle in automated post-correction. As such, many corrections deal with *real-word* errors (3–6, 10–14, 16–21, 38), i. e. tokens that match an existing word, which means that their erroneous nature can only be revealed through their context or by comparison with the facsimile. A similarly tricky issue is dehyphenation, which cannot be performed mechanically in a linguistically unaware fashion (see example 28 vs. 30). We tried to estimate the amount of different real word error types for words (excluding their non-alphanumeric characters such as quotation marks or hyphenation characters). About 19% of all word types that were corrected at least once can be found in the corrected version of our corpus, and therefore, might be considered as real word errors.

Table 4 shows the most frequent edit operations. Most of the top modifications are concerned with fixing word boundaries through insertion and deletion of spaces and hyphens. Some operations had been carried out using the search and replacement interface, such as the global replacement of quotation marks or the removal of squares and bullets. 35.9% of the modifications are deletions of one or more characters (mostly punctuation and whitespace characters). Many corrections are related to letters with

Table 3: Frequent word corrections.

German			French		
OCR	corr.		OCR	corr.	
nnd	und	(1)	ll	ll	(31)
zn	zu	(2)	ä	à	(32)
sieh	sich	(3)	öü	où	(33)
Ton	von	(4)	complètement	complètement	(34)
Über	über	(5)	mètres	mètres	(35)
lieber	Ueber	(6)	privilège	privilège	(36)
Eichtung	Richtung	(7)	sécher	sécher	(37)
Kichtung	Richtung	(8)	Aimer	Almer	(38)
Bedaktion	Redaktion	(9)	Ford	Forel	(39)
Hessen	liessen	(10)	Nsegeli	Nægeli	(40)
massig	mässig	(11)	Tun	l'un	(41)
Händen	Handen	(12)	Us	Ils	(42)
Centralcomite	Centralcomité	(13)	à l'etude	à l'étude	(43)
Bureau	Büreau	(14)	See	Scé	(44)
Thaies	Thales	(15)	Mordes	Morcles	(45)
grossenteils	grossentheils	(16)	VOfenpass	l'Ofenpass	(46)
altern	ältern	(17)	Ehône	Rhône	(47)
Schütze	Schutze	(18)	Lütschine	Lutschine	(48)
Schlüsse	Schlusse	(19)	Saas-Fee	Saas-Fée	(49)
Eimer	Elmer	(20)	Palü	Palu	(50)
Gesehenen	Geschenen	(21)	S*-Bernard	S ^t -Bernard	(51)
Unterwaiden	Unterwalden	(22)	S ^t -Gothard	S ^t -Gothard	(52)
Bergeil	Bergell	(23)			
Bubi	Dübi	(24)			
Imfeid	Imfeld	(25)			
111.	Ill.	(26)			
Franche	Francke	(27)			
Ueber-gang	Uebergang	(28)			
all-mälig	allmällig	(29)			
Schnee-und	Schnee- und	(30)			

language-independent		
OCR	corr.	
')	')	(53)
m 8	m ³	(54)
-f-	+	(55)
°/o	%	(56)
Va	½	(57)

diacritic marks, which appear to be particularly challenging for OCR in a multilingual corpus. 7.6% of the observed edit operations differ only by diacritics (e.g. a→ä or vice versa).

3.2 Quality of Corrections

In order to assess the quality of the corrected pages, we decided to carefully validate them using a representative sample. For having a wide base, sampling units should be as small as possible, e.g. words or even single characters. However, proofreading individual characters is tedious, and judging words also often requires additional context. In the trade-off between coverage and sufficient context, we set the unit size to a span of 1–2 printed lines, on which the proofreaders consented that it is considerably less tiring than

Table 4: Most frequent edit operations.

German						French					
freq.	OCR	corr.	freq.	OCR	corr.	freq.	OCR	corr.	freq.	OCR	corr.
13,970	< >	< >	528	<a>	<ä>	2024	< >	< >	49	<»>	<«>
10,006	<->	< >	496	<Y>	<V>	701	<e>	<é>	45		<R>
7175	<^>	<“>	486	<é>	<e>	526	<->	< >	44	<œ>	<æ>
3669	< >	< >	461	<•>	< >	417	< >	< >	42	< >	< >
2644	<i>	<l>	411	<u>	<n>	372	<^>	<“>	41	<*>	<l>
1942	<e>	<c>	407	<ii>	<ü>	183	< >	< >	39	<O>	<0>
1354	< >	< >	383	<o>	<ö>	141	<->	< >	38	<V>	<l>
1147	<K>	<R>	380	<ii>	<n>	128	<ä>	<â>	38	<*>	<t>
1146	<E>	<R>	358	<ti>	<ü>	126	< >	< >	37	<->	< >
1079	<u>	<ü>	353	<™>	< m>	124	<e>	<è>	36	< >	< >
1070	<^>	< >	337	<0>	<O>	114	<n>	<„>	35	< >	< >
912	< >	< >	332	< >	< >	113	<i>	<l>	35	<l>	<l>
847		<R>	294	< >	< >	112	<e>	<c>	34	<se>	<æ>
824		<h>	275	<*>	<l>	98	<é>	<e>	34	<->	<←>
783	<->	< >	269	<a>	<u>	98	<■>	< >	33	<l>	< >
782	<U>	<ü>	268	<—>	<->	95	<•>	< >	33	<è>	<é>
766	<n>	<u>	236	<«>	<e>	94	<l>	< >	32		<h>
741	<ö>	<o>	229	<ii>	<u>	91	<E>	<R>	32	<0>	<O>
722	<ü>	<u>	226	<->	< >	85	<←>	<->	32	<Y>	<V>
713	<l>	< >	225		<D>	82	<ii>	<ü>	32	<l>	<t>
655	<■>	< >	221	<tt>	<ü>	82	<K>	<R>	32	<e>	<è>
625	<ä>	<a>	218	<*>	< >	80	<l>	<l>	31	< >	< >
604	<l>	<l>	214	<a>	<n>	76	<™>	< m>	31	<a>	<s>
573	<e>	<é>	213	<i>	< >	73	<n>	<u>	27	<k >	< >
533	<m>	<rn>	202	<»>	<s>	50	<^>	< >	27	<l>	< >

reading randomly sampled words. Therefore, we divided the filtered corpus into snippets with a soft target size of roughly 100 visible characters (117 including whitespace, on average), which was adjusted accordingly to meet word and page boundaries.

For determining the minimal required size of the sample, we regarded the problem as an application of empirical probability of one character being incorrectly recognized. Preliminary investigation suggested that the rate of remaining errors did not exceed 0.1% on the level of characters. Since the distribution of the two classes (correctly vs. incorrectly recognized) are very skewed (999:1), we chose a narrow error band of $\pm 0.02\%$. With a significance level of $p < 0.01$ (and thus $z^2 \approx 6.635$), the minimal required sample size in terms of characters is:²⁷

$$\frac{z^2}{0.0002^2} \times 0.001 \times (1 - 0.001) = 165706.54$$

²⁷Assuming that recognition errors are independent of each other, so that sampling sequences of contiguous characters yields the same distribution as sampling individual characters.

As this number is close to $\frac{1}{200}$ of the corpus, we divided the corpus into 200 stratified folds and picked one for proofreading. Each fold contained approximately 1440 snippets that were randomly sampled, but with a distribution representative for the entire corpus with regard to yearbook and language.

The selected sample had a size of approximately 25,000 word tokens. It was independently proofread by two German native speakers with good knowledge of French. They were asked to correct the snippets according to the guidelines of the crowd correctors. For each snippet, an appropriately cropped facsimile image was provided for collation.

5 % of the snippets were modified by at least one proofreader. Well over half of the modifications were done by both correctors in agreement, the rest was contributed by either of them in similar parts. When both proofreaders modified the same word, their modifications were always identical, i. e. they never disagreed on how to correct an error, but only on its mere presence. It is most likely that the disagreements arose from varying attentiveness, rather than differences in judgment: Some of the errors just slipped through, as they already had for our crowd correctors.

We sought for a way to quantify the agreement between the correctors with a standard measure. If we assume that the correction of each detected error is unambiguous, the modifications can be modeled as a binary classification task (namely error *detection*) on the word level. Thus, each word in the sample is a data point, comparing the correctors' decisions to change this word or leave it unchanged. Measuring the agreement with Fleiss' κ (Fleiss, 1971) yielded a value of 0.67. Discussions between the proofreaders revealed that the guidelines were not detailed enough concerning whitespace (e. g. spaces separating the integer and fractional part in decimal numbers). By applying appropriate adjudication to these cases, κ raised to 0.73.

We then merged the proofread snippets into a single gold standard. Judging from the κ score, a few errors may have remained undetected, but we expect them not to be more than a handful, as the number of errors found by only one of the proofreaders and missed by the other was small.

By comparing to the gold standard, we measured the spelling quality of the sample as corrected by the online collaborators. In total, the proofreaders corrected 113 characters in 72 words, which means that the crowd-corrected texts achieved a high accuracy of 99.71 % on the level of words and 99.93 % on the level of characters. Qualitatively, most of the remaining errors were hard-to-spot details, such as missing commas or diacritics (e. g. *avance/avancé*) or substitutions of similarly looking letters (e. g. *Clnbhütte/Clubhütte*, *Generalyersammlung/Generalversammlung*).

Based on the accuracy figures, we estimate the proportion of errors removed by the crowd correctors. The modification rate of the filtered corpus tells us that 141,592 characters were edited (see Table 1). Under the assumption that every modification by the correctors effectively contributed to an error correction, this can be considered as the number of removed character errors. Extrapolating the observed rate of remaining errors in the sample (0.07 %²⁸) to the entire corpus (33.8 million characters), we can

²⁸The exact observed character error rate is $\frac{113}{169619}$.

estimate the amount of remaining errors to be approximately 22,500 character errors. This means that our online collaborators removed a proportion of $\frac{141592}{141592+22500}$ of all errors, which is a reduction rate of 86 %. For the word level, an analogous computation yields an estimated reduction rate of 85 %.

3.3 OCR19thSAC: an OCR Resource for Training and Testing

While correcting the OCR errors in our heritage corpus of German and French texts from the Alpine domain, we created a large OCR ground truth that can be either used as OCR training and testing material, or for optimizing automatic OCR post-correction, or as a resource for lexicon extraction. The estimated word-level accuracy of 99.7 % provides a good basis for evaluating systems that either process the scanned images of the pages or try to improve upon the output of a standard OCR system. We provide the scanned images, the initial snapshot of the extracted text, and the crowd-corrected ground truth (final snapshot).

We distribute the textual portion of our OCR19thSAC corpus in three different versions:

1. Complete multilingual corpus without filtering, page-wise aligned with the scan images. Provided in two variants: text only and text plus image coordinates of word boundaries. The latter is suited for extracting training material for an OCR engine which often requires image snippets of lines and their corresponding ground truth text.
2. Corpus with page-wise filtering (as described in Section 3.1), paragraph-wise aligned across snapshots; suitable for training a post-correction system.
3. Same as 2, but with additional paragraph-wise filtering, that is, without paragraphs that changed more than 10 % (measured in characters) between the two snapshots, also suitable for post-correction training.

All versions are provided as UTF-8 encoded plain text for both snapshots, that is OCR output quality and crowd-corrected quality, under a Creative Commons Attribution 4.0 International License.²⁹

Although many projects are involved in the digitization of heritage text material and OCR correction, surprisingly few OCR datasets are available in a suitable form for training and testing. In the course of the *IMPACT* project (Tumulla, 2008), a collection of ground-truth texts was created from digitized historical printed texts. The resource is advertised on the *Impact Centre of Competence's* website,³⁰ however, it is only accessible to members.

The open-source OCR framework *OCROPUS* (Breuel, 2008) could be trained with our resource. More recently, Breuel, Ul-Hasan, Al-Azawi, and Shafait (2013) have shown that an OCR system based on LSTM neural networks is able to outperform commercial

²⁹ See <http://pub.cl.uzh.ch/purl/OCR19thSAC> for download

³⁰ <http://www.digitisation.eu/tools-resources/image-and-ground-truth-resources/>

systems even with a rather small training set. Berkeley’s GPU-enabled state-of-the-art historical OCR system *Ocular* (Berg-Kirkpatrick & Klein, 2014)³¹ can easily be trained for documents with simple one-column book-like layouts and performs substantially better on historical data than commercial off-the-shelf systems.

4 Conclusion

We have shown that interested volunteers can effectively solve annoying OCR quality problems for the scientific community. In our case we were able to recruit volunteers from an inherently interested community that additionally has a long tradition of citizen science. Other success factors that we consider relevant for projects like ours are: (a) simple and concise guidelines, (b) an easy to use user interface with intuitive user interactions, (c) visual aids for quickly moving between the textual representation and its location in the facsimile image, (d) support for different ways of accessing the text and detecting possible errors, for instance, by reading sequentially or by investigating search results, (e) constant feedback about the correction progress on the level of validated pages, (f) personal correction statistics and high score rankings. The latter is needed in order to keep motivation up for the top volunteers who typically show an incredible amount of dedication to the task. The achieved accuracy of 99.93% on character-level comes close to the performance of double-keying transcription methods.

Acknowledgements

We would like to thank our volunteer correctors who invested a lot of time and engagement into our crowd-correction project. We also thank Adrian Althaus and Matthias Fluor for implementing the Kokos web platform. SC has been supported by the Swiss National Science Foundation under grant CRSII5_173719.

References

- Berg-Kirkpatrick, T., & Klein, D. (2014). Improved typesetting models for historical OCR. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)* (pp. 118–123). Association for Computational Linguistics. Retrieved from <http://aclweb.org/anthology/P14-2020> doi: 10.3115/v1/P14-2020
- Breuel, T. M. (2008). The OCRopus open source OCR system. In *Proc. spie* (Vol. 6815, p. 68150F-68150F-15). Retrieved from <http://dx.doi.org/10.1117/12.783598> doi: 10.1117/12.783598
- Breuel, T. M., Ul-Hasan, A., Al-Azawi, M. A., & Shafait, F. (2013, Aug). High-performance OCR for printed English and Fraktur using LSTM networks. In *2013*

³¹<http://nlp.cs.berkeley.edu/projects/ocular.shtml>

- 12th international conference on document analysis and recognition (p. 683-687). doi: 10.1109/ICDAR.2013.140
- Carrasco, R. C. (2014). An open-source OCR evaluation tool. In *Proceedings of the first international conference on digital access to textual cultural heritage (DATeCH '14)* (pp. 179–184). New York, NY, USA: ACM. doi: 10.1145/2595188.2595221
- Chronos, O., & Sundell, S. (2011). Digitalkoot: Making Old Archives Accessible Using Crowdsourcing. In *Proceedings of the 2011 AAAI Workshop on Human Computation* (pp. 20–26). Association for the Advancement of Artificial Intelligence (AAAI). Retrieved from <http://cdn3.microtask.com/assets/download/chronos-sundell.pdf>
- Deutsches Textarchiv – Qualitätssicherung.* (2016). Retrieved from <http://www.deutschestextarchiv.de/dtaq/>
- Dunning, T. (1994). *Statistical identification of language* (Tech. Rep. No. CRL MCCS-94-273). Computing Research Lab, New Mexico State University.
- Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5), 378–382. doi: 10.1037/h0031619
- Geyken, A., & Gloning, T. (2015). A living text archive of 15th-19th-century German corpus strategies, technology, organization. In J. Gippert & R. Gehrke (Eds.), *Historical corpora. challenges and perspectives* (p. 165-179). Tübingen: Narr.
- Göhring, A., & Volk, M. (2011). The Text+Berg Corpus: An Alpine French-German Parallel Resource. In *Proceedings of Traitement Automatique des Langues Naturelles (TALN 2011)*. Montpellier. Retrieved from http://www.atala.org/taln_archives/TALN/TALN-2011/taln-2011-court-017
- Haaf, S., Wiegand, F., & Geyken, A. (2013, March). Measuring the correctness of double-keying: Error classification and quality control in a large corpus of TEI-annotated historical text. *Journal of the Text Encoding Initiative [Online]*, 4. Retrieved from <http://jtei.revues.org/739> doi: 10.4000/jtei.739
- Haaf, S., Wiegand, F., & Geyken, A. (2014, March). The dta "base format": A tei subset for the compilation of a large reference corpus of printed text from multiple sources. *Journal of the Text Encoding Initiative [Online]*, 8. Retrieved from <http://jtei.revues.org/1114> doi: 10.4000/jtei.1114
- Holley, R. (2009a). How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs. *D-Lib Magazine*, 15(3/4). Retrieved from <http://www.dlib.org/dlib/march09/holley/03holley.html>
- Holley, R. (2009b). *Many hands make light work: Public collaborative OCR text correction in Australian historic newspapers*. National Library of Australia.
- Holley, R. (2010). Crowdsourcing: How and why should libraries do it? *D-Lib Magazine*, 16(3/4). Retrieved from <http://dx.doi.org/10.1045/march2010-holley> doi: 10.1045/march2010-holley
- Holley, R. (2013). *Crowdsourcing text correction and transcription of digitised historic newspapers: a list of sites*. Retrieved 2016/01/05, from <http://rose-holley.blogspot.ch/2013/04/crowdsourcing-text-correction-and.html>

- Kettunen, K. (2016). Keep, change or delete? setting up a low resource OCR post-correction framework for a digitized old Finnish newspaper collection. In D. Calvanese, D. De Nart, & C. Tasso (Eds.), *Digital libraries on the move: 11th italian research conference on digital libraries, iredl 2015, bolzano, italy, january 29-30, 2015, revised selected papers* (pp. 95–103). Cham: Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-41938-1_11 doi: 10.1007/978-3-319-41938-1_11
- Koskeniemi, K., & Haapalainen, M. (1996). GERTWOL – Lingsoft Oy. In R. Hausser (Ed.), *Linguistische Verifikation: dokumentation zur ersten Morpholympics 1994* (pp. 121–140). Tübingen: Niemeyer.
- Levenshtein, V. I. (1966, February). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10(8).
- Long, F. A. (1993). Electronic composition and the typesetter. *American Journal of Economics and Sociology*, 52(2), 223–226. Retrieved from <http://dx.doi.org/10.1111/j.1536-7150.1993.tb02536.x> doi: 10.1111/j.1536-7150.1993.tb02536.x
- Piotrowski, M. (2012). *Natural language processing for historical texts* (Vol. 5) (No. 2). Morgan & Claypool. Retrieved from <http://dx.doi.org/10.2200/S00436ED1V01Y201207HLT017> doi: 10.2200/S00436ED1V01Y201207HLT017
- Reffle, U. (2011). Efficiently generating correction suggestions for garbled tokens of historical language. *Natural Language Engineering*, 17(2), 265–282. Retrieved from http://www.journals.cambridge.org/abstract_S1351324911000039
- Rice, S. V. (1996). *Measuring the Accuracy of Page-Reading Systems* (Doctoral dissertation, University of Nevada, Las Vegas). Retrieved from <http://www.cs.olemiss.edu/~rice/rice-dissertation.pdf>
- Scientific library services and information systems (LIS): DFG practical guidelines on digitisation.* (2009). electronic. Retrieved from http://www.dfg.de/download/pdf/foerderung/programme/lis/praxisregeln_digitalisierung_en.pdf
- Seidman, M. J., Flanagan, M., Rose-Sandler, T., & Lichtenberg, M. (2016, jul). Are games a viable solution to crowdsourcing improvements to faulty OCR? – the purposeful gaming and BHL experience. *code4lib*, 33. Retrieved from <http://journal.code4lib.org/articles/11781>
- Simeoni, M. M. J.-A., Mazzei, A., & Kaplan, F. (2014). *Semi-automatic transcription tool for ancient manuscripts*. IC Research Day 2014: Challenges in Big Data, SwissTech Convention Center, Lausanne, Switzerland, June 12, 2014. Retrieved from <https://infoscience.epfl.ch/record/199578/files/Semi-Automatic%20Transcription%20Tool%20for%20Ancient%20Manuscripts%20-%20The%20Venice%20Atlas.pdf>
- Tumulla, M. (2008, July). IMPACT: Improving Access to Text. *Dialog mit Bibliotheken*, 20(2), 39–41. ((German article))
- Vobl, T., Gotscharek, A., Reffle, U., Ringlstetter, C., & Schulz, K. U. (2014). 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 DATeCH '14* (pp. 57–61). ACM Press. Retrieved from

<http://dl.acm.org/citation.cfm?doid=2595188.2595197>

- Volk, M., & Clematide, S. (2014, oct). Detecting code-switching in a multilingual Alpine heritage corpus. In M. Diab, J. Hirschberg, P. Fung, & T. Solorio (Eds.), *Proceedings of the first workshop on computational approaches to code switching* (p. 24-33). Doha, Qatar: Association for Computational Linguistics. Retrieved from <http://www.aclweb.org/anthology/W14-39>
- Volk, M., Marek, T., & Sennrich, R. (2010, August). Reducing OCR Errors by Combining Two OCR Systems. In *ECAI 2010 Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities* (pp. 61–65). Retrieved from <http://dx.doi.org/10.5167/uzh-35259>
- von Ahn, L., Maurer, B., McMillen, C., Abraham, D., & Blum, M. (2008). reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, 321(5895), 1465–1468. Retrieved from <http://science.sciencemag.org/content/321/5895/1465> doi: 10.1126/science.1160379
- Wang, S., Wang, M., & Chen, K. (2013). Boosting OCR accuracy using crowdsourcing. In *Human computation and crowdsourcing: Works in progress and demonstration abstracts, an adjunct to the proceedings of the first AAAI conference on human computation and crowdsourcing, november 7-9, 2013, palm springs, ca, USA* (Vol. WS-13-18). AAAI. Retrieved from <http://www.aaai.org/ocs/index.php/HCOMP/HCOMP13/paper/view/7538>

Supervised OCR Error Detection and Correction Using Statistical and Neural Machine Translation Methods

Abstract

For indexing the content of digitized historical texts, optical character recognition (OCR) errors are a hampering problem. To explore the effectivity of new strategies for OCR post-correction, this article focuses on methods of character-based machine translation, specifically neural machine translation and statistical machine translation.

Using the ICDAR 2017 data set on OCR post-correction for English and French, we experiment with different strategies for error detection and error correction. We analyze how OCR post-correction with NMT can profit from using additional information and show that SMT and NMT can benefit from each other for these tasks. An ensemble of our models reached best performance in ICDAR's 2017 error correction subtask and performed competitively in error detection.

However, our experimental results also suggest that tuning supervised learning for OCR post-correction of texts from different sources, text types (periodicals and monographs), time periods and languages is a difficult task: the data on which the MT systems are trained have a large influence on which methods and features work best. Conclusive and generally applicable insights are hard to achieve.

1 Introduction

Optical character recognition (OCR) is an important processing step for text digitization, especially with the growing interest in digital humanities. Unfortunately, the output of OCR systems for historical documents is often faulty due to both orthographic and typographic variation as well as due to poor condition of the source material (especially for newspapers). Therefore, it is necessary to take measures to improve the quality of existing OCR-generated text if re-OCRing of the image material is not an option. Recently natural language processing (NLP) tasks have profited from neural methods. Neural machine translation (NMT) outperformed statistical machine translation (SMT), thus, NMT now supersedes SMT in research and practice. Since string-to-string translation methods have been used to correct OCR errors for a long time, it is interesting to explore how character-based NMT can be employed for this task.

This paper focuses on two questions: How does character-based NMT perform compared to character-based SMT in the case of OCR post-correction? How can these approaches be improved by using more information during training and translation?

Note that our historical data we are working with represents the task of correcting OCR output into the spelling found in the original documents and does not include any modernization of the texts. The rest of this article is structured as follows: In the next section, we discuss previous work relevant to OCR post-correction, neural modeling of related NLP tasks and character-based MT. Section 3 describes the data used in our experiments and Section 4 presents the methods. Section 5 gives details on the results and discusses the major findings. In Section 6, we offer our ideas for future work and finally, we conclude in Section 7.

2 Related Work

2.1 OCR Post-Correction

Volk et al. (2011) discuss three possible ways to tackle the problem of OCR errors: Modifying the input images, altering the OCR system, or post-processing the output text. Since it does not involve re-OCRing, a considerable number of approaches have focused on the last option. Kukich (1992) discusses various automatic methods to find and correct errors, such as n-gram or dictionary-based techniques. Eger et al. (2016) compare traditional spelling error correction techniques with general string-to-string translation methods, and evaluate on an OCR data set. They show that the latter methods achieve significantly better results than (weighted) edit distance or the noisy channel model (Brill and Moore, 2000).

Based on the idea that OCR post-correction can be cast as a translation task, Afli et al. (2016) have successfully trained an SMT system to translate historical French texts with OCR errors into corrected text. Their model outperforms language model based techniques. However, they use a large training set of more than 60 million tokens. This exceeds by far the size of the ICDAR training sets used in our experiments. In previous experiments, Afli et al. concluded that word-level SMT systems perform slightly better than character-level systems for OCR post-correction (Afli et al., 2015). The size of the training set there was over 90 million tokens.

2.2 OCR Correction of Historical Texts

The introductory book by Piotrowski (2012) resumes the problems of OCR, historical spelling and correction. The TICCL system by Reynaert (2016) is an unsupervised corpus-based approach to OCR post-correction that has been developed over many years and works for several languages. However, it requires high-quality lexicons in order to be effective. Silfverberg et al. (2016) present an interesting supervised approach for historical Finnish OCR correction of isolated words that formulates the problem as a sequence labeling task and uses weighted finite-state transducer techniques to implement it. The official ICDAR OCR post-correction paper (Chiron et al., 2017) contains concise descriptions of the different approaches that the shared task participants used for their solutions.

The recent paper of Schulz and Kuhn (2017) presents a complex architecture for OCR post-correction of historical texts that includes token and character-level SMT as well as specialized tools for merging and splitting of erroneous tokens. Their related work section gives a good overview on older and newer approaches. Due to space restrictions and our focus on experimenting and evaluation, we refer the interested reader to their discussion.

2.3 Neural Networks for NLP Tasks

Recently, neural networks have been given much attention in the field of computational linguistics. Especially the work of Sutskever et al. (2014) has prompted much research which focused on employing sequence-to-sequence (seq2seq) neural networks in various NLP tasks. Examples include effectively using neural networks for transliteration (Rosca and Breuel, 2016), for grapheme-to-phoneme conversion (Yao and Zweig, 2015), for historical spelling normalisation (Bollmann and Søggaard, 2016) and language correction of second language learners (Xie et al., 2016).

All of these tasks can (at least partly) be viewed as string-to-string translation problems and are, thus, closely related to OCR post-correction. The models were all successful, performing at least equally but mostly better than traditional NLP methods such as methods based on (weighted) edit distance (Xie et al., 2016).

However, Schnober et al. (2016) express their doubt whether neural networks are already at the point where they can entirely replace traditional approaches. They evaluated the performance of encoder-decoder neural networks against established methods for spelling correction, OCR post-correction, grapheme-to-phoneme conversion and lemmatisation. Some of the string-to-string translation systems they used as baselines were also evaluated by Eger et al. (2016). It is particularly interesting to see how neural network approaches performed compared to these models which were previously shown to exceed (weighted) edit distance and other techniques.

In the experiments of Schnober et al. on OCR post-correction, the neural network systems did not manage to outperform a system built with Pruned Conditional Random Fields on a training set of 72,000 misrecognized words. Despite these negative results, the work gives valuable insight into model selection for neural network approaches. Attention-based models (Bahdanau et al., 2015) show significant improvements over plain seq2seq models. On a smaller training set, the attention-based models also performed better than the Pruned Conditional Random Fields. In contrast to the experiments conducted by Schnober et al., our data does not only contain misrecognized words. Therefore, the task becomes harder since the systems also need to detect erroneous words before trying to correct them.

2.3.1 Character-based NMT

NMT (Koehn, 2017) needs a fixed vocabulary to generate fixed-size vectors as input to the neural network. The larger this vocabulary is, the less unknown words occur, but the longer it takes to train the model and to apply it. To address this drawback,

Language	Monograph		Periodical	
	fr	en	fr	en
Total # of Tokens in Ground Truth	597967	624678	255527	249483
Total # of Tokens in Original OCR Output	604444	649604	268742	274413
Erroneous Tokens	6.34%	6.22%	11.89%	12.11%

Table 1: Overview of the training set: Percentage of erroneous tokens (relative to the number of tokens in the OCR output using whitespace tokenization).

many approaches have been proposed to design open vocabulary NMT systems. These range from simple dictionary lookup techniques (Luong et al., 2015) over integrating SMT features (He et al., 2016) to specific design choices for the NMT architecture itself. Luong and Manning (2016) use a hybrid-system which translates primarily on word-level and falls back to a character-based representation for OOV words. Similarly, Sennrich et al. (2016) propose to translate on subword-level. They use byte pair encoding to create a vocabulary that is built bottom up, starting with characters and then adding larger subwords made from already known entries. The resulting vocabulary will contain characters, subwords as well as whole words.

Chung et al. (2016) designed a character-based decoder without explicitly segmenting the character sequences to match words. They motivate their approach with the following arguments: There is always a risk of introducing artificial errors when sentences are explicitly segmented into words. Furthermore, a character-based model will not suffer as much from data sparsity since stem forms and affixes can be treated separately. Finally, the system has a much better chance at generalization for unseen word forms. Extending this idea, Lee et al. (2017) aimed to design a fully character-based NMT system without explicit segmentation. They observe that their model is capable of locating spelling errors and still producing the correct translation. This finding is shared by Zhao and Zhang (2016).

Motivated by the mentioned approaches and their outcomes, we employ character-based NMT for post-correcting OCR errors. Since character-based NMT proved to handle spelling errors we assume that it will also perform well on OCR errors.

3 ICDAR 2017 OCR Post-correction Data

Our experimental data comes from the OCR post-correction Shared Task¹ of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR 2017) (see (Chiron et al., 2017) for more information). The data set is a subpart from a corpus

¹<https://sites.google.com/view/icdar2017-postcorrectionocr/home>

Language	Monograph		Periodical	
	fr	en	fr	en
Total # of Chars in Ground Truth	3560500	3592543	1637087	1574796
Total # of Chars in Original OCR Output	3569285	3592763	1640237	1575613
Insertions	0.32%	0.51%	0.68%	0.63%
Deletions	0.57%	0.52%	0.87%	0.68%
Substitutions	0.75%	0.78%	1.88%	2.46%
Errors Total	1.64%	1.81%	3.43%	3.77%
Unrecognizable	0.42%	3.32%	5.27%	9.33%

Table 2: Edit operations needed to correct training set: Percentage of total characters (relative to the characters in the OCR output) that need to be inserted, deleted, or substituted.

that was built in the AmeliOCR project². The documents in the corpus originate from different digital collections and vary in terms of their condition and date of origin as well as the OCR engine and the post-correction initiative used to create the corpus (e.g. project-internal correction or external projects such as Gutenberg or Wikisource).

The equally sized English and French data consists of 12M OCRred characters and their alignments to a ground truth (GT)³. The training set has 10M characters (83%) and the official test set 2M (17%). The data is distributed as raw text files (one paragraph per file) where the first line contains the OCR output. On the second line, the OCRred text is vertically aligned with the GT. Wherever a character has to be inserted to match the length of the GT (which means there is no possible alignment) an "@" is inserted. On the third line that contains the GT, an "@" is inserted for all characters that appear in the OCR output but cannot be aligned with a GT character. The following example from 1860 illustrates the format:

- (1) ATRAVELLER STOPPED AT A WIDOW'S GATE. [OCR]
 A@TRAVELLER STOPPED AT A WIDOW'S GATE. [OCR aligned]
 A TRAVELLER STOPPED AT A WIDOW@S GATE. [GT aligned]

Additionally, "unrecognizable" character sequences that cannot be identified with certainty in the original image are aligned with the "#" character in the GT. Probably due to different processing of soft hyphens, the GT for hyphens is very inconsistent. Following the decision of the shared task organizers, we ignore all hyphens in evaluations and error analyses, however, we keep them in the training data.

²<https://bit.ly/2BLsN7B>

³We use terms "gold standard" and "ground truth" interchangeably.

	Monograph		Periodical		fr		en		both	
	fr	en	fr	en	form	freq	form	freq	form	freq
1.	f → s	l → l	t → l	fi → fî	f → s	2346	l → l	2279	f → s	2653
2.	u → v	é → e	, → .	b → h	é → e	870	b → h	1250	l → l	2310
3.	é → e	U → ll	e → é	u → n	e → é	790	fi → fî	993	é → e	1647
4.	i → j	e → é	o → e	- → -	u → v	740	u → n	898	b → h	1338
5.	v → u	h → b	. → ,	ff → f̄f	t → l	677	é → e	777	u → n	1320
6.	e → é	b → h	é → e	f → f̄	, → .	639	- → -	720	. → ,	1225
7.	l → t	d → ll	i → l	o → e	l → t	562	ff → f̄f	712	, → .	1214
8.	c → e	e → c	a → e	li → h	. → ,	535	c → e	701	e → é	1209
9.	l → !	f → s	u → n	c → e	o → e	521	. → ,	690	c → e	1157
10.	è → e	' → e	l → t	. → ,	i → j	504	li → h	666	o → e	1110

Table 3: 10 most common substitutions of characters ngrams (excluding hyphens). $x \rightarrow y$ means x was recognized instead of y .

The data set is difficult since the documents (a) origin from different collections (the BnF and the British Library), (b) were published either in periodicals or monographs, (c) cover a large time span (1654 to 2000). 92% of the documents with a known publication date are from the 19th century.

Diachronic document collections pose several challenges for OCR post-correction. OCR quality is generally lower for historical documents, typically worse for documents typeset in black letter fonts than in antiqua fonts. Additionally, old and modern spellings coexist, for instance, *compleated* (1744) vs *completed* (1894). Some characters as long s “f” disappear over time.

Tables 1 and 2 summarize the size of our training data, the proportion of “unrecognizable” characters, and the edit operations needed for error correction. Overall, periodical texts need more correction operations, as OCR quality for these texts is worse than for monographs. The better the OCR quality is, the harder it is to improve it with post-correction. Consequently, it will be easier to correct the periodical texts than the monographs. Note that English texts contain many more unrecognizable characters than the French texts.

Statistics on character level Table 3 shows the ten most frequent substitutions over character n-grams aggregated according to different text criteria. The substitutions are derived from the precomputed alignments of the data, and most of them can be explained due to the visual similarity of the glyphs. Note that the frequencies of the substitutions are not only specific to language but also to text type. Additionally,

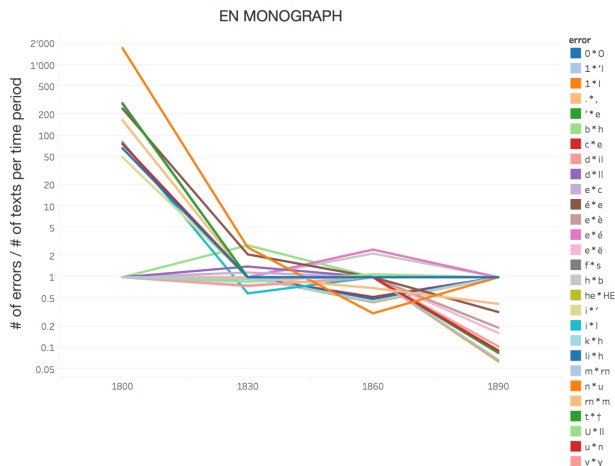


Figure 1: Changes of frequencies of the most frequent error types over time in English monograph data sampled in buckets of 30 years.

error frequencies also differ due to font type, paper quality and other document-related characteristics. However, since the source images of the data set were not provided, the influence of these factors on error frequencies cannot be assessed here. The confusion pair “e/é” in English is related to code-switching.

How do error types develop over time? Figure 1 is a screenshot from an interactive visualization of the frequency of the ten most frequent errors per 30 years in English monographs. The y axis is logarithmic and shows the number of errors normalized by the number of files for the specific time period. It is not essential to know which error belongs to which line. Instead, it is interesting to see the error development, for instance, the most frequent error from 1800 appears 1000 times less in 1830. Many errors in early documents occur rarely in later periods. Even though not all data sets show such extreme differences between the periods, similar observations can be made. Thus, it can be concluded that the different time periods all offer distinct challenges for OCR post-correction.

Statistics on word level 87.2% (en) and 90.2% (fr) of the wrong tokens only appear once in the training set (see Table 4). These numbers are computed by excluding any punctuation characters, e.g. a comma at the end of a word. The large percentage of

	fr			en		
	OCR	GT	Freq	OCR	GT	Freq
1.	font	sont	470	1	l	2245
2.	a	à	331	tbe	the	513
3.	l	!	286	thé	the	510
4.	d	de	167	tlie	the	303
5.	!.	!...	164	aud	and	211
6.	do	de	145	l	'l	195
7.	ta	la	139	tho	the	188
8.	ie	je	129	he	be	172
9.	vn	un	116	hut	but	165
10.	dé	de	101	ail	all	165
all			51699			47583
hapax legomena			46644			41487

Table 4: Distribution of most frequent word errors

hapax legomena suggests that supervised post-correction on character level will be more beneficial than on word level due to data sparsity issues.

Table 4 also shows the ten most frequently misspelt words. Both datasets contain frequently occurring real word errors that cannot be corrected in isolation. Considering the Zipfian distribution of words, it is probably not surprising that four OCR error types of “the” are among the most frequently misrecognized English words. The mean of spelling versions per misrecognized GT type is 1.7 for English and 1.77 for French. The English word “the” has the highest number of spelling variants (275 unique types), for French it is the word “de” (214 types). Again, these findings suggest that a character-based approach is suitable for our task.

4 Methods

4.1 Experimental Setup

In order to keep the official test data untouched while exploring many different experimental settings, we randomly split the official training data in an internal training (80%), development (10%) and test set (10%). Unrecognizable characters are excluded from our data. Table 5 characterizes the resulting data sets.

Our baseline approach for character-level OCR correction preprocesses the data into a verticalized text format as follows: Whenever two space characters are aligned between the OCR output and the GT, a newline is inserted. After each actual text character

	Periodical		Monograph		Combined		
	fr	en	fr	en	fr	en	both
Train	1320728	1268194	2862351	2887538	4183079	4155732	8338811
Dev	165449	158748	360090	362068	525539	520816	1046355
Test	165032	158492	356580	360228	521612	518720	1040332

Table 5: Character counts of internal train, dev and test sets (including whitespace and hyphens) for all four document types. The combined data sets contain periodicals and monographs.

(@ can be ignored), we add a space to force the MT systems to translate on character level. Example 2 illustrates the conversion into the common line-based training format of MT systems. Having one word per line makes the training of the MT model simple, however, it results in a context-insensitive approach.

- (2) `This@course@was agreed to, [...]` Aligned OCR output
`This course was agreed to, [...]` Aligned GT

OCR (Source)	GT (Target)
<code>T h i s c o u r s e w a s a g r e e d t o ,</code>	<code>T h i s _ c o u r s e _ w a s a g r e e d t o ,</code>

Character-based SMT We train a baseline character-based SMT model analogous to (Pettersson et al., 2013) using the Moses toolkit (Koehn et al., 2007), GIZA++ character alignment (Och and Ney, 2003), and MERT optimization (Och, 2003). We use a 10-gram language model in all experiments. For a more extensive introduction to SMT, please refer to Koehn (2009).

Character-based NMT For a more in-depth and general introduction to NMT, we refer to Koehn (2017).

We first tested the convolutional framework described in Lee et al. (2017), which is a character-level NMT system by design. However, as early experiments did not show satisfactory results we discarded this option.

Nematus⁴ is a high-performing NMT framework (Sennrich et al., 2017). Character-level MT can be enforced by using the same whitespace insertion “trick” in the data format as for Moses. Initial experiments showed positive results and we relied on this framework for all subsequent experiments.

⁴<https://github.com/EdinburghNLP/nematus>

Nematus – as most neural translation frameworks – has many hyperparameters that influence its performance. Given training times of several hours even on fast GPUs, it is not feasible to tune these parameters systematically for each data set. We decided to explore them on the French periodical data for two reasons: First, as one of the smaller data sets it is faster to train, and, second, prior SMT experiments had shown that this set is harder to correct than others. In the following, we quickly introduce our base hyperparameter settings that we determined by experiments on the French periodicals.

Input or output embeddings are low-dimensional dense continuous vector representations of one-hot encoded categorical data with typically much larger dimensionality. The dimensionality of characters is orders of magnitude lower than that of words. We choose an embedding size of 256, tested against 32 and 512. We configure them as tied embeddings, meaning that the embedding of a character on the source side is the same as on the target side. This seems reasonable given that both sides are the same language apart from OCR errors. According to our experiments, not using tied embeddings achieves better results in error detection but worse results in error correction. Eventually, we decided to use tied embeddings as they speed up training.

Dropout is a useful regularization method for neural architectures. During training, some units in the network are switched off. Consequently, the model does not have access to all its information and is less prone to overfit. The dropout rate specifies how many units are switched off at the same time. We set it to 0.2.

Batch size refers to the number of training examples which are used to compute the gradient and update the weights in the network. A larger batch size leads to a speed-up in training, but it can also have a negative impact on learning. We set it to 100, tested against 50 and 200.

For a few hyperparameters, we use the default values from Nematus: hidden layer size (1000), optimizer (adadelata), gradient clipping threshold (1) and learning rate (0.0001). Furthermore, we set the maximum sequence length to 23 for all context-insensitive experiments and to 53 for context-sensitive experiments. These limits cover 99.99% of all training examples.

4.2 Using More Training Material

In MT, the most straightforward method to increase translation quality is by adding more training material. Since OCR errors mostly occur due to the visual similarity of characters, the same errors occur in periodicals as in monographs and also in both languages. Therefore, it makes sense to combine the training sets of the individual text types (henceforth, “medium” data set size), maybe also combining all the available data across languages (henceforth, “large” data set). The latter especially because we noticed code switching effects in Section 3.

4.3 Using More Context

Some OCR errors introduce wrong tokens that are valid words of the document language. For example, in Table 4 the most frequent French OCR error on word-level is “sont”

wrongly recognized as “font” (probably because of the long s glyph). However, “font” cannot be corrected in isolation since it is a frequent French word. We need some context in order to change it appropriately. Our context-sensitive format with two preceding and one succeeding words with respect to a focus word inbetween is shown in Example 4.3.

- (3) Si ces principes font fondés sur le goût Aligned OCR
 Si ces principes sont fondés sur le goût Aligned GT

OCR (Source)
S i # c e s # p r i n c i p e s # f o n t
c e s # p r i n c i p e s # f o n t # f o n d é s
p r i n c i p e s # f o n t # f o n d é s # s u r
GT (Target)
S i # c e s # p r i n c i p e s # f o n t
c e s # p r i n c i p e s # s o n t # f o n d é s
p r i n c i p e s # s o n t # f o n d é s # s u r

An artificial word boundary marker which otherwise does not occur in the training data needs to be chosen.⁵ Alternatively, we could have chosen to translate only the focus word, however, forcing the MT system to produce the context on the target side as well, produced better results. This representation increases the training material, which is good, but unfortunately also extends training time.

4.4 Factored Character-based NMT

Nematus supports “factored” NMT models (Sennrich and Haddow, 2016) where structured information can be included on the source side: for instance, the time span from which the text originates, or its text type, or even its language if we want to merge both languages in order to have more training material.⁶ In Nematus, factors are implemented as embedding vectors that are concatenated with the input character embeddings.⁷

Factors can be conveniently expressed in the input format. Example 4 with a text snippet from English periodicals between 1800 and 1849 shows the context-insensitive input format with factors. The symbol | marks the beginning of a factor. The first factor refers to the text type, and the second stands for the time span⁸ when the text was written. This “feature” can be helpful to model time-dependent OCR errors which occur due to orthographic or typographical changes as illustrated in Figure 1.

- (4) who may wish Aligned OCR
 who may wish Aligned GT

⁵If the boundary marker does not appear 3 times on the translated target side, we select the word with the smallest edit distance to the input focus word as its translation.

⁶OCR errors for French and English can be similar due to their common Latin script system.

⁷Due to a problem in Nematus, we could not use tied embeddings in combination with factors.

⁸Split in bins of 50 years and represented by YYYY//50 (integer division).

OCR (Source)	GT (Target)
w peri 36 h peri 36 o peri 36	w h o
m peri 36 a peri 36 y peri 36	m a y
w peri 36 i peri 36 s peri 36 h peri 36	w i s h

4.5 Glyph Embeddings

State-of-the-art NMT often uses pre-trained distributional word embeddings, which mainly put words closer together in the vector space if they share the same contexts. For OCR post-correction, characters do not necessarily need to be substituted with characters that often share the same context, but rather share similar character shapes. Therefore, we generate pre-trained embeddings that express some visual similarity between characters. As no source images are available for our data, we simply use 16 by 16 grey-scale pixel values of every character from the Helvetica font as a proxy.⁹

Of course, there are better ways how visual information can be included in an NMT system. However, this is a straightforward and time-saving technique and is therefore used in this experiment.

4.6 Error-focused Models

Most of the OCR output is simple to process for an MT system since it is already correct and does not need to be translated. This means that for the most part the NMT system just learns how to copy characters from the source to the target side. If the system is trained on a data set which contains a larger proportion of errors, it will become better at detecting errors and will do this more aggressively at the cost of over-correction. However, these systems might over-correct. Therefore, it is essential to determine a reasonable amount of non-error tokens in the training data to boost the error detection recall but not affecting precision too much. For our experiments, we first tried a split of 75% error and 25% non-error tokens. Since there are fewer errors in the monographs, we adjust to 50% errors for French monograph and 37.5% for the English. We randomly filter out examples without errors in this subsampling process.

4.7 Ensemble Decoding

Ensemble decoding is a common technique in NMT where the individual probability distributions of different models are averaged. The rationale is that the biases of single models' outputs even out by the combination of several models. With Nematus it is possible to do ensemble decoding by using either models from the same training run at different time steps or multiple models. For our experiments, we try both. In order to distinguish them, the former will be called "single ensemble" (using different time stamp models of the same system) and the latter "multi ensemble" (using the best model of different systems). All experiments with ensemble decoding are conducted on

⁹The values are normalized into a range between -1 and 1 and then scaled by 0.01 as done in Nematus for randomly initialized values.

the context-sensitive input formats. Single ensembles are tested for the base context-sensitive NMT system and the error-focused NMT system. For single ensembles, we combine the best model with the models from two previous time stamps. Multi ensembles use the base context-sensitive NMT system, the one with glyph embeddings and the error-focused system.

4.8 System Combinations

The final method explored in our experiments is combining the outputs of multiple MT systems. The output from such post-translation combination of systems was submitted to the ICDAR shared task.

Different strategies are used for error detection and error correction. Keep in mind that for the error correction test set, the shared task organizers published the positions of erroneous tokens. Therefore, we translate all data for error detection, however, for the error correction we only translate the erroneous tokens.

A flow diagram of our decision procedure is shown in Figure 2. The error detection algorithm uses the output of five MT systems for a specific data set which worked best in combination as tested on the dev set. We have four conditions which trigger the error detection. First, if the model with the smallest Levenshtein distance proposes a change, we assume there is an error. Second, we check the five best systems, and if the most frequently proposed token is different from the OCR output, we also assume an error. Third, if the original OCR token does not occur in the GT training and dev sets for both text types, we assume an error. Finally, if the current token and one of its neighbors (both can be translated or untranslated) do not occur in the training set, but their concatenation does, we assume that the tokens have to be merged.

The algorithm for error correction works slightly different. The five models with the smallest Levenshtein distance on the dev set are used to generate correction candidates. Since the evaluation script for the shared task evaluates two scenarios, a “fully-automated” one where only one correction candidate is given and a “semi-automated” one where a ranked list of candidates with confidence scores is given, we pay special attention to our choice of candidates. If the system with the smallest Levenshtein distance on the dev set suggests a correction, we take this as an exclusive correction candidate. Otherwise, we look at the candidates of all five systems and model the weights according to the frequency distribution of their suggestions which are different from the OCR output.

5 Results and Discussion

5.1 Evaluation Setup

All configurations are evaluated on our internal test set by macro-averaging the scores of three individual models using the same configuration. This reduces effects caused by system variance. Error detection is measured on word level using the following measures:

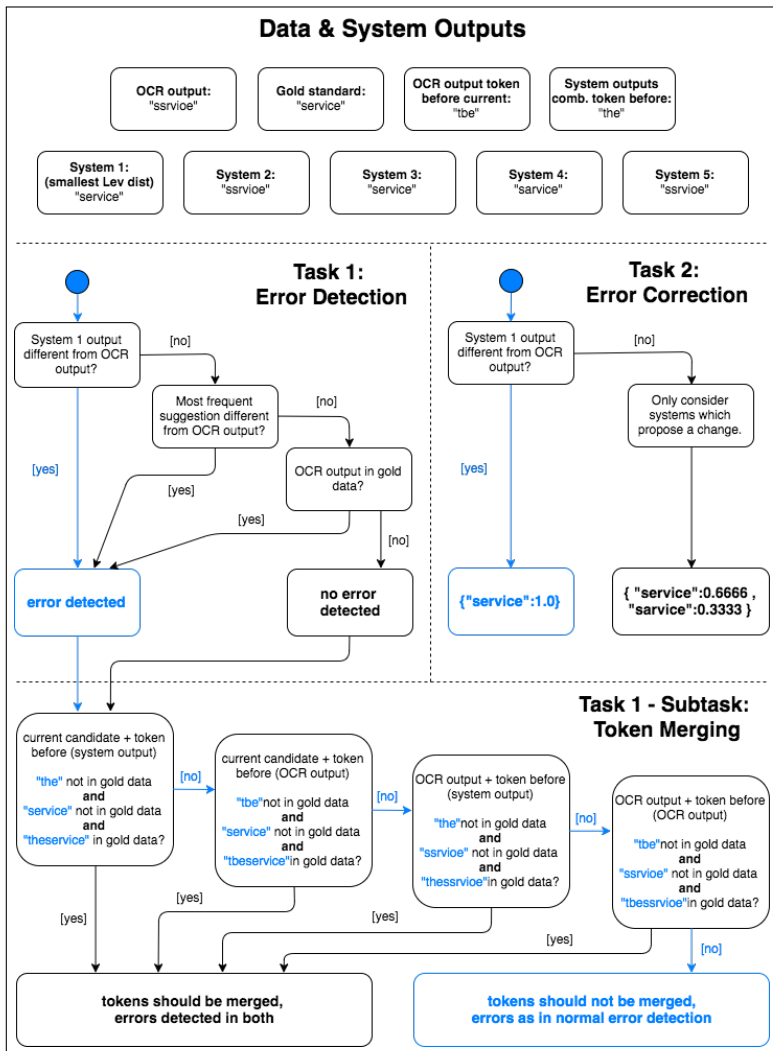


Figure 2: Algorithms for error detection and correction explained with an example.

Supervised OCR Error Detection and Correction Using SMT and NMT

	Error Detection			Error Correction		
	P ↑	R ↑	F1 ↑	Lev. ↓	% Rel. Imp. ↑	% Correct ↑
English Periodicals						
OCR baseline	Char/Tok ER: 3.47% / 11.28%			0.1898	0%	-
SMT baseline *	83.82	61.84	71.17	0.1347	40.96	53.42
SMT medium	91.33	46.87	61.94	0.1461	29.96	57.28
SMT large	93.26	<i>40.51</i>	<i>56.47</i>	0.1510	25.71	56.78
NMT baseline	87.33	59.27	70.61	0.1472	28.95	49.00
NMT medium	90.60	46.60	61.53	0.1584	19.85	47.52
NMT large	91.47	42.51	58.04	<i>0.1593</i>	<i>19.13</i>	47.42
SMT context *	85.19	58.87	69.62	0.1339	41.83	58.98
NMT context *	89.31	59.56	71.46	0.1406	34.95	51.34
NMT time factor no context	87.38	60.27	71.33	0.1475	28.68	48.24
NMT time factor context	89.99	57.77	70.37	0.1420	33.65	51.38
NMT factor medium	88.82	58.94	70.85	0.1448	31.12	49.94
NMT factor large	89.32	57.28	69.79	0.1415	34.10	50.95
NMT glyph embeddings *	89.62	59.16	71.27	0.1422	33.44	51.17
NMT error-focused	<i>76.95</i>	68.01	72.20	0.1591	19.28	<i>41.09</i>
NMT single ensemble *	89.92	59.54	71.64	0.1388	36.73	52.22
NMT single ensemble error	78.41	68.28	72.99	0.1533	23.80	43.35
NMT multi ensemble *	89.20	60.79	72.30	0.1369	38.68	52.37
English Monographs						
OCR baseline	Char/Tok ER: 1.79% / 6.38%			0.0830	0%	-
SMT baseline *	92.84	33.92	49.68	0.0631	31.50	72.03
SMT medium	86.32	36.58	51.37	0.0649	27.84	64.52
SMT large	89.41	<i>32.84</i>	<i>48.03</i>	0.0652	27.30	68.56
NMT baseline *	91.77	35.50	51.19	0.0652	27.33	65.94
NMT medium	87.56	36.74	51.76	0.0668	24.18	61.54
NMT large	87.82	33.92	48.93	0.0678	22.45	62.52
SMT context *	88.85	39.07	54.27	0.0628	32.25	70.34
NMT context *	87.97	39.93	54.92	0.0632	31.27	65.38
NMT time factor no context	90.19	40.64	56.03	0.0668	24.26	58.39
NMT time factor context *	85.31	45.04	58.94	0.0644	28.77	61.06
NMT factor medium	92.33	39.66	55.48	0.0652	27.27	61.22
NMT factor large	91.62	39.24	54.95	0.0654	26.83	61.91
NMT glyph embeddings *	87.75	40.27	55.20	0.0631	31.46	65.73
NMT error-focused	50.51	50.81	50.60	0.0884	-6.01	35.80
NMT single ensemble	89.95	39.51	54.90	0.0625	32.87	67.39
NMT single ensemble error	<i>49.38</i>	51.70	50.43	<i>0.0888</i>	<i>-6.31</i>	<i>35.58</i>
NMT multi ensemble *	86.85	41.14	55.82	0.0627	32.38	65.78

Table 6: English macro-averaged experiments results. Best results are marked in blue, worst results in red. Asterisks mark indicates systems used for ICDAR submission. (ER = error rate)

	Error Detection			Error Correction		
	P ↑	R ↑	F1 ↑	Lev. ↓	% Rel. Imp. ↑	% Correct ↑
French Periodicals						
OCR baseline	Char/Tok ER: 3.41% / 11.85%			0.1930	0%	-
SMT baseline *	83.48	35.31	49.62	0.1656	16.54	47.39
SMT medium	86.93	38.00	52.87	0.1663	16.06	46.97
SMT large	87.18	35.37	50.31	0.1675	15.22	44.43
NMT baseline *	87.87	43.52	58.21	0.1700	13.53	39.25
NMT medium	88.49	41.75	56.72	0.1695	13.88	38.93
NMT large	88.01	40.07	55.06	0.1735	11.21	35.84
SMT context	81.75	40.00	53.71	0.1614	19.64	53.66
NMT context	88.53	44.26	59.01	0.1645	17.32	42.53
NMT time factor no context	85.81	44.58	58.66	0.1755	10.02	37.77
NMT time factor context *	87.78	46.74	61.00	0.1655	16.68	40.88
NMT factor medium	87.51	44.20	58.72	0.1699	13.64	39.09
NMT factor large	86.28	43.83	58.11	0.1704	13.30	39.24
NMT glyph embeddings *	88.30	43.22	58.03	0.1657	16.45	41.81
NMT error-focused *	67.47	60.72	63.91	0.1956	-1.32	28.99
NMT single ensemble	88.30	44.26	58.96	0.1631	18.34	43.45
NMT single ensemble error	69.54	61.55	65.30	0.1903	1.40	30.99
NMT multi ensemble *	86.79	47.84	61.68	0.1621	19.08	42.17
French Monographs						
OCR baseline	Char/Tok ER: 1.61% / 6.18%			0.0692	0%	-
SMT baseline *	82.69	38.55	52.52	0.0558	24.00	55.04
SMT medium	80.81	36.30	50.10	0.0574	20.59	51.95
SMT large	82.44	36.24	50.18	0.0587	18.07	48.84
NMT baseline	84.22	40.10	54.38	0.0602	15.00	44.13
NMT medium	83.59	38.23	52.37	0.0593	16.71	43.89
NMT large	82.81	37.15	51.05	0.0610	13.66	40.66
SMT context *	82.49	40.51	54.18	0.0568	22.15	58.47
NMT context *	84.17	42.95	56.85	0.0586	18.26	46.10
NMT time factor no context	84.37	41.53	55.75	0.0591	17.28	45.38
NMT time factor context	81.60	43.04	56.45	0.0583	18.85	45.68
NMT factor medium	85.94	40.97	55.46	0.0580	19.31	47.32
NMT factor large	85.63	39.36	53.86	0.0577	20.11	47.48
NMT glyph embeddings *	83.55	43.30	57.00	0.0583	18.73	45.43
NMT error-focused	58.53	54.56	56.48	0.0756	-8.28	30.23
NMT single ensemble *	85.72	42.98	57.29	0.0575	20.43	48.10
NMT single ensemble error	59.71	55.15	57.37	0.0744	-6.99	31.87
NMT multi ensemble *	82.27	45.73	58.91	0.0581	19.09	46.30

Table 7: French macro-averaged experiments results. Best results are marked in blue, worst results in red. Asterisks mark indicates systems used for ICDAR submission. (ER = error rate)

Precision P How many of the tokens predicted as incorrect actually needed a correction?

Recall R How many of the incorrect tokens were actually predicted as incorrect?

F1-Measure Harmonic mean of precision and recall: $F1 = \frac{2*P*R}{P+R}$

Error correction is measured on character level using Damerau-Levenshtein distance averaged over all tokens.¹⁰ Levenshtein distance between tokens t_1 and t_2 measures the minimal amount character substitutions, deletions or inserts needed to turn t_1 into t_2 (Levenshtein, 1966). Damerau distance (Damerau, 1964) additionally allows to swap two adjacent characters.¹¹ Note that for the official submissions in task 2, a list of candidate corrections combined with their probability was evaluated. In this setup, the evaluation reports the weighted sum of edit distances for all candidates (weight equals probability). For the results shown on our internal test set, we always stucked to only one correction candidate.

For task 2, i.e. the error correction, the positions of erroneous tokens were given and only those were evaluated. Of course, evaluating error correction only on erroneous tokens is an artificial scenario. In practice, it is not known if a token is correct or not. Any potential corrections may also introduce new errors to correct tokens. This evaluation is therefore not ideal but the evaluation setup was given by the competition. Note that we used the same training data for both tasks.

In the following, we present and discuss the results for different experimental settings in the same order as they were introduced in the preceding section. Table 6 summarizes all results on English data sets, Table 7 on French. In addition to the F1-score and the relative Levenshtein distance improvement, these tables also report character- and token-level error rates as well as the Levenshtein distance for the original OCR output, precision and recall scores for all experiments as well as the percentage of translated words that are correct.

5.2 More Training Material

What is the effect of combining the different data sets for training? Against our expectations, error detection and correction performance mostly decreases with medium training sets, especially in the case of English periodicals (see Tables 6 and 7). Only SMT can sometimes profit from more data. This suggests that OCR errors are particular to a text type and that NMT approaches specifically adapt to the specific OCR error distribution (frequency and types) of a training set. This trend is even stronger when cross-linguistically combining larger training sets. There, performance drops even more.

More general observations about context-insensitive correction can be made: First, SMT models perform better than NMT in error correction and worse in error detection.

¹⁰The official evaluation script was released via <https://git.univ-lr.fr/gchiro01/icdar2017>. Although the official ICDAR report refers to Levenshtein distance, the implementation actually uses Damerau-Levenshtein distance.

¹¹Damerau-Levenshtein actually is more useful for measuring human spelling distance than OCR spelling distance.

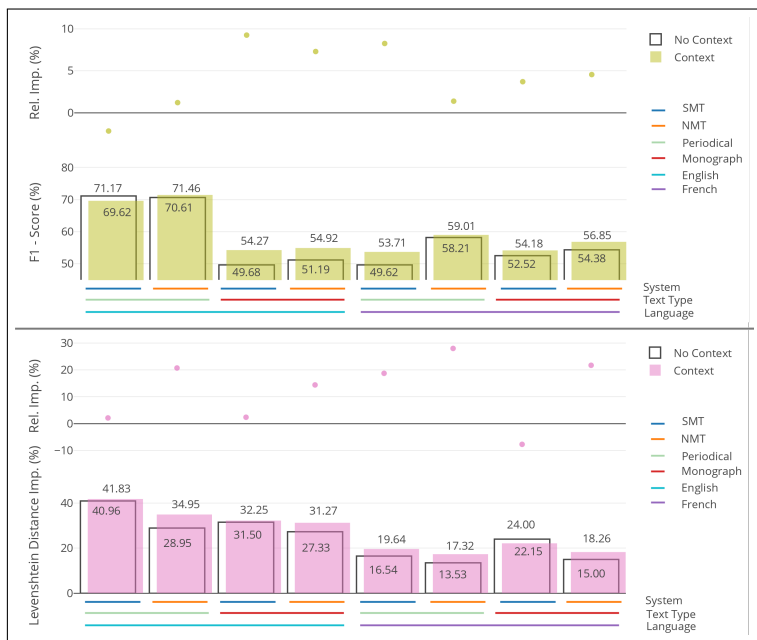


Figure 3: Effect of the inclusion of context (two preceding and one succeeding token) on performance

Second, error correction works better on English than on French. Third, error detection works better on the periodical data, probably due to the higher a priori error rate of this text type.

5.3 Using More Context

The results of the experiments described in Section 4.3 are shown in Figure 3. As expected, more context for translation effectively improves results in all but one data set. The transparent columns with black borders show the context-sensitive baseline trained separately on language-specific text types. The colored columns show the results using the context-insensitive sets. The scatter plots above the bar plots give the relative improvement (scales are not comparable between figures).

The overall performance pattern as discussed before still applies to the individual data sets, and SMT still performs better for error correction and NMT for error detection.

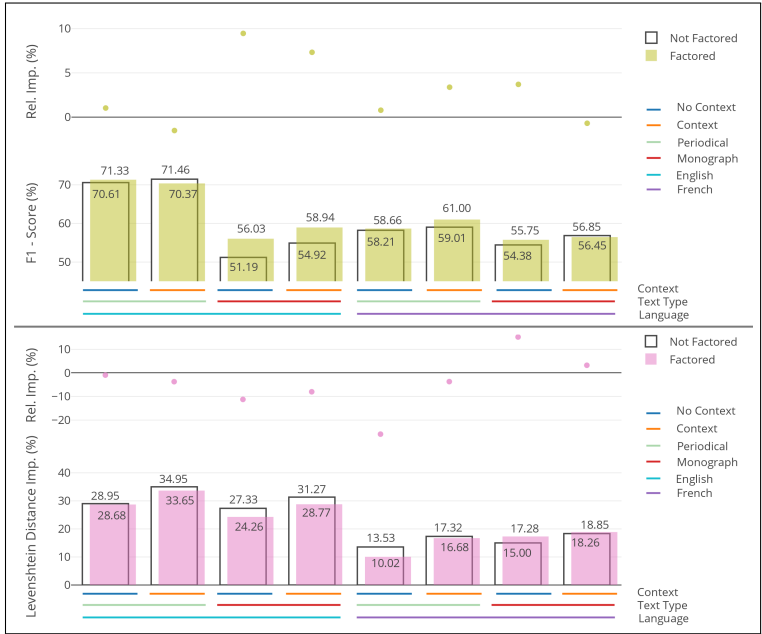


Figure 4: Effect of time-factored training material

5.4 Factored Character-based NMT

Using factors as described in Section 4.4 is another strategy to include more information when training an NMT system. Figure 4 shows the influence of the factor on the publication time. Overall, time factors result in a better performance in error detection for most data sets, especially for English monographs. This phenomenon can be explained with a quick look at Figure 1. Most English monograph texts do not need many corrections. However, this is problematic for the earlier files (around 1800), which contain many more errors than the later ones. Time factors allow the model to detect errors more aggressively on texts that were published around 1800. Therefore, time factors control the aggressiveness of an NMT model’s error detection. In contrast to error detection, there is a decrease in the relative Levenshtein distance improvement in error correction for all but two data sets. This development could be simply because there is an improvement in error detection. Detecting more errors does not automatically

mean providing accurate corrections. Trying to correct more tokens can lead to an overall higher distance to the GT. Another explanation could be that, especially, with smaller data sets there might be data sparsity issues.

Unfortunately, also adding factors for languages and combining all data folds into a large data set still does not generally improve the performance (results not shown).

5.5 Glyph Embeddings

The use of glyph images as pre-trained embeddings only shows a small improvement for monographs in both F1-score and relative Levenshtein distance (results not shown). For periodicals, the results are even slightly worse than with randomized embeddings. Adding the intuitively promising information about glyph similarity therefore needs another modeling approach than ours for more general applicability. Maybe using glyph images with closer resemblance to the actual fonts helps, maybe we should concatenate the glyph vector representation with the character embeddings instead of using them as initialization weights.

5.6 Error-Focused Models

We have seen in Table 2 that periodicals contain many more errors in need of correction than monographs. Therefore, we subsample error-focused training sets as described in Section 4.6. As expected, error correction suffers a lot from such unrepresentative training material and over-corrects the texts. As intended, error detection F1 score improves for periodicals as recall and precision get more balanced. Without subsampling precision is typically 20 to 40 percentage points higher than recall. Unfortunately, F1 decreases for monographs (especially for English). Maybe an improved tuning of the optimal error proportion for subsampling can remedy these rather unexpected differences.

5.7 Ensemble Decoding

The results confirm that in almost all cases error detection and correction improve when single model ensembles are used. Relative improvement for error correction is much higher than for error detection. This suggests that at different training moments the models provide different correction candidates which in combination converge to better suggestions.

Context-sensitive neural multi ensembling as described in Section 4.7 is strong for error detection and competitive for NMT error correction. However, SMT generally outperforms neural multi ensembles on error correction by a large margin. The performance pattern of multi ensembles across languages and text types is still similar to the patterns seen for the context-insensitive baseline, suggesting that these patterns are not due to specific idiosyncrasies of one of our models.

Data Set	Error Detection (Task 1)		Error Correction (Task 2)	
en Periodical	73.0	NMT single ensemble error	41.8	SMT context
en Monograph	58.9	NMT time factor context	32.8	SMT baseline
fr Periodical	65.3	NMT single ensemble error	19.6	SMT context
fr Monograph	58.9	NMT multi ensemble	24.0	SMT context

Table 8: The best systems for each data set. For error detection, macro-averaged F1 score is considered and for error correction macro-averaged relative Levenshtein distance improvement.

5.8 Summary on Systems

Generally, the performances across data sets vary a lot, except for error detection on monographs. Table 8 collects all best-performing systems for error detection (macro-averaged F1-measure) and error correction (macro-averaged relative Levenshtein distance improvement). All best systems for error detection use NMT, while all best systems for error correction use SMT. Context-sensitive SMT consistently works best in all but one case. For error detection, all but one system are ensembles. We explain the exception of English monographs by its high error variation over time, something that time-factored models capture effectively. Our results indicate that the optimal choice of an MT systems for OCR post-correction strongly depends on the task (detection or correction), language, text type, time span and error distribution. Ensemble decoding in NMT models almost always boosts performance. Ensembling of SMT output could also help, but we did not conduct such experiments.

5.9 Our ICDAR 2017 OCR Post-Correction Competition Submission and Results

As described in Section 4.8, our submission combines the outputs from different systems. Unfortunately not all system types presented in this article were available at submission time of the shared task, for instance, the well-performing ensembles of the error-focused models. The systems for submission were selected such that their combination performs best on an internal tuning set (see Table 9 for a complete listing for each data set). Note that the selected systems are not necessarily the top five systems on that data set as their combined performance is considered.

Table 10 summarizes the results of the best 6 teams (out of 11) from the official paper on the ICDAR 2017 OCR post-correction shared task (Chiron et al., 2017).¹² Our approach (Char-SMT/NMT) achieved the best results of all submitted methods in error correction (task 2). The relative improvement columns report the relative

¹²Probably due to the rather complex output format defined by the shared task organizers, many teams seem to have produced inconsistent data.

Error Detection (Task 1)		Error Correction (Task 2)	
en Periodical	en Monograph	en Periodical	en Monograph
NMT multi ensemble	NMT baseline	SMT context *	SMT baseline *
SMT baseline *	SMT context *	SMT baseline	SMT context
NMT single ensemble	NMT multi ensemble	NMT multi ensemble	NMT baseline
NMT glyph embeddings	NMT context	NMT single ensemble	NMT glyph embeddings
NMT context	NMT time factor context	NMT context	NMT multi ensemble
fr Periodical	fr Monograph	fr Periodical	fr Monograph
NMT multi ensemble	NMT multi ensemble	NMT multi ensemble *	SMT context *
NMT baseline *	SMT context *	NMT time factor context	SMT baseline
NMT time factor context	NMT single ensemble	NMT error-focused	NMT multi ensemble
NMT error-focused	NMT context	NMT glyph embeddings	NMT single ensemble
NMT glyph embeddings	NMT glyph embeddings	SMT baseline	NMT context

Table 9: Systems which were included in our system combination for task 1 and 2. Systems marked with an asterisk have the smallest Levenshtein distance of the systems in that combination.

token-level improvement if the top-ranked correction candidate of a system is applied. Compared to the other approaches, we consistently deliver best results across all data sets. A hyphen in the cells indicates that a system actually deteriorated more tokens than it improved. In contrast to our submission, quite a lot of system failed to improve the texts, which is an indication of the difficulty of the task.

In error detection (task 1), we performed comparatively to other submissions. The best approach for task 1 applies a noisy channel model to the OCR post-correction, and uses the Google Books Ngrams for their vocabulary and language model. Therefore, error detection probably profits from external data. In contrast, our approach does not need additional resources to train the MT systems. Other models that achieve similar results as our method use character-based NMT with context, SMT on character and token level, spell checkers, error frequency patterns, or a 2-pass RNN architecture where the first RNN works on character level and the second on token level. The official shared task report (Chiron et al., 2017) contains short summaries of the chosen approach from all participating teams.

Table 11 shows the results of task 1 in more detail. Accuracy and recall can be compared directly. Our method achieved by far the highest accuracy. However, even though we tried to increase the recall for error detection via system combination, our recall is not as high as it is for other methods. Still, the amount of documents that our system actually improves (102 documents) is much higher than for all other systems. The best system for error detection is only able to improve about 60% of documents

Teams/Data Sets	Task 1 (F-measure)					Task 2 (%Improvement) (top-ranked correction candidate)			
	en mono	en peri	fr mono	fr peri	mean	en mono	en peri	fr mono	fr peri
# tokens (error rate)	63371 (10%)	33176 (15%)	32274 (5%)	48356 (7%)		63371 (10%)	33176 (15%)	32274 (5%)	48356 (7%)
WFST-PostOCR	0.73	0.68	0.55	0.69	0.66	28%	-	-	-
2-pass-RNN ‡	0.66	0.66	0.43	0.60	0.59	x	-	x	x
EFP	0.69	0.54	0.40	0.54	0.54	13%	-	23%	5%
Char-SMT/NMT	0.67	0.64	0.31	0.50	0.53	43%	37%	44%	29%
CLAM	0.67	x	0.36	0.54	0.52	29%	22%	1%	5%
MMDT ‡	0.66	0.44	0.36	0.41	0.47	20%	-	3%	2%
Post-Submission	0.62	0.59	0.35	0.51	0.52	18%	21%	40%	24%

Table 10: Official ICDAR OCR post-correction results (only 6 best teams out of 11 shown). Systems marked with “‡” had partially malformed submission format. Cell entries with “x” indicate missing or malformed submissions for this data set. Cell entries with “-” indicate that no improvement was achieved, meaning there was a deterioration of the texts. The last row shows the performance of our high-performing single system post-submission.

compared to our submission.

Table 11 provides valuable information on what could be improved with our method and in what cases it makes sense to use it. Our approach is optimal for an automatic OCR correction scenario with high precision requirements for error detection and high correction quality. On the other hand, if recall is more important, for instance, if manual verification is applied to the correction candidates, our method might need to be adapted further, or another approach should be used for the error detection.

There is a practical issue with our approach if we want to apply it to new datasets. Building all the necessary systems for our system combination is a bit complex and laborious. In a post-submission experiment, we therefore tried to build the most promising single system according to our experimental experience.

Our post-submission system is a combination of the strategies with the greatest potential to profit from each other¹³: For each language, we combined the context-sensitive (two preceding and one following token) training material for periodicals and monographs and used factors to encode information on the time period and text type of the documents. The last row in Table 10 shows that such a single system performs reasonably across all data sets, however, there is a substantial drop in comparison to the best reported result.

¹³We did not empirically test all combinations systematically, though.

System	en		fr		en		fr		all
	mono	peri	mono	peri	mo	pe	mo	pe	all
	Acc./Rec.	Acc./Rec.	Acc./Rec.	Acc./Rec.	#	#	#	#	#
Char-SMT/NMT	0.98 /0.51	0.88 /0.50	0.74 /0.19	0.93 /0.34	40	4	46	12	102
CLAM	0.93/0.52	-/-	0.48/0.28	0.71/0.44	36	4	31	7	78
MMDT	0.84/0.55	0.72/0.32	0.62/0.25	0.71/0.28	37	3	28	7	75
EFP	0.62/0.77	0.54/0.55	0.29/ 0.60	0.49/0.58	31	1	24	11	67
WFST-PostOCR	0.67/ 0.82	0.68/ 0.68	0.51/0.59	0.72/ 0.66	36	0	20	3	59
2-pass RNN	0.58/0.77	0.64/ 0.68	0.33/0.60	0.09/0.04	x	0	x	x	0
Total # of Documents in Test Set ⇒					41	4	54	12	110

Table 11: Official ICDAR results break down (only 6 best performing teams out of 11 shown) on accuracy, recall, and number of documents where a net improvement resulted from the corrections of task 2.

6 Future Work

The comparison of the best ICDAR systems in error detection with our results indicates that our methods do not yet spot the optimal amount of OCR issues that need correction. The resources that our models use are strictly limited to the official training material. The best ICDAR error detection system uses additional material from Google Books n-grams. For a practically oriented OCR error detection, we should think about ways to integrate external data for error detection. However, for solving the full OCR post-correction task one still has to be able to actually generate the correct spelling.

An important question for practical applicability is the amount of training data needed for our approach. Ablation experiments where the training material is systematically reduced would offer some insights about the correlation of training data size and the post-correction improvement, which is probably also dependent on the original error rate.

A better tuning of the subsampling for error-focused models is worth trying. For an optimal F1 score in error detection, recall and precision should be perfectly balanced. However, this goal is difficult to optimize directly and needs further experimentation.

In our work, we experimented with only one way (two preceding and one succeeding token) for introducing textual context into the translation model. Context is needed for high-quality OCR post-correction with character-based MT, but other ways of integrating it might work as well.

A more sophisticated model representation of glyph shapes that are more similar to the historical fonts used for typesetting is worth exploring. Furthermore, our experiments

suggest that it would be interesting to train models on data from smaller time periods, e.g. only documents from 1800 to 1850.

Finally, one should better analyze how the training material size, relative error frequency, language, text type, time span and typeface influence the OCR post-correction quality. Our experiments with different data sets with different properties often showed strongly varying or inconsistent effects. Conclusive and generally applicable insights are hard to achieve, but they are strongly needed for a practical application of supervised OCR post-correction.

7 Conclusion

This article presented a broad overview as well as extensive experiments and evaluations on how character-based neural and statistical machine translation techniques can be used for OCR post-correction. We showed that SMT systems perform better in error correction, while NMT systems achieve higher results in error detection. This is important to know for anyone who plans to employ character-based MT for any of these subtasks.

We tested both state-of-the-art and novel strategies to include more information in the training and translation process of NMT systems. Giving enough context to the systems for a correction candidate allows better detection of real word errors and increases the training material. We found that no improvement in OCR post-correction can be achieved when data sets with different error characteristics are combined. However, when the individual training examples are labelled with their data set characteristics as factors, neural systems are able to generalize from the increased training sets and produce better results, especially for error correction. Specifically, data sets with considerably varying error rates profit from time factors. Another insight is that error-focused models can boost error detection, but have a rather negative influence on error correction. Correctly calibrating the error threshold for subsampling the training material is essential for these models. We showed that the decrease in error correction with error-focused models can be mitigated by using ensemble decoding. In fact, normal ensembling of single systems' output is useful for OCR correction, even more so, if different systems are combined for ensemble decoding. Finally, we experimented with a novel, straightforward approach how visual information on glyphs can be included in the training process of character-based NMT systems.

However, we also saw that for the given training sizes and the highly varied training material it is hard to achieve conclusive and generally applicable insights about the best settings and hyperparameters. We often observed some improvements on some data sets and at the same time performance deterioration or no effect on others. Thus, it is hard to reuse models for out-of-domain data sets.

A carefully compiled ensemble of our models reached best performance in ICDAR's 2017 error correction subtask and performed competitively in error detection. Due to the individual systems' strengths and weaknesses, we proposed an algorithm that combines different system outputs. The results of the shared task show that our

approach is competitive in error detection and strongly outperforms other approaches in error correction, even though we do not use any external resources. We also presented post-submission results on the ICDAR data set for the best single-model configuration, which is more practical to adapt and apply on new data sets than complex ensemble solutions. Our evaluation suggests that future work on NMT for OCR post-correction should focus on improving error detection.

Acknowledgements

SC has been supported by the Swiss National Science Foundation under grant CR-SII5_173719.

References

- Afi, H., Barrault, L., and Schwenk, H. (2015). OCR error correction using statistical machine translation. In *16th International Conference on Intelligent Text Processing and Computational Linguistics, Cairo, Egypt*.
- Afi, H., Qiu, Z., Way, A., and Sheridan, P. (2016). Using SMT for OCR error correction of historical texts. In *Proceedings of LREC-2016, Portorož, Slovenia*, pages 962–965.
- Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.
- Bollmann, M. and Søgaard, A. (2016). Improving historical spelling normalization with bidirectional LSTMs and multi-task learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 131–139, Osaka, Japan.
- Brill, E. and Moore, R. C. (2000). An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, ACL '00*, pages 286–293.
- Chiron, G., Doucet, A., Coustaty, M., and Moreux, J.-P. (2017). ICDAR2017 competition on post-OCR text correction. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1423–1428.
- Chung, J., Cho, K., and Bengio, Y. (2016). A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany.
- Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176.
- Eger, S., Mehler, A., et al. (2016). A comparison of four character-level string-to-string translation models for (ocr) spelling error correction. *The Prague Bulletin of Mathematical Linguistics*, 105(1):77–99.

- He, W., He, Z., Wu, H., and Wang, H. (2016). Improved neural machine translation with SMT features. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 151–157. AAAI Press.
- Koehn, P. (2009). *Statistical machine translation*. Cambridge University Press.
- Koehn, P. (2017). Neural machine translation. *CoRR*, abs/1709.07809.
- Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.
- Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439.
- Lee, J., Cho, K., and Hofmann, T. (2017). Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.
- Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8).
- Luong, M.-T. and Manning, C. D. (2016). Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany.
- Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015). Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China.
- Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.
- Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.
- Pettersson, E., Megyesi, B., and Tiedemann, J. (2013). An SMT approach to automatic annotation of historical text. In *Proceedings of the workshop on computational historical linguistics at NODALIDA 2013; May 22-24; 2013; Oslo; Norway. NEALT Proceedings Series 18 / Linköping Electronic Conference Proceedings 87*, pages 54–69. Linköping University Electronic Press.
- Piotrowski, M. (2012). Natural language processing for historical texts. *Synthesis Lectures on Human Language Technologies*, 5(2).
- Reynaert, M. (2016). OCR post-correction evaluation of early dutch books online - revisited. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA).

- Rosca, M. and Breuel, T. (2016). Sequence-to-sequence neural network models for transliteration. *CoRR*, abs/1610.09565.
- Schnober, C., Eger, S., Do Dinh, E.-L., and Gurevych, I. (2016). Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714, Osaka, Japan.
- Schulz, S. and Kuhn, J. (2017). Multi-modular domain-tailored ocr post-correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2716–2726, Copenhagen, Denmark. Association for Computational Linguistics.
- Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Miceli Barone, A. V., Mokry, J., and Nadejde, M. (2017). Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain.
- Sennrich, R. and Haddow, B. (2016). Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany.
- Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany.
- Silfverberg, M., Kauppinen, P., and Lindén, K. (2016). Data-driven spelling correction using weighted finite-state methods. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, pages 51–59, Berlin, Germany. Association for Computational Linguistics.
- Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.
- Volk, M., Furrer, L., and Sennrich, R. (2011). *Strategies for Reducing and Correcting OCR Errors*, pages 3–22. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., and Ng, A. Y. (2016). Neural language correction with character-based attention. *CoRR*, abs/1603.09727.
- Yao, K. and Zweig, G. (2015). Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. In *16th Annual Conference of the International Speech Communication Association (INTERSPEECH 2015)*, pages 3330–3334.
- Zhao, S. and Zhang, Z. (2016). An efficient character-level neural machine translation. *CoRR*, abs/1608.04738.

Resource and tool descriptions

Comparison of OCR Accuracy on Early Printed Books using the Open Source Engines Calamari and OCRopus

Abstract

This paper proposes a combination of a convolutional and an LSTM network to improve the accuracy of OCR on early printed books. While the default approach of line based OCR is to use a single LSTM layer as provided by the well-established OCR software OCRopus (OCRopy), we utilize a CNN- and Pooling-Layer combination in advance of an LSTM layer as implemented by the novel OCR software Calamari. Since historical prints often require book specific models trained on manually labeled ground truth (GT) the goal is to maximize the recognition accuracy of a trained model while keeping the needed manual effort to a minimum.

We show, that the deep model significantly outperforms the shallow LSTM network when using both many and only a few training examples, although the deep network has a higher amount of trainable parameters. Hereby, the error rate is reduced by a factor of up to 55%, yielding character error rates (CER) of 1% and below for 1,000 lines of training. To further improve the results, we apply a confidence voting mechanism to achieve CERs below 0.5%. A simple data augmentation scheme and the usage of pretrained models reduces the CER further by up to 62% if only few training data is available. Thus, we require only 100 lines of GT to reach an average CER of 1.2%. The runtime of the deep model for training and prediction of a book behaves very similar to a shallow network when trained on a CPU. However, the usage of a GPU, as supported by Calamari, reduces the prediction time by a factor of at least four and the training time by more than six.

1 Introduction

The best OCR engines on early printed books like Tesseract (4.0 beta)¹ or OCRopus² currently use Long Short Term Memory (LSTM) based models, which are a special kind of recurrent neural networks. In order to achieve low CERs below e.g. 1% or 2% on early printed books these models must be trained individually for a specific book due to a high variability among different typefaces used (see Springmann et al. 2016 or Springmann and Lüdeling 2017). Thereto, a certain amount of GT, in the case of OCRopus that is a pair of text line image and transcribed text, must be manually labeled. The primary goal is to reduce the number of labeled text lines to achieve a certain error rate. A

¹<https://github.com/tesseract-ocr>

²<https://github.com/tmbdev/ocropy>

secondary goal is to continuously retrain the model, if more GT becomes available because e.g. all lines of a book are reviewed and corrected to achieve a final error rate of near 0%. The default OCRopus implementation uses a shallow one layer bidirectional LSTM network combined with the CTC-Loss to predict a text sequence from the line image. Since convolutional neural networks (CNN) showed an outstanding performance on many image processing tasks, see e.g. Mane and Kulkarni (2017), our aim is to train a mixed CNN-LSTM network to increase the overall performance of OCR. Therefore, we compare the default OCRopus implementation with the deep network architectures provided by the novel OCR engine Calamari³ which is based on TensorFlow⁴. It is well known that voting the outputs of several different models improves the accuracy by a significant margin, which is why we use Calamari’s cross fold training approach proposed by Reul et al. (2018). This approach trains five different models whose outputs are combined by a voting mechanism that considers the confidence values of each output. Moreover, Calamari offers data augmentation and pretrained models to increase the accuracy especially on small datasets.

The rest of the paper is structured as follows: Section 2 introduces and discusses related work regarding OCR on early printed books including deep models and voting. The used data and the applied methods are described in detail in Section 3. In Section 4, we evaluate and discuss the results achieved on three early printed books. After summing up the results and insights in Section 5 we conclude the paper with some ideas regarding future work.

2 Related Work

This section lists related work concerning the application of CNN-LSTM hybrids in the areas of speech, vision, and text processing. Furthermore, related work covering improvements on OCR using different voting algorithms are itemized.

2.1 Combinations of CNN-LSTM

Currently CNN-LSTM hybrids are used in a high diversity of fields to achieve state-of-the-art results. The combination of those diverse network structures is promising because CNNs are suited for hierarchical but location invariant tasks, whereas LSTMs are perfect at modelling temporal sequences.

For example, in the domain of speech processing Sainath et al. (2015) use a combination of LSTMs, CNNs and Fully Connected Layer for an automatic speech recognition task, or else Trigeorgis et al. (2016) train a CNN-LSTM for speech emotion recognition. Another excellently suited area for CNN-LSTM networks is video processing, since CNNs are perfect for handling images and the video itself is a sequence of images. For instance, in this area Donahue et al. (2015) propose a CNN-LSTM as basic structure

³<https://github.com/Calamari-OCR/calamari>

⁴<https://www.tensorflow.org/>

to automatically generate video descriptions or Fan et al. (2016) use this structure for recognizing emotions in videos.

In the field of text recognition, a combination of CNNs and LSTM was most recently proposed by Breuel (2017). The deep models yield superior results on the University of Washington Database III⁵, which consists of modern English prints with more than 95,000 text lines for training. However, the effect of deep networks on historical books and using only a few dozens of training lines for training book individual models has not been considered, yet.

A very similar task to OCR is handwriting recognition, e.g. by Graves and Schmidhuber (2009) or Bluche (2015) or scene text recognition, e.g. by Shi et al. (2017). These applications usually act on contemporary data, which is why it is meaningful to include a language model or a dictionary to achieve a higher accuracy. However, for early printed books, e.g. medieval books, general language models are less effective mostly due to variability in spelling.

2.2 Voting

Using voting methods to improve the OCR results of different models was investigated by many different researchers, an overview is given by Handley (1998). Here, we list only a subset of the most important and the most recent work in the field of OCR and focus mostly on historical documents.

Rice et al. (1996) showed that voting the outputs of a variety of commercial OCR engines reduces the CER from values between 9.90% and 1.17% to 0.85%. The voting algorithm first aligns the outputs by using a Longest Common Substring algorithm (Rice et al., 1994) and afterwards applies a majority voting to determine the best character including a heuristic to break ties.

Al Azawi et al. (2015) trained an LSTM network as voter of the aligned output of two different OCR engines. A comparison using printings with German Fraktur and the University of Washington Database III the LSTM approach led to CERs around 0.40%, while the ISRI voting tool achieved CERs around 2%. A major reason for this high improvement is that the LSTM-based voting algorithm learns some sort of dictionary. Thus, the voter was able to predict a correct result even in cases where each individual voter failed. However, this behaviour might not be desired since the method not only corrects OCR errors but also normalizes historical spellings.

Most recently, in Reul et al. (2018) we showed that a cross-fold training procedure with subsequent confidence voting reduces the CER on several early printed books by a high amount of up to and over 50%. This voting procedure not only takes the actual predictions of a single voter into account, but also their confidence about each individual character. Therefore, instead of a majority voting for the top-1 output class the best character is chosen for the top-N outputs. This led to improvements by another 5% to 10% compared to the standard ISRI sequence voting approach.

⁵<http://isis-data.science.uva.nl/events/dlia/datasets/uwash3.html>

2.3 Data Augmentation and Pretraining

A crucial point for the performance of DNNs is the availability of huge labeled datasets. However, if only a few data points are available or if the GT has to be labeled manually the GT can be augmented to generate new examples. The applied operations must be label preserving, e.g. a line image must still show the same content after augmentation. The augmentations used in Calamari are provided by OCRodeg⁶ and consist of padding, distortions, blobs, and multiscale noise.

Furthermore, in Reul et al. (2017b), we successfully applied transfer learning on early printed books using OCRopy: Instead of training a network from scratch with random weights, the weights can be copied from a network that was trained on different GT. Thus, the new network starts its training already with knowledge about the task, i.e. the transferred features are meaningful on the new data as-well. The network only has to adapt for the new typeface and possibly some new characters in the codec. Therefore, in general, the network requires less lines of GT to reach the CER of a network trained from scratch.

3 Material and Methods

The OCR pipeline of Calamari is based on the OCRopus workflow, whose fundamental idea is to use a full text line as input and train an LSTM network to predict the GT character sequence of that line. This is achieved by the usage of the CTC-Loss during training and a CTC-Decoder for prediction. For a deeper insight in this pipeline, in this section, we first introduce the used datasets. Afterwards, we explain our training and evaluation procedure. Finally, the differences of OCRopy and Calamari concerning implementation, hyperparameters, and network architectures are listed.

3.1 Datasets

For our experiments we employ three different early printed books (see Table 1). Only lines from running text are used, whereas headings, marginalia, page numbers etc. are excluded, because these elements vary in line length, font size or their characters e.g. numbers are underrepresented. Thus, the actual data is not affected by unwanted side effects resulting from these elements. 1505 represents an exception to that rule as we chose the extensive commentary lines instead, as they presented a bigger challenge due to very small inter character distances and a higher degree of degradation. Figure 1 shows one line per book as an example.

1505 is an edition of the *Ship of Fools* (*Narrenschiff* by Sebastian Brant) and was digitized as part of an effort to support the *Narragonien digital* project at the University of Würzburg⁷. 1488 was gathered during a case study of highly automated layout

⁶<https://github.com/NVLabs/ocrodeg>

⁷<http://kallimachos.de/kallimachos/index.php/Narragonien>

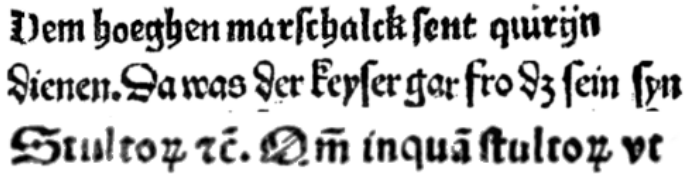


Figure 1: An example line from each of the used books. From top to bottom: 1476, 1488, 1505.

Table 1: Books used for evaluation and their respective number of ground truth lines available for training and validation.

Year	Language	GT Train	GT Validation
1476	German	2,000	1,000
1488	German	3,178	1,000
1505	Latin	2,289	1,000

analysis Reul et al. (2017a) and 1476 is part of the Early New High German Reference Corpus⁸. The GT data of all three books was published⁹ by Springmann et al. (2018).

3.2 Training and Evaluation

To train the essential book specific models the human effort to annotate GT should be minimized. Therefore, we examine the effect of adding only a few dozen lines of GT on the CER of an individual book. The aim of a real world application is to train incrementally improved models when more GT becomes available to support the human annotator with increasingly better models.

As setup, each book in the dataset is first split into an evaluation and a training set. While the evaluation set is fixed, the training set size is chosen as 60, 100, 150, 250, 500, and 1,000 lines, with each set subsuming the smaller sets entirely. Then, each training set is divided into a 5-fold, where four parts are used for the actual training and one part for validation. For example, if 100 lines are chosen randomly from the full set, each fold uses 80 lines for training and the remaining 20 lines for validation. These 20 validation lines are distinct from each of the five folds resulting in five diverse models. Based on the validation set the best performing model for each fold is determined.

In summary, for three books and six different numbers of lines in the training set, we train five models, respectively. For each run, we compute the CER on the validation set every 1,000 iterations to determine the best model of this fold. This model is then evaluated on the initial evaluation dataset to obtain the final CER. Since the results of each of the five folds vary, we compute the average to compare the outcomes of the different network architectures, books, and number of lines. Furthermore the five

⁸<http://www.ruhr-uni-bochum.de/wegera/ref/index.htm>

⁹<https://zenodo.org/record/1344132/>

$$A \left\{ \begin{array}{ll} n(0.6) & u(0.1) \\ n(0.2) & u(0.3) \\ n(0.4) & u(0.5) \end{array} \right\} \text{ example sentence}$$

Figure 2: An example for the improvement by using the confidence voting mechanism. Here only three voters are considered. Although the character "u" is the best character twice the "n" is chosen for the final output because of its higher average confidence.

predictions are used to compute a voted result using the voting mechanism described in Section 3.3.

At each iteration during training one line is randomly chosen out of the data fold to compute the gradient update. The number of iterations during training is chosen as 10,000, 12,000, 15,000, 20,000, 25,000, and 30,000 for the training set size of 60, 100, 150, 250, 500, and 1,000, respectively. These values are fixed for both OCRopus and Calamari.

3.3 Voting

In order to further improve the predictions, we implement the confidence voting scheme as proposed by Reul et al. (2018) to vote on the label sequences that are produced by the CTC-Greedy-Decoder for each fold considering the confidence of each predicted character. An example for only three voters is shown in Figure 2.

In a first step, all sentences are aligned as far as possible. Afterwards, all differences are resolved by averaging the confidence of all equal options and keeping the character with the highest value. Naturally, this procedure cannot guarantee a flawless output, but it significantly improves the overall result.

An important prerequisite for the models that are used to vote is that they are similarly performant, but diverse in their predictions. Errors that occur randomly in one or another sentence can easily be corrected, whereas errors that appear in every output cannot be identified. The above Cross-Fold-Training as proposed by Reul et al. (2018) yields different models that can be used for voting although a high amount of training data is shared among each pair of models.

The number of models that are used for voting in our experiments is set to five. Those are the individual models produced by the 5-fold Cross-Validation on each training set. A higher number of voters is expected to yield better results with the drawback to require a higher effort in training and prediction. Experiments showed that a fold size of five is reasonable trade-off.

3.4 Comparison of Calamari and OCRopy

While both Calamari and OCRopy are written in Python and their interfaces are identical (both require images of lines and their respective GT) there are many distinct

differences regarding the implementation, supported features and default settings.

To fasten up the computations of the neural network, OCRopy supports usage of CLSTM¹⁰ which is a C++ implementation of the fixed LSTM network architectures and only runs on the CPU. Calamari however uses Googles TensorFlow library that both allows to design custom network architectures and to utilize a GPU.

Calamari supports the described Cross-Fold-Training mechanism and confidence voting as well as integrated data augmentation. To choose the best model based on the validation data set as provided by the Cross-Folds, Calamari implements early stopping so that an upper limit for the number of training iterations tests whether the current model improves the accuracy on the validation data. If the accuracy has not improved for a given number of models (default 10), the training is stopped. Data augmentation of the training data is provided by a variable factor n_{aug} , i.e. each line is replicated n_{aug} times. Other features are the support of bidirectional text, and an automatic codec adaptation if a pretrained model is loaded that shares only a subset of characters.

Calamari employs by default an Adam solver with an initial learning rate of 0.001, whereas OCRopy utilizes a learning rate of 0.0001 and a Momentum solver with a momentum of 0.9. The batch size of Calamari can be chosen arbitrarily, and is 1 by default. OCRopus does not allow to train or predict batchwise. Moreover, Calamari implements an automatic gradient clipping to tackle the exploding gradient problem of LSTMs as proposed by Pascanu et al. (2013).

The standard OCRopus network uses a single hidden LSTM layer with 100 nodes. Calamari extends this shallow structure by introducing two pairs of convolutional (40 and 60 kernels) and pooling layers before the default LSTM-Layer with 200 nodes. Calamari uses a convolutional kernel of size 3×3 with a stride of 1 and equal padding. The network uses max pooling layers with a kernel size and stride of 2×2 . A stride or a kernel size of 2 in the first dimension, that is the time dimension, halves the width of the intermediate picture and therefore the number of LSTM operations. On the one hand, this makes the network faster but on the other hand repeated characters might not get resolved, because the CTC-Decoder requires an intermediate blank label. Finally, Calamari adds dropout with a factor of 0.5 to the last layer.

Even though the network architecture of Calamari can be adapted, preliminary experiments showed that Calamari's default network yields competitive results in both accuracy and speed, hence it will be used as the *deep* network in the following. The *shallow* network architecture is the default OCRopus network.

An overview of the differences and default values is listed in Table 2.

4 Experiments

In the following, we present our findings regarding the improvements of the deeper architecture, and the usage of voting. Additionally, we compare the time required for

¹⁰<https://github.com/tmbdev/clstm>

Table 2: Comparison and default values of OCRopy and Calamari regarding various aspects.

	OCRopy	Calamari
Language	Python 2	Python 3
Network Backend	Native/CLSTM	Tensorflow
GPU Support	No	Yes
Default Network Architecture	LSTM 100	CNN 40, Pool, CNN 60, Pool, LSTM 200
CNN Kernel Size	–	3×3
Pool Size	–	2×2
Dropout	No	Yes
Solver	Momentum (0.9)	Adam
Default Learning Rate	0.0001	0.001
Voting	No	Yes
Pretrained Models	Yes (identic codec)	Yes
Data Augmentation	No	Yes

training and prediction of the different architectures, and extend the training corpus to a size of up to 3,000 lines to investigate the behaviour of the deep models on many lines. Finally, we use data augmentation or pretraining to minimize the CER with the focus on only a few lines of GT. All experiments are conducted by using the default hyperparameters and network architectures of Calamari and OCRopy as described in Sec. 3.4.

4.1 Results of the Deep Network

Table 3 shows exemplarily the CER for each of the five folds on the three books for 60, 100 and 1,000 lines. The average of these five models is used to compute the average improvement of the deep network. Note, that in practice the individual results of the model for each fold must be combined e.g. by voting (see Section 3.3) in order to obtain a usable result. Without voting, only one fold could be used to predict a sequence.

The results of the individual folds show no obvious correlation between CERs on the same fold, that is the same training data, and its CERs on different network architectures. For example, the worst fold of the shallow network on book 1488 using 100 lines is the fourth fold with a CER of 4.79%. However, the same fold results in the second best model for the deep network. The same can be observed for book 1505 for the second fold: The best outcome of the deep network with CER of 3.02% is the worst fold for the shallow network.

The relative improvements that are shown in Table 3 are listed for all the different number of lines in Table 4. In the left section the relative improvements of the average

Table 3: This table shows exemplarily the improvements of a deep CNN-LSTM compared to the default shallow architecture for 60, 100, and 1,000 lines in the training dataset. Both the individual results of each Cross-Fold plus their average, and the voting improvements are entirely listed. The last columns of the fold average and the voting average state the relative improvements of the deep network architecture. All numbers are given in percent.

60 Lines										
Book	Network	CERs per Fold					Avg.		Conf. Voted	
		1	2	3	4	5	CER	Imp.	CER	Imp.
1476	Shallow	8.19	8.62	9.23	6.64	7.91	8.12		4.72	
	Deep	6.71	5.48	7.63	5.74	6.28	6.37	21.5	4.63	1.92
1488	Shallow	8.86	6.25	5.87	8.61	6.79	7.28		4.38	
	Deep	5.34	4.63	4.75	4.48	5.31	4.90	32.6	3.84	12.4
1505	Shallow	8.86	6.25	5.87	8.61	6.79	7.28		4.58	
	Deep	4.96	5.21	4.60	4.77	4.56	4.81	33.8	4.02	12.3
Avg.								29.3		8.9
100 Lines										
Book	Network	CERs per Fold					Avg.		Conf. Voted	
		1	2	3	4	5	CER	Imp.	CER	Imp.
1476	Shallow	8.19	4.96	5.30	8.02	7.64	6.82		3.49	
	Deep	3.94	3.33	3.33	3.03	4.23	3.57	47.6	2.68	23.2
1488	Shallow	4.30	3.72	3.79	4.79	4.38	4.20		2.73	
	Deep	2.79	3.15	3.20	2.97	3.21	3.06	27.0	2.38	12.7
1505	Shallow	4.74	4.67	4.32	4.59	4.40	4.54		3.16	
	Deep	3.13	3.02	3.23	3.22	3.29	3.18	30.0	2.49	21.3
Avg.								34.9		19.1
1,000 Lines										
Book	Network	CERs per Fold					Avg.		Conf. Voted	
		1	2	3	4	5	CER	Imp.	CER	Imp.
1476	Shallow	1.46	1.52	1.52	1.56	1.68	1.55		0.97	
	Deep	0.74	0.91	0.68	0.71	0.85	0.78	49.7	0.56	42.1
1488	Shallow	1.15	1.20	1.08	1.03	1.38	1.17		0.71	
	Deep	0.54	0.59	0.63	0.60	0.57	0.59	49.7	0.42	40.7
1505	Shallow	1.96	1.87	1.77	1.85	1.77	1.84		1.35	
	Deep	1.34	1.31	1.32	1.31	1.32	1.32	28.4	1.12	17.2
Avg.								43.6		33.3

of the folds is shown, on the right hand side, the improvement when using voting (see Section 4.2). A single deep network architecture yield an increasingly better average CER. For only 60 lines the average improvement is 29%, while for 1,000 lines the best improvement on a single book is 50% and the average over all books is increased by 43%.

A plot of the relative increase dependent on the number of lines is shown in Figure 3 (solid points). The increasing slope is flattening which indicates that more lines used for training a deep model will still yield a better model compared to the default model, but its relative improvement is eventually constant. An even deeper network might increase the relative improvement if more lines are available.

In general, as to expected, the shallow and the deep network yield better results with an increasing number of training data. Surprisingly, although the deep network has a higher amount of trainable parameters which increases the vulnerability for overfitting, it outperforms the shallow net even for a very small amount of training data.

4.2 Evaluation of Voting

The average error rates based on applying the confidence voting algorithm are shown in Table 3. As expected, voting improves the accuracy on all experiments by a significant amount and even reaches a optimum value of 0.42% CER on book 1488. The shallow network benefits by a higher margin compared to the deep network, especially when using only a few training examples. Yet, the deep network of Calamari always outperforms the shallow network of OCRopy before and after voting. The relative improvements shown in Table 4 and Figure 3 clarify this behaviour.

When training on just 60 lines the deep network architecture performs only slightly better than the default network, yielding an average improvement factor of 9%. However, if 1,000 lines are used an average improvement of 33% is obtained. Considering the slopes of Figure 3 it is observed that the improvement gap between voting and non-voting is narrowed from $29\% - 9\% = 20\%$ to $42\% - 33\% = 9\%$ for 60 and 1,000 lines, respectively. Furthermore, it is to be expected that the relative improvement approaches a limit value. Hence, the used deep model is expected to still perform better than the default model by absolute values, but the relative gap appears to be constant. The limit value, i.e. the performance for infinite data, is influenced by the network architecture. Thus, if more lines are available for training more complex models must be considered.

As shown, voting has a higher impact on the default OCRopus network than on the used deep network, especially if only a few lines are used for training. Thus, the individual models must be more diverse to allow for a more effective voting. The evaluation of errors (see Table 5 in Section 4.3) shows that, apart from insertion and deletions of spaces, the errors of deep networks are mostly missing characters, while the errors of the default network are mostly confusions, e.g. $e \leftrightarrow c$ or $n \leftrightarrow r$ in both directions. Therefore, voting of deep models that all omit single character predictions (compare Table 5) and thus suffer from similar errors can not benefit by such a high amount compared to the default models whose errors are more random.

Table 4: Relative improvement of the deep CNN compared to default OCRopus listed for all three books and the six variations of the amount of training data. The left and right halves show the relative improvements without and with voting, respectively. All numbers are given in percent.

Lines	Improvement over Folds				Improvement over Voting			
	1476	1488	1505	Avg.	1476	1488	1505	Avg.
60	21.5	32.7	33.8	29.3	1.9	12.4	12.3	8.9
100	47.6	27.0	30.0	34.9	23.2	12.7	21.3	19.8
150	38.4	40.1	32.9	37.1	19.7	30.7	19.7	23.4
250	49.7	54.6	30.9	45.0	27.0	34.2	23.5	28.2
500	50.4	49.6	32.1	44.1	35.2	43.7	21.3	33.3
1,000	49.7	49.7	28.4	42.6	42.1	40.7	17.2	33.3

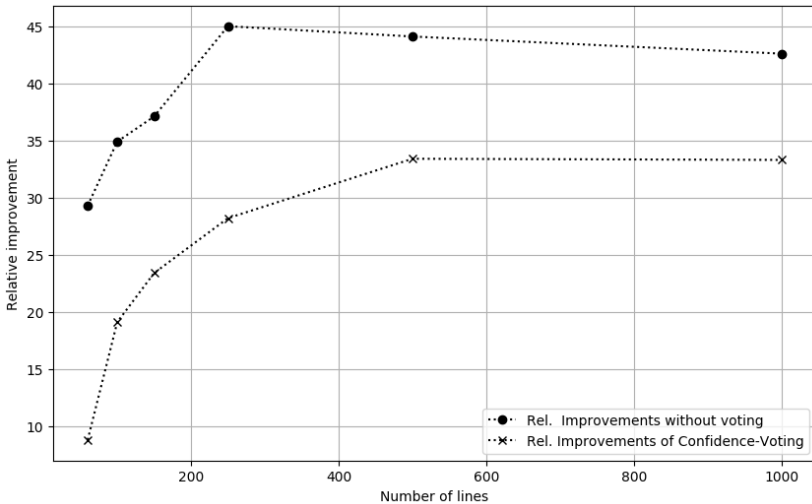


Figure 3: Relative averaged improvements of the deep network versus the default OCRopus network. The solid points indicate the relative improvements based on the averages of the individual Cross-Fold results. The crosses mark the relative improvements when using confidence voting.

Table 5: The 10 most common errors made by the deep network after voting on the evaluation dataset of book 1476. The left and right halves lists the results for 150 and 1,000 training lines, respectively. Both the count (Cnt.) of occurrences and their relative (Rel.) contribution to the total CER are shown. An underscore represents deletions or insertions of characters.

150 Lines				1000 Lines			
Cnt.	Rel.	True	Predicted	Cnt.	Rel.	True	Predicted
56	8.9%	SPACE	—	32	18.0%	SPACE	—
54	8.6%	—	SPACE	26	14.6%	—	SPACE
47	7.5%	i	—	11	6.2%	i	—
18	2.9%	l	—	6	3.8%	n	—
12	1.9%	G	E	3	1.7%	r	t
12	1.9%	n	—	2	1.1%	r	v
10	1.6%	r	—	2	1.1%	S	—
9	1.4%	r	t	2	1.1%	r	—
8	1.3%	d	—	2	1.1%	f	f
7	1.1%	o	—	2	1.1%	—	i

4.3 Error Analysis

Table 5 lists the ten most common errors of the deep network after voting when predicting the evaluation dataset of 1476. The results are shown for 150 lines and 1,000 lines on the left and right columns, respectively. For both networks the most common errors consist of insertions or deletions of spaces. Their relative contribution account for one fifth for 150 lines, but almost one third for 1,000 lines, which underlines the difficulty of the task of inserting correct spaces between narrowly printed text if no language model or dictionary is used.

The model trained with 150 lines then mostly misses the prediction of characters which shows the expected behaviour for CTC-trained networks that are only trained with few data: To minimize the CTC-Loss it is more profitable to predict a blank rather than an actual uncertain character. The confusion of G and E vanished because the G occurred rarely in the dataset with only a few lines, but it was successfully learned if more examples were available. The deep network trained with 1,000 lines shows errors among the top ten that are expected, e.g. confusions of f and f (long s).

4.4 Training and Prediction Times

In this section we compare the time required for training and for the prediction of an entire book of the default OCRopus network and our deep network. All times were measured on an Intel Core i7-5820K processor using 1, 2, 4, or 8 threads for each experiment. A NVIDIA Titan X is utilized to gauge the GPU times. During training the parallelism is internally used in Numpy for the default OCRopus implementation and in the TensorFlow-Backend for our deep networks. TensorFlow supports cuDNN

Table 6: Average times for training and prediction of a single line for all three books. Note that during prediction each line has to be processed five times due to the voting of the five folds. The timing procedure was conducted for a various number of threads.

Threads	Training in seconds					Prediction in seconds				
	1	2	4	8	GPU	1	2	4	8	GPU
Shallow	0.28	0.27	0.30	0.40	–	0.89	0.48	0.25	0.16	–
Deep	0.57	0.40	0.32	0.33	0.05	0.29	0.21	0.16	0.12	0.03

when using the GPU. For predicting, the default OCRopus implementation copies the individual model for each thread and uses only one thread in the internal operations of Numpy. Calamari instead creates only one model and predicts one batch consisting of 20 lines in our setup using all desired threads.

Note that each line has to be processed five times during the prediction due to the voting algorithm. Table 6 reports our findings for the averages across the three used books.

First of all, the results for the training time show that, despite the deeper network consists of way more parameters and more operations, the optimized TensorFlow implementation is only slightly slower than the default implementation based on Numpy, when only one thread is used. However, the shallow default LSTM net cannot benefit from a higher number of threads, instead it even suffers from a too high count. The reason is, that the underlying matrix multiplications are too low dimensional in order to be relevant for multiprocessing. As expected, the deep network benefits from up to 4 threads, as the training time is decreased by an average factor of approximately 40%. The reason is, that the convolution operations that are carried out on each pixel on the full image profit from a parallel computation. As a result, the deep network is faster than the default implementation when allowing several cores during the training. Our results show, that a number of 4 is sufficient.

The average time required to predict one line for the three books shows that the deep network requires less than half the time compared to the default OCRopus implementation, whereby the time for voting can be neglected. Using a higher number of threads, the required time for prediction almost shrinks linearly for the default implementation, since each thread computes a single independent model by construction. The TensorFlow implementation of Calamari still benefits from a higher thread count, but by a reduced factor due to the core sharing of batch versus convolutional operation. Yet, for all the tested thread counts the TensorFlow implementation is a bit faster than default OCRopus.

Usage of a GPU reduces the training and prediction times further by a factor of at least six and four, respectively. These times can easily be reduced when processing a whole batch of lines instead of a single line.

Table 7: Decrease of the CER for using more than 1,000 lines for training the deep network. All values are given in percent.

Lines	Averaged CER of folds				CER using voting			
	Book			Avg.	Book			Avg.
	1476	1488	1505		1476	1488	1505	
1,000	0.78	0.59	1.3	0.90	0.56	0.42	1.1	0.70
1,500	0.69	0.50	1.3	0.82	0.48	0.35	1.0	0.62
2,000	0.62	0.49	1.2	0.76	0.45	0.35	1.0	0.60
3,000	—	0.43	—	0.43	—	0.34	—	0.34

4.5 Increasing the Training Data above 1,000 Lines

The available amount of data for the three books allows us to increase the training set size up to 2,000, 3,000, and 2,000 lines for the books 1476, 1488, and 1505, respectively (compare Table 1). The averaged CER of the folds and after voting is shown in Table 7 by usage of the deep network. As expected, even more lines for training reduces the CER even further in all experiments with and without voting by a significant amount. Thus, we reached an average CER of 0.6% after voting.

4.6 Data Augmentation

In this section we examine the effect of data augmentation with a factor of $n_{\text{aug}} = 10$ on the performances of deep network. In a first step, the training data consists of both the augmentations and the original data. A second step uses the resulting model of the first step as starting point and continues the training on solely the original data.

The results using confidence voting of five different voters each with its own augmentations are shown in Table 8. As expected, especially for only a few lines in the training data set data augmentation has a huge impact of up to 55%. Increasing the number of lines decreases the benefit of augmentations to a point where it even worsens the result (up to -19% on book 1488). This can be explained by the fact that the mixture of real lines and augmented lines forces the network to focus on both types of data which is why it loses its specific training on just the real lines. Thus, when continuing the training on only the real lines (step 2) the results clearly improve for many lines, but even for a few lines. On average, data augmentations in combination with the deep network leads to a CER of below 2% and approximately 1% when using only 60 and 150 lines for training, respectively. The shallow network as provided by OCRopus requires more than 1,000 lines to reach this level of accuracy (compare Table 3).

In summary, the experiments show the expected behaviour that the data augmentation as implemented in Calamari yields slightly improved results for a large data set and very high improvements of up to 55% for only a few lines.

Table 8: Relative improvements of data augmentation with $n_{\text{aug}} = 10$. The first step trains on both the real and augmented data, the second step continues the training only on the real data. The last column shows the absolute CER of the final model (step 2) averaged over all three books.

Lines	Improvement after Step 1				Improvement after Step 2				Avg. CER.
	Book			Avg.	Book			Avg.	
	1476	1488	1505			1476	1488		1505
60	56.2	60.4	40.2	52.3	61.4	62.1	41.2	54.9	1.87
100	43.8	46.8	28.0	39.5	46.6	52.0	29.6	42.7	1.43
150	43.4	44.6	21.5	36.5	44.8	48.1	26.4	39.7	1.04
250	32.0	22.3	19.7	24.6	37.9	31.4	25.4	31.5	0.83
500	25.9	6.7	14.2	15.6	34.7	23.6	17.1	25.1	0.64
1,000	21.5	9.7	8.7	13.3	22.0	14.6	16.2	17.6	0.58
1,500	12.6	-6.5	9.5	5.2	15.2	3.9	13.9	11.0	0.54
2,000	4.3	-5.8	5.2	1.2	17.0	11.6	11.6	13.4	0.52
3,000	—	-18.5	—	-18.5	—	7.3	—	7.3	0.32

Table 9: Improvement of using both pretraining (PT) and data augmentation (AUG). All values are voted and shown as percentage.

Lines	Improvement after PT				Improvement after PT and AUG				Avg. CER.
	Book			Avg.	Book			Avg.	
	1476	1488	1505			1476	1488		1505
60	51.1	46.8	30.0	42.7	70.5	68.8	48.0	62.4	1.6
100	37.9	41.9	22.7	34.2	63.6	55.8	39.8	53.1	1.2

4.7 Incorporating Pretrained Models

To evaluate the effect of pretraining, we use three different pretrained Calamari models¹¹ designed for Fraktur (FRK), historical Latin (LH), and Modern English (EN). Two models of the five voters are trained with FRK, two more with LH and only one with EN since the trained typefaces of FRK and LH are closer to the target font. These outcomes are voted by the confidence voter to obtain the final CER. Furthermore, we combine pretraining with a data augmentation of 10. Both setups are only trained for 60 and 100 lines of GT, because here on the one hand the effect of pretraining and data augmentation is expected to be the largest and on the other hand we want to simulate the lack of manual annotated GT to train a book specific model.

As shown in Table 9, the CER for both 60 and 100 lines benefits from pretraining (left) and the combination with data augmentation (right). We reach an improvement of over 62% for 60 lines of GT compared to the model without pretraining or data

¹¹https://github.com/Calamari-OCR/calamari_models

Table 10: Improvements of voting, data augmentation, and pretraining (PT) for 60 and 100 lines comparing Calamari to OCRopy. All values are given in percent.

60 Lines					
Model	Books			Averages	
	1476	1488	1505	CER	Imp.
Shallow (OCRopy)	8.1	7.3	7.3	7.6	–
Deep (Calamari)	6.4	4.9	4.8	5.4	28.9
Deep + Voting	4.6	3.8	4.0	4.2	44.7
Deep + Voting + Data aug.	2.0	1.5	2.4	2.0	73.7
Deep + Voting + Data aug. + PT	1.4	1.2	2.1	1.6	78.9
100 Lines					
Model	Books			Averages	
	1476	1488	1505	CER	Imp.
Shallow (OCRopy)	6.8	4.2	4.5	5.2	–
Deep (Calamari)	4.7	4.4	4.6	4.6	11.5
Deep + Voting	2.7	2.4	2.5	2.5	51.9
Deep + Voting + Data aug.	1.5	1.3	1.8	1.5	71.2
Deep + Voting + Data aug. + PT	1.0	1.1	1.5	1.2	76.9

augmentation, with a final CER of 1.6%. An increase to only 100 lines drops the CER to 1.2%.

Comparing pretraining (Table 9) to the results of only data augmentation (Table 8) shows that data augmentation has a higher impact on the performance (e.g. 55% vs 43% using 60 lines). However, data augmentation requires more training time than using pretrained models, because the initial random weights must be trained from scratch.

5 Conclusion and future work

In this paper, we compared the combinations of CNN- and LSTM-Networks as implemented by Calamari to the default LSTM network of OCRopy achieving optimum values considerably below 1% CER and a relative improvement of above 50% compared to standard models. The enhancements are increased by a larger amount of available training data and the introduction of voting mechanisms, data augmentation, and pretrained models. Thus, to train a book specific model, only 60 or 100 lines of GT are sufficient to achieve an average CER of below 2% or about 1%, respectively, as shown in Table 10.

Although the proposed deeper network has a higher number of parameters the absolute training and prediction time is in the same order of magnitude compared to the standard model. However, the usage of a GPU quickens these times by a factor of at least four depending on the lines processed in parallel.

To further improve the voted results, distinct voters are required. These voters could be created by variations of the network architectures and the usage by several different datasets for pretraining.

Moreover, training of an even deeper network using many different books sharing similarities in typeface, is expected to result in a generic model that has very low error rates on a high variety of fonts. To reduce its training time, a GPU combined with batch-wise training should be considered to achieve a high throughput of lines per second.

Recently, the popular OCR engine Tesseract¹² published a new version that implements Deep Neural Networks. We expect similar improvements compared to shallow networks, however voting, data augmentation, and pretraining in the form of Calamari are not supported, yet. Moreover, GPUs can not be used to speed up the training time.

In summary, it can be stated that the application of Calamari implementing deep CNN-LSTM-networks, Cross-Fold-Voting, data augmentation, and pretraining opens the door to a very promising approach to establish a new benchmark for OCR both on early printed books and despite the historical focus of this paper also on any other print.

References

- Al Azawi, M., Liwicki, M., and Breuel, T. M. (2015). Combination of multiple aligned recognition outputs using WFST and LSTM. In *Document Analysis and Recognition (ICDAR), 2015 13th Int. Conf. on*, pages 31–35. IEEE.
- Bluche, T. (2015). *Deep Neural Networks for Large Vocabulary Handwritten Text Recognition. (Réseaux de Neurones Profonds pour la Reconnaissance de Texte Manuscrit à Large Vocabulaire)*. PhD thesis, University of Paris-Sud, Orsay, France.
- Breuel, T. M. (2017). High performance text recognition using a hybrid convolutional-lstm implementation. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 11–16. IEEE.
- Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.
- Fan, Y., Lu, X., Li, D., and Liu, Y. (2016). Video-based emotion recognition using cnn-rnn and c3d hybrid networks. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction, ICMI 2016*, pages 445–450, New York, NY, USA. ACM.
- Graves, A. and Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552.
- Handley, J. C. (1998). Improving OCR accuracy through combination: A survey. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE Int. Conf. on*, volume 5, pages 4330–4333. IEEE.

¹²<https://github.com/tesseract-ocr/tesseract>

- Mane, D. T. and Kulkarni, U. V. (2017). A survey on supervised convolutional neural network and its major applications. *IJRSDA*, 4(3):71–82.
- Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1310–1318. JMLR.org.
- Reul, C., Dittrich, M., and Gruner, M. (2017a). Case study of a highly automated layout analysis and ocr of an incunabulum: 'der heiligen leben' (1488). In *Proceedings of the 2Nd Int. Conf. on Digital Access to Textual Cultural Heritage, DATECH2017*, pages 155–160, New York, NY, USA. ACM.
- Reul, C., Springmann, U., Wick, C., and Puppe, F. (2018). Improving OCR accuracy on early printed books by utilizing cross fold training and voting. In *13th IAPR International Workshop on Document Analysis Systems, DAS 2018, Vienna, Austria, April 24-27, 2018*, pages 423–428. IEEE Computer Society.
- Reul, C., Wick, C., Springmann, U., and Puppe, F. (2017b). Transfer learning for OCRopus model training on early printed books. *027.7 Zeitschrift für Bibliothekskultur / Journal for Library Culture*, 5(1):38–51.
- Rice, S. V., Jenkins, F. R., and Nartker, T. A. (1996). *The fifth annual test of OCR accuracy*. Information Science Research Institute.
- Rice, S. V., Kanai, J., and Nartker, T. A. (1994). An algorithm for matching OCR-generated text strings. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(05):1259–1268.
- Sainath, T. N., Vinyals, O., Senior, A., and Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584.
- Shi, B., Bai, X., and Yao, C. (2017). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304.
- Springmann, U., Fink, F., and Schulz, K. U. (2016). Automatic quality evaluation and (semi-) automatic improvement of mixed models for ocr on historical documents. *CoRR*.
- Springmann, U. and Lüdeling, A. (2017). OCR of historical printings with an application to building diachronic corpora: A case study using the RIDGES herbal corpus. *Digital Humanities Quarterly*, 11(2).
- Springmann, U., Reul, C., Dipper, S., and Baiter, J. (2018). GT4HistOCR: Ground Truth for training OCR engines on historical documents in German Fraktur and Early Modern Latin.
- Trigeorgis, G., Ringeval, F., Brueckner, R., Marchi, E., Nicolaou, M. A., Schuller, B., and Zafeiriou, S. (2016). Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5200–5204.

Ground Truth for training OCR engines on historical documents in German Fraktur and Early Modern Latin

Abstract

In this paper we describe a dataset of German and Latin *ground truth* (GT) for historical OCR in the form of printed text line images paired with their transcription. This dataset, called *GT4HistOCR*, consists of 313,173 line pairs covering a wide period of printing dates from incunabula from the 15th century to 19th century books printed in Fraktur types and is openly available under a CC-BY 4.0 license. The special form of GT as line image/transcription pairs makes it directly usable to train state-of-the-art recognition models for OCR software employing recurring neural networks in LSTM architecture such as Tesseract 4 or OCRopus. We also provide some pretrained OCRopus models for subcorpora of our dataset yielding between 95% (early printings) and 98% (19th century Fraktur printings) character accuracy rates on unseen test cases, a Perl script to harmonize GT produced by different transcription rules, and give hints on how to construct GT for OCR purposes which has requirements that may differ from linguistically motivated transcriptions.

1 Introduction

The conversion of scanned images of printed historical documents into electronic text by means of OCR has recently made excellent progress, regularly yielding character recognition rates by individually trained models beyond 98% for even the earliest printed books (Springmann et al., 2015; Springmann and Fink, 2016; Springmann and Lüdeling, 2017; Springmann et al., 2016; Reul et al., 2017a,b, 2018, see also this volume). This is due to (1) the application of recurrent neural networks with LSTM architecture to the field of OCR (Fischer et al., 2009; Breuel et al., 2013; Ul-Hasan and Breuel, 2013), (2) the availability of open source OCR engines which can be trained on specific scripts and fonts such as Tesseract¹ and OCRopus², and (3) the possibility to train recognition models on real printed text lines as opposed to generating artificial line images from existing computer fonts (Breuel et al., 2013; Springmann et al., 2014).

What is missing, however, are robust pretrained recognition models applicable to a wide range of typographies spanning different fonts (such as Antiqua and Fraktur with long s), scripts and publication periods, which would yield a tolerable OCR result of >95% character recognition rate without the need of any specific training. Accurate ground truth and better individual OCR models could be constructed from the output of these pretrained models much more easily than by transcriptions from scratch. The feasibility to construct such mixed models able to generalize to previously unseen books that have not contributed to model training has been shown in Springmann and Lüdeling

¹<https://github.com/tesseract-ocr/tesseract/wiki/Training-Tesseract>

²<https://github.com/tmbdev/ocropy/wiki>

(2017) with diachronic German Fraktur printings (compare their Fig. 6 and Fig. 7): Character recognition rates of individual models quickly fall below 80% when applied to books printed with different fonts at different periods, whereas mixed models show an average rate of 95% (see their Table 2).

The construction of pretrained mixed models crucially depends on available ground truth data for a wide variety of historical documents. In this paper we describe training material of historical ground truth which has been collected and produced by us over the course of several years. The training of OCRopus models is described in detail in the CIS workshop on historical OCR³ and the OCRoCIS tutorial⁴.

All of our ground truth is made available in the GT4HistOCR (*Ground Truth for Historical OCR*) dataset under a CC-BY 4.0 license in Zenodo (Springmann et al., 2018). The repository contains the compressed subcorpora, some pretrained mixed OCRopus models for subcorpora, and a Perl script which can be adapted to harmonize GT produced by different transcriptions guidelines in order to have a common pool of training data for mixed models.

In the following we describe our GT4HistOCR dataset and its constituent subcorpora (Sect. 2), mention other existing sources of historical GT which have not yet been mined for model construction (Sect. 3) together with a description of a crowdsourcing tool for GT production using public APIs of the Internet Archive (Sect. 3.1), make some remarks about transcription guidelines and their relevance to the production of GT for OCR purposes (Sect. 4), and end with a conclusion (Sect. 5).

2 The *GT4HistOCR* dataset

In the following we introduce the five subcorpora of our *GT4HistOCR* dataset (see Table 1). The transcription of these corpora was done manually (partly by students) and later checked and corrected by trained philologists within projects in which we participated:

The *Reference Corpus Early New High German*⁵ is a DFG funded project, the *Kallimachos* corpus derives from work done in the BMBF funded Kallimachos project⁶, the *Early Modern Latin* corpus was produced during projects on OCR postcorrection funded by CLARIN and DFG⁷, *RIDGES*⁸ has been built by students at HU Berlin as part of their studies in historical corpus linguistics and *DTA19* has been extracted from the DFG-funded *Deutsches Textarchiv* (DTA)⁹. An overview of the contribution of these subcorpora to our dataset is shown in Table 1.

The text line images corresponding to the transcribed lines have been prepared and matched by us using OCRopus segmentations routines or, in the case of DTA19, the segmentation of ABBYY Finereader. The ground truth in the form of paired line images and their transcriptions are an excerpt from the books in a corpus.

Because the transcription guidelines for each subcorpus differ in the amount of typographical detail that has been recorded we chose not to construct corpora according to language or period by merging

³<http://www.cis.uni-muenchen.de/ocrworkshop/program.html>

⁴<http://cistern.cis.lmu.de/ocrocis/tutorial.pdf>

⁵<https://www.linguistics.ruhr-uni-bochum.de/ref/>

⁶<http://kallimachos.de>

⁷<http://www.cis.lmu.de/ocrworkshop/>

⁸<http://korpling.org/ridges/>

⁹<http://www.deutschestextarchiv.de/>

Table 1: Overview of the subcorpora of *GT4HistOCR*. For each subcorpus we indicate the number of books, the printing period, the number of lines, and the language.

Sect.	Subcorpus	# Books	Period	# Lines	Language
3.1	Reference Corpus ENHG	9	1476-1499	24,766	ger
3.2	Kallimachos Corpus	9	1487-1509	20,929	ger, lat
3.3	Early Modern Latin	12	1471-1686	10,288	lat
3.4	RIDGES Fraktur	20	1487-1870	13,248	ger
3.5	DTA19	39	1797-1898	243,942	ger
Sum:				313,173	

File name: EarlyModernLatin/1564-Thucydides-Valla/00001.bin.png

fuerunt,tum uetustæ,tum sequentiũ temporum.Non minimam

File name: EarlyModernLatin/1564-Thucydides-Valla/00001.gt.txt

fuerunt, tum uetustæ, tum sequentiũ temporum. Non minimam

Figure 1: Example GT line pair of line image (upper line) and its transcription. A blank after each punctuation symbol has been added and the OCR model will consequently learn to map a punctuation symbol to the sequence punctuation, blank.

and harmonizing material from these subcorpora. However, because the directory containing the GT of each book is named with publishing year and book title, a user can remix our data and construct new corpora according to his needs after the transcriptions have been harmonized. An example of a GT line pair is given in Fig. 1.

2.1 Incunabula from the *Reference Corpus Early New High German*

The Reference Corpus Early New High German (ENHG) is being created in an ongoing project which is part of a larger initiative with the goal of creating a diachronic reference corpus of German, starting with the earliest existing documents from Old High German and Old Saxon (750–1050), and including documents from Middle High German (1050–1350) and Middle Low German and Low Rhenish (1200–1650), up to Early New High German (1350–1650). The Reference Corpus Early New High German contains texts published between 1350 and 1650. From 1450 on, prints are included in the corpus besides manuscripts. The last part, 1550–1650, consists of prints only. The texts have been selected in a way as to represent a broad and balanced selection of available language data. The corpus contains texts from different time periods, language areas, and document genres (e.g. administrative texts, religious texts, chronicles). From the Reference Corpus Early New High German we got ground truth for the incunabula printings in Table 2. Specimen of line images which

Table 2: The Early New High German incunabulum corpus. Given are the printing year, the GW number, the short title, the number of ground truth lines for training and evaluation, and the character recognition rate (CRR) in % of a mixed model trained on all other books.

Year	GW	(Short) Title	# Lines	CRR
1476	M51549	Historij	3160	96.11
1478	04307	Biblia	2745	91.90
1485	M09766	Gart der Gesundheit	2520	96.37
1486	M45593	Eunuchus	3403	92.61
1486	5077	Jherusalem	2232	97.83
1490	10289	Pfarrer vom Kalenberg	2503	98.07
1490	5793	Leben und Sitten	3099	93.59
1497	5593	Cirurgia	3476	96.16
1499	6688	Cronica Coellen	1628	95.98
Sum:			24,766	

give an impression of the fonts are shown in Fig. 2. Full bibliographic details for these documents can be retrieved from the *Gesamtkatalog der Wiegendrucke*¹⁰ via the GW number.

While in principle we would like to have as large a corpus as possible and reuse all transcriptions from 1450 up to 1650, the process of generating accurately segmented printed lines from scanned book pages and matching them to their corresponding transcriptions is still laborious. Because OCR ground truth for periods later than 1500 is provided in other subcorpora we just used the incunabula printings of the reference corpus.

We also wanted to explore the feasibility to construct a mixed model and test its predictive power for unseen works from this period. For the about 30,000 incunabula printings, about 2000 print shops (*officinae*) using about 6000 typesets have been identified in the *Typenrepertorium der Wiegendrucke*¹¹, so a mixed model trained on only a few books might not generalize well to other incunabula printed in one of the many other and possibly much different fonts. On the other hand, even in this early period a division of labour between punchcutters and printers took place and commercially successful printing types were available for sale (Carter, 1969), so it might be expected that not all 6,000 identified fonts employed in the print shops were totally different from each others.

To get an idea of how well mixed models work for incunabula we trained nine models on eight books each and applied this model to the one book left out of the training set. The resulting CERs are given in the last column of Table 2. The previous finding of Springmann and Lüdeling (2017) that mixed models generalize better than individual models is corroborated: The worst recognition rate is 91.90% with an average rate of 95.40% on unseen books.

We provide a mixed model that was trained on the combined training set of all books and evaluated against a previously unseen test set taken from the same books. The resulting character recognition

¹⁰<http://www.gesamtkatalogderwiegendrucke.de/GWEN.xhtml>

¹¹<http://tw.staatsbibliothek-berlin.de/>

Op den burgeroyrten maendach
vnde nathan vnde zachariam vnde
die brust vnd den menschen in züfessigen
¶ Das ist ein entdeckung · oder lautere erkläring in der
iagten sie vnd erflügen yr eyn teyl · welches
Wānyndert lebt vff erden kein man
stenget · So nymmet sy auch nicht ab · Den
molcken der geysß · vñ werde gespise
eme dat so gesacht wurde / dat men eme gene

Figure 2: Example lines of the Early New High German incunabulum corpus in chronological order (see Table 2).

rate is above 97% for each book in this corpus (a higher value than the previous average because for this model each book contributed to the training set).

2.2 The *Kallimachos* corpus

The *Kallimachos corpus* consists of the 1488 printing of *Der Heiligen Leben* and eight books from the *Narragonien digital* subproject¹² dealing with the second most popular book in its time after the bible, the *Narrenschiff* (*ship of fools*) by Sebastian Brant. There are four Latin printings (*Stultifera nauis*) translated by Locher and Badius, respectively, two Early New High German printings, one Early Low German work (*Der narrenscip*), and one Latin/English document (Barclay) of which we just provide the Latin part. Whereas the German documents use a broken script, some Latin works are printed with Antiqua types similar to our modern types (Fig. 3). We do not provide a mixed model of these rather diverse types but leave it to the reader to construct his own models for his specific interests. The transcription of Badius is less accurate than that of the other books because it has not yet been checked to the same level of detail.

2.3 An *Early Modern Latin* corpus

In Springmann et al. (2016) we introduced a Latin data set of manual transcriptions from books that were either of interest to us or to scholars who requested an OCR text for a complete book for which we had to train an individual recognition model. The Early Modern Latin corpus is essentially the same, but leaves out the 1497 *Stultifera Nauis* (belonging to the *Kallimachos* corpus) and adds the 1543 *Psalterium* of Folengo (see Table 4). The printings are mostly in Antiqua types (except the

¹²<http://kallimachos.de/kallimachos/index.php/Narragonien>. Because annotated transcriptions of the *Narrenschiff* works have not yet been published the single lines of these works have been randomly permuted and do not provide a coherent text in their enumerated order [04.03.2018].

Table 3: The Kallimachos corpus

Year	GW	(Short) Title	# Lines
1488	M11407	Der Heiligen Leben (Winterteil)	4178
1495	5049	Das neu narrenschiff	2114
1497	5051	Das nuw schiff von narragonia	1197
1497	5056	Stultifera nauis	1424
1497	5061	Stultifera Nauis	1092
1499	5064	Stultifera nauis	721
1500	5066	Der narrenscip	2500
1505		Nauis stultifera (Badius)	4713
1509		The Shyp of Folys (Barclay)	2990
Sum:			20,929

Sienen. Da was der Keyser gar fro dz sein syn
 Et pudor: atq; fides/probitas/constantia
 hymel vmd fallen wider ab bis zū
 n̄r wißheit ist verschluct/sie haben geirt in
 nis nunq̄ satis laudata Nauis p̄ Sebastia
 rē Florētīnū atq; Franciscū Petrārcham
 Men diincker/men s̄incker/men specker op snaren
 Feria quinta post Martīn post Saturnus per
 uī cui id dedicarē: tū quia saluberrīma tua

Figure 3: Example lines of the Kallimachos corpus in chronological order (see Table 3). Both Antiqua fonts (Latin) and broken fonts (German) are present.

Table 4: The Early Modern Latin corpus

Year	(Short) Title	Author	# Lines
1471	Orthographia	Tortellius	417
1476	Speculum Naturale	Beauvais	2012
1483	Decades	Biondo	915
1522	De Septem Secundadeis	Trithemius	201
1543	De Bello Alexandrino	Caesar	830
1543	Psalterium	Folengo	314
1553	Carmina	Pigna	297
1557	Methodus	Clenardus	350
1564	Thucydides	Valla	1948
1591	Progymnasmata vol. I	Pontanus	710
1668	Leviathan	Hobbes	1078
1686	Lexicon Atriale	Comenius	1216
Sum:			10,288

Speculum Naturale of Beauvais, Fig. 4). The two provided models are those of the above mentioned publication.

2.4 The *RIDGES* Fraktur corpus

The use of broken scripts dates back to the 12th century and was once customary all over Europe. It is therefore of considerable interest to be able to recognize this script in order to OCR the large amount of works printed in a variety of Fraktur. This dataset collects Fraktur material from 20 documents of the *RIDGES* corpus of herbals (Odebrecht et al., 2017) which has been proofread for diplomatic accuracy and matched by us against lines images of the best available scans. OCR experiments on this corpus were reported in Springmann and Lüdelling (2017). The two mixed models used in that publication are provided and give a good base model covering about 400 years of Fraktur printings. Note that the author of the 1543 printing was erroneously attributed to Hieronymous Bock in Springmann and Lüdelling (2017) and has been corrected to Leonhart Fuchs in Table 5.

2.5 The *DTA19* corpus of 19th century German Fraktur

The use of broken scripts in the 19th century and later was mostly restricted to Germany and some neighboring countries. There is a large amount of scans available from 19th century documents (newspapers, long-running journals such as *Die Grenzboten*¹³ or *Dabeim*, encyclopedias¹⁴, dictionaries, novels, and reprints of classical works from previous centuries) which are of considerable interest to philologists and historians.

¹³<http://brema.suub.uni-bremen.de/grenzboten>

¹⁴<https://www.zedler-lexikon.de/>

Igitur apud antiquiores nostros & graecos eiusdem prorsus
gogica dicitur & unificā. Unificā quia dispersos i
revertit & postea philippo cum eiusdem nominis filio fraude
tatum excepisti & quidem liberalissime! plusq̄
 his erant quinqueres & quadriremes X. reliquæ infra
 set in tabulis lapideis populo lapidibus ipsis duriori,
 Et qui Pontificis maximi ad Arcana uocatus es,
 quod sanguam circumstet propositam senten-
 statim ab eo moto: sperans etiam fore
 neratione taceam, illi maximè scriptis
 te Actionis diversas producit Apparitiones!
 etiam ex Atrio nostro, desumpta esse

Figure 4: Example lines of the Early Modern Latin corpus in chronological order (see Table 4).

Table 5: The RIDGES Fraktur corpus.

Year	(Short) Title	Author	# Lines
1487	Garten der Gesunthait	Cuba	747
1532	Artzney Buchlein der Kreutter	Tallat	504
1532	Contrafayt Kreüterbuch	Brunfels	366
1543	New Kreüterbuch	Fuchs	483
1557	Wie sich meniglich	Bodenstein	995
1588	Paradeißgärtlein	Rosbach	795
1603	Alchymistische Practic	Libavius	473
1609	Hortulus Sanitatis	Durante	696
1609	Kräutterbuch	Carrichter	677
1639	Pflantz-Gart	Rhagor	1091
1652	Wund-Artzney	Fabricius	601
1673	Thesaurus Sanitatis	Nasser	733
1675	Curioser Botanicus	Anonymous	567
1687	Der Schweitzerische Botanicus	Roll	520
1722	Flora Saturnizans	Henckel	562
1735	Mysterium Sigillorum	Hiebner	470
1764	Einleitung zu der Kräuterkenntniß	Oeder	916
1774	Unterricht	Eisen	562
1828	Die Eigenschaften aller Heilpflanzen	Anonymous	658
1870	Deutsche Pflanzennamen	Grassmann	868
		Sum:	13,248

vnd tr:ffen in dem zwayten grad
Cinamomum. Symetrörlin/neuße sie fast/ist gut für
igen feüchtrigkeit/also das es küler biz vff den anfang
XII/oder Dyllkraut würdt zu Latein vnd auff Griechisch
rach der müter / treibt auß die an
Weens in das Erdreich sind gestelle/
er desto mehr öhl denn sonst.
langlechte schmahle Wurzeln/
süchtig Wasser zun Selen herauf / also
gelegt / dem rechten Verstand der Wörren niche
der vnder das Gebirch der Löblichen berühmten.
Sund mit dem Sarne sitzen/darnach sich wol auß:
fer ist kalt und trucken im
Mond verpfanget/ man kan dise
gang bläß-grünliche Farbe; Sal com. kasset es unge
durch empfinden, inmassen der Iouialishe Geist, so
nach unten zu eingerollet ist, und wdhrenden
Strünke selber, so weit sie zart sind,
Wilsenkraut, Saubohnen, Schlaßkraut.
er über die deutschen Namen zu jeder Gattung hinzufügt

Figure 5: Example lines of the RIDGES Fraktur corpus in chronological order (see Table 5).

Because of this high interest, some prominent works have been converted into electronic form by manual transcription (keyboarding, double-entry transcription) in low-wage countries¹⁵. Given the sheer amount of available material, faster and less costly alternatives are sought after and both commercial (ABBYY Finereader with a special Fraktur licence¹⁶) and open source OCR engines (Tesseract and OCRopus) are capable of recognizing Fraktur printings. What motivated us to look at 19th century Fraktur separately was the question whether we could beat the available general recognition models of the mentioned OCR engines. This is currently an open research topic.

It is tempting to use synthetic training materials, as a variety of Fraktur computer fonts is readily available on the internet. In fact, the Fraktur recognition model of Tesseract is completely based upon synthetic material, the model of OCRopus mostly. However, closer inspection shows that many fonts are either lacking some essential characteristics of real Fraktur types (such as long *f*, or *ch* and *tz* ligatures) or have obviously been constructed for calligraphic use and do not reflect the most frequently used historical types. For best OCR results we have to rely on transcriptions of real data, at least as an addition to any synthetic data set one might construct.

In the following we describe a collection of transcriptions from *Deutsches Textarchiv* for which line segmentations from ABBYY Finereader are available. The corresponding scans of these transcriptions are held by *Staatsbibliothek zu Berlin*¹⁷. We produced line images by cutting page scans into lines using the line coordinates contained in the ABBYY XML output. In this way a corpus of 63 books, some belonging to multi-volume works, could be assembled fully automatically. From these we selected just one volume of each multi-volume edition to provide a balanced multi-font corpus and did some quality checks on correct segmentations by hand.

The resulting *DTA19* corpus of 39 works is detailed in Table 6. To our knowledge there does not exist a similar extensive collection of ground truth for German 19th century Fraktur. We also provide a model trained on this corpus.

Because most Fraktur fonts do not differentiate between the alphabetic characters *I* and *J* and use the same glyph for both, we harmonized the transcription of *DTA* that employs different symbols to just use *J*. Otherwise, a model trained on the original transcription would randomly output either *I* or *J* for the same glyph. As a side effect, however, Roman numerals with the *I* glyph in the image will now be recognized with the *J* letter in the OCR output. This is a systematic error resulting from ground truth that is incorrect for these cases. A better model would result from training on handcorrected ground truth where only Roman numerals have the *I* letter.

3 Other historical ground truth corpora

In the following we mention other historical ground truth corpora which are not part of GT4HistOCR. Only the *Archscribe* corpus of 19th century German Fraktur is directly usable for OCR model training whereas the others would need various amounts of effort to be aligned as line image/transcription pairs. We also give estimates on the amount of material (number of line pairs) potentially available.

¹⁵E.g. Krünitz' Ökonomische Enzyklopädie: <http://www.kruenitz1.uni-trier.de/>,

¹⁶https://abbyy.technology/en/features/ocr/old_font_recognition

¹⁷<http://staatsbibliothek-berlin.de/>

Table 6: The DTA19 Fraktur corpus.

Year	(Short) Title	Author	# Lines
1797	Herzensergießungen	Wackenroder	5150
1802	Offerdingen	Novalis	6198
1804	Fliegeljahre vol. 1	Paul	5332
1815	Elixiere vol. 1	Hoffmann	8008
1816	Buchhandel	Perthes	861
1817	Nachstücke vol. 1	Hoffmann	6578
1819	Revolution	Görres	6178
1821	Waldhornist	Müller	2343
1826	Taugenichts	Eichendorff	7662
1827	Liebe	Clauren	6724
1827	Reisebilder vol. 2	Heine	5980
1827	Lieder	Heine	5873
1828	Gedichte	Platen	5103
1828	Literatur vol. 1	Menzel	8124
1832	Gedichte	Lenau	4446
1832	Paris vol. 1	Börne	5329
1834	Feldzüge	Wienbarg	7805
1835	Wally	Gutzkow	5728
1852	Ruhe vol. 1	Alexis	9314
1852	Gedichte	Storm	2038
1853	Ästhetik	Rosenkranz	14062
1854	Heinrich vol. 1	Keller	9343
1854	Christus	Candidus	2095
1861	Problematische Naturen vol. 2	Spielhagen	6445
1863	Menschengeschlecht	Schleiden	1788
1871	Bühnenleben	Bauer	12008
1877	Novellen	Saar	6354
1879	Auch Einer vol. 2	Vischer	10492
1880	Hochbau	Raschdorff	661
1880	Heidi	Spyri	6210
1882	Sinngedicht	Keller	11209
1882	Gedichte	Meyer	6262
1886	Katz	Eschstruth	6601
1887	Künstlerische Tätigkeit	Fiedler	4983
1888	Irrungen	Fontane	7079
1891	Bittersüß	Frapan	7008
1897	Gewerkschaftsbewegung	Poersch	1476
1898	Fenitschka	Andreas-Salomé	4753
1898	Erinnerungen vol. 2	Bismarck	10339
		Sum:	243,942

da der Eeelige eine sehr kleine, unles
 und blies starke Dampfvolken von sich, daß wir
 sich auch mit freudigem Muthe zu vollbringen in kurtz
 mit einem Sprunge hinter dem Wagen, der Kutscher
 ihr Schweigen immer mehr ausdrücken, als
 zu leiden, denn die Nothheit und Gemeinheit, die Abform
 Karten und Pläne zu zeichnen. Schon waren von
 der mit diesem lästigen Leiden so fatal verwachsen war,
 einem gewissen Stolz wies sie ihm das hochgethürmte
 nach nicht, sondern ganz andere Dinge spielen

Figure 6: A selection of lines of the DTA19 corpus. From top to bottom: 1815, 1817, 1819, 1826, 1835, 1853, 1861, 1879, 1891, 1897.

3.1 The *Archiscribe* corpus

A prime obstacle for generating ground truth for OCR training purposes consists in the segmentation of textual elements on a printed page into text lines. To circumvent this problem, we made use of several open APIs of the Internet Archive¹⁸ to directly retrieve line images from historical books that can be used as image sources for creating ground truth.

The Internet Archive hosts a collection of over 15 million texts, whose scans are sourced from Google Books as well as a number of volunteers and cooperating institutions.¹⁹ For every scanned book, an automated process creates OCR with ABBYY FineReader. While the actual OCR output of this engine for text with Fraktur typefaces is of very low quality, the resulting line segmentation is usually fairly accurate.

To create ground truth from the Internet Archive corpus, a simple crowd sourcing web application, Archiscribe²⁰, is provided. First-time users of the application have to read through a simplified version of the transcription guidelines of the Deutsches Textarchiv²¹. They are then offered the option to pick a certain year between 1800 and 1900 and set a number of lines they want to transcribe.

In order to retrieve these lines from a suitable book, Archiscribe uses the publicly available search API of the Internet Archive²² to retrieve a list of 19th century German language texts and randomly picks a volume that has not yet been transcribed. To determine whether a given text is actually set in Fraktur, a heuristic is used: The OCR text is downloaded and searched for the token *ift*, a common misinterpretation by OCR engines trained on Antiqua fonts of the actual word *ist* (German *ist* = English *is*), which has a high frequency in any German text (of course, real books also contain quotations and other material in Antiqua, as is seen in the second line of Figure 8). If this heuristic results in a false positive (there are some books printed in Antiqua employing a long s), one can just start over. Once a suitable book is found, the desired number of lines²³ are picked at random from the book.

¹⁸<http://archive.org>

¹⁹<https://archive.org/scanning>

²⁰<https://archiscribe.jbaiter.de>, source code: <https://github.com/jbaiter/archiscribe> (MIT license)

²¹<http://www.deutschestextarchiv.de/doku/basisformat/transkription.html>

²²<https://archive.org/advancedsearch.php>

²³user-defined, by default 50



Figure 7: Transcribing a line with Archiscribe

To serve the images to the user, Archiscribe uses the publicly available IIIF Image API endpoint²⁴ of the Internet Archive. As the API allows the cropping of regions out of a given page image hosted by the archive.org server, the application can directly use it for rendering the line images in the user's browser, and no image processing on the Archiscribe server is necessary.

Once a suitable volume has been picked and the lines to be transcribed have been determined, the user is presented with a minimal transcription interface consisting of the line to be transcribed, a text box to enter the transcription and an on-screen keyboard with a number of commonly occurring special characters not available on modern keyboards. To offer more context in difficult cases, the user may opt to display the lines above and below the line to be transcribed (Fig.7).

When all lines have been transcribed, they are submitted to the Archiscribe server, where they are stored alongside with their corresponding line images in a Git repository that is published to the corpus repository on GitHub on every change²⁵.

To ease maintenance of the ground truth corpus a simple review interface is available (Fig.8) where existing transcriptions can be filtered and edited. Due to the use of a Git repository as the storage backend, it is also very easy to keep track of changes in the dataset or to revert some changes in case of vandalism.²⁶

Currently the application is restricted to 19th century German language books from the Internet Archive, but it is planned to add support for the transcription of books sourced from any repository that offers a IIIF API, the number of which is steadily increasing.

The *Archiscribe* corpus of ground truth generated by crowdsourcing with the Archiscribe tool currently consists of 4145 lines from 109 works published across 72 years²⁷ evenly distributed across the whole 19th century. All of the data is available under a CC-BY 4.0 license.

²⁴<https://iiif.archivelab.org/iiif/documentation>

²⁵<https://github.com/jbaiter/archiscribe-corpus>

²⁶Although the application does not require authentication or registration of any kind, this has not been an issue so far.

²⁷[last accessed 31th August 2018]

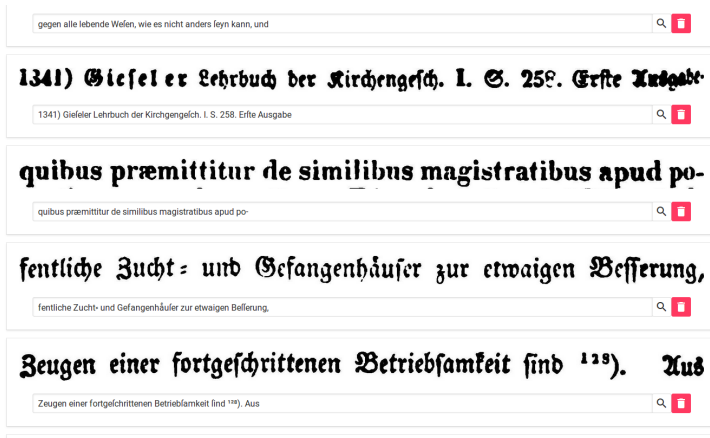


Figure 8: Reviewing an existing transcription with Archiscribe. Often books printed in Fraktur also contain lines in Antiqua, mostly quotations in Latin (second line from top). If they are transcribed as well, the model will be able to recognize mixed Fraktur-Antiqua texts.

3.2 The *OCR-D* ground truth corpus

The OCR-D project funded by Deutsche Forschungsgemeinschaft (DFG) created ground truth of Latin and German printings published between 1500 and 1835 in Germany. This corpus currently consists of one to four pages each of 94 works.²⁸ Data are provided in both TIFF format (page images) and an XML representation in both ALTO and PAGE XML containing the segmentation of the pages in text zones as well as their transcription. In order to produce OCR training data from these files, the text zones of the TIFF images need to be identified by their coordinates contained in the XML files, then these subimages have to be segmented into text lines and matched with the corresponding transcription, also contained in the XML files. We estimate that this dataset currently contains 300 pages and a total of approximately 10,000 lines.

3.3 The full *DTA* corpus

There is also the complete DTA corpus of currently 4,422 volumes in German with transcriptions on page level covering the period 1500 to 1900. To produce OCR ground truth fully automatically one needs to segment page images and heuristically match the existing line transcriptions against segmented text line images. Work along this direction is already under way. The amount of available lines is approximately 30 million²⁹.

²⁸<http://www.ocr-d.de/sites/all/GTDaten/IndexGT.html> [last accessed 26 August 2018]

²⁹<http://www.deutschestextarchiv.de/doku/ueberblick>

3.4 Ground truth from the *IMPACT* project

The EU-funded *IMPACT* project (2008–2012) collected historical ground truth in the form of semantic regions of page images (such as text, image, footnote, marginal notes, page number etc.) for the task of automatic page segmentation (*document analysis*) as well as transcriptions for the text regions of ca. 45,000 pages³⁰. Transcribed ground truth is available for several European languages³¹. There may be as many as 1 million lines available, but unfortunately the ground truth comes under a variety of licenses depending on the contributing institution and can currently only be downloaded page by page.

4 Notes on transcription guidelines for OCR

To produce training data for OCR where a machine will decide what label to attach to a printed glyph, the golden rule is: *The same glyph must have the same transcription, even if the glyph has different context dependent meanings*. Otherwise, the machine will get confused and randomly output one of the different characters or character sequences it has learnt to associate with the glyph. Consequently the single Fraktur glyph for the letters *I* and *J* can only have one character representation, not two, and ambiguous and context dependent abbreviations must not be resolved. E.g., a vowel with tilde above in Early Modern Latin could either mean (vowel+m) or (vowel+n). A further example is provided by the r-hook above letter *d* in Table 7. Also, ignoring line endings of printed lines and merging hyphenated words will destroy the correspondence between printed line image and transcription needed for model training.

This makes most of the existing transcriptions of historical documents which resolve abbreviations, merge hyphenated words at line endings, correct printing errors, and modernize historical spellings unusable for OCR purposes. What is needed instead is a *diplomatic transcription*, i.e. a transcription of printed glyphs to characters with no or minimal editorial intervention³².

But even if we transcribe diplomatically, there is still room for a decision on the level of detail we want to transcribe, e.g. if we want to record the usage of long s (*ſ*) or rounded r (*r rotunda*). The collection of explicit recordings of such decisions are called transcription guidelines. They are indispensable to ensure a consistent text, both over time and between different people transcribing parts of same document. They are also necessary if you want to pool data from different corpora which have been transcribed by different guidelines. You have to inspect the guidelines in order to regularize different data sets to a common norm.

Explicit transcription guidelines exist for the Reference Corpus ENHG³³, texts from DTA³⁴, and RIDGES³⁵. All other corpora had to be made internally consistent with our Perl script. The correctness of the data will determine the predictive power of any machine model trained on it. We define the correctness of a transcription as its adherence to predefined, internally consistent transcription

³⁰<https://www.primaresearch.org/www/media/datasets/ImpactRepositoryPoster.pdf>

³¹See <https://www.digitisation.eu/>






³²<http://www.stoa.org/epidoc/gl/latest/trans-diplomatic.html>

³³Not yet publically available.

³⁴See Footnote 22

³⁵https://www.linguistik.hu-berlin.de/de/institut/professuren/korpuslinguistik/forschung/ridges-projekt/documentation/download-files/pubs/ridgesv8_2018-04_06.pdf, pp. 248 ff.

Table 7: Extract from the transcription guidelines of the Reference Corpus ENHG. The transcript column shows examples of the linguistically motivated transcription, the UTF-8 column represents our interpretation for OCR purposes.

Example	Transcript	UTF-8	Description
	o\e	ö	vowel modifier
	o_r	or	ligature
	d'	ð	d with abbreviation of <er>, <r>, <ir>, <re>, <ri>
	me<t>	met	letters that are difficult to read
	A=	A=	word-internal line break

guidelines and not as the level of detail which it records. We emphasize this point because we have been set back by inconsistent data produced by researchers, students and the public alike.

Note that a linguistically motivated transcription (such as in the *Reference Corpus Early New High German* or the *Deutsches Textarchiv*) might very well choose to transcribe similar looking glyphs by differently looking characters for a specific use case such as search. In order to use these transcriptions for OCR model training one needs to normalize to just one alternative (*j*, in our case). Examples of differences between a linguistic transcription and a transcription for OCR training are shown in Table 7 for the Reference Corpus ENHG.

5 Conclusion

Historical OCR has been advanced to a state where even very early printings from the 15th century can be recognized by individually trained models with a character recognition rate of 98% and above. To be practical on a large scale, however, pretrained models are needed that result in recognition rates >95% without any prior training requirement. As long as we lack an automatic method to revive historical fonts to build large synthetic corpora the construction of pretrained models rests on the availability of historical ground truth. The *GT4HistOCR* dataset is put forward to allow experimentation and research under a permissive CC-BY 4.0 license and is a first step for the construction of widely applicable pretrained models for Latin and German Fraktur. We hope that other researchers will follow our example and make their ground truth available under an open source license in directly usable form for OCR training.

Acknowledgments

We are grateful to our colleagues Phillip Beckenbauer for aligning the ground truth of the *Early New High German Corpus* to the printed text lines of the respective books and the training of a mixed

model, to the *Narragonien digital* project (Joachim Hamm/Brigitte Burrichter) providing corrected transcriptions as ground truth, especially by Christine Grundig and Thomas Baier and their collaborators, and to the many students and collaborators of the *RIDGES Corpus* at Humboldt University Berlin. Uwe Springmann produced most of the Early Modern Latin Corpus, Johannes Baiter is the author of the Archiscribe tool. The Kallimachos project has been funded by BMBF under grant no. 01UG1415A and 01UG1715A.

References

- Breuel, T. M., Ul-Hasan, A., Al-Azawi, M. A., and Shafait, F. (2013). High-performance OCR for printed English and Fraktur using LSTM networks. In *2th International Conference on Document Analysis and Recognition (ICDAR), 2013*, pages 683–687. IEEE.
- Carter, H. (1969). *A View of Early Typography: Up to about 1600: the Lyell Lectures 1968*. Clarendon Press.
- Fischer, A., Wüthrich, M., Liwicki, M., Frinken, V., Bunke, H., Viehhauser, G., and Stolz, M. (2009). Automatic transcription of handwritten medieval documents. In *15th International Conference on Virtual Systems and Multimedia, 2009 (VSM'09)*, pages 137–142. IEEE.
- Odebrecht, C., Belz, M., Zeldes, A., and Anke (2017). RIDGES herbology: designing a diachronic multi-layer corpus. *Language Resources and Evaluation*, 51(3):695–725.
- Reul, C., Dittrich, M., and Gruner, M. (2017a). Case study of a highly automated layout analysis and ocr of an incunabulum: 'der heiligen leben' (1488). In *Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage, DATeCH2017*, pages 155–160, New York, NY, USA. ACM.
- Reul, C., Springmann, U., Wick, C., and Puppe, F. (2018). Improving OCR accuracy on early printed books by utilizing cross fold training and voting. In *13th IAPR International Workshop on Document Analysis Systems, DAS 2018, Vienna, Austria, April 24-27, 2018*, pages 423–428.
- Reul, C., Wick, C., Springmann, U., and Puppe, F. (2017b). Transfer learning for OCRopus model training on early printed books. *027.7 Zeitschrift für Bibliothekskultur / Journal for Library Culture*, 5(1):38–51.
- Springmann, U. and Fink, F. (2016). CIS OCR Workshop v1.0: OCR and postcorrection of early printings for digital humanities.
- Springmann, U., Fink, F., and Schulz, K. U. (2015). Workshop: OCR & postcorrection of early printings for digital humanities.
- Springmann, U., Fink, F., and Schulz, K. U. (2016). Automatic quality evaluation and (semi-) automatic improvement of OCR models for historical printings. *ArXiv e-prints*.

- Springmann, U. and Lüdeling, A. (2017). OCR of historical printings with an application to building diachronic corpora: A case study using the RIDGES herbal corpus. *Digital Humanities Quarterly*, 11(2).
- Springmann, U., Najock, D., Morgenroth, H., Schmid, H., Gotscharek, A., and Fink, F. (2014). OCR of historical printings of Latin texts: problems, prospects, progress. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, DATECH '14*, pages 57–61, New York, NY, USA. ACM.
- Springmann, U., Reul, C., Dipper, S., and Baiter, J. (2018). GT4HistOCR: Ground Truth for training OCR engines on historical documents in German Fraktur and Early Modern Latin.
- Ul-Hasan, A. and Breuel, T. M. (2013). Can we build language-independent OCR using LSTM networks? In *Proceedings of the 4th International Workshop on Multilingual OCR, ICDAR 2013, Washington, D.C., USA, August 24, 2013*, pages 9:1–9:5.

Author index

Chantal Amrhein
Institute of Computational Linguistics
University of Zurich, Switzerland
`chantal dominique.amrhein@uzh.ch`

Johannes Baiter
Munich Digitization Center
Bavarian State Library, Munich, Germany
`johannes.baiter@bsb-muenchen.de`

Simon Clematide
Institute of Computational Linguistics
University of Zurich, Switzerland
`simon.clematide@cl.uzh.ch`

Stefanie Dipper
Department of Linguistics
Ruhr University Bochum, Germany
`dipper@linguistics.rub.de`

Lenz Furrer
Institute of Computational Linguistics
University of Zurich, Switzerland
`furrer@cl.uzh.ch`

Frank Puppe
Department for Artificial Intelligence and Applied Computer
Science
University of Würzburg, Germany
`puppe@informatik.uni-wuerzburg.de`

Christian Reul
Department for Artificial Intelligence and Applied Computer
Science
University of Würzburg, Germany
`christian.reul@uni-wuerzburg.de`

Uwe Springmann
Center for Information and Language Processing
Ludwig Maximilian University of Munich, Germany
`uwe@springmann.net`

Martin Volk
Institute of Computational Linguistics
University of Zurich, Switzerland
`volk@c1.uzh.ch`

Christoph Wick
Department for Artificial Intelligence and Applied Computer
Science
University of Würzburg, Germany
`christoph.wick@uni-wuerzburg.de`