Supervised OCR Post-Correction of Historical Swedish Texts: What Role Does the OCR System Play?

Dana Dannélls[1] and Simon Persson[2]

[1] Språkbanken Text, Department of Swedish
University of Gothenburg, Sweden
dana.dannells@svenska.gu.se
[2] Department of Computer Science and Engineering
Chalmers University of Technology, Sweden
simonpersson@flaskpost.org

Abstract. Current approaches for post-correction of OCR errors offer solutions that are tailored to a specific OCR system. This can be problematic if the post-correction method was trained on a specific OCR system but have to be applied on the result of another system. Whereas OCR post-correction of historical text has received much attention lately, the question of what role does the OCR system play for the post-correction method has not been addressed. In this study we explore a dataset of 400 documents of historical Swedish text which has been OCR processed by three state-of-the-art OCR systems: Abbyy Finereader, Tesseract and Ocropus. We examine the OCR results of each system and present a supervised machine learning post-correction method that tries to approach the challenges exhibited by each system. We study the performance of our method by using three evaluation tools: PrimA, Språkbanken evaluation tool and Frontiers Toolkit. Based on the evaluation analysis we discuss the impact each of the OCR systems has on the results of the post-correction method. We report on quantitative and qualitative results showing varying degrees of OCR post-processing complexity that are important to consider when developing an OCR post-correction method.

Keywords: Historical Newspaper Corpus, OCR, Machine Learning.

1 Introduction

Different cultural institutions rely on state-of-the-art Optical Character Recognition (OCR) systems to automatically process scanned documents and images. For historical documents and images the quality of the results of the OCR vary greatly depending on the system that was used to process the text. One way to address this has been to apply post-processing computational methods for correcting the errors that were caused by the OCR system [4, 7, 8, 13].

Current approaches for correcting OCR errors offer solutions that are tailored to a specific OCR system [7]. This can be problematic if the post-correction method was trained on a specific OCR system but has to be applied on the results of another. Whereas OCR post-correction of historical text has received much attention lately, the

question of what role does the OCR system play for the post-correction method has not been addressed. In this paper we present a post-processing method that has been developed on the basis of results of three state-of-the-art OCR systems: Abbyy Finereader,[3] Tesseract,[4] and Ocropus.[5] We discuss how the errors made by each system effect the results of the post-processing method.

Our method utilizes a supervised machine learning approach. Supervised machine learning is an approach which uses ground truth material, i.e. manually transcribed material, to train a model and then test how well the model performs on a new material. Thus, given a set of training data such as a set of words an algorithm can learn to predict whether the word contains an error and tries to correct it. There have been considerable efforts to develop a machine leaning models for OCR post-processing [1, 9, 12, 16]. These attempts consider post-processing as an integrated part of the OCR system. Other approaches consider the task as an independent step that is applied on the results of any OCR system regardless of the system [18]. Successful attempts on Devnagari and historical English considered the supervised machine learning model Support Vector Machine (SVM) [1, 9] and have shown that the performance of an SVM model is equivalent to the performance of a neural network. To our knowledge, an SVM model has not been applied for correcting OCR errors of historical Swedish text.

Methods that have been developed to post-process the results from an OCR system have reported on the performance of the method against one OCR system. Observations were merely based on the results of one system, most notably Tessearct or Abbyy. Furthermore, authors showed that the performance of the method differs depending on the evaluation tool in question [11]. In this work we therefore study the performance of our method by using three evaluation tools: PrimA,[6] Språkbanken evaluation tool [2] and Frontiers Toolkit [5].

Språkbanken evaluation tool uses two metrics, *Character error rate* (CER) and *Word error rate* (WER). Both of the metrics are calculated the same way, by figuring out how many character/word errors there are compared to the total number of characters/words. PrimA and Frontiers Toolkit measure character accuracy (CACC) and word accuracy (WACC). While CACC is the result of the comparison of two characters, WACC is defined by comparing two lists of words, one from the OCR output and the another from the ground truth.

2 Datasets

We have experimented with three datasets of 19th century Swedish text. Two of them which are outlined in Section 2.1 were used for training and testing our SVM method. The third dataset which is outlined in Section 2.2 comprises a collection of historical corpora and lexicons from which we build frequency and tri-gram databases that were used to develop our method.

[3] `http://finereader.abbyy.com/` (version 12)

[4] `https://github.com/tesseract-ocr/` (version 4.0)

[5] `https://github.com/tmbdev/ocropy` (version1.3.3)

[6] `https://github.com/PRImA-Research-Lab` is an extension of a tool that has been developed as a part of the IMPACT project

2.1 Training and Testing Material

Then Swänska Argus is a work that was written by Olof von Dalin and was published in Stockholm in 1732–1734. The text marks a great change in Swedish language mainly because it has been written in a point were Swedish transformed from the historical Old Swedish to the Modern Swedish. This was a period when changes in spellings and inflectional morphology took place. The material contains 177 images which are dominated with deterioration, mixture of typefaces (Blackletter and Schwabacher) and difference in font size. Access to the ground truth of the material, comprising 43,640 words is available.

Grepect is a collection of randomly selected images from Gothenburg University library spanning the period from 17th to 19th century. Similar to Then Swänska Argus, the quality of the scanned images is low. The scans were manually transcribed by an external company called Grepect and therefore henceforth is referred to as Grepect. The dataset comprises 48,112 words.

2.2 Corpora and Lexicons

Project Runeberg is an OCR project that started in 1992 at Linköping University with the aim to scan books primarily from Sweden but also from other Nordic countries. After the books are scanned the text is often proofread by volunteers. Therefore, these texts are reliable for statistical analysis.[7] We have extracted the pages ranging from 17th and 19th century to build our frequency database, in total this set contains nearly 3 million unique words with their frequencies. The benefit of having a large database is that there exist many potential candidates and the chance of finding the correct word is higher. The shortcoming is that the method might be greedy if the search space is not limited.

In addition, we added to this database more data that was extracted from two lexicons: Swedberg, an electronic lexicon over 18th century, covering 16.158 entries, and Dalin, an electronic lexicon over 19th century, covering the morphology of 509,924 entries of late modern Swedish [3].

2.3 Baselines

The two datasets we describe in Section 2.1 were processed with three OCR systems: Abbyy FineReader 12, Tesseract 4.0 and Ocropus 1.3.3. The amount of tokens produced by each system varies considerably. Table 1 provides a summary of the amount of tokens each system produced and the differences between the amount of tokens containing non-alphanumeric characters, vowels and more than one uppercase letter that have been recognized by each system. We learn that not only the amount of tokens that were produced by each system varies but also the amount of tokens with different characteristics differs between each system.

When we inspect some of the word features that are typical for each OCR system we find that there is a huge difference between the output results for each system.

[7] At the time of writing, i.e. November 2019, the Project Runeberg database contained about 384,000 proofread pages.

Table 1. Analysis of the tokens that were recognized by each OCR system.

Fearure	Abbyy	Tesseract	Ocropus
token count	15992	51852	92690
non-alphanumeric characters	16102	51350	92231
vowels	22871	79714	14645
uppercase letters	1904	6891	9556

The amount of tokens that was produced by Ocropus is six times as high compared to Abbyy and double as high compared to Tesseract. The amount of tokens containing non-alphanumeric characters exhibits the same imbalance between the systems.

When we compare the amount of tokens containing more than one vowel we find that Tesseract has the highest amount. The amount of tokens containing more than one uppercase letter is highest for Ocropus which is twice the amount of Tesseract. Given these differences we assume that if these features are captured in our method the method will yield good results for the different systems.

Further, we examine the length of the tokens that were produced by each system. Figure 1 presents the results of our analysis. The majority of tokens recognized by all three systems are four characters long. All systems tend to recognize longer tokens or rather merge two or more tokens into one, seven and eight characters long tokens are common. Compared to the ground truth where the majority of tokens is less than seven characters.

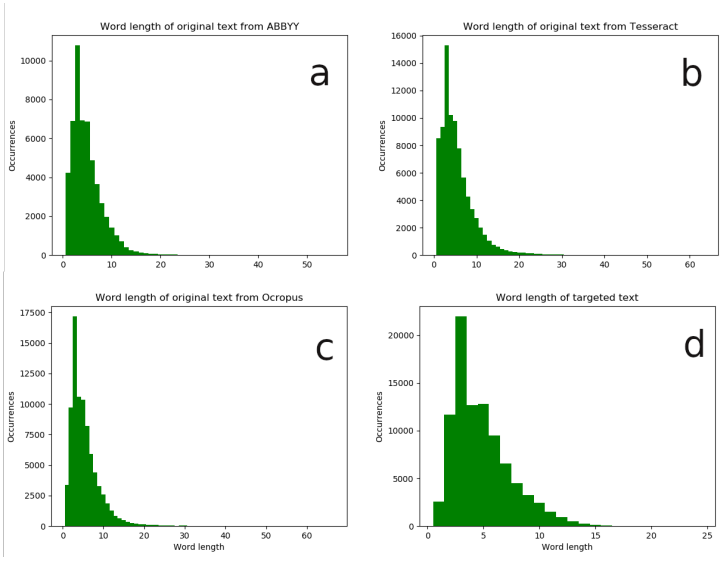

Fig. 1. OCR performance with respect to token length for three different OCR systems: Abbyy (a), Tesseract (b), and Ocropus (c), and the ground truth material (d).

We evaluated the results of each system against the ground truth material, the results are shown in Table 2. These results form the baseline on which our method will try to improve upon. Because this is the first work that experiment with three OCR systems against this particular dataset, there is no other previous baseline to compare these baseline results with. According to these evaluation results, Ocropus is the system with the best performance.

Table 2. The baseline results by three evaluation scripts for each OCR system.

	OCR Frontier Toolkit		Språkbanken evaluation script		PrimA Text Evaluation	
System	**CACC**	**WACC**	**CER(%)**	**WER (%)**	**CACC**	**WACC**
Ocropus	0.80	0.58	17.68	59.60	0.80	0.41
Tesseract	0.64	0.35	34.08	76.37	0.64	0.25
Abbyy	0.58	0.24	39.30	86.59	0.62	0.2

3 Post-Processing Method

Our post-processing method follows a simple system architecture. In the first step the data is pre-processed to remove unnecessary tags such as <h1>, <fr>, <aq> which have been inserted into the text to indicate visual changes such as font-change or headings. Also some irregularities are removed from the results of the OCR system like redundant white-spaces. In the second step the processing is divided into two blocks, one for detecting and one for correcting OCR errors. The detection stage utilizes an SVM algorithm. The error correction uses frequencies from corpora and lexicons in order to extract a list of candidates for replacing the erroneous words,[8] and levenshtein distance algorithm to find the most likely candidate to replace the erroneous one.

3.1 Error Detection

There exist a number of different methods for deciding whenever a word is erroneous or valid. The most obvious approach is to perform a lexicon lookup [12]. For historical text this approach has proven to be insufficient because of out-of-vocabulary words [8]. Another approach is to represent the word with a set of features (also called metrics) and train a model to classify the word based on different measurements that indicate whether a word is erroneous or valid [11].

Features could be divided into two classes, depending on what is required to compute them. Classes that previous authors have experimented with are either word based or statistical based [12, 15]. Khirbat [12] has experimented with word based features that, among others, return the number of non-alphanumeric characters in the given word. Statistical based features have been explored by [15]. Following the experiments

[8] We use the terms tokens and words interchangeably because resulted string of an OCR system is not necessarily a lexical word.

from previous work and the quantitative analysis of the output of the three OCR systems (as shown in Table 1) we computed 8 features to train our word classifier on:

- Number of non-alphanumeric characters in a word. Returns the number of characters in the word that are not alphanumeric.
- Number of vowels in a word. Returns the number of vowels in the word.
- Frequency of the word. A large list of the 100000 most frequent words is extracted from the Runeberg dataset. If the word is present in the list, the frequency is returned, otherwise the value is set to zero.
- "Swedishness" of a word. A list of 15000 tri-gram character frequencies extracted from the Runeberg dataset. For a given word frequencies are multiplied, the value of unseen tri-grams is set to 0.1.
- Word length. Returns 1 if the word is over 13 characters, otherwise 0.
- Number of uppercase characters. Returns the number of uppercase characters in the word.
- Contains numbers. Returns 1 if the word contains one or more numbers, otherwise 0.
- Validity of the word. Returns 1 if the word is valid and 0 otherwise. This value is only present in the training-data.

We complied a training data consisting av 104,008 feature vectors representing both error words and correct ones. Correct words are extracted from the manual transcription. Error words are extracted from the output results of Abbyy, Tesseract and Ocropus. Since this is done completely automatically, some false negatives are present in the set.

3.2 Error Correction

Our approach for correcting errors is divided in two stages. First, we search for word candidates to replace the erroneous one. Second, we rank the candidates and choose the word with the highest probability [12]. A fundamental part of the correction step is access to a large database of potential word candidates.

Another important fact that is illustrated in Figure 1 is the tendency of the OCR systems to merge two or more adjacent words into one [6]. This was addressed before by applying a probabilistic noisy channel model that combines an error model and a language model [13]. This approach was tested on Swedish but has shown to have high computational costs [2]. Our method simplifies the approach by performing binary splits, thus the word is split only once, though in different possible ways. As a result, the method finds a number of candidates in each split. For each split for the given word a *candidate search* is performed. The search starts by calculating the Levenshtein distance (LD) [14] between all the words in the dataset and splits. The algorithm choose the word with the lowest LD in the dataset. If the split consists of two words, an additional edit-distance is added to accommodate for the action of splitting the word.

When considering this candidate search it is crucial to specify the minimum and maximum LD for each candidate. If the minimum LD is set to 0 the system will not correct real-word errors since there will always be a candidate with LD$=0$ and that will be the winning candidate. But since there exist false negatives, the error detection stage does not yield perfect result, some false negatives and a minimum LD of 0 would potentially not be corrected. Contrary, a minimum LD of 1 would potentially correct some

real-word errors but would wrongfully correct all false negatives. When considering a maximum LD the question becomes about computational time rather then correctness. It is very unlikely that candidates with large LD of the considered word will be the winning candidate or the correct word. Therefore a reasonable high threshold number is needed if the candidate should not be considered. To address this, the maximum LD between the considered word and a candidate is set to 8.

The ranking of candidates is done primarily by the LD but further distinction is needed when two or more candidates have the same LD. This distinction is done by comparing the frequency of the candidates. The frequencies are taken from the aforementioned word database presented in Section 2.2. If the candidate consists of two words an average of the two is calculated.

4 Evaluation of the Method

We propose a modular method. Each part of the method is evaluated separately. Also the performance of the method as whole is evaluated.

4.1 Error Detection

The quality of the features. When generating the training data each word is processed by two filters called *last character* filter and only *non-alfa* filter before the feature vector is calculated. Last character filter removes the last character of the word if it is a punctuation, e.g. “example,” becomes “example”. Non-alfa filter removes all words that only contains symbols or numbers i.e. “?” or “1888”. We evaluate the effect of these filters on a subset of the training data. First, the performance without applying any filters was measured. Second, each filter was tested individually. Finally both filters were used together. During these tests the size of both the tri-gram and word database was set to a default value of 10000. Each feature is evaluated by considering the greatest percentage difference according to the following equation:

$$1 - \frac{\text{avg}_{low}}{\text{avg}_{high}}$$

where avg_{high} is the numerical greatest of the two averages, avg_{low} is the contrary lowest of the two averages. We call this measure *Percent difference mean*. Figure 2 shows the result of each feature with different configurations. As we can see, the performance of most features remains unchanged except for the *#Alphanumeric* and the *#Numbers* features which are improved. In (d) we see that both *#Alphanumeric* and *#Numbers* are improved compared to the initial results without any filters in (a). We conclude that the optimal configuration for filtering the training data is to use both *last-char* and *only-alfa* filters.

When we look closer on the effect of the features given the characteristics of the data that was produced by the three OCR system we do not find it surprising that the combination of both filters produce the best results. It would have become clearer if we tested these configuration on each of the system’s results separately. The performance of the SVM based on these four different word feature configurations is presented in Table 3.

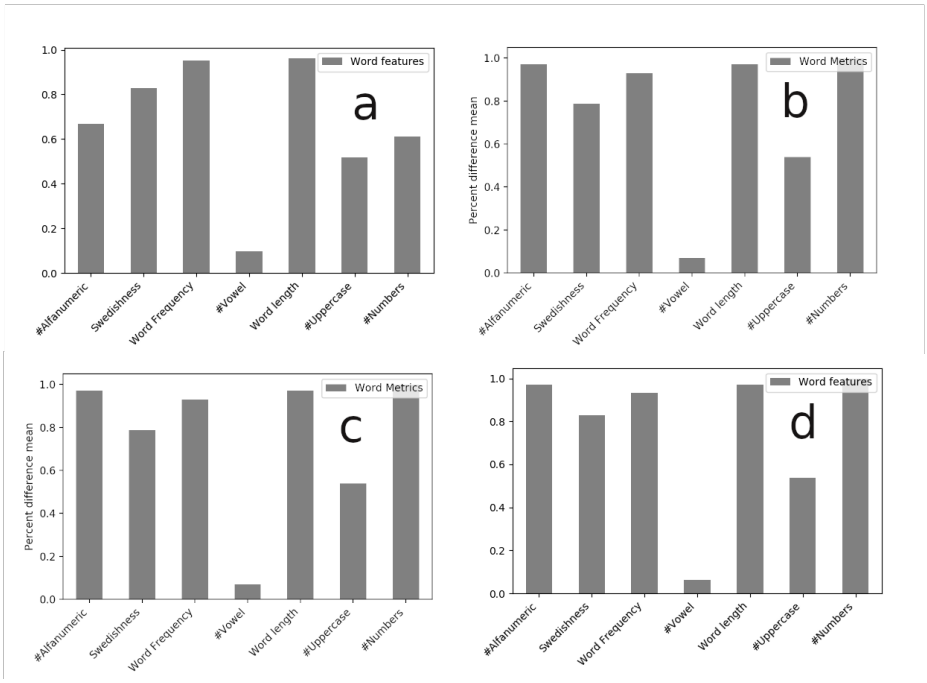

Fig. 2. Word features with different configurations: (a) without any filters, (b) with the *last character* filter only, (c) with the *only-alfa* filter only, (d) with both *last-char* and *only-alfa* filters. The performance (y-axle) is measured with *percent difference mean*.

The SVM model. The performance of the SVM is highly dependent on the selection of parameters, but since the choice of the parameters depends on the choice of the kernel function, the kernel function should be selected first. Hence, the evaluation of the SVM was done in two steps, first the choice of kernel was made, second the search of optimal parameters was conducted. Both of these steps were evaluated using the three metrics: Precision, Recall and F1-score. Precision is the percentage of the systems' provided correct examples, recall is the percentage of examples actually present in the input that were identified correctly by the system. F1-score is the harmonic average between precision and recall. In the case of OCR, precision is the value we are mostly interested in, but of course finding as many correct examples as possible is also important in this context, therefore the F1-score will be considered.

When the SVM solves a classification problem the data points (x), i.e. the tokens, are mapped to a higher dimensional space where the problem becomes linear separable by a linear hyperplane. The mapping to the higher dimension is achieved using *kernel function*, $\phi(x)$ [17].

$$\text{i.e. } x \longrightarrow \phi(x)$$

There exist a number of different kernels, each of which maps the points differently. The method for selecting the kernel function consists of testing all available kernels

Table 3. The impact of four word feature configurations on the performance of the SVM.

Word metric configuration:	**Precision**	**Recall**	**F1-score**
Only #Alfanumeric and #Word frequency	0.67	0.62	0.56
All metrics except #Vovel and #Uppercase	0.77	0.65	0.63
Only high performing metrics	0.77	0.65	0.61
All metrics	0.77	0.67	0.63

and comparing their performance. The most common kernel functions that we experimented with are displayed below, where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ and γ, r and d are kernel parameters [10].

- Linear function: $K(x_i, x_j) = x_i^T x_j$
- Polynomial function: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
- Radial based function: $K(x_i, x_j) = exp(-\gamma ||x_i - x_j||^2), \gamma > 0$
- Sigmoid function: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

The kernel selection was done by an exhaustive search of all available kernels. Any parameters that are offered by the different kernels were set to default values in order to isolate the choice to be only about kernel selection. The training data was set to 10000 input vectors. The size of the training data was set to a relative low value in order to save some computational time. The performance of the *radial basis* kernel function yielded the best result with 0.77 precision 0.70 Recall and 0.67 F1-score, and thus was the one we considered for the parameter selection.

Parameter selection When the kernel was chosen the other parameters had to be determined. This was done with the grid search algorithm that is a part of the scikit-learn library,[9] which conducts an exhaustive search over predefined lists of possible values for parameters. The grid search was done in an iterative fashion by first selecting few values in a big range and then narrowing the range in stages until a satisfactory value is found. The radial basis kernel have two parameters, c-value and γ-value. We found that the optimal values for these are $c = 1000$ and $\gamma = 1000$, with the resulting performance of F1-score 0.75.

4.2 Error Correction

The resulting performance of the error correction was evaluated using the optimal configuration for all previous parts of the system. To save some computational time only a sample of all the available texts was considered in the different tests. We perform an evaluation on 10 pages from Argus and 10 pages from Grepect, these are combined with three OCR systems: Abbyy, Tesseract, and Ocropus. The six possible combinations give the sample size of 60 pages.

[9] `https://scikit-learn.org/stable/`

Input filtering. Similar to the training data the input to the system is filtered through two filters: *last character* and *non-alfa* filters. The results of applying these filters are given in Table 4. The two latter configurations, *non-alpha* filter and both filters perform very similar but *both filters* configuration was selected as the optimal configuration since the difference in word accuracy from PrimA text evaluation is substantial.

Table 4. The results of applying different filters.

	OCR Frontier Toolkit		Språkbanken evaluation script		PrimA Text Evaluation	
Filter:	**CACC**	**WACC**	**CER(%)**	**WER (%)**	**CACC**	**WACC**
No filter	0.63	0.36	36.63	84.10	0.61	0.18
Last character filter	0.65	0.39	33.83	82.38	0.65	0.20
Only non-alfa filter	0.66	0.39	32.89	78.90	0.65	0.23
Both filters	0.65	0.39	33.72	77.68	0.65	0.26

Error correction algorithm. There exist two versions of the error correction algorithm. The first and most simple algorithm does not consider any splits of the erroneous words. The second and upgraded version considers all splits of the erroneous word and can therefore correct words that are wrongfully merged. Because the size of the database effects the performance of the error correction algorithm we experimented with different database sizes. We found that the most optimal configuration is achieved with a database size of 10,000 candidates. Considering that computational time is proportional to the amount of words in the database, this was the fixed size during the evaluation of the error correction algorithm. Furthermore, both algorithms were tested with different minimum LD threshold. The only thresholds that are reasonable to test are 0 and 1, and thus those are the only values we considered. The result of the different configurations is displayed in Table 5. According to Språkbanken evaluation script and PrimA text evaluation the no split algorithm yields better result on word level but worse on character level. However, according to the OCR frontier toolkit split performs better on both word and character levels. The split algorithm performs better on character level according to all three evaluation-tools, it also performs better on word level according to the OCR frontier toolkit. We conclude it is the optimal algorithm combined with the minimum edit-distance 0.

Table 5. Evaluation of two error correction algorithms combined with different minimum edit distance values.

	OCR Frontier Toolkit		Språkbanken evaluation script		PrimA Text Evaluation	
Algorithm:	**CACC**	**WACC**	**CER(%)**	**WER (%)**	**CACC**	**WACC**
No split, minimum edit-distance=0	0.64	0.37	34.96	74.84	0.63	0.29
No split, minimum edit-distance=1	0.64	0.37	35.19	75.51	0.63	0.29
Split, minimum edit-distance=0	0.65	0.39	33.72	77.68	0.64	0.27
Split, minimum edit-distance=1	0.65	0.38	33.90	77.56	0.64	0.27

5 Results

Table 6 shows the results of applying our method with optimal configurations on the results from the three OCR systems. Contrary to previous evaluations this evaluation was conducted on all available pages in both datasets. The value that is displayed to the right, in either green or red, is how much higher or lower the result is compared to the baseline, given in Table 2. Out of the eighteen different values, seven show an improvement. Six of the improvements were made on data generated from Tesseract. Unfortunately eleven values have not been improved, but the performance has rather decreased. The results are visualized with histograms for each system in Figure 3.

Table 6. The performance of our post-processing method compared to the baseline, in parentheses.

	OCR Frontier Toolkit		Språkbanken evaluation script		PrimA Text Evaluation	
System	**CACC**	**WACC**	**CER(%)**	**WER (%)**	**CACC**	**WACC**
Ocropus	0.75 (red-0.05)	0.56 (red-0.02)	24.37 (red+6.69)	66.49 (red+6.89)	0.75 (red-0.05)	0.38 (red-0.03)
Tesseract	0.65 (green+0.01)	0.36 (green+0.01)	33.98 (green-0.10)	70.86 (green-5.51)	0.65 (green+0.01)	0.27 (green+0.02)
Abbyy	0.56 (red-0.02)	0.26 (green+0.02)	92.34 (red+5.75)	42.79 (red+3.49)	0.55 (red-0.06)	0.09 (red-0.1)

A closer look at the successfully generated tokens in the post-processed texts shows that the method successfully recognizes shorter tokens that are up to 4 characters. A statistical overview of the results shows that 4 character long tokens constitute half of the strings in each of the three OCR results. When we compare the correctly post-processed tokens that are up to 4 characters long, we learn that the method successfully recognizes and corrects them regardless of the result from the OCR system, some examples are listed in Table 7.

Table 7. Examples of correctly recognized words for each OCR system.

OCR-token	**Corrected-word**	**Ocropus**	**Tesseract**	**Abbyy**
cn	en	58	2	26
pd	på	34	1	38
rn	en	11	1	2
mcd	med	29	5	0
ndr	när	16	2	8
ych	och	17	4	0
Hwad	Hwar	30	29	0
fttt	fått	0	0	29

The OCR results from each system vary significantly, when we look at the statistics over how many of the correct words are included in our database, we learn that the percentage of words that are found in the database is highest for Tesseract. This also explains why the performance of our post-processing method only improves the OCR result from Tesseract. A more objective evaluation would be to evaluate the performance only on tokens that are up to 4 characters long.

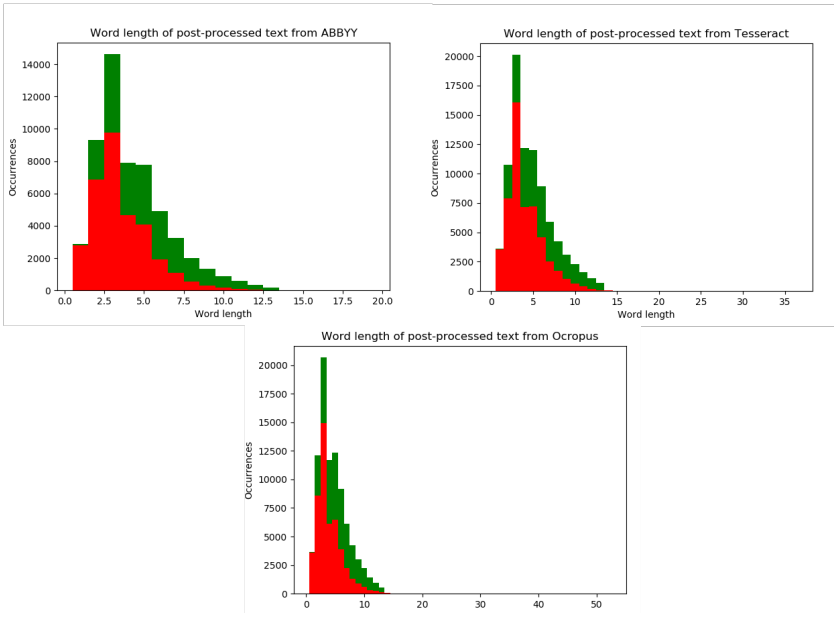

Fig. 3. Histograms showing the token length distributions for three OCR systems after the post-processing step. The red area indicates how many of the words have been correctly recognized.

6 Conclusions and Future Work

In this paper we show the great challenge that is involved in trying to develop a post-correction method that is not tailored to any particular OCR system. We present a post-correction method for improving historical Swedish text that tries to address some of the differences we observed. The overall performance is evaluated using three different evaluation systems. The method shows an improvement for seven of the eighteen possible metrics. Most of these improvements are made on the text that was processed by Tesseract. One reason for that is because our lexicons seem to cover most of the words that have been misrecognized by this system. Our analysis shows that our method seems to perform best on short words up to 4 characters. The conclusion is that our OCR method is not biased towards any OCR system but excels at correcting specific types of errors.

Although we anticipated that our filters and features should result in a significant improvement, they have not been sufficient for improving the results of the OCR systems, at least not for the type of text we examined in this study. In this work we only experimented with two types of word features: word and statically based. For future work, features that cover consecutive consonants as well as context based features [15] should also be considered. Another possible improvement is to change the *word length* feature to simply return the word length. This would possibly give a more nuanced feature that could provide more information to the SVM. This feature could in particular be useful for tailoring the result from each system.

The error correction algorithm lacks the ability to remove words. This is a desired feature when correcting historical texts especially because the OCR systems, most notably Abbyy tend to misrecognize figures appearing in the image, decorative patterns or ink bleed-through. This causes many erroneous words that should not be corrected but rather removed. As mentioned previously, the input filters partially solves this problem but a more effective solution is needed. During error correction the algorithm splits the word one time. An improvement here would be to increase the number of possible splits. Introducing several splits is in particular relevant for improving the results of Abbyy and Ocropus.

Acknowledgements

The research presented here was partially supported by Språkbanken and Swedish CLARIN (Swe-Clarin), a Swedish consortium in Common Language Resources and Technology Infrastructure (CLARIN), grant agreement 821-2013-2003.

References

1. Arora, S., Bhattacharjee, D., Nasipuri, M., Malik, L., Kundu, M., Basu, D.K.: Performance comparison of SVM and ANN for handwritten devnagari character recognition. IJCSI International Journal of Computer Science Issues **7**(6) (2010).
2. Borin, L., Bouma, G., Dannélls, D.: A free cloud service for OCR / En fri molntjänst för OCR. Tech. rep., Department of Swedish, University of Gothenburg (2016).
3. Borin, L., Forsberg, M.: A diachronic computational lexical resource for 800 years of Swedish. In: Language technology for cultural heritage, pp. 41–61. Springer:Berlin (2011).
4. Breuel, T.M., Ul-Hasan, A., Al-Azawi, M.A., Shafait, F.: High-performance OCR for printed english and fraktur using lstm networks. In: 12th International Conference on Document Analysis and Recognition. pp. 683–687. IEEE (2013).
5. Carrasco, R.C.: An open-source OCR evaluation tool. In: Proc. of the First International Con-ference on Digital Access to Textual Cultural Heritage. pp. 179–184. DATeCH '14, NY, USA (2014).
6. Clematide, S., Furrer, L., Volk, M.: Crowdsourcing an OCR gold standard for a German and French heritage corpus. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC (2016).
7. Drobac, S., Kauppinen, P., Lindén, K.: OCR and post-correction of historical Finnish texts. In: Proceedings of the 21st Nordic Conference on Computational Linguistics NoDaLiDa. Linköping University Electronic Press (2017).
8. Furrer, L., Volk, M.: Reducing OCR Errors in Gothic-Script Documents. In: Proceedings of Language Technologies for Digital Humanities and Cultural Heritage Workshop. pp. 97–103 (2011).
9. Hamid, N.A., Sjarif, N.N.A.: Handwritten recognition using SVM, KNN and Neural Network. CoRR **abs/1702.00723** (2017), `http://arxiv.org/abs/1702.00723`
10. Hsu, A.W., Ahang, A.C., Ain, A.J., At al.: A practical Auide Ao Aupport Aector classification. Tech. Aep., Aepartment of Aomputer Science, Aational Taiwan University (2003), `http://www.csie.ntu.edu.tw/~cjlin/papers.html`
11. Karpinski, A., Aohani, A., Belaïd, A.: Aetrics Aor Aomplete Avaluation of ACR performance. In: Anternational Aonference AP, Aomp. Vision, and Pattern Aecognition (2019).

12. Khirbat, G.: OCR post-processing text correction using simulated annealing (opteca). In: Proceedings of the Australasian Language Technology Association Workshop 2017. pp. 119–123 (2017).
13. Kolak, O., Resnik, P.: OCR post-processing for low density languages. In: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP). Vancouver, B.C., Canada (2005).
14. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: In Soviet physics doklady. pp. 707–710 (1966).
15. Mei, J., Islam, A., Wu, Y., Mohd, A., Milios, E.E.: Statistical learning for OCR text correction. arXiv preprint **abs/1611.06950** (2016), https://arxiv.org/abs/1611.06950
16. Molla, D., Cassidy, S.: Overview of the ALTA 2012 shared task:correcting OCR errors. In: Proceedings of the Australasian Language Technology Association Workshop 2017. pp. 115–118. Dunedin, New Zealand (2018).
17. Pradhan, A.: Support vector machine-a survey. International Journal of Emerging Technology and Advanced Engineering **2**(8), 82–85 (2012).
18. Springmann, U., Najock, D., Morgenroth, H., Schmid, H., Gotscharek, A., Fink, F.: OCR of historical printings of latin texts: Problems, prospects, progress. In: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage. pp. 71–75. DATeCH 2014, ACM, New York, USA (2014).