A Supervised Machine Learning Approach for Post-OCR Error Detection for Historical Text

Dana Dannélls
Språkbanken Text
University of Gothenburg
dana.dannells@svenska.gu.se

Shafqat Mumtaz Virk
Språkbanken Text
University of Gothenburg
shafqat.virk@svenska.gu.se

Abstract

Training machine learning models with high accuracy requires careful feature engineering, which involves finding the best feature combinations and extracting their values from the data. The task becomes extremely laborious for specific problems such as post Optical Character Recognition (OCR) error detection because of the diversity of errors in the data. In this paper we present a machine learning approach which exploits character n-gram statistics as the only feature for the OCR error detection task. Our method achieves a significant improvement over the baseline reaching state-of-the-art results of 91% and 89% F1 score on English and Swedish datasets respectively. We report various experiments to select the appropriate machine learning algorithm and to compare our approach to previously reported traditional approaches.

1 Introduction

Post processing is a conventional approach for correcting errors that are caused by Optical Character Recognition (OCR) systems. Traditionally, the task is divided into two subtasks: (1) Error detection, classify words as either erroneous or valid, and (2) Error correction, find suitable candidates to correct the erroneous words (Kolak and Resnik, 2005; Kissos and Dershowitz, 2016; Mei et al., 2016). Previous research has shown that machine learning based approaches are suitable for both subtasks (Schulz and Kuhn, 2017; Nguyen et al., 2018, 2019a; Dannélls and Persson, 2020). In the current work we aim to improve on the first task for historical texts by using machine learning techniques.

Training an accurate machine learning model requires handcrafted feature engineering,[1] which involves finding the best feature combinations and parameter settings. In the context of post-OCR error detection, finding a suitable set of features is challenging because of the diversity of OCR errors (Amrhein and Clematide, 2018). At the same time, it is well-known that feature computation is often time and labour expensive. This raises the question: Do we always need a rich feature set for achieving better results or, depending on the task at hand, fewer features could lead to better or equally good results? To our knowledge, this question has not been addressed before.

Unlike OCR errors for modern material, the error rates for historical texts are very high, resulting from a large amount of unseen characters in the output text. This has been observed for several languages (Springmann et al., 2014; Drobac et al., 2017; Adesam et al., 2019). To address the challenges for post-OCR error detection for historical text, a number of feature combinations have previously been explored with varying success rates (more details in Section 2). In this paper, we take a different approach, and instead of trying to find the optimal set of features for the task at hand, we experimented with one n-gram character feature (Sections 3 and 4). Our method achieves a significant improvement over the baseline reaching state-of-the-art results of 91% and 89% F1 on English and Swedish datasets respectively. In addition to being simple, our approach is less expensive for feature value computations. Finally, we discuss the strengths of the method and provide pointers to future work (Section 5).

2 Related work

There are two approaches to OCR detection and correction. One approach incorporates fine-tuned methods for improving the OCR system. For example, Tesseract (Smith, 2007) has built-in post-

[1] The *handcrafted* part of this process is finding the most suitable features and feature combinations by examining the data manually.

	Character	Word n-gram	Context	Features tot.	Method	Recall (%)
Mei et al. (2016)	✓	✓	✓	6	RM	73.9
Khirbat (2017)	✓	✓	✓	3	SVM	44.2
Nguyen et al. (2019b)	✓	✓	✓	13	GTB	61 & 76
Dannélls and Persson (2020)	✓	✓		6	SVM	67

Table 1: Feature combinations reported in previous work on post-OCR detection using machine learning models and the percentage of detected OCR errors reported by each author (RM = Regression Model, SVM = Support Vector Machine, GTB = Gradient Tree Boosting).

correction functions for improving the OCR results for different languages. Another approach, that is taken here and has been adapted by the majority of previous works, builds on the output results of a specific OCR system – the one being referred to as post-OCR processing. The obvious advantage of the latter approach is that the developed method is not tailored to a particular system and could be applied to any OCR output regardless of the OCR system. One must bear in mind, however, that post-OCR processing is a complicated task because of the nature of the different errors produced by various OCR systems.

The majority of post-OCR methods of error detection exploits supervised (Evershed and Fitch, 2014; Drobac et al., 2017; Khirbat, 2017) or unsupervised (Hammarström et al., 2017; Duong et al., 2020) machine learning techniques, depending on whether the ground truth data is available or not. In this paper we focus on supervised methods. The methods described below have been trained on each word of the document. Words have been classified as either erroneous or correct. Precision, recall and F-score have been calculated based on the predicted erroneous words.

Mei et al. (2016) have experimented with 6 features containing character, word n-gram and context information. They have reported a recall for bounded (true punctuation) detection of 73.5% using regression models. Khirbat (2017) has trained a support vector machine (SVM) model with 3 features: presence of non alpha-numeric characters, bi-gram frequencies of the word and context information, that is if the word appears with its context in other places. He reported 69.6% precision, 44.2% recall and 54.1% F1. Nguyen et al. (2019b) experimented with 13 character and word features on two datasets of handwritten historical English documents (monograph and periodical) taken from the ICDAR competition (Chiron et al., 2017). The features they have experimented with include character and word n-gram frequencies, part-of-speech, and the frequency of the OCR token in its candidate generation sets which they generated using edit-distance and regression model. They trained a Gradient Tree Boosting classifier and achieved a recall of 61% and 76% and an F1 of 70% and 79% on each dataset respectively. Their results are the highest reported on the ICDAR English dataset.

Dannélls and Persson (2020) have trained an SVM model and experimented with 6 statistical and word based features, including the number of non-alphanumeric characters, number of vowels, word length, tri-gram character frequencies, number of uppercase characters and the amount of numbers occurring in the word. They reported 67% recall, and 63% F1, which is the highest results reported on Swedish text from the 19th century.

An overview of the feature sets previous authors have experimented with and the recall of the error detection machine learning models reported by each is provided in Table 1.

3 Method

3.1 Datasets

We experimented with three datasets, two for English and one for Swedish.

The first English dataset (henceforth Sydney) comprises newspaper text from the Sydney Morning Herald 1842-1954, consisting of 10,498,979 tokens and a ground truth data of randomly sampled paragraphs (Evershed and Fitch, 2014). The material was processed with Abbyy Finereader 14. The training and testing sets compiled from this material contain instances from this particular OCR system only.

The second English dataset (henceforth ICDAR2017) is the monograph dataset from the ICDAR 2017 competition (Chiron et al., 2017), which accounts for 754,025 OCRed tokens with their cor-

responding ground truth.[2] The dataset has been collected from national libraries and university collections. It was processed with Abbyy Finereader 11, and the ground truth comes from various European project initiatives.

The Swedish dataset (henceforth Fraktur&Olof) consists of a selection of digitized versions of older Fraktur prints from 1626-1816,[3] and all pages from Olof v. Dalin's Swänska Argus from 1732-1734,[4] all amounting to 261,323 tokens. The ground truth for this dataset was produced through double-keying. The material was processed with three OCR systems: Abbyy Finereader 12, Tesseract 4.0 and Ocropus 1.3.3. Each one of these systems is using their own built-in dictionary and the quality of the OCR results differs significantly between the systems. When we compiled the training and testing sets in our experiments, described in Section 4, we included instances from all three systems to avoid the risk of developing a method that is biased towards a particular OCR system (Dannélls and Persson, 2020).[5]

In our experiments (see Section 4), we chose randomly selected subsets of 50K tokens from the Sydney and the Fraktur&Olof datasets. A balanced set of 92K instances was selected from the ICDAR2017 dataset. All three subsets were then divided into training (80%) and test (20%) sets. Depending on the vocabulary size, it can take days to run the models. Because of this constraint the complete datasets were not used in the experiments.

3.2 Preprocessing

All of the above datasets come in different formats, therefore we had to preprocess them before we could proceed. For our experiments we needed to first align the OCRed and ground truth data at the token level and secondly convert the aligned data to feature vectors.

In the ICDAR2017 and Sydney datasets, the OCRed and ground truth data are aligned at the character level. To align them at the token level, the ground truth was tokenized on space, and for each token the same number of characters was extracted from the OCRed version. After removing the special alignment symbols ('@' and '#') that were inserted by the competition organizers, the resulting OCRed and ground truth tokens were compared to set the labels: '0' if the token was erroneous or '1' if the token was valid.[6] These labels are to be learned and predicted by the machine learning models during training and testing. Learning is based on a set of feature combinations to help the model detect the errors in the output of the OCR, described in Section 4.

The tokens in the Swedish dataset were computed by first removing duplicate white-spaces and second, replacing all non-space white-spaces such as tab with space. Then, valid tokens were extracted from the ground truth data and were assigned label '1'. Erroneous tokens were extracted from the OCRed data and were compared to a large scale computational Swedish lexicon (Borin and Forsberg, 2011). If the token appeared in the lexicon it was assigned label '1' otherwise '0'.[7]

Table 2 shows a few instances from the data produced after the preprocesscing step both for Swedish and English. The resulting full data-sets were then used to compute various features and train/test models as explained in Section 4.

English			Swedish		
Token	GT	Label	Token	GT	Label
matter	matter	1	nytta	nytta	1
the	the	1	sassvanter	-	0
king@	king	0	angenämt	angenämt	1
very	very	1	p-å	-	0
glad	glad	1	föreställa	föreställa	1
hereof,@	hereof,	0	behöfwesr	-	0
@Hkewise	likewise	0	Lärdomar	lärdomar	1

Table 2: A sample from the English and Swedish datasets after the preprocessing step (GT = Ground Truth).

All the machine learning models we experimented with are part of the Sci-kit Python library (Pedregosa et al., 2011). Input data to all the algorithms in the sklearn library should be in numerical form, but only some of the features we experimented with are numeric (e.g. the token frequencies), the others are non-numeric (e.g. bigrams). For the non-numeric features, we used one-hot encoding for data transformation. While the details are beyond the scope of this paper, the

[2] https://drive.google.com/file/d/1-pNT00vvIqh0ss_5b2aHo-nG8advaFJi/view

[3] https://spraakbanken.gu.se/en/resources/svensk-fraktur-1626-1816

[4] https://spraakbanken.gu.se/en/resources/dalin-then-swaanska-argus-1732-1734

[5] Datasets are available under CC-BY license and can be accessed from https://spraakbanken.gu.se/en/resources#refdata.

[6] Valid OCRed tokens are identical to the GT token.

[7] Because preprocessing of the datasets is completely automatic, we noticed that a small proportion of instances was miss-classified.

	Fraktur&Olof			Sydney		
	Precision	Recall	F1	Precision	Recall	F1
Logistic Regression	0.82	0.74	0.76	0.74	0.60	0.65
Decision Tree	0.84	0.79	**0.80**	0.71	0.73	**0.71**
Bernoulli Naive Bayes	0.84	0.78	0.79	0.79	0.58	0.63
Naive Bayes	0.67	0.54	0.37	0.70	0.60	0.59
SVM	0.84	0.79	**0.80**	0.74	0.60	0.66

Table 3: Evaluation results of error detection for English and Swedish datasets trained with different models on one feature. The best performing models are highlighted in bold (Experiment I).

major idea behind one-hot encoding is to add an extra dimension in the feature vector for each unique feature value. This produces an N dimensional feature vector (the learned encoding), where N is the total number of unique values of the complete feature set. An instance is then encoded by setting the dimension corresponding to the feature value to '1', while the remaining dimensions are set to '0'. We used sklearn's 'CountVectorizer' and 'SVC' classifiers with default parameter to learn the encoding and train the different machine learning models. In all the experiments we used the default SVM *radial basis* kernel function.

4 Experiments and results

We devised three experimental settings. The first experiment is set up to learn which machine learning algorithm performs best on the OCR error detection task. In the second experiment we create our baseline and train a machine learning model with different feature configurations. Given our findings in the second experiment we further explore the best performing configuration with simple character n-gram features.

4.1 Experiment Setup

Experiment I Machine learning classifiers are known to have pros and cons depending on the task. To our knowledge, there are no previous studies to examine the performance of different machine learning techniques for detecting OCR errors. We compared between 5 popular state-of-the-art machine learning classifiers to learn which of them is most suitable for this task. More specifically, we explored Logistic Regression, Decision Tree, Bernoulli Naive Bayes, Naive Bayes and SVM.

Logistic Regression has been very common for binary tasks because of its success in linearly separating data. Decision Tree is a predictive classifier, most widely used for solving inductive problems. It has also proven to be efficient for detecting OCR errors (Abuhaiba, 2006). Both Bernoulli Naive Bayes and Naive Bayes are probabilistic classifiers. Bernoulli Naive Bayes includes a probability for whether a term is in the data or not, and therefore has been shown useful for document classification. SVM is a supervised machine learning method that is very effective in high dimensional spaces. It has gained high popularity for detecting OCR errors partially because its performance has proven to be as robust and accurate as of a neural network (Arora et al., 2010; Hamid and Sjarif, 2017; Amrhein and Clematide, 2018).

In this experimental setting, we trained all machine leaning classifiers on one feature that is the actual word. For training and testing, 5-cross validation was applied. Because of the time needed to train the models, the classifiers were only trained on two datasets, Fraktur&Olof and Sydney.

Experiment II We experimented in three different settings. First, we form our baseline by training the best performing model (from experiment I) on the 6 features reported by Dannélls and Persson (2020). This set of features forms our baseline, it includes: (1) whether the word contains an alphanumeric character, (2) the word tri-gram frequency, (3) whether the word contains a vowel, (4) whether the word length is over 13 characters (5) whether the first letter appears in upper case, (6) whether the word contains a number. Since all of the features are numeric in nature, no encoding was required for this setting.

Second, analogous to previous approaches (Mei et al., 2016; Khirbat, 2017; Nguyen et al., 2019b), we enhanced the feature set with 4 additional features (referred to as the 10-feature model): (1) the actual word (2) the actual word length, (3) context, i.e. the word proceeding and following the actual word, (4) whether the word appears in the word2vec model, here we apply a simple look-up

	Fraktur&Olof			Sydney			ICDAR2017		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Baseline	0.82	0.68	0.70	0.73	0.59	0.60	0.85	0.85	0.85
10-feature	0.80	0.71	0.73	0.81	0.62	0.63	NA	NA	NA
1-word-feature	0.78	0.83	0.79	0.80	0.62	0.63	0.86	0.84	0.84

Table 4: Evaluation results of error detection with SVM, once computed with the 10-feature model and once with the 1-word-feature model (Experiment II). Baseline was computed with the 6-feature model.

method against the pre-trained model by Hengchen et al. (2019). In this case, some of the features (e.g. the word itself) are non-numeric, hence one-hot encoding was applied for those features. As mentioned previously, this means adding an extra dimension for each unique word in the training data to learn the encoding and then encoding each instance by setting the corresponding dimension values accordingly. The same applies for the context feature.

Third, we removed all features and trained the model only on one feature, the actual word (referred to as the 1-word-feature model).[8] This potentially means turning the model into a dictionary look-up kind of system, with the major restriction that the system is not scalable and is restricted to only those words which have been seen in the training data.

Experiment III To overcome the above mentioned limitation of using the word as the only feature, we experimented further with n-gram feature sets. For each candidate word, we generated character uni-, bi-, and tri-grams first, and then their counts within the word were used as feature values to train the model. To take an example, suppose our candidate word is 'passenger', the computed uni-, bi-, and tri-gram features vectors will be as follows:

- **uni-gram** {'a':1, 'e':2, 'g':1, 'n':1, 'p':1, 'r':1, 's':2}
- **bi-gram** {'p':1, 'as':1, 'en':1, 'er':1, 'ge':1, 'ng':1, 'pa':1, 'r ':1, 'se':1, 'ss':1}
- **tri-gram** {'pa':1, 'ass':1, 'eng':1, 'er ':1, 'ger':1, 'nge':1, 'pas':1, 'sen':1, 'sse':1}

The intuition is simple: It is more probable that the corresponding uni-, bi-, and tri-grams have been seen in the training data as opposed to the complete word. This can remove the above described limitation and make the system more scalable. The models were then trained on the resulting feature vectors and then tested on the test data.

[8] We write 'word' although, in practice, it actually refers to a token because 'a word' is not necessarily a lexical word, for example if we consider an instance from our training data, i.e. 'ycsteidas'.

4.2 Results

Experiment I The results from the first experiment, where only one feature was used to train different machine learning models, are presented in Table 3. We can observe that both Decision Tree and SVM outperform the other models on the Swedish dataset, achieving 80% F1. Bernoulli Naive Bayes is almost as good with an F1 of 79%. Decision Tree is the best performing model on the English dataset with the highest F1 of 71%. These results strengthen previous successful attempts to train an SVM model for detecting OCR errors (Arora et al., 2010; Hamid and Sjarif, 2017; Clematide and Ströbel, 2018).

Experiment II The results from the second experiment are presented in Table 4. Even though we experimented with the same feature combination as reported in Dannélls and Persson (2020), our baseline yields 70% F1 compared to their reported 63% F1 probably owing to parameter settings and the chosen sub datasets. The results on Fraktur&Olof show that the model trained on 1-word-feature outperforms the model trained on 6 (baseline) and 10 feature sets respectively.

Interestingly, the results on the Sydney dataset show no difference in performance between the 10-feature and the 1-word-feature datasets. In contrast to the Fraktur&Olof dataset where F1 increases with 5%. We believe the difference in the results between Fraktur&Olof and Sydney can be characterized by the nature of the data. A manual inspection of the datasets reveals that Fraktur&Olof is representative with regards to its vocabulary. Hence, more words in the Swedish dataset were seen in the training set as compared to the English counterpart.

Our baseline results on the ICDAR2017 dataset

	Fraktur&Olof			Sydney			ICDAR2017		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Uni-gram	0.81	0.78	0.78	0.83	0.68	0.70	0.89	0.87	0.87
Bi-gram	0.87	0.87	0.87	0.86	0.76	**0.79**	0.91	0.91	**0.91**
Tri-gram	0.89	0.89	**0.89**	0.84	0.74	0.77	0.88	0.88	0.88

Table 5: The accuracy scores of the SVM classifier trained with the n-gram feature sets. Best results for each dataset in bold (Experiment III).

are not as high compared to the F1 reported by Mei et al. (2016) and Nguyen et al. (2019b). The reason for this is because we are experimenting with completely different datasets with respect to both size and content. Training the SVM classifier on 1-word-feature did not improve the baseline. This, again, may be due to the nature of the data.

Experiment III The results from the experiments with the n-gram feature sets are shown in Table 5. When we compare between the results of the 1-word-feature and the n-gram feature models, we see there is an improvement for all three datasets: Fraktur&Olof, Sydney, and ICDAR2017.

The best performance achieved on Fraktur&Olof is 89% F1 with the tri-gram model. This is the highest results on 19th century Swedish text reported so far. The best performing model for Sydney is 79% F1, achieved with the bi-gram model. The best results achieved on the ICDAR2017 data are also with the bi-gram model. For all datasets the n-gram models show an incremental improvement. One explanation for the difference between the results might be the differences between the types of OCR errors in each dataset. The most obvious errors on Fraktur&Olof are due to appearance of long *s*, uppercase letters and miss-recognition of the Swedish vowels (‘å’ and ‘ä’), while obvious errors in ICDAR2017 are due to hypens and non-alphanumeric characters.

5 Discussion and Conclusion

Training supervised machine learning models with large number of features is a computationally expensive task. This has been demonstrated in previous work where carefully crafted features were considered at the expense of high computational costs. In our experiments we trained an SVM model on a number of feature sets consisting of 6 features, 10 features, one word feature and three n-gram character level features, and compared their results. By training the model on the word itself, we are necessarily turning the machine learning model into a dictionary look-up kind of system. The results show that the 1-word-feature model trained on word level is sufficient, not only for improving over the baseline, but also for reaching better results than previously reported for historical Swedish data. The results on the English datasets show that the 1-word-feature model is as good as the 10-feature model. This proves that with the dictionary of words over the training data alone we can better predict whether a word contains an OCR error or not. However, this type of approach has its own limitations as mentioned previously, and for that purpose, we turned to character level n-gram based approach, which improved the results further.

What makes the proposed approach interesting is that it eliminates the need to compute many features for detecting OCR errors. On the other hand, we are aware that it relies on the availability of large amount of training data which is also costly, and will in turn also increase the training time.

Notwithstanding, in this work we kept the datasets rather small mostly because of time constraints and memory issues. This leaves several open questions regarding the representativeness of the chosen data. Correspondingly, we are unable to make direct comparisons with the results reported by others. In the future, we plan to experiment with bigger datasets, and our hope is to improve on the results reported in this study. Parameter optimization of the chosen machine learning algorithms is another direction which can be explored further to improve the results in future. Another possible way to improve the results is to use the back-off approach in the n-gram setting. Taking a back-off approach we will use a bi-gram if a tri-gram is not in the vocabulary in a tri-gram setting, and likewise a uni-gram if a bi-gram is not in the vocabulary.

Acknowledgments

The work presented here was funded by (1) the *Dictionary/Grammar Reading Machine: Compu-*

tational Tools for Accessing the World's Linguistic Heritage (DReaM) Project awarded 2018–2020 by the Joint Programming Initiative in Cultural Heritage and Global Change, Digital Heritage and Riksantikvarieämbetet, Sweden; (2) the Swedish Research Council as a part of the project *South Asia as a linguistic area? Exploring big-data methods in areal and genetic linguistics* (2015–2019, contract no. 421-2014-969); (3) *From Dust to Dawn: Multilingual Grammar Extraction from Grammars* project funded by Stiftelsen Marcus och Amalia Wallenbergs Minnesfond 2007.0105, Uppsala University; (4) the Swedish Research Council as part of the project *Evaluation and refinement of an enhanced OCR-process for mass digitisation* (2019–2020, grant agreements IN18-0940:1 and 421-2014-969). It is also supported by Språkbanken Text and Swe-Clarin, a Swedish consortium in Common Language Resources and Technology Infrastructure (CLARIN) Swedish CLARIN (grant agreement 821-2013-2003). The authors would like to thank the SLTC anonymous reviewers for their valuable comments and suggestions on how to improve the paper.

References

Ibrahim S I Abuhaiba. 2006. Efficient OCR using simple features and decision trees with backtracking. *Journal for science and engineering*, 31(2):223–244.

Yvonne Adesam, Dana Dannélls, and Nina Tahmasebi. 2019. Exploring the quality of the digital historical newspaper archive kubhist. In *Proceedings of the 4th Conference of The Association Digital Humanities in the Nordic Countries (DHN)*, pages 9–17, Copenhagen, Denmark. University of Copenhagen, Faculty of Humanities.

Chantal Amrhein and Simon Clematide. 2018. Supervised OCR error detection and correction using statistical and neural machine translation methods. *Journal for Language Technology and Computational Linguistics (JLCL)*, 33(1):49–76.

Sandhya Arora, Debotosh Bhattacharjee, Mita Nasipuri, Latesh L. G. Malik, Kundu Mahantapas, and Dipak Kumar Basu. 2010. Performance comparison of SVM and ANN for handwritten Devnagari character recognition. *IJCSI International Journal of Computer Science Issues*, 7(6).

Lars Borin and Markus Forsberg. 2011. A diachronic computational lexical resource for 800 years of Swedish. In Caroline Sporleder, Antal van den Bosch, and Kalliopi Zervanou, editors, *Language technology for cultural heritage*, pages 41–61. Springer, Berlin.

Guillaume Chiron, Antoine Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. 2017. ICDAR2017 competition on post-OCR text correction. *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 01:1423–1428.

Simon Clematide and Phillip Ströbel. 2018. Improving OCR quality of historical newspapers with handwritten text recognition models. In *Workshop DARIAH-CH*, Neuchâtel. University of Zurich.

Dana Dannélls and Simon Persson. 2020. Supervised OCR post-correction of historical Swedish texts: What role does the OCR system play? In *Proceedings of the Digital Humanities in the Nordic Countries 5th Conference*, volume 2612 of *CEUR Workshop Proceedings*, pages 24–37, Riga, Latvia. CEUR-WS.org.

Senka Drobac, Pekka Kauppinen, and Krister Lindén. 2017. OCR and post-correction of historical Finnish texts. In *Proceedings of the 21st Nordic Conference on Computational Linguistics (Nodalida)*, pages 70–76, Gothenburg, Sweden. Association for Computational Linguistics.

Quan Duong, Mika Hämäläinen, and Simon Hengchen. 2020. An unsupervised method for OCR post-correction and spelling normalisation for Finnish. *arXiv*, abs/2011.03502.

John Evershed and Kent Fitch. 2014. Correcting noisy OCR: Context beats confusion. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, pages 45–51, New York, NY, USA. Association for Computing Machinery.

Norhidayu Abdul Hamid and Nilam Nur Amir Sjarif. 2017. Handwritten recognition using SVM, KNN and Neural Network. *ArXiv pre-print*, abs/1702.00723.

Harald Hammarström, Shafqat Mumtaz Virk, and Markus Forsberg. 2017. Poor man's OCR post-correction: Unsupervised recognition of variant spelling applied to a multilingual document collection. In *Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage*, DATeCH 2017, pages 71–75, NY, USA.

Simon Hengchen, Ruben Ros, and Jani Marjanen. 2019. A data-driven approach to the changing vocabulary of the 'nation' in English, Dutch, Swedish and Finnish newspapers, 1750-1950. In *Proceedings of the Digital Humanities (DH) conference 2019, Utrecht, The Netherlands*.

Gitansh Khirbat. 2017. OCR post-processing text correction using simulated annealing (OPTeCA). In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pages 119–123. Association for Computational Linguistics.

Ido Kissos and Nachum Dershowitz. 2016. OCR error correction using character correction and feature-based word classification. In *Document Analysis Systems 12th IAPR Workshop*, pages 198–203. IEEE.

Okan Kolak and Philip Resnik. 2005. 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)*, pages 867–874, Vancouver, B.C., Canada. Association for Computational Linguistics.

Jie Mei, Aminul Islam, Yajing Wu, Abidalrahman Mohd, and Evangelos E Milios. 2016. Statistical learning for OCR text correction. *arXiv preprint*, abs/1611.06950.

Thi-Tuyet-Hai Nguyen, Mickaël Coustaty, Doucet Antoine, and Nhu-Van Nguyen. 2018. Adaptive edit-distance and regression approach for post-OCR text correction. In *20th International Conference on Asia-Pacific Digital Libraries, ICADL*, volume 11279 of *Lecture Notes in Computer Science*, pages 278–289. Springer.

Thi-Tuyet-Hai Nguyen, Adam Jatowt, Mickael Coustaty, Nhu-Van Nguyen, and Antoine Doucet. 2019a. Deep statistical analysis of OCR errors for effective post-OCR processing. In *Proceedings of the 18th Joint Conference on Digital Libraries*, JCDL '19, page 29–38. IEEE Press.

Thi-Tuyet-Hai Nguyen, Adam Jatowt, Mickaël Coustaty, Nhu-Van Nguyen, and Antoine Doucet. 2019b. Post-OCR error detection by generating plausible candidates. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 876–881. IEEE.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12:2825–2830.

Sarah Schulz and Jonas Kuhn. 2017. Multi-modular domain-tailored OCR post-correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2716–2726, Copenhagen, Denmark. Association for Computational Linguistics.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE.

Ume Springmann, Dietmar Najock, Hermann Morgenroth, Helmut Schmid, Annette Gotscharek, and Florian Fink. 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 2014, pages 71–75, New York, USA. Association for Computing Machinery.