Automatic Semantic Text Tagging on Historical Lexica by Combining OCR and Typography Classification

A Case Study on Daniel Sander's *Wörterbuch der Deutschen Sprache*

Christian Reul*
Centre for Philology and Digitality
University of Würzburg
christian.reul@uni-wuerzburg.de

Sebastian Göttel
Berlin-Brandenburg Academy of
Sciences and Humanities
sebastian.goettel@bbaw.de

Uwe Springmann
Center for Information and
Language Processing; LMU Munich
uwe@springmann.net

Christoph Wick
Chair for Artificial Intelligence
University of Würzburg
christoph.wick@uni-wuerzburg.de

Kay-Michael Würzner
Berlin-Brandenburg Academy of
Sciences and Humanities
wuerzner@bbaw.de

Frank Puppe
Chair for Artificial Intelligence
University of Würzburg
frank.puppe@uni-wuerzburg.de

ABSTRACT

When converting historical lexica into electronic form the goal is not only to obtain a high quality OCR result for the text but also to perform a precise automatic recognition of typographical attributes in order to capture the logical structure. For that purpose, we present a method that enables a fine-grained typography classification by training an open source OCR engine both on traditional OCR and typography recognition and show how to map the obtained typography information to the OCR recognized text output. As a test case, we used a German dictionary (Sander's *Wörterbuch der Deutschen Sprache*) from the 19th century, which comprises a particularly complex semantic function of typography. Despite the very challenging material, we achieved a character error rate below 0.4% and a typography recognition that assigns the correct label to close to 99% of the words. In contrast to many existing methods, our novel approach works with real historical data and can deal with frequent typography changes even within lines.

CCS CONCEPTS

• Applied computing → Optical character recognition.

KEYWORDS

OCR, typography recognition, semantic tagging, historical lexica

ACM Reference Format:

Christian Reul, Sebastian Göttel, Uwe Springmann, Christoph Wick, Kay-Michael Würzner, and Frank Puppe. 2019. Automatic Semantic Text Tagging on Historical Lexica by Combining OCR and Typography Classification: A Case Study on Daniel Sander's *Wörterbuch der Deutschen Sprache*. In *3rd International Conference on Digital Access to Textual Cultural Heritage (DATeCH2019)*, May 8–10, 2019, Brussels, Belgium. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3322905.3322910>

*Corresponding author.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
DATeCH2019, May 8–10, 2019, Brussels, Belgium
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7194-0/19/05.
<https://doi.org/10.1145/3322905.3322910>

1 INTRODUCTION

Despite recent progress in the area of historical OCR (see e.g. [4]), mainly due to the introduction of a line-based recognition approach incorporating recurrent neural networks (see [1]), the thorough indexing of a lexicon remains a challenging task. To fully exploit the content, it does not suffice to just gather the text by performing OCR or by transcribing it, since most lexica contain a complex structure within the text consisting not only of the lemmata themselves, but also definitions, grammatical information, or possible word formations. Usually, to encode this information, printed lexica rely on typographical variations of the text (compare Figure 1). Therefore, if the typography within a dictionary article changes, e.g. from Fraktur to Antiqua, it carries semantic meaning. Gathering this information is not a trivial task, since manual tagging is cumbersome and machine learning approaches like *GROBID dictionaries*¹, while being promising, are usually not flexible enough to smoothly adapt to new material or user requirements.

Our proposed approach treats the gathering of the information stored in historical lexica as two separate sequence classification tasks: textual OCR and typography recognition. To ensure flexibility and adaptability, we rely on the application of an open source OCR engine which can easily be trained and geared towards recognizing different text and typography classes. Analogously to the textual OCR, we also produce line-based typography ground truth (GT) by assigning a distinct label to each of the typography classes. After training two separate models (one for text and one for typography) using the respective GT, each model recognizes the line images and the outputs are aligned on word level. Our case study on Daniel Sander's *Wörterbuch der Deutschen Sprache* showed very promising results leading to low character error rates (CER), both for OCR and typography recognition.

The remainder of the paper is structured as follows: after discussing related work in section 2, we introduce the used material in section 3 and describe our approach in section 4. The outcome of the experiments performed in section 5 are discussed in section 6 before section 7 concludes the paper.

¹<https://github.com/MedKhem/grobid-dictionaries>

2 RELATED WORK

In this section, an overview over the related work regarding font identification is given and then the associated topic of script identification is briefly discussed. Moreover, we point out some refinements regarding our OCR training and recognition procedure.

2.1 Font Identification

An early approach by Zramdini and Ingold in 1993 [11] extracted features from the horizontal and vertical projection profiles of text lines and applied a Bayesian classifier to discriminate the font weight, slope, and size. Experiments on 100 French text lines each printed in 48 fonts showed encouraging results.

In [12], the same authors extracted features like black pixel density, x-height, and the presence of serifs from connected components and their bounding rectangles and then trained 280 font models on about 100 English text lines. Evaluation on a test set consisting of at least 100 French text lines for each font showed an character recognition rate of close to 97%.

Other approaches consider font identification as a global texture analysis problem, e.g. Zhu et al. [10] first normalize an input document in order to obtain a uniform block of text. Afterwards, they extract font features by applying multi-channel Gabor filters to the text blocks and utilize a simple weighted Euclidian distance classifier for font recognition. Experiments on computer-generated images using 56 Chinese and English fonts resulted in an average character recognition rate of 99.1%.

Nicoulaou et al. [3] also considered font recognition as a texture classification problem. They relied on Local Binary Patterns extracted from text blocks as features and a simple Nearest Neighbour classifier. For training and evaluation, the wide-spread APTI database² consisting of over 100,000 Arabic words was used. During evaluation, the Local Binary Pattern approach achieved recognition rates between 93 and well over 99%, depending on the subtask.

Naturally, the topic of font identification has also been addressed by many (deep) neural network based techniques. Tao et al. [6] interpreted the task of classifying the font of single Chinese characters as a sequence classification problem. Therefore, they rearranged the input ordering of the points on a stroke and applied a 2-D *long short term memory* network to it. Their system outperformed five other approaches on a synthetic dataset.

The winners of the most recent third edition of the ICDAR competition on multi-font and multi-size digitally represented Arabic text (see [5]) applied a CNN based approach. They used ResNet-18 as their base model and only replaced the final layer with fully connected layers which fitted the individual subtasks of font and font-size recognition as well as joint font and font-size recognition.

2.2 Script Identification

A very comprehensive and up-to-date overview over the various methods of script identification is given in [7]. First, the authors discuss script detection methods categorized in different domains (e.g. printed and handwritten documents). Then, a vast amount of local and global features is described and the performance of the corresponding systems is evaluated and discussed.

Fujii et al. [2] performed sequence-to-label script identification for multilingual OCR on 232 languages and 20 scripts. Their system is trained end-to-end and consists of an encoder which converts a line image into a feature sequence and a summarizer which then aggregates the obtained sequence in order to classify the line. After training their system on over 1.1 million lines evaluation on more than 180,000 lines showed a script identification error rate of 3.1%.

Despite our focus on font identification, the sequence learning method for script identification proposed by Ul-Hasan et al. [8] represents an important piece of preparatory work for our purposes. The authors used OCRopus³ and its underlying LSTM implementation (see [1]) to separate English and Greek script. First, they generated close to 100,000 synthetic text lines in a variety of fonts using the OCRopus line generator and freely available English-Greek text. On an evaluation set consisting of 9,500 synthetically generated text lines, a CER of below 1.81% was achieved.

2.3 Optical Character Recognition

We chose Calamari⁴ (see [9]) as our OCR engine for several reasons. First, its Tensorflow backend enables GPU support for very fast training and recognition. Second, Calamari natively supports several techniques developed for higher recognition rates which we will briefly explain in the following.

To reap the well-known benefits of voting, we implemented a method which allows to train several strong, yet diverse voters using a single OCR engine and to combine their respective outputs by a confidence based voting. In addition, we showed that building from already available mixed models instead of starting training from scratch improves the recognition performance. The combination of pretraining and voting led to a reduction of errors by 53% resulting in an average CER of 1.65% when applied to six early printed books [4].

Apart from the textual OCR result, Calamari also allows the optional output of extended prediction data which we utilize throughout this paper. These data include the start and end position of each character which are identified solely based on the output of the network. Additionally, confidence values for the most likely character as well as for all noteworthy alternatives are provided.

2.4 Differences To Our Approach

The approach proposed in this paper differs from all methods described above in one or several of the following aspects:

- While many approaches deal with synthetically generated images, we are working on **real data**. We therefore do not only present a theoretical proof-of-concept but give a fully realistic example of our method.
- Due to the age of the material, we are not dealing with modern standard fonts but with **historical non-uniform fonts**.
- The **frequent change of typography even within lines** presents an additional challenge since each word has to be classified by only using a small amount of information.
- In consequence of the usage of an open source OCR engine, our approach is **easily adaptable** to other works.

²<https://diuf.unifr.ch/diva/APTI>

³<https://github.com/tmbdev/ocropy>

⁴<https://github.com/Calamari-OCR>

3 MATERIAL: SANDER’S DICTIONARY

The German dictionary of Daniel Sander can still be considered an extraordinary linguistic “œuvre” of 19th century lexicography. Three part-volumes in two volumes written in 1859 until its last delivery and completion in 1865 it is a dictionary “completely rounded” which comprises a particularly complex semantic function of typography (see Figure 1) being explained in the following (see also Figure 2, 3 for the typographical class labels).

The main lemmata at the beginning of each dictionary-article are always in bold and indented. The corresponding GT representation we are aiming to reproduce with our method is shown below with lemmata color-coded in red (assigned label of typographical class *l*). Their typographic appearance is unique. In this case, as with most dictionary articles, the lemma is just followed by grammatical properties, regarding this example for verbs, such as “tr.” (transitive), printed in Antiqua (green, *a*). The semantic paraphrases are annotated as a simple Fraktur type (black, *f*). The typeface of the quotations is divided into two types consisting of the author’s name (yellow, *n*) and the page number (again Antiqua). At the end of a reading possible word formations may be listed. The typeface again uses Fraktur but in letter-spacing (blue, *F*) followed by a meaning.

Figure 1: Input (top) and color-coded output (middle) of our method for an example article depicting the five typography labels. The raw OCR output together with its aligned typography label determines the color code as described in Figure 3. This output can then be further transformed into a (reduced) TEI representation (bottom), explicitly encoding not only the lemma (*orth*) but also all additional information about grammatical properties (*gram*), sources (*bibl*), and compositions (*re*).

4 METHODS

In this section, we describe our approach in detail. After a brief look at the required preprocessing and segmentation steps, we first discuss the OCR and the typography recognition as separate subtasks and afterwards combine them in order to obtain the final result.

4.1 Preprocessing and Segmentation

To be able to work with Calamari, some segmentation steps had to be performed. First, the scans are preprocessed by applying the *ocropus-nlbin* script resulting in a deskewed binary image for each page. Next, the columns have to be detected and segmented into lines. For the first step, we used a rule-based implementation⁵ based on the detection of whitespace separators. Before the line segmentation is performed by applying *ocropous-gpageseg*, the columns are once again deskewed separately. For every extracted line, the coordinates of its enclosing rectangle are stored for later use.

4.2 OCR

To maximize the OCR accuracy while keeping the computational effort reasonable, we trained an ensemble of five book specific models using an existing and freely available⁶ strong voting ensemble for the recognition of 19th century Fraktur as a starting point. The required GT was produced by transcribing a selection of lines which gave a good representation of the occurring characters and typography classes within the book.

During the recognition step, the start and end positions of the recognized characters and the voted probabilities both for the most likely character as well as for the top alternatives are stored.

Due to the complexity of the material at hand, we had to handle a significantly bigger alphabet compared to average 19th century Fraktur printing. 150 distinct characters occurred in the training and evaluation set. For comparison, the mixed models we used as a starting point were trained on GT comprising only 93 characters.

4.3 Typography

In general, the typography recognition sub task is very similar to the OCR one. The main difference lies in the GT production, as manually assigning abstract typography labels to each character is even more cumbersome and error prone than producing textual GT. Therefore, we first introduce a method that considerably simplifies this task. Additionally, we describe the training and recognition process including data augmentation.

4.3.1 Ground Truth Production. Regarding the typography GT, we followed the approach proposed by Ul-Hasan et al. (see [8]) and assigned a distinct label to each of the five classes (*f*, *F*, *n*, *l*, *a*). Since the typography never changes during the course of a word, we developed a GUI (see Figure 2) which uses the OCR GT as input and consequently allows for an efficient typography GT production on word level. After the transcription of the OCR GT, it is first split into words according to the occurring whitespaces. Then, the words are processed from left to right. The currently active word is highlighted by an additional label. Next, the user can assign one of

⁵<https://github.com/wrznr/column-detect>

⁶<https://github.com/chreul/19th-century-fraktur-OCR>

Figure 2: Typography GT production process of a single line. The OCR GT (green border) and the line image are shown at the very top. The remaining steps demonstrate the word-wise typography labelling: 1) The first word is highlighted as active and a single mouse click assigns the selected class label (here: *f*) to each letter. 2) The step is repeated for the following words with the to-be-labeled word always being highlighted in red. 3) Final OCR and typography GT result after assigning the *f* label to the last four words at once.

the predefined classes to each of the characters of the active word by performing a single mouse click. After the insertion of a whitespace, the next word of the line becomes active and is processed in the same way. Moreover, it is possible to assign a specific label to the remainder of the line at once. Figure 2 demonstrates this process.

To ensure native compatibility to OCRopus and Calamari our labeling procedure works on raw line images and produces textual GT files. Naturally, the input and output could also easily be imported from or exported to other (e.g. XML-based) representations.

4.3.2 Training and Recognition. With the obtained typography GT, we performed another training run. In order to add robustness to the models, some data augmentation techniques from the OCRopus3 *ocrodeg* module⁷ were utilized. For every line in the training data set, we created augmented lines by artificially degrading the original image using random distortion and blurring operations. With the obtained data, we performed a pretraining and refined the resulting models by training on real lines exclusively.

4.4 Combining the Outputs

4.4.1 Whitespace Pruning. To combine the output of the two recognition steps, we first prune the results by removing leading and trailing whitespaces from the lines. Furthermore, chains of whitespaces are combined into a single one. The start and end positions are updated accordingly and the confidence is set to 1.

4.4.2 Alignment. Next, we perform an alignment on word level by iterating over the whitespaces recognized by the OCR. For each word in the OCR, we compute the corresponding word in the typography result by checking the typography labels which got recognized since the last whitespace in the OCR. If the bounds of a label end before the next whitespace it is added to the current word. Otherwise, the label is dropped, the current word is concluded, and the next one is started. Additional whitespaces in the typography

⁷<https://github.com/NVlabs/ocrodeg>

Figure 3: Typography alignment for an example line: 1) Line image with whitespace positions (red). 2) Textual OCR output. 3) Typography output with whitespace and character positions (green). 4) Raw (and slightly flawed) textual typography output. 5) Final combined output with voted typography classes assigned on word level.

are also dropped. Figure 3 explains the alignment step and the upcoming typography voting.

4.4.3 Typography Voting. Since the typography does not change during a word, each word is assigned a unified label by selecting the most likely one from the labels matched to the characters of the word. Therefore, we iterate over all recognized characters and sum up the confidence values for all classes. Finally, the label with the highest confidence sum is assigned to the word as a whole. With respect to our example this approach not only covers up the misclassification of the last two characters in the fifth word “Frauen” but also shows its robustness against the insertion or deletion of characters: Despite the typography model only recognizing three characters of type n in the four letter word “Sch.”, the method provides the correct end result of class n .

4.4.4 Final Output. The final output of the recognition process is a JSON file which stores a list of words for each line. Each word is represented by the OCR result and the assigned typography label. Additionally, the minimal character confidence which occurred during the OCR of the word is stored. If needed, this allows for an efficient correction of suspicious semantic key words, e.g. lemmata. Finally, the bounding box of each word is determined from the y -coordinates of its containing line and the x -coordinates of its preceding and subsequent whitespaces. Reversing the cutting and rotational operations during the column segmentation and deskewing steps allows to map the word bounding boxes to the original scan image, e.g. for presentation purposes.

5 EXPERIMENTS

The proposed approach is thoroughly evaluated in this section. After a brief explanation of the data used, we first take a look at the OCR performance before evaluating the typography recognition, both with and without the inclusion of data augmentation.

5.1 Data

To examine the influence of the number of GT lines on the OCR and typography recognition accuracy, we performed the upcoming experiments not only with all available GT (765 lines) but also with subsets of 400, 200, 100, and 50 lines respectively. The lines of each subset were randomly selected from the next bigger set of lines. For evaluation purposes, we selected six columns consisting of 630

Table 1: Statistics of the evaluation data including the distribution of the number of words (W) and characters (C) as well as the average word length (L) for the five typography classes.

	a	f	F	n	l	All
W	1,007	4,378	105	314	53	5,857
	17.2%	74.7%	1.8%	5.4%	0.9%	100%
C	2,580	21,936	747	1,768	333	27,364
	9.4%	80.2%	2.7%	6.5%	1.2%	100%
L	2.56	5.01	7.11	5.63	6.28	4.67

Table 2: CER of the line based OCR dependent on the number of lines used for training. The first row shows the results achieved without book specific training.

# Lines	Calamari	ABBY
-	3.69	10.28
50	1.83	-
100	1.05	-
200	0.67	-
400	0.43	-
765	0.35	-

lines and created the corresponding OCR and typography GT. Table 1 provides additional information about the evaluation data.

5.2 OCR

In this section, we briefly evaluate the OCR performance. Apart from varying the number of training lines, the results are compared to the ones achieved by the state-of-the-art tool for document analysis and recognition ABBYY FineReader⁸ and the freely available raw Fraktur 19th century polyfont model we used for pretraining. For evaluation, we apply the built-in Calamari evaluation script which first computes the Levenshtein distance between the OCR result and the GT and then divides it by the number of GT characters in order to obtain the CER. Table 2 sums up the results.

Unsurprisingly, the CER decreases considerably with a growing amount of training lines. It is worth mentioning that, despite the very challenging material at hand, a quite manageable number of 100 lines of GT is already enough to get close to a very low CER of 1%. Furthermore, the familiar saturation effect becomes clearly visible as almost doubling the number of training lines from 400 to 765 only results in a smaller relative reduction in CER compared to previous steps. The mixed model performs reasonably well, especially when considering that over 20% of the errors occur due to different representations of hyphens and can mostly be attributed to diverging transcription guidelines. Yet, these results demonstrate that a book specific training is reasonable. This is underlined even further by the mediocre recognition accuracy achieved by ABBYY, which struggled with the diversity of the alphabet as well as with the peculiarities of the material, differing considerably from other historical sources such as novels, where ABBYY’s strong dictionary and language modelling features are much more useful.

⁸<https://www.ABBYY.com/finereader>

Table 3: Performance of the line based typography recognition dependent on the number of GT lines.

# Lines	Raw	Voted	
	CER	CER	WER
50	8.48	8.53	9.82
100	3.16	3.12	4.08
200	2.46	2.40	2.66
400	1.39	1.38	1.72
765	1.17	1.18	1.47

5.3 Typography

After showing the OCR capabilities of Calamari and the pretrained Fraktur models, the next step is to evaluate the performance of the typography recognition. Therefore, we first train exclusively on real data and investigate the influence of a postprocessing step, i.e. the confidence voting in order to unify the labels within a word. Then, we perform data augmentation to synthetically enrich the training corpus. Analogously to the OCR evaluation, we examine the influence of the number of training lines for every step. We use the Levenshtein distance as a performance measure for the typography classification task. Since the typography does not change within a word it makes sense to utilize the word error rate (WER).

After determining the most likely label by confidence voting we collapse each word to a single character and remove all whitespaces, e.g. a voted output of “aaaa fffffff fff llllll ffff” becomes “afff”. Subsequently, we simply calculate the CER by applying Calamari’s eval script using an analogously preprocessed GT.

5.3.1 Training Exclusively on Real Data. For the first typography experiment, we used the same set of lines as for the OCR training. After the recognition we first measured the raw CER and then performed the postprocessing described above to obtain the voted CER and the WER. The results are shown in Table 3. There are many interesting things to take away from these results. First, as expected and analogous to the OCR results, a strong negative correlation between the number of lines and the CER can be observed.

Interestingly, the postprocessing does not lead to noteworthy improvements which is at first surprising since intuitively one expects a regular occurrence of single wrong characters within an otherwise correct word, which should easily be fixed by the postprocessing, e.g. “ffffnf” → “ffffff”. However, in reality the dominating errors are insertions and deletion of single characters which, naturally, cannot be eradicated by our postprocessing.

To come along on top, the postprocessing can even worsen the results because of deleted whitespaces between two words with different labels. For example, two words “aaaa ffff” apart from a single error, i.e. the missed whitespace, got correctly recognized as “aaaaffff”, but the confidence voting turned the recognized string into “aaaaaaa” introducing four additional errors.

Concerning the WER the same tendencies can be observed as for the CER. Incorporating all available training lines results in a model which is capable of assigning the correct typography label to over 98.5% of the words.

5.3.2 Incorporating Data Augmentation. In this last experiment, we investigate the influence of adding synthetically altered data to

Table 4: Effect of data augmentation on the line based typography recognition. Each cell shows the resulting WER as well as the improvement in percent compared to training exclusively on real lines. For example, augmenting 200 lines 5 times leads to a WER of 1.89 which represents an improvement of 29% over the real-line-only WER (2.66).

# Real Lines	# Augmentations			
	-	5	20	100
50	9.82	6.37	5.34	4.90
	-	35%	46%	50%
200	2.66	1.89	1.67	1.66
	-	29%	37%	38%
765	1.47	1.45	1.43	1.38
	-	1%	3%	6%

our training set. Therefore, we augmented each line 5, 20, and 100 times for a small (50) and medium (200) set of lines, as well as for the complete set. Table 4 sums up the results. Unsurprisingly, the effectiveness of data augmentation highly depends on the number of available real lines. While for 50 and 200 lines significant improvements in recognition accuracy can be observed, only a moderate reduction in WER of 6% can be achieved when using all available real training lines.

6 DISCUSSION

During our experiments, we showed that typography recognition using the open source OCR engine Calamari is not only possible but very precise. Naturally, a high quality recognition of several very similar typography classes, just like the ones used in Sander’s dictionary, can only be achieved by performing a thorough book-specific training. However, the results showed that even a manageable number of GT lines (200) results in an excellent WER of 1.5% when utilizing data augmentation. Due to the proposed approach to semi-automatically produce the typography GT by building from the available OCR GT, the required manual effort compared to a fully manual transcription is considerably reduced as is the susceptibility to errors.

While the OCR was not the main focus of this paper, the achieved recognition results are still worth mentioning, getting very close to 1% CER with a mere 100 training lines and on a particularly demanding material. This also demonstrates the impressive recognition capabilities of Calamari and underlines the usefulness of the established approach of combining pretraining and confidence voting with a combined CNN/LSTM network. The unequal distribution of typography types in the GT, which is typical for historical lexica, caused a problem, as the OCR model overfits the very frequent classes and underfits the rare ones. This problem can be addressed during the GT selection but only to a certain degree, since lines which contain a less frequent type are still pretty much always dominated by the main types *a* and *f*.

The high quality OCR result combined with the reliable typography based tagging lays the foundation for methods to fully automatically transform comprehensive (in Sander’s case ca. 800,000 text lines) scanned lexica with reasonable effort into a fine-structured

output, e.g. TEI (see bottom of Figure 1 for an example). Appropriate rule-based algorithms have been developed at the Berlin-Brandenburg Academy of Sciences and Humanities, now allowing for complex search queries like “find all lemmata which include Goethe as a source”.

7 CONCLUSION AND FUTURE WORK

We proposed an approach to apply typography recognition to a real world dictionary comprising a particularly complex semantic function of typography. While we only performed a case study on a single lexicon, both the high quality results for OCR and tagging as well as the heavily assisted method for producing typography GT imply that our method can serve as a generic workflow together with a specific typographic model geared towards a complete electronic representation of historical lexica.

There are several promising ideas to improve the proposed method further. For example, the errors caused by misrecognized whitespaces could be reduced by introducing a meta learner which exclusively deals with accepting or rejecting whitespaces proposed by the textual and typographical outputs. Moreover, training type-specific OCR models and applying them on word instead of line level would allow the models to specialize on recognizing a single type, and therefore improving the accuracy even further. Naturally, this approach is most effective when being applied to font or even script types which differ considerably. Despite the five typography classes occurring in Sander’s dictionary being very similar to each other, first experiments yielded very encouraging results.

REFERENCES

- [1] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait. 2013. High-Performance OCR for Printed English and Fraktur Using LSTM Networks. *12th International Conference on Document Analysis and Recognition* (2013), 683–687.
- [2] Yasuhisa Fujii, Karel Driesen, Jonathan Baccash, Ash Hurst, and Ashok C Popat. 2017. Sequence-to-Label Script Identification for Multilingual OCR. In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, Vol. 1. IEEE, 161–168.
- [3] Angelos Nicolaou, Fouad Slimane, Volker Maergner, and Marcus Liwicki. 2014. Local binary patterns for arabic optical font recognition. In *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*. IEEE, 76–80.
- [4] Christian Reul, Uwe Springmann, Christoph Wick, and Frank Puppe. 2018. Improving OCR Accuracy on Early Printed Books by combining Pretraining, Voting, and Active Learning. *JLCL* 33, 1 (2018), 3–24.
- [5] Fouad Slimane, Rolf Ingold, and Jean Hennebert. 2017. ICDAR2017 Competition on Multi-Font and Multi-Size Digitally Represented Arabic Text. In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, Vol. 1. IEEE, 1466–1472.
- [6] Dapeng Tao, Xu Lin, Lianwen Jin, and Xuelong Li. 2016. Principal component 2-D long short-term memory for font recognition on single Chinese characters. *IEEE transactions on cybernetics* 46, 3 (2016), 756–765.
- [7] Kurban Ubul, Gulzira Tursun, Alimjan Aysa, Donato Impedovo, Giuseppe Pirlo, and Tuergen Yibulayin. 2017. Script Identification of Multi-Script Documents: A Survey. *IEEE Access* 5 (2017), 6546–6559.
- [8] Adnan Ul-Hasan, Muhammad Zeshan Afzal, Faisal Shafait, Marcus Liwicki, and Thomas M Breuel. 2015. A sequence learning approach for multiple script identification. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, 1046–1050.
- [9] Christoph Wick, Christian Reul, and Frank Puppe. 2018. Comparison of OCR Accuracy on Early Printed Books using the Open Source Engines Calamari and OROPUS. *JLCL* 33, 1 (2018), 79–96.
- [10] Yong Zhu, Tieniu Tan, and Yunhong Wang. 2001. Font recognition based on global texture analysis. *IEEE Transactions on pattern analysis and machine intelligence* 23, 10 (2001), 1192–1200.
- [11] Abdelwahab Zramdini and Rolf Ingold. 1993. Optical font recognition from projection profiles. *Electronic Publishing* 6, 3 (1993), 249–260.
- [12] Abdelwahab Zramdini and Rolf Ingold. 1998. Optical font recognition using typographical features. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 20, 8 (1998), 877–882.