2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

anyOCR: A Sequence Learning Based OCR System for Unlabeled Historical Documents

Martin Jenckel
University of Kaiserslautern
German Research Center for Artificial Intelligence (DFKI)
Kaiserslautern, Germany.
Email: martin.jenckel@dfki.de

Syed Saqib Bukhari
German Research Center for Artificial Intelligence (DFKI)
Kaiserslautern, Germany.
Email: saqib.bukhari@dfki.de

Andreas Dengel
University of Kaiserslautern
German Research Center for Artificial Intelligence (DFKI)
Kaiserslautern, Germany.
Email: andreas.dengel@dfki.de

***Abstract*—Institutes and libraries around the globe are preserving the literary heritage by digitizing historical documents. However, to make this data easily accessible the scanned documents need to be transformed into search-able text. State of the art OCR systems using Long-Short-Term-Memory networks (LSTM) have been applied successfully to recognize text in both printed and handwritten form. Besides the general challenges with historical documents, e.g. poor image quality, damaged characters, etc., especially unknown scripts and old fonds make it difficult to provide the large amount of transcribed training data required for these methods to perform well. Transcribing the documents manually is very costly in terms of man-hours and require language specific expertise. The unknown fonds and requirement for meaningful context also make the use of synthetic data unfeasible.**
We therefore propose an end-to-end framework anyOCR that cuts the required input from language experts to a minimum and is therefore easily extendable to other documents. Our approach combines the strengths of segmentation-based OCR methods utilizing clustering on individual characters and segmentation-free OCR methods utilizing a LSTM architecture. The proposed approach is applied to a collection of 15^{th} century Latin documents. Combining the initial clustering with segmentation-free OCR was able to reduce the initial error of about 16% to less than 8%.

Keywords*—*OCR, Historical Documents, Clustering, OCRopus, LSTM Networks

I. Introduction

Advances in the capturing technology have significantly reduced the cost of preserving old documents. With an increasing global effort in making digital copies of historical documents in order to preserve the literary heritage, an increasing demand for robust and low effort OCR systems has emerged.

However, automatic recognition of historical documents is very challenging. The three main problems can be categorized as:

- bad document condition (e.g. highly degraded text, torn pages),

Fig. 1. Samples of the 15^{th} century novel "Narrenschiff" in Latin script. Images are taken from the German government funded project, *Kallimachos*[2].

- ancient scripts and archaic orthography,
- costly or unavailable transcribed training data.

There are multiple approaches to OCR documents with little available training data.

One approach would be to use segmentation-based OCR which does not require much training data, e.g. by extracting the unique characters from the document and train a shape-based classifier [1] that assigns the shapes to their respective digital representations. Another approach would be to train a segmentation-free OCR system on the partial data and then use this model to OCR the rest of the corpus.

In practice the first approach does not generalize well for new documents [2] and the amount of training data required for the second approach in order to generalize well is also very high. The creation of transcribed data however, is tedious and costly, and would require to manually transcribe every document that is to be digitized. Another option is the use of artificial data [3], though generating such data often also requires some already transcribed data. Additionally artificial data rarely contains meaningful context and limits the advantages of segmentation-free OCR approaches.

This paper reports our approach to minimize the re-

[2]http://kallimachos.de/kallimachos/index.php/Narragonien:Main

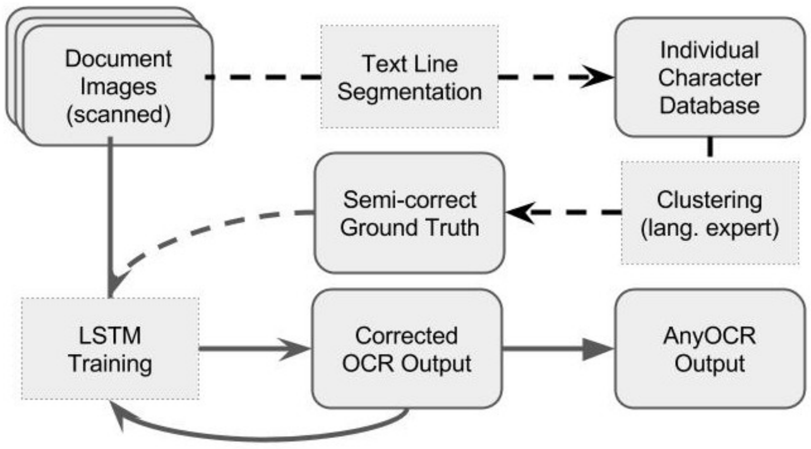
Document Images (scanned)
Text Line Segmentation
Individual Character Database
Semi-correct Ground Truth
Clustering (lang. expert)
LSTM Training
Corrected OCR Output
AnyOCR Output

Fig. 2. The complete pipeline for anyOCR. First text lines and unique symbols are extracted from the scanned data. These symbols are then clustered. After a language expert has identified the resulting clusters, semi-correct ground truth data for each text line is generated. In a second phase a LSTM-based OCR model is trained using the clustering output (using the OCRopus software suit). The trained model can then be used to generate better ground truth data for iterative retraining. When no better OCR model can be trained or gains become too small (e.g. less than 1%), training should be stopped.

quirement of a language expert for manually transcribing documents, esp. historical documents. The only requirement in the proposed framework anyOCR is for a language expert to identify a limited amount of character images through a GUI and provide their digital representation, e.g. Unicode. The proposed approach is capable of reducing the recognition errors by combining the strengths of different OCR systems.

Using multiple OCR types in tandem, specifically segmentation-based and segmentation-free approaches, to increase the overall performance has been reported by [4], however the use of Tesseract as the segmentation-based OCR results in several problems. Tesseract needs a complete list of unique characters. Providing this can result in a lot of additional work for language experts and often involves clustering of some kind. Additionally, Tesseract itself needs some amount of training data for which artificial “meaningless” text has been generated, which leads to the same problems with artificial data as has already been mentioned [5].

Our hypothesis is similar, such that a segmentation-based approach can be used in tandem with a segmentation-free approach to provide an OCR system for documents where no training data is available. However instead of Tesseract, which also needs some amount of training data, we decided to incorporate a totally unsupervised clustering algorithm as our segmentation-based OCR approach. This way we retain the strengths of combining the two approaches, namely not needing transcribed training data for the clustering and utilizing the context sensitive learning and the good generalization properties of the LSTM-based OCR as reported by [6]. The described methodology is designed for 15^{th} century printed Latin documents. As shown in figure 1 these documents provide comparatively few challenges with regards to the first category, however we use no transcribed training data and therefore our framework can easily be adapted to other data sets, with no available training data.

II. Methodology

Similar to [4] our concept is to use both segmentation-based and segmentation-free OCR approaches in tandem to design a high performance OCR system for documents having no or very limited ground truth (GT) data. However, we do not require a unique list of symbols beforehand. The complete pipeline of our procedure is shown in Figure 2 The idea is to use character clustering on the segmented symbols. The clustering results obtained are then used to generate semi-correct ground truth, which is used subsequently to train the segmentation-free OCR system.

First the text lines are semi-automatic extracted from the scanned documents and then segmented into individual characters. After clustering them in an iterative manner, a language experts has to identify which character is represented by each cluster. The segmentation-based approach is finished by producing erroneous text lines for all text line images. These text lines serve as the semi-correct ground truth to train OCRopus [7], a segmentation-free open-source OCR system based on Long Short-Term Memory (LSTM) neural networks. The trained LSTM model is then used again to provide a second iteration of more correct ground truth information. This iterative process can be repeated for any number of iterations. However, with each iteration the gains become less and less and therefore should be stopped when gains become too small (e.g. less than 1%).

Before proceeding further, it is appropriate to understand the internal working of the two OCR systems that have been used to validate our hypothesis.

A. Character Clustering: Iterative K-Means Clustering

As the segmentation-based approach we chose a combination of text line segmentation and character clustering. Only after clustering has finished, a language expert has to identify which character each cluster represents and assign their corresponding Unicodes. For convenience we created a simple GUI for this as shown in Figure 3.

Before clustering, the binarized text line images are segmented into single characters based on connected components. To account for multi-component-characters as well as some degree of noise and artifact filtering, some heuristic rules have been implemented. These are optimized for the $(15)^{th}$ century Latin novel “Narrenschiff”:

- If a connected-component has less than 20 pixels total, it is considered noise and discarded.
- If a connected-component is larger than 200 pixel in either dimension, it is considered an artifact and also discarded.
- If one connected-component completely overlaps with another horizontally (e.g. one is above the other), they are merged into a single component.
- If one connected-component only partially overlaps with another horizontally, but the total displacement is less than 10 pixel, they are merged into a single component (e.g. i with the dot slightly shifted to the left).
- If the horizontal distance between two subsequent connected-components is larger than 11 pixel, a space (empty image) is added between the two components.

K-means clustering is one of the most used clustering algorithms, because it is well known and relatively easy to compute. Especially for these type of historical documents using more sophisticated clustering algorithm or combining them with more sophisticated features does not necessarily improve performance [8]. We chose to use k-means clustering combined with raw features. All characters are centered and downscaled to 32x32 leading to a 1024-dimensional feature space.

However normal k-means produces unsatisfactory results on historical documents, such that clustering based OCR systems often require additional steps for manual inspection. For this task we came up with a fully automated iterative clustering scheme which we call “iterative K-Means clustering”. After an initial clustering each clusters average image is evaluated by how “blurry” it is. If most members of a cluster are the same character, than that clusters average image would be a slightly noisy picture of that character. However, if a cluster contains images of different characters, then the average image becomes more and more “blurry”. We quantize a clusters “blurriness” by dividing all pixel into 4 classes as shown in Figure 3.

- $corepixel$ are shared between almost all cluster members and therefore their value is above p_{core}
- $noisepixel$ are within n_d pixels of a core pixel and have a value between p_{bg} and p_{core}
- $non-corepixel$ are farther than n_d pixels of a core pixel and also have values between p_{bg} and p_{core}
- $backgroundpixel$ have values smaller than p_{bg}

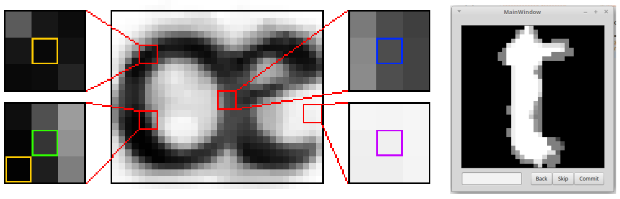

Fig. 3. Left: Examples for all 4 types of pixel, core pixel (yellow), pixel in a core pixels vicinity (green), non-core pixel (blue) and background pixel (purple). Right: The GUI for assigning Unicodes to the clusters. One can either “Commit” the Unicode, get “Back” to the last cluster or “Skip” the current cluster to put it at the back of the list.

Since we are taking averages of binarized images, the pixel values can be understood as the percentage of cluster members that share this pixel. So noise pixel are not present in almost all of the cluster members, but their closeness to core pixel indicates that they describe small variations in the characters shape. In contrast non-core pixel are not close to core pixel and therefore represent an overlap of the shapes of multiple different characters in the average image.
We define blurriness of an average image therefore as the relative amount of core pixel to the total number of core and non-core pixel:

$$b = \frac{\#core}{\#core + \#non\text{-}core} \tag{1}$$

If an average image has too high blurriness, that cluster is re-clustered. The new k is estimated to be the number of overlapping shapes. To assist the language expert at assigning a Unicode to each cluster we created a GUI showing the average images. Lastly the positional information of all characters and their Unicodes are used to generate the text corresponding to each image.

B. LSTM-Modell

OCRopus [7] is an open-source document analysis and recognition system that implements contemporary LSTM-networks, which have shown very good results on printed OCR tasks [6], [2]. This LSTM implementation uses a $1 - pixel$ wide sliding window where each pixel value is fed to one LSTM-unit in the input layer. The height of the image therefore has to be normalized for all inputs. OCRopus offers a normalization method which is based on Gaussian filtering and affine transformation to preserves the baseline and x-height, which are important especially for distinguishing Latin characters [9]. The size of the hidden layer is variable and defines the complexity of the model. During training the error is calculated by a *Connectionist Temporal Classification*-layer (CTC-layer), which tries to align the network output with the provided ground truth by maximizing the expectation over all possible alignments. This can be calculated efficiently by using a forward-backward algorithm [10]. The network learns through a modified backpropagation algorithm, where the weights and biases are updated according to the backpropagated errors. The modified algorithm is called backpropagation through time (BGT) [11]. The training is therefore fully supervised and requires the corresponding ground-truth for each text line image.

The simplified LSTM cell (without the peephole connections) shown in Figure 4 corresponds to the following equations describing the forward path of the training:

$$\begin{aligned}
f_t &= \sigma(W_{xf} \cdot x_t + W_{hf} \cdot h_{t\ 1} + b_f) \\
i_t &= \sigma(W_{xi} \cdot x_t + W_{hi} \cdot h_{t\ 1} + b_i) \\
v_t &= tanh(W_{xv} \cdot x_t + W_{hv} \cdot h_{t\ 1} + b_v) \\
o_t &= \sigma(W_{xo} \cdot x_t + W_{ho} \cdot h_{t\ 1} + b_o) \\
C_t &= f_t * C_{t\ 1} + i_t * v_t \\
h_t &= o_t * tanh(C_t)
\end{aligned}$$

where

- f, i and o are the **forget gate**, **input gate** and **output gate** respectively,
- C is the **Cell State** and h is the **cell output**, while t refers to the current time step and $t-1$ refers to the previous time step
- $b_y, y \in \{f, i, o, v\}$ is the bias unit for the forget gate, input gate, output gate as well as the input squashing respectively,
- W_{ij} is the weight connection between i and j, e.g., W_{xf} is the weight between the external input and the forget gate,
- σ is the *logistic sigmoid* function $\sigma = \frac{1}{1+exp(\ x)}$ and *tanh* is the *tangent hyperbolic* function $tanh(x) = \frac{e^{2x}\ 1}{e^{2x}+1}$,

III. Experimental Evaluation

The basic concept behind the proposed framework is to use segmentation-based and segmentation-free approaches in combination to OCR documents for which GT data is unavailable. For evaluation, first the iterative K-Means clustering is used on the data described in Section III-A. Then the proposed approach is used to improve the semi-correct ground truth (GT) data generated by the clustering by training a LSTM network. For comparison a LSTM network is trained directly on the manually transcribed data.

The performance is estimated in terms of the *Character Error Rate (CER)* which is measured using the "edit distance" metric. This distance is defined as the ratio of insertion, deletion and substitution actions compared to the total length of the text line.

A. The Database

The 15^{th} century novel "Narrenschiff" is part of the German government funded project Kallimachos. A sample document image is shown in Figure 1. We see some degradations due to aging, but also some handwritten annotations and side notes, which makes the data set more challenging for text line segmentation as well as training a LSTM-network.

In order to validate our research hypothesis, we have selected a subset of 100 images from one Latin novel for

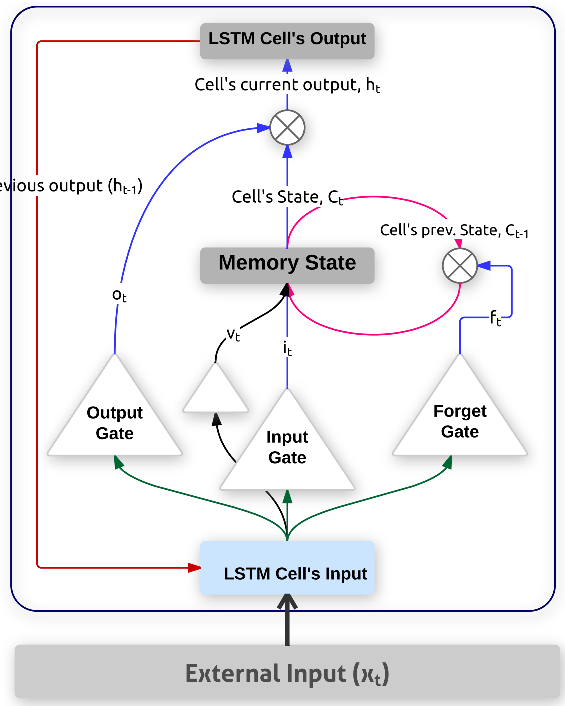

Fig. 4. A shematic LSTM cell: The forget gate decides which inputs to retain at any given time step. The input gate determines which input is processed by the cell, and the output gate determines what output the cell has. Image taken from [4]

training and selected 8 pages from another Latin novel for testing. For performance evaluation on the full data set, the manually transcribed text for 100 pages with a total of 3329 text-lines has been provided by our project partners.

For the iterative K-Means clustering we need no prior information, but a language expert has to assign Unicodes to each cluster. To train the OCRopus model we also needed text line images, that we have semi-automatically cropped from the training data set.

B. The System Parameters

For the initial k-means clustering we largely overestimated the number of character classes and set $k = 200$. For the evaluation of the average images we chose $p_{core} = 0.90$. The threshold for background pixel is set to $p_{bg} = 0.05$. A cluster is subject to re-clustering if the blurriness $b > 0.9$. These values have been found heuristically for the described data set. However re-clustering is only applied if the average size of child clusters is at least 5 elements. For OCRopus the number of LSTM units in the hidden layer is set to 100. With this value the network is complex enough to learn the desired concepts while retaining its good generalization properties. Experience shows that an image height of 48 pixels is a reasonable choice. Smaller images might lead to resolution problems during normalization while bigger images increase the network size and therefore the training time. The learning rate and momentum are the remaining parameters and they are set to $1e-4$ and 0.9 respectively. Similar values are used for various machine

TABLE I. Comparison of applying OCRopus, iterative K-Means clustering, OCRoRACT and anyOCR. The results using manually transcribed GT in OCRopus are only marginally better than using the erroneous ground truth generated by Tesseract and clustering during the OCRoRACT and anyOCR framework respectively.

	Character Error Rate, CER (%)			
Dataset	**OCRopus**	**it. K-Means**	**OCRoRACT**	**anyOCR**
T0	7.37	16.48	9.34	7.33
T1	6.29	–	6.57	6.53
T2	10	–	10.6	10.16

learning algorithm and allowed us to train the models in a reasonable time frame of a few days.

C. Results

This section reports the results of the three evaluations that have been employed to test the performance of the proposed framework. There are three test sets used in the evaluation process. The first data set consists of three pages randomly selected from our training data. These three pages are used for training in all mentioned models and represent our training errors. We call this data set 'T0'. It contains 111 lines and 2686 characters. The second data set 'T1' consists of two randomly selected pages from the same book that is used for training the anyOCR and OCRoRACT systems; however, these pages were not part of any training or clustering. The total number of text-lines in these two pages are 104 and the total number of characters are 2877. The last data set consists of 8 pages randomly selected from a second book that has not been used in training. It consists of 270 text-lines has a total of 7017 characters. This data set is termed as 'T2'.

We did not evaluate our initial steps of the page- and text-line segmentation, because the first one was semi-automatic and for the latter most things that would be considered errors would not be problematic. Even if characters were not segmented successfully, if the same errors are clustered into the same cluster and the language expert assigns the correct Unicode(s) to that cluster, then for our purposes the result will be the same. As mentioned before, first we used our iterative K-Means training scheme on the segmented characters. Instead of having a language expert assigning the Unicodes we used a complete list of unique characters to assign the Unicodes ourselves. Clustering yields a CER of 16.58% on the 'T0' data set and a CER of 12.6% on the complete training data (CTD), which is lower than the CER reported in [4] after applying Tesseract. This means we trained our LSTM-based OCR model with a 12.6% erroneous ground truth data. The first iteration LSTM model thus trained outputs a CER of 7.33% on the 'T0' data set and corrects the CTD to a CER of $\approx 6\%$. We tried to further improve our accuracy by training a second iteration (*iteration-2*) LSTM model using the corrected ground truth. However during the second iteration we got a CER of 7.59% on the 'T0' data set and could not further improve the performance.

The LSTM models obtained after the 1st iteration are used to OCR the 'T2' data set and are compared to the 3rd iteration OCRoRACT model and the OCRopus model, that was trained on the manually transcribed GT data. The results are listed in Table I.

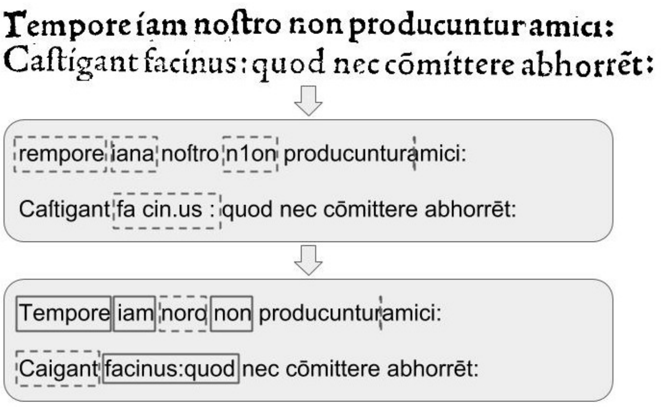
Tempore iam noſtro non producuntur amici:
Caſtigant facinus: quod nec cōmittere abhorrēt:
rempore iana noſtro n1on producunturamici:
Caſtigant fa cin.us : quod nec cōmittere abhorrēt:
Tempore iam noro non producunturamici:
Caigant facinus:quod nec cōmittere abhorrēt:

Fig. 5. Two extracted sample text line images (top) and their output after clustering (middle) and LSTM training (bottom). After clustering there are a total of 8 errors. LSTM training reduces these errors to 1, however two new errors are introduced.

IV. Error Analysis and Discussions

The top confusions of the OCR systems on the 'T0' data set are shown in Table II.

The OCRopus system produces errors related to missing or extra characters mainly. Most prominent is the missing ſ. In another model these missing ſ were instead replaced with *f* resulting in similar overall errors, however these characters have very similar shapes. More training data and even longer training should allow the LSTM to overcome most of these *deletion* and *insertion* errors. Even still the performance of LSTM networks is quite satisfactory.

Our iterative K-Means clustering approach has mostly made errors in differentiating *f* and ſ as well, followed by space related errors. The former can probably be explained due to us assigning Unicodes to the clusters rather than a language expert. The shapes are quite similar and can easily be confused. Any space related error could be reduced simply by excluding spaces from clustering. We detect them by a specific heuristic rule and could apply them directly to the output.

The anyOCR system, which has been trained on the erroneous GT data corrects many of these errors. However,

TABLE II. Top errors generated by the OCRopus, iterative K-Means, OCRoRACT and anyOCR systems when applied on the 'T0' data set. **GT** denotes the ground-truth, **Pred** denoted the predicted output and **Error** shows the number of errors. __ shows the *insertion* or *deletion* of the corresponding character. Total number of errors made by a particular OCR system are shown below the name of that system.

OCRopus			it. K-Means			OCRoRACT			anyOCR		
Total Errors		198	Total Erros		443	Total Erros		255	Total Errors		197
Pred.	**GT**	**Errors**	**Pred.**	**GT**	**Errors**	**Pred.**	**GT**	**Errors**	**Pred.**	**GT**	**Errors**
__	ſ	50	f	ſ	38		__	15	__	ſ	38
	__	8		__	31	__		12	__		26
__	l	5	__		25	__	i	10	__	l	18
i	__	3	ſt		21	ā	ē	6	__	i	5

TABLE III. Top errors generated by OCRopus and the anyOCR system when applied on the 'T1' (right) and 'T2' (left) data set. **GT** denotes the ground truth, **Pred** denoted the predicted output and **Error** shows the number of errors. _ shows the *insertion* or *deletion* of the corresponding character. Total number of errors are shown below the name of that system. OCRoRACT performance has been evaluated in [4]

OCRopus			anyOCR			OCRopus			anyOCR		
Total Errors		181	Total Erros		187	Total Erros		702	Total Errors		713
Pred.	**GT**	**Errors**	**Pred.**	**GT**	**Errors**	**Pred.**	**GT**	**Errors**	**Pred.**	**GT**	**Errors**
_	ſ	77	_	ſ	76	_	ſ	142	_	ſ	143
_	l	11	_	ſt	22	_		29	_	ſt	50
_		8	_	ę	9	_	l	20	_		38
.	_	5	.	_	16	.	_	16	_	ę	22

it struggles with the character ſ the same way OCRopus does. It also inherits many of the space related errors introduced by the clustering. Lastly the errors seem to be quite similar to OCRopus overall. A lof of the *deletion* errors are related to thin and small characters like "i" and "l". Since the LSTM is fed the text-lines in columns it might be a technical limitation of the LSTM approach used in OCRopus and anyOCR.

Comparing the errors made by these three system when applied on unseen data from the same book, given in the 'T1' data set, and from another book, given in the 'T2' data set, it becomes apparent, that OCRopus, which was trained on manually transcribed data performs only slightly better than our anyOCR approach, which has been trained on the semi-correct clustering output with a CER of $\approx 12\%$ on the CTD.

AnyOCR also performed better than the OCRoRACT system which utilizes Tesseract. Also we only used a single iteration of LSTM training, effectively cutting training time by two thirds. The top confusions as listed in Table III, show that the errors on this data is in nature very similar to the confusions on the 'T0' data set.

V. Conclusion and Future Work

For our anyOCR approach we combined the strength of segmentation-based and segmentation-free OCR systems to overcome the difficulties that arise, if no or very little training data is available. In contrast to similar approaches we further reduced the required input necessary from language experts while increasing overall performance simultaneously. Even when evaluating documents with similar script, our OCR approach still performed satisfactory. Overall our anyOCR approach achieved a CER of less than 8% and performed very similar to classic OCRopus systems trained on manually transcribed data. At the same time we reduced the necessary input from language experts regarding transcription to a few hours.

Compared to OCRoRACT, we managed to overcome the limitations given by Tesseract. No list of unique characters has to be provided nor is any artificial data used in the training process. From further comparison with the results in [4] we conclude, that iterative LSTM training is able to compensate for a high amount of errors in the ground truth data used for training. However, when the errors are small enough, it becomes unnecessary and its effects are limited. Further improvement can most likely be achieved by investigating the nature behind the reoccurring errors of the LSTM-based training with certain characters.

With our approach, however, we could not overcome the general weakness when dealing with scripts like Arabic and Devanagari, where character/ligature segmentation is non-trivial. Arabic-like scripts contain a huge amount of ligatures and the reliable segmentation of ligatures requires more research efforts.

For further improvement we plan to use a deep-learning convolutional architecture to extract better and more robust features from the individual character images before clustering and before feeding the text line images into the LSTM-network. Also we plan on further researching the properties and limitations of training LSTMs with erroneous ground truth data.

Acknowledgment

This work was partially funded by the BMBF (German Federal Ministry of Education and Research), project Kallimachos (01UG1415C). For clustering the ELKI software suit was used [12].

References

[1] N. White, "Training Tesseract for Ancient Greek OCR," *Eutypon*, pp. 1–11, 2013.

[2] F. Simistira, A. Ul-Hasan, V. Papavassiliou, B. Gatos, V. Katsouros, and M. Liwicki, "Recognition of Historical Greek Polytonic Scripts Using LSTM Networks," in *ICDAR*, Tunisia, 2015.

[3] A. Ul-Hasan, S. S. Bukhari, and A. Dengel, "Meaningless text ocr model for medieval scripts," in *2nd International Conference on Natural Sciences and Technology in Manuscript Analysis*, Germany, 2016.

[4] A. UlHasan, S. S. Bukhari, and A. Dengel, "Ocroract: A sequence learning ocr system trained on isolated characters," in *The 12th IAPR Workshop on Document Analysis Systems (DAS'16)*, Greece, 2016, pp. 174–179.

[5] B. Nunamaker, S. S. Bukhari, D. Borth, and A. Dengel, ""a tesseract-based ocr framework for historical documents lacking ground-truth text"," in *ICIP'16*, USA, 2016.

[6] T. M. Breuel, A. Ul-Hasan, M. Al Azawi, F. Shafait, "High Performance OCR for Printed English and Fraktur using LSTM Networks," in *ICDAR*, Washington D.C. USA, aug 2013.

[7] "OCRopus – Open Source Document Analysis and OCR System." [Online]. Available: https://github.com/tmbdev/ocropy

[8] M. Jenckel, S. S. Bukhari, and A. Dengel, "Clustering benchmark for characters in historical documents," in *12th IAPR International Workshop on Document Analysis Systems DAS 2016*, Greece, 2016.

[9] M. R. Yousefi, M. R. Soheili, T. M. Breuel, and D. Stricker, "A Comparison of 1D and 2D LSTM Architectures for Recognition of Handwritten Arabic," in *DRR–XXI*, USA, 2015.

[10] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks." in *ICML'06*, 2006.

[11] P. Werbos, "Backpropagation through time: what does it do and how to do it," in *Proceedings of IEEE*, vol. 78, no. 10, 1990.

[12] E. Achtert, H. Kriegel, E. Schubert, and A. Zimek, "Interactive data mining with 3d-parallel-coordinate-trees," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2013.