Project Number: **770299**

NewsEye:

A Digital Investigator for Historical Newspapers

Research and Innovation Action
Call H2020-SC-CULT-COOP-2016-2017

D2.5: Automatic Text Recognition (final)

Due date of deliverable: M24 (30 April 2020)

Actual submission date: 14 April 2020

Start date of project: 1 May 2018 **Duration:** 36 months

Partner organization name in charge of deliverable: UROS

Project co-funded by the European Commission within Horizon 2020		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	-
RE	Restricted to a group specified by the Consortium (including the Commission Services)	-
CO	Confidential, only for members of the Consortium (including the Commission Services)	-

Revision History

Document administrative information		
Project acronym:	NewsEye	
Project number:	770299	
Deliverable number:	D2.5	
Deliverable full title:	Automatic Text Recognition (final)	
Deliverable short title:	Automatic Text Recognition (final)	
Document identifier:	NewsEye-T22-D25-ATR-Submitted-v3.0	
Lead partner short name:	UROS	
Report version:	V3.0	
Report preparation date:	14.04.2020	
Dissemination level:	PU	
Nature:	Report	
Lead author:	Johannes Michael (UROS)	
Co-authors:	Roger Labahn (UROS)	
Internal reviewers:	Florian Krull (UIBK-DEA), Leo Leppänen (UH-CS)	
Status:		Draft
		Final
	x	Submitted

The NewsEye Consortium partner responsible for this deliverable has addressed all comments received, making changes as necessary. Changes to this document are detailed in the change log table below.

Change Log

Date	Version	Editor	Summary of changes made
02/02/2020	0.1	Johannes Michael (UROS)	First full draft
11/02/2020	0.2	Johannes Michael (UROS)	Modifications after internal review
24/02/2020	1.0	Johannes Michael (UROS)	Minor changes prior to internal review
20/03/2020	1.1	Johannes Michael (UROS)	Incorporating reviewer comments
23/03/2020	2.0	Johannes Michael (UROS)	Final revision
14/04/2020	2.1	Johannes Michael (UROS)	Minor modifications following quality check
14/04/2020	3.0	Antoine Doucet (ULR)	Final proofreading and submission

Executive summary

This report describes the second step of the information processing pipeline of work packages 2–5, the automatic text recognition. Based on the results of a layout analysis algorithm, we apply models to automatically transcribe the contents of segmented text line images. These models involve deep artificial neural networks designed for sequence labeling tasks.

The present deliverable is the second and last version of our work on Automatic Text Recognition. The incorporation of Language Models, as an outlook of the first deliverable, did not benefit the sequence-to-sequence models. Nevertheless, minor adaptations to last year's systems as well as modified training strategies and additional training data improved the overall performance of our models.

Contents

1 Introduction

Last year we introduced two different kinds of deep Artificial Neural Networks (NNs) designed for sequence labeling tasks that we apply to the task of Automated Text Recognition (ATR): Systems utilizing the Connectionist Temporal Classification (CTC) [1] objective function and Sequence-to-Sequence (Seq2Seq) [2] models. Both rely on an efficient usage of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). The proposed architectures built a solid baseline and reached satisfactory performance on a newspaper collection from the Austrian National Library (ONB)[1].

In the second year we tried to improve upon this foundation. Although no major changes to the underlying models were done, minor hyperparameter adaptations, as well as modified training strategies and additional training data have brought further improvements. Our ATR systems reach new peak accuracies in terms of Character Error Rate (CER). Overall, we observe a relative improvement of about $15 - 23\,\%$ over the initial benchmarks of year one, despite the fact that the initial error rates were already down to 1-2 absolute percent points. The evaluation was done on an updated version of the ONB dataset, in which several ground-truth errors were corrected.

Our CTC models still slightly outperform our Seq2Seq models. Experiments on combining the latter with Language Models (LMs) were not able to bridge the minor performance gap. Nevertheless, they still reached competitive performance compared to the state-of-the-art on two popular handwritten datasets and significantly improved over any recent Seq2Seq approaches. The corresponding work was published as a paper [2] at the 15th International Conference on Document Analysis and Recognition (ICDAR 2019) and is appended at the end of this deliverable.

This deliverable is structured as follows: After a reminder of the original description of Task 2.2 to conlude this introduction, Section 2 explains the adaptations to the ATR systems and training procedures. Afterwards, Section 3 introduces the different concepts of combining a Seq2Seq model with a LM. Section 4 presents the updated newspaper collection from the ONB and evaluates the models. Finally, Section 5 concludes this deliverable.

Task 2.2 – Automatic Text Recognition (ATR)

The following is copied from the grant agreement:
In this task we will develop methods and tools to read text line images for finally providing their textual information. This will start by adapting and incorporating existing tools, thus further enabling the entire NewsEye workflow to be run from an early project phase already. For the result of the text recognition, however, two approaches are to be followed: (a) String transcription approach directly delivers strings, and in order to achieve a highly reliable text quality, we will incorporate semantic context by means of (Neural) Language Models. The required essential contribution to the application domain will be delivered by the consortium partners ULR and UH-CS. (b) Statistical decoding approach will investigate and apply advanced decoding schemes of the posterior (character) probabilities output generated by the Neural Networks based ATR. Exploiting its convincing search capabilities for simple keywords and regular expressions, we will investigate the new approach of Topic Modeling based on statistical ATR output. This will mainly start as a research topic in synergy with WP4, developed in joint work with UH-CS and ULR.

[1] Österreichische Nationalbibliothek

2 ATR improvements

2.1 Model overview

We want to reintroduce our two different models. This section is mostly based on D2.2 (Section 2), where a more detailed description is available. Both models tackle the general sequence labeling task of taking sequences of fixed-size, real-valued vectors as input and outputting sequences of discrete labels. In this case the inputs are grayscale text line images and the targets are the corresponding character sequences present in the image. They do this in a supervised setting, where input-target pairs are presented and the goal is to minimize some task-specific error measure.

CTC-based models compute a probability distribution over all possible output sequences, given an input sequence. They do so by dividing the input sequence into frames and emitting, for each frame, the likelihood of each character of the target alphabet expanded by an artificial blank character. The probability distribution can be used to infer the actual output greedily, either by taking the most likely character at each time step or using beam search. Our ATR-CTC model combines a CNN as a generic feature extractor with an RNN to encode the visual information as well as the temporal context between characters in the input image. The CNN converts segmented text line images of variable length and fixed height into a sequence of visual feature vectors. The RNN reads the sequence of convolutional features and tries to encode temporal context between them. The model is fully differentiable and can be trained end-to-end in a supervised manner via the backpropagation algorithm. A general overview of the model is shown in Figure 1.

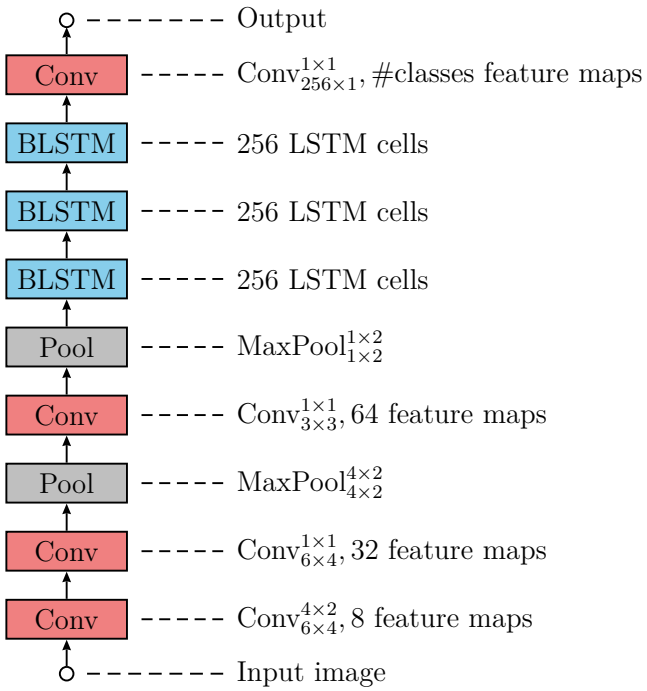

Figure 1: **The ATR-CTC model –** A stacked Conv-BLSTM architecture with interweaved pooling layers. The sub- and superscript in Conv/MaxPool$^{s_y\times s_x}_{k_y\times k_x}$ describe the size of the kernel and strides along both dimensions, respectively. The number of character channels present in the output is denoted by #classes.

Seq2Seq models that follow the encoder-decoder framework decouple the decoding from the feature extraction. First, an encoder reads and builds a feature representation of the input sequence, then a decoder emits the output sequence one token at a time. Our ATR-Seq2Seq model consists of a deep CNN-RNN-encoder and an RNN-decoder. The former matches the general architecture of previously presented ATR-CTC model. The latter employs an attention mechanism to gather context information and search for relevant parts of the encoded features. The model is fully differentiable and can be trained end-to-end in a supervised manner via the backpropagation algorithm. A more detailed look at the Seq2Seq model can be found in [2] and a general overview of the model is shown in Figure 2.

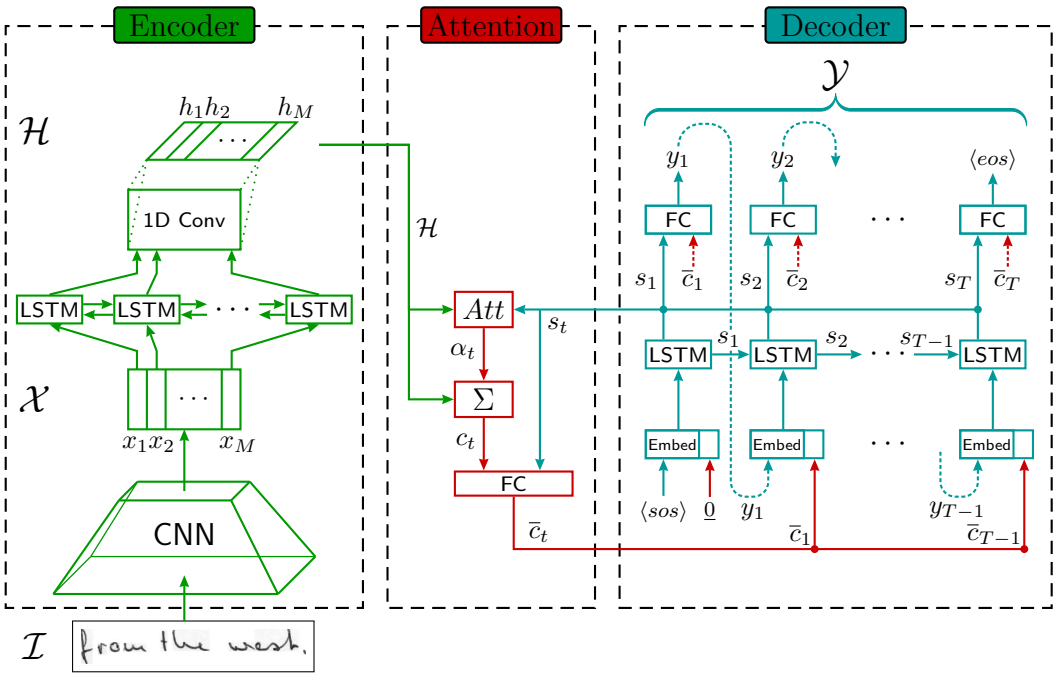

Figure 2: **The ATR-Seq2Seq model –** The encoder (which matches the ATR-CTC architecture) converts an input image $\mathcal{I}$ into a sequence of fixed-shaped feature vectors $\mathcal{H} = (h_1, \ldots, h_M)$. The decoder emits the output sequence $\mathcal{Y} = (y_1, \ldots, y_T)$ one character at a time. At each time step t, it employs an attention mechanism Att to generate a context vector $\bar{c}_t$ based on the encoded feature vectors and the time-dependant decoder hidden state s_t. This context vector is used to produce the decoders output y_t for the current time step via a fully-connected layer (FC). A concatenation of the context vector and the embedded output serves as the decoders next input.

2.2 Architectural adaptations

Only minor changes to the underlying models were done. We simplified some of the convolutional blocks, by excluding utility layers like batch normalization and local response normalization. They did not seem to affect the final performance of the evaluations and by removing them we could speed up the training and inference times.

More important was the replacement of the final model parameters (the actual weights that were trained) with an exponentially weighted moving average over their values across the training cycle. The problem is that during training, the gradients and thus the weight adaptaptions are often noisy. Therefore it can

be beneficial to smooth out the final parameters, by using averaged weights for inference. The moving averages are computed using exponential decay, i.e.

$$v_t = \delta v_{t-1} + (1 - \delta) * w_t, \tag{1}$$

where w_t are the models weights at training step t, v_t is the corresponding moving average and δ is the decay factor, which is usually close to 1.0.

2.3 Modified training strategies

Both of our ATR models are entirely differentiable and can be trained end-to-end in a supervised manner via the backpropagation algorithm. We implement them using TensorFlow, Google's C++ based open-source software library for numerical computations, mainly used for machine learning applications. This allows us to experiment with a wide variety of optimization algorithms, like ADAM or RMSPROP. One of the most important hyperparameters that ensures a robust and stable training is the learning rate, controlling how much the network parameters are adjusted during training. It is independent of the chosen optimizer and amongst other things, it is responsible for a smooth convergence of the training and can improve the final model performance. If the learning rate is too small, the training will take too long, since the network adapts very slowly. If the learning rate is too big, the network may fail to converge or even diverge. We experimented with different learning rate schemes, to improve upon a basic constant learning rate. A simple exponential decay of the learning rate over the course of the training helps the network to adapt quickly in the early stages of training and to finetune its parameters in the later stages of training. Another experimental approach was to lock the learning rate to a constant value in the beginning and to apply a cosine-like decay at the end. Overall the network converges faster with the exponential decay, but reaches better final performance with the latter approach, which is why we chose that setup in combination with the ADAM optimization algorithm as the new standard for ATR training.

We also experimented with the concept of reinitialization strategies. The core idea is to fully retrain an already pretrained model after reinitializing one or more specific layers of the network. Especially deeper layers should in theory benefit from a fresh training when they get more meaningful features from the layers below them early on. The standard case comprises a full training run, after which a layer gets randomly reinitialized, the learning rate gets reset and a second full training run is executed. Theoretically there are no constrains on how many layers can be reinitialized in this manner or how often one can repeat the training process. We did not find any literature that explored a similar technique, but even with just one reinitialization loop, we observed a performance boost of up to $10\,\%$, depending on the use case, when applied on the output or last Bidirectional Long Short-Term Memory (BLSTM) layer. Doing multiple loops can be expensive, because you have to fully retrain the model each time, but exploiting GPU capabilities, it seems feasible when one wants to really optimize a particular model.

3 Language Model integration

The decoder part of the Seq2Seq approach can be interpreted as to implicitly model language data conditioned on the encoder features. Its training requires paired data of text line images and their matching transcriptions. Often times there is additional raw text data available, that can be used to build distinct language models (LMs), which in turn may be able to improve the final prediction. A LM tries to model a probability distribution $P(w_1, \ldots, w_n)$ over sequences of token (e.g., characters or words). This

probability can be rewritten as

$$P(w_1, \ldots, w_T) = P(w_1) \prod_{i=2}^{T} P(w_i \mid w_1, \ldots, w_{i-1}). \tag{2}$$

A common type are n-gram LMs. They only consider a history of $n-1$ tokens, assuming a Markov property of order $n-1$, i.e.

$$P(w_1, \ldots, w_T) = P(w_1) \prod_{i=2}^{T} P(w_i \mid w_{i-n+1}, \ldots, w_{i-1}). \tag{3}$$

The actual conditional probabilities are modeled by counting frequencies:

$$P(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) = \begin{cases} \frac{c(w_{i-n+1}, \ldots, w_i)}{c(w_{i-n+1}, \ldots, w_{i-1})} & \text{, if } c(w_{i-n+1}, \ldots, w_{i-1}) > 0 \\ 0 & \text{, otherwise} \end{cases} \tag{4}$$

where $c(\cdot)$ counts the frequency of a sequence given in the text. In contrast, Neural LMs are NNs that are trained on text corpora.

If we want to combine our Seq2Seq model with the text-predictive probabilities given by a LM, one can imagine various ways of doing so. Popular examples in the literature include Shallow Fusion [3], Deep Fusion [3], Cold Fusion [4] and Simple Fusion [5]. These approaches differ mainly in two points: At what point the LM is integrated into the ATR models computation and at what point it is integrated into the ATR models training. Since we use n-gram LMs, which are not trainable in the common sense because they just count frequencies, we can only use a late training integration, i.e. both the ATR model and the LM need to be trained stand-alone and may only be combined afterwards during inference. And since we are not working with a neural LM, where internal states of the different NNs can be combined, we can only use a late computation integration. What we end up with is the Shallow Fusion approach. We use a fixed pretrained Seq2Seq model and a fixed n-gram LM. At inference time we then apply a log-linear interpolation of the Seq2Seq scores $p(\mathbf{y} \mid \mathbf{x})$ and the LM scores $p_{LM}(\mathbf{y})$

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \log p(\mathbf{y} \mid \mathbf{x}) + \lambda \log p_{LM}(\mathbf{y}), \tag{5}$$

and approximate $\mathbf{y}^*$ using beam search.

4 Experiments

Experiments on newspapers demonstrate minor performance gains in terms of CER compared to year one's models. Further comparisons between the two different architectures suggest that the well established CTC-based architectures should be the models of choice for the ATR task on newspaper data. We briefly go over some changes made to the datasets for the evaluation in the following and report on the corresponding results afterwards.

4.1 Data

We refer to Deliverable D2.2 (Section 3) for the introduction of the **ONB dataset**. After some thorough reevaluation of the ground-truth data during the last year we recognized various errors. These contain layout errors on baseline level (e.g., falsely split or merged baselines, too long or too short baselines etc.) as well as transcription errors on line level. We tried to manually correct these errors. The main

focus was to polish the layout information about the baselines present in the images, since an erroneous baseline will inevitable result in an incorrect extracted text line and thus in an incorrect transcription. Obvious transcription errors were fixed along the way as well, but we reckon that the corrected data is still not perfect. The updated statistics are depicted in Table 1.

For the ATR scenario the data is split into three disjoint subsets for training, validation and testing. Since the number of baselines in the updated dataset does not match the old one, new subsets were created. The text lines were shuffled and two random selections of $10\,\%$ of the lines were split off for the validation and test set. The number of segmented text lines for each subset is shown in Table 2.

Table 1: **Updated ONB statistics –** Various statistics for each of the four newspapers are shown.

		# pages	# text lines		avg # $\frac{\text{text lines}}{\text{page}}$	
			old	corrected	old	corrected
Arbeiter Zeitung	(ONB-A)	30	12993	13072	433	436
Innsbrucker Nachrichten	(ONB-I)	70	16282	17031	233	243
Illustrierte Kronen Zeitung	(ONB-K)	32	6726	6780	210	212
Neue Freie Presse	(ONB-N)	100	50830	51903	508	519
ONB overall	(ONB-All)	232	86831	88786	374	383

Table 2: **Updated ONB subsets –** The number of segmented text lines for the new subsets is shown.

Set	Number of segmented text lines				
	ONB-A	ONB-I	ONB-K	ONB-N	ONB-All
Training	10458	13625	5424	41523	71030
Validation	1307	1703	678	5190	8878
Test	1307	1703	678	5190	8878

For the upcoming LM experiments we used the so-called **3 Million German Sentences**[2] dataset. It contains german language data from the *Leipzig Corpus Collection* [6], specifically 3 million sentences taken from newspaper texts in 2015. Thus the language data represents modern german, but we could not find any other text corpora closer representing the ONB data. We preprocessed the data to exclude most of the non-latin characters. Therefore lines including mostly chinese and arabic characters, or very rare occuring symbols were pruned from the original text.

4.2 Comparative results

In Deliverable D2.2 (Section 3) we established that training a single model on the entire dataset increases the generalization and final performance compared to multiple specialized models. We therefore only focus on general models in this deliverable.

[2] available at https://www.kaggle.com/rtatman/3-million-german-sentences

Table 3: **Comparative results –** The validation and test CER for both ATR models across both years are shown on each ONB newspaper as well as the entire data set. ONB-All refers to the evaluation on the joined validation and test sets. The depicted values represent average error rates over three seperate training runs.

Dataset	CER: Validation / Test [%]			
	ATR-CTC (Y1)	ATR-Seq2Seq (Y1)	ATR-CTC (Y2)	ATR-Seq2Seq (Y2)
ONB-A	0.78 / 0.71	1.02 / 0.68	0.67 / 0.92	0.62 / 0.88
ONB-I	1.04 / 1.02	1.14 / 1.15	1.16 / 0.99	1.16 / 1.00
ONB-K	1.24 / 1.06	1.64 / 1.34	0.82 / 0.91	0.88 / 0.90
ONB-N	1.27 / 1.13	1.37 / 1.18	0.93 / 0.88	0.93 / 0.88
ONB-All	1.14 / 1.04	1.29 / 1.11	**0.93 / 0.91**	**0.93 / 0.91**

We compare our established CTC-based architecture to the Seq2Seq model, both for year one and year two. Since the underlying data was changed for the new models, the comparison is not perfectly fair. Mostly layout errors were corrected in the data, e.g., falsely split or joined baselines and missing baselines etc., which resulted in the two datasets containing a different number of baselines. Ideally we would run the older models on the corrected data, but because we work with random data splits for training and evaluation, it is highly likely that there would be an overlap between the training sets of the older models and the new validation and test sets. This in turn would produce overly positive results for the older models because of overfitting. Instead we remain with the results of the models on their respective datasets. Evaluation is made by comparing the estimated transcription of the model with the target character sequence. At this point we do not use a language model to improve the final prediction. As is common in ATR, we measure the CER, i.e. the edit distance (Levenshtein distance) normalized by the number of characters in the target. Note, that the validation CER is a representative measure, since we train our models for a fixed number of epochs and do not perform any early stopping based on the validation error. The results are shown in Table 3.

Both model variations trained on the entire corpus perform well across all subsets contained in the ONB dataset. In contrast to year one, the Seq2Seq model pulls on par with the CTC model. Overall both approaches are able to reduce the CER below $1\,\%$ on almost every subset, with the validation set of ONB-I being the only exception. Comparing the CER of ONB-All to year one's models shows a relative improvement of about $15\,\%$ for the CTC model and an improvement of about $23\,\%$ for the Seq2Seq model.

4.3 Shallow fusion

To try to further improve the Seq2Seq model, we combine it with a LM based on the previously mentionend 3 Million German Sentences dataset. In particular we use character based n-gram LMs with Kneser-Ney-Smoothing [7]. These were built using the *KenLM Language Model Toolkit*[3].

First experiments with different n-grams, beam widths and interpolation weightings showed no performance improvements on the ONB dataset. These results can be explained by the fact that the probability

[3] available at https://github.com/kpu/kenlm

distribution of the optical model is really peaky, i.e. the model is very confident in its own predictions and thus the integration of the LM probabilities has almost no effect on the beam search output. To try to alleviate this problem, two approaches were considered:

- *Label Smoothing*: For the purpose of regularization, the ground-truth label distribution is smoothed, with some fraction of the probability mass assigned to classes other than the correct one.
- *Softmax Temperature*: By scaling the logits (divide them by hyperparameter T) before applying the softmax function, the neural network produces a softer probability distribution over the classes, resulting in more diversity.

Experiments with a smoothing mass in the range from 0.01 to 0.06 were very unstable during training and thus not usable. Therefore we focused mostly on the second approach.

When increasing the softmax temperature, one has to compensate for the effects of a soft output probability distribution. We observed that the increased likelihood of the $< eos >$ token (a special token that declares the end of the sentence) makes the model cut off its predictions too early, resulting in horrendeous error rates. We tried to combat this by introducing a length normalization [8] in the beam search process, to deal efficiently with the problem of comparing hypotheses of different lengths during decoding. Otherwise regular beam search will favor shorter results over longer ones on average. Concretely, the log-probabilities get normalized in the following way:

$$lp(\mathbf{y}) = \frac{(5 + |\mathbf{y}|)^\alpha}{(5+1)^\alpha}, \quad 0 < \alpha < 1 \tag{6}$$

$$\log(p(\mathbf{y} \mid \mathbf{x})) \;\rightarrow\; \frac{\log(p(\mathbf{y} \mid \mathbf{x}))}{lp(\mathbf{y})} \tag{7}$$

This regularization technique helped lengthening the predictions to better match the targets again. Unfortunately the CER could still not be improved compared to the baseline model. It should be noted that the baseline model performs below $1\,\%$ CER, so there is not a lot of room for improvement anyway. The remaining errors could be due to bad image quality or errors in the ground-truth data. Also the text data that was used for the LM might not be able to represent the specific language found in the ONB data. A lot of possible reasons for why the Seq2Seq model was not able to surpass the CTC model in our particular use-case.

5 Conclusion

The Task 2.2 (Automatic Text Recognition) dealt with the problem of reading extracted text line images, in order to provide their textual transcription. The underlying problem is a sequence labeling task, where a system needs to find an appropriate alignment between input and output sequences of variable length. In this case, one needs to identify the correct characters at each time step without any prior knowledge about the alignment between the image pixels and the target characters.

We presented two different deep learning methods to solve this task. Firstly systems utilizing the CTC objective function and secondly Seq2Seq models. We experimentally showed the applicabilty of both models on the ATR task, specifically on a dataset of historical german newspapers. They reach very good performance rates, if enough ground-truth data of adequate quality is available. Comparing the baseline set in year one with the models from year two, we can observe a relative improvement of about $15\,\%$ for the CTC models and about $23\,\%$ for the Seq2Seq models. In absolute terms, the current models reach CERs below 1 percent.

We were also able to show that the models are able to generalize across different kinds of newspapers, given that the language does not differ too much. This is because both models are language dependant, in the sense that they can only transcribe text consisting of characters that they have seen in the training data. For example, it would not be possible to apply a german model on arabic newspapers, since the underlying character sets differ completely (latin vs non-latin).

Comparing the CTC model with the Seq2Seq model shows no clear winner in terms of CER on this particular dataset, even when combining the latter with an n-gram LM. The former comes out ahead in terms of training and inference time though, since it is a smaller model with less parameters.

References

[1] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks”. In: *ICML*. Jan. 2006, pp. 369–376.

[2] Johannes Michael, Roger Labahn, Tobias Grüning, and Jochen Zöllner. “Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition”. In: *ICDAR*. Sept. 2019. DOI: 10.1109/ICDAR.2019.00208.

[3] Caglar Gülcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. *On Using Monolingual Corpora in Neural Machine Translation*. 2015. arXiv: 1503.03535 [cs.CL].

[4] Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. *Cold Fusion: Training Seq2Seq Models Together with Language Models*. 2017. arXiv: 1708.06426 [cs.CL].

[5] Felix Stahlberg, James Cross, and Veselin Stoyanov. *Simple Fusion: Return of the Language Model*. 2018. arXiv: 1809.00125 [cs.CL].

[6] D. Goldhahn, T. Eckart, and U. Quasthoff. “Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages”. In: *LREC*. 2012.

[7] Hermann Ney, Ute Essen, and Reinhard Kneser. “On structuring probabilistic dependences in stochastic language modelling”. In: *Computer Speech and Language* 8 (1994), pp. 1–38.

[8] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. *Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 [cs.CL].