READ

RECOGNITION & ENRICHMENT
OF ARCHIVAL DOCUMENTS

D7.7
HTR Engine Based on NNs P1

Concepts, Integration and Experiments

Tobias Grüning, Gundram Leifert, Tobias Strauß, Roger Labahn
URO

Distribution: http://read.transkribus.eu/

READ
H2020 Project 674943

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 674943

Project ref no.	H2020 674943
Project acronym	READ
Project full title	Recognition and Enrichment of Archival Documents
Instrument	H2020-EINFRA-2015-1
Thematic priority	EINFRA-9-2015 - e-Infrastructures for virtual research environments (VRE)
Start date/duration	01 January 2016 / 42 Months

Distribution	Public
Contract. date of delivery	31.12.2016
Actual date of delivery	30.12.2016
Date of last update	21.12.2016
Deliverable number	D7.7
Deliverable title	HTR Engine Based on NNs P1
Type	Demonstrator
Status & version	Final
Contributing WP(s)	WP7
Responsible beneficiary	URO
Other contributors	UPVLC
Internal reviewers	George Retsinas (NCSR), Joan Puigcerver (UPVLC)
Author(s)	Tobias Grüning, Gundram Leifert, Tobias Strauß, Roger Labahn
EC project officer	Martin Majek
Keywords	RNN, segmentation free, context, influence of training data

Contents

Executive Summary

The first years deliverable describes the importance of the Neural Network based HTR and its basic concepts. It is mentioned that the focus of the first year was the integration of UROs existing Neural Network based HTR into Transkribus. Finally, results of two experiments regarding the influence of the amount of training data on the error rate and the combination of Neural Network based HTR with HMM language modeling are described.

1. Introduction

Since 2009 artificial Neural Networks (NN), more precisely so-called *Recurrent Neural Networks* (RNN), are becoming more and more important in the fields of *Handwritten Text Recognition* (HTR) or *Keyword Spotting* (KWS). Starting at the ICDAR2009 conference, nearly all competitions organized in the fields of HTR or KWS were won by teams using RNNs. Therefore, a fast ingegration of this technology into Transkribus was one focus of the first year of READ. Besides, further research and development to improve HTR accuracy as well as to simplify the usage of RNNs are necessary – Task 7.3 formulates this.

Task 7.3 - Neural Network based HTR

The following is copied from the Grant Agreement:
This task comprises all investigations concerning Neural Network (NN) based HTR technology. It focuses on the NN training based upon a sufficiently large number of line images along with their textual transcripts, and how the NN output can be effectively decoded into word graphs for further processing in the overall workflow. This covers both training and decoding rely on well-known machine learning techniques that were extensively used and extended by the UPVLC and URO. Further flexibility, improvement and particular issues require ongoing research: maintaining stability of the training process for avoiding degenerated NNs; speeding up the training process for advanced flexibility; adapting to specific data for effective extendibility; researching on word-based graphs and character-based graph generated from NN-based decoders for interactive transcription and KWS; incorporating advanced language models into the decoding procedures for improved overall performance in transcription and search tasks. Furthermore the NN based HTR engine available at URO will be customized for use in the project workflow. From a research or an implementation point of view, we will then investigate the interactions of the algorithms or the engine, respectively, across the interfaces to adjacent workflow modules.

2. RNN Integration

As already mentioned in the introduction, a lot of work was done to integrate UROs RNN based HTR engine into Transkribus. To facilitate a smooth integration as well as a

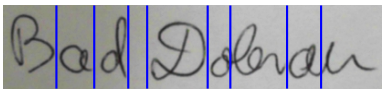


Figure 1: text-line image with a difficult character segmentation scenario.

parallel usage of HMMs and RNNs, it is essential to define interfaces, which are then to be implemented by all modules. The according interface definition was a collaborative work of UIBK and all technical partners (UPVLC, ASV, NCSR, CVL, URO), their implementation was subsequently done by URO, and they are now publicly available via GitHub at https://github.com/Transkribus/TranskribusInterfaces.

Interface Implementation

The defined interfaces had to be implemented for the URO based HTR to make it available in Transkribus. The resulting implementations are on GitHub at https://github.com/Transkribus/CITlabModule (Up to now it is a private repository, but will be publicly available in the near future.)

3. Concepts of RNN Based HTR

In this section, basic concepts of the RNN based HTR framework are introduced. Basically, RNN based HTR models allow a segmentation free, context sensitive processing of text-line images. The model parameters are adjusted via training, and therefore it is possible to generate models for different corpora easily. Finally, the output of such an RNN based model is a so-called *confidence matrix* (ConfMat), which is a very flexible output structure, enabling us to incorporate HMM approaches (to improve HTR accuracy) or to do a very powerful regular expression decoding (resulting in a state of the art keyword spotting system). These concepts will be explained roughly in the next subsections.

3.1. Segmentation Free Processing

Classical approaches for making the content of scanned documents automatically accessible rely on the segmentation of text into isolated characters. Such so-called OCR (Optical Character Recognition) approaches fail to tackle problems were a segmentation into single characters is not easily accessible or simply not possible (see Fig. 1). This scenario often occurs in handwritten texts (and especially historical handwritten texts). Hence, RNNs (as well as HMMs) process a text-line image as a data sequence (see Fig. 2), what is meaningful and highly motivated by the way we humans process text.

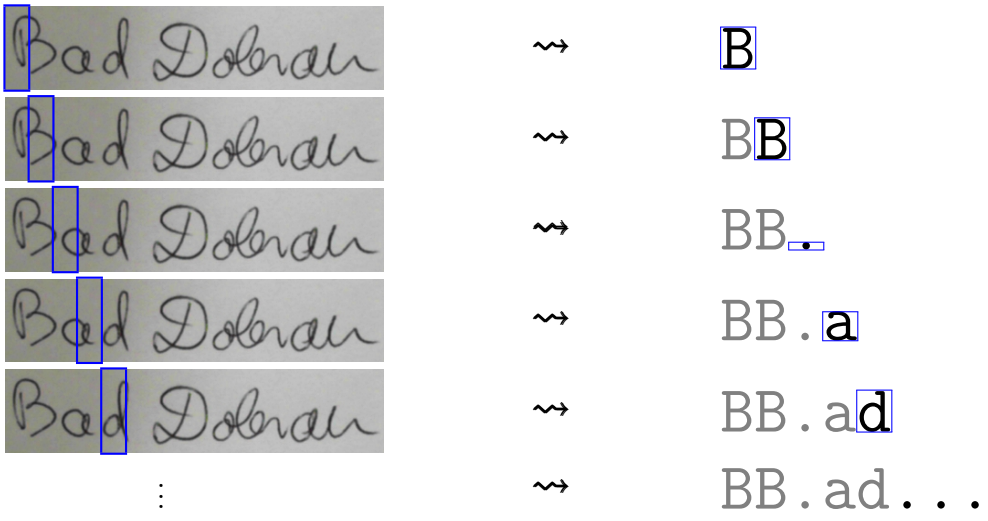


Figure 2: Schematic depiction of the sequential processing of a text-line image.

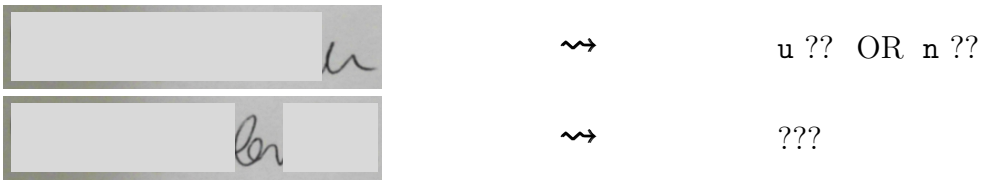


Figure 3: Context is essential for a high HTR accuracy. Isolated characters or character sequences are sometimes hard or even impossible to transcribe.

3.2. Context Sensitive Processing

Since the recognition of isolated characters or sequences of characters is hard or even not possible in various situations (see Fig. 3), a context sensitive processing is essential. In the HTR scenario, context sensitive processing means that the already processed signal influences the further processing of the signal. In the RNN based HTR framework, this is realized via so-called *recurrent connections*. This means that at each time step (position in the input image), besides the visual input, the system also gets its own previous results of the last time-step as input. This is shown in Fig. 4 (blue arrows). Further information can be found in [2].

3.3. RNN Model Parameter Adaption via Learning

The RNN based HTR model's applicability and accuracy depend on the model parameters (whose number can easily reach millions). Since the model is fully differentiable, paradigms of machine learning are directly usable, which means that the model parameters are adjusted by showing the model a certain amount of training samples. These are simply pairs of text line images as inputs and the corresponding text transcripts. Basically, an error between the model's output (given a training sample image) and the correct transcript is calculated for each training pair. The derivative of this error is propagated through the model and used to adapt the model parameters slightly.

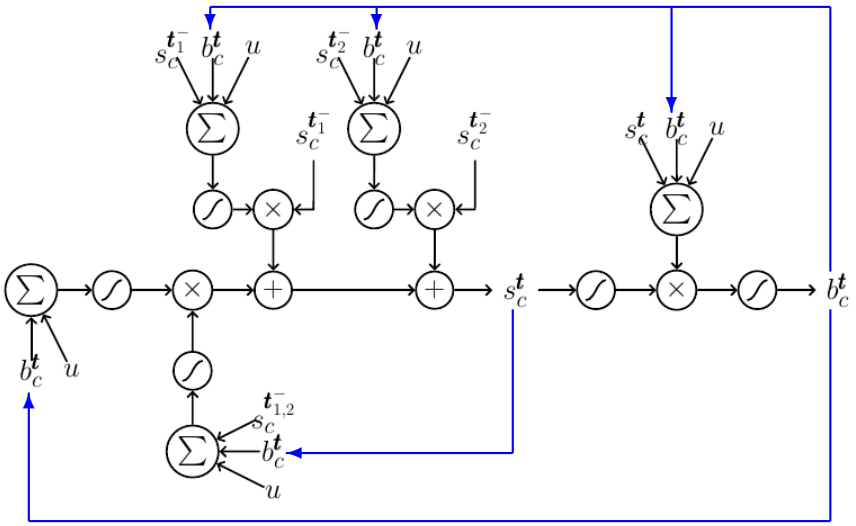


Figure 4: Schematic representation of a complex recurrent cell - blue arrows depict recurrent connections, these facilitate context sensitive processing of the input sequence.

The amount of training data is a crucial value and differs among different writing styles, time periods, ... as well as limitations to the amount of usable training data is often an issue. Consequently, various experiments were performed to investigate the influence of the amount of training data on the HTR accuracy. Furthermore, we also investigated the influence of different approaches to increase the amount of training data, see Sec. 4.

3.4. ConfMat – RNN Model Output

A great benefit of the RNN model is the output structure - the so-called *confidence matrix* (ConfMat). An example ConfMat and the corresponding text image are shown in Fig. 5. In general, each row of a ConfMat belongs to a position in the image and contains a conditional distribution over all possible (fixed due to the model) characters. Thus, an entry of the ConfMat is a real number between 0 and 1 representing the probability of a specific character at a certain image position (in Fig. 5 darker means more likely). Because no hard binary (true-false) decision is made, no information is lost and ConfMats contain all information necessary to allow a powerful and fast decoding process. A decoding approach using regular expressions [4] leads to very flexibel systems, as proven by UROs system [3] when winning the ImageCLEF 2016 competition [5]. Furthermore the ConfMats can be used as inputs for HMMs. An example and results of such a hybrid system are explained in Sec. 5.

4. Training Data and Noise Impact

Since the required amount of training data and the effect of more training data regarding the HTR accuracy are of immense importance for users, experiments were performed to investigate this relation. The so-called *German Konzilsprotokolle* collection was used.

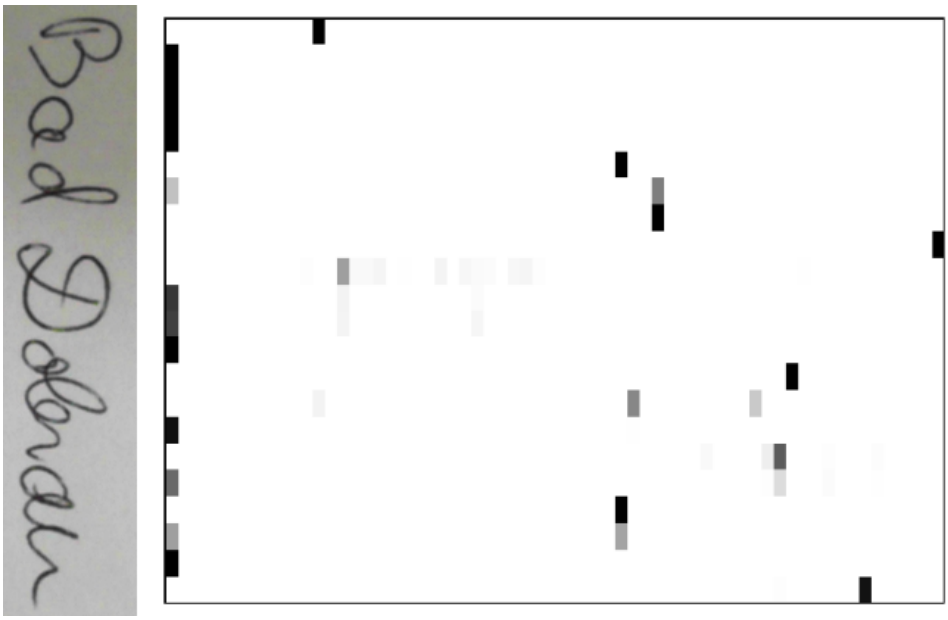


Figure 5: Image writing with corresponding ConfMat.

This collection consists of hundreds of pages and is maintained by a READ MoU partner. 8770 text-line images (with their transcripts) were extracted. This set was divided into 6857 samples to train the model (training set $\mathcal{T}$) and 1913 samples to validate the model (validation set). The dataset is publicly available [1]. In different experiments, models were trained on subsets of $\mathcal{T}$ to investigate the impact of the reduction of training data. Furthermore different types of noise to artificially increase the amount of training data (to reduce overfitting/increase generalization) were introduced. We differentiate between preprocessing noise (pp - which is noise on the parameters of the image processing algorithms preparing the writing image for the neural network) and neural network noise (nn - which is dropout like noise on the neural network activations during training). These types of noise will not be explained in more detail here, but basically they aim at artificially augmenting the amount of data. All models were trained under comparable conditions: same learning rate, momentum, minibatch size, as well as a consistent number of training samples shown per epoch. Results are depicted in A.1.

5. RNN/HMM Hybrid System

In a joint work of UPVLC and URO during UPVLC's Researcher Internship at URO the combination of RNNs and HMMs was investigated. Collaborative activities were held to improve the RNN-based handwritten text recognition system by incorporating and testing different n-gram language models at word and at character level. Final results have shown that the best HTR performence is obtained by using a 5-gram language model at character level, which additionally deals with the problem of out-of-vocabulary-words, i.e. words not known from the training set. The results are outlined in details in the READ wiki (and attached as A.2 here).

References

[1] Tobias Grüning et al. *read_dataset_german_konzilsprotokolle*. Dec. 2016. URL: https://doi.org/10.5281/zenodo.215383.

[2] Gundram Leifert et al. "Cells in Multidimensional Recurrent Neural Networks". In: *Journal of Machine Learning Research* 17.97 (2016), pp. 1–37. URL: http://jmlr.org/papers/v17/14-203.html.

[3] Tobias Strauß et al. "CITlab ARGUS for Keyword Search in Historical Handwritten Documents: Description of CITlab's System for the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task". In: CEUR Workshop Proceedings. Évora, Portugal, Sept. 2016. URL: http://ceur-ws.org/Vol-1609/16090399.pdf.

[4] Tobias Strauß et al. "Regular expressions for decoding of neural network outputs". In: *Neural Networks* 79 (2016), pp. 1–11. URL: http://www.sciencedirect.com/science/article/pii/S0893608016000447.

[5] M Villegas, J Puigcerver, and AH Toselli. "Overview of the ImageCLEF 2016 handwritten scanned document retrieval task". In: CEUR Workshop Proceedings. Évora, Portugal, Sept. 2016. URL: http://ceur-ws.org/Vol-1609/16090233.pdf.

A. Appendix

A.1. Training Data and Noise Impact – Results

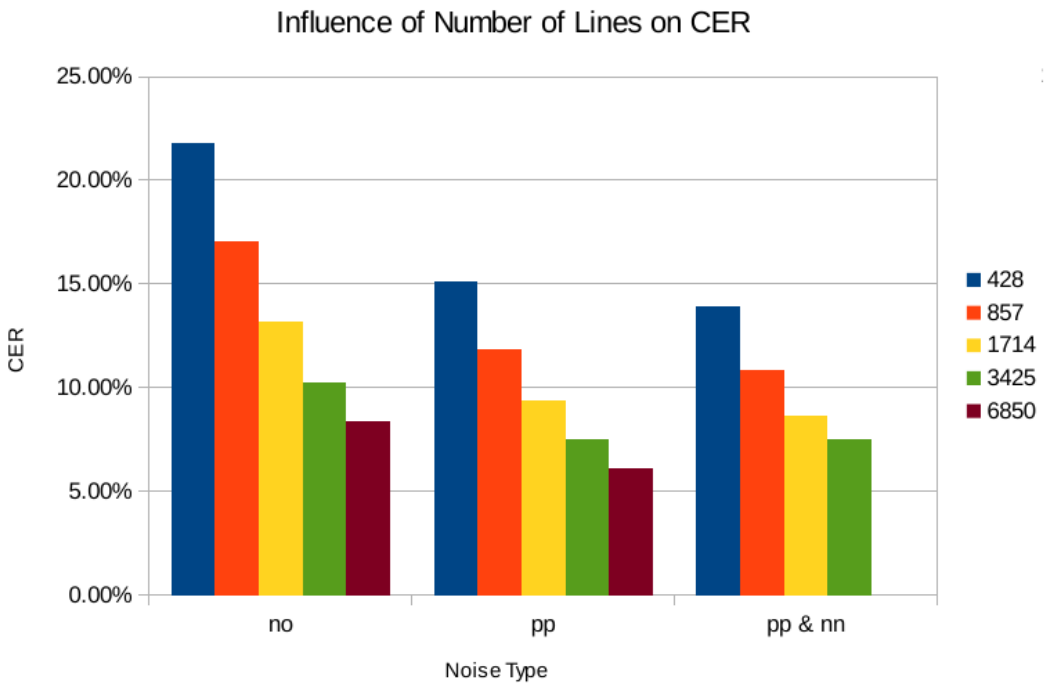


Figure 6: This figure shows the influence of the number of lines used for training and of noise on the HTR error (CER). The models were trained using the number of lines shown in the right legend and validated on a set of 1913 lines for all experiments. It is easy to see, that more training data yields lower error. This behaviour saturates at a certain point. Furthermore, noise leads to a significant improvement of the HTR accuracy for the same amount of training data.

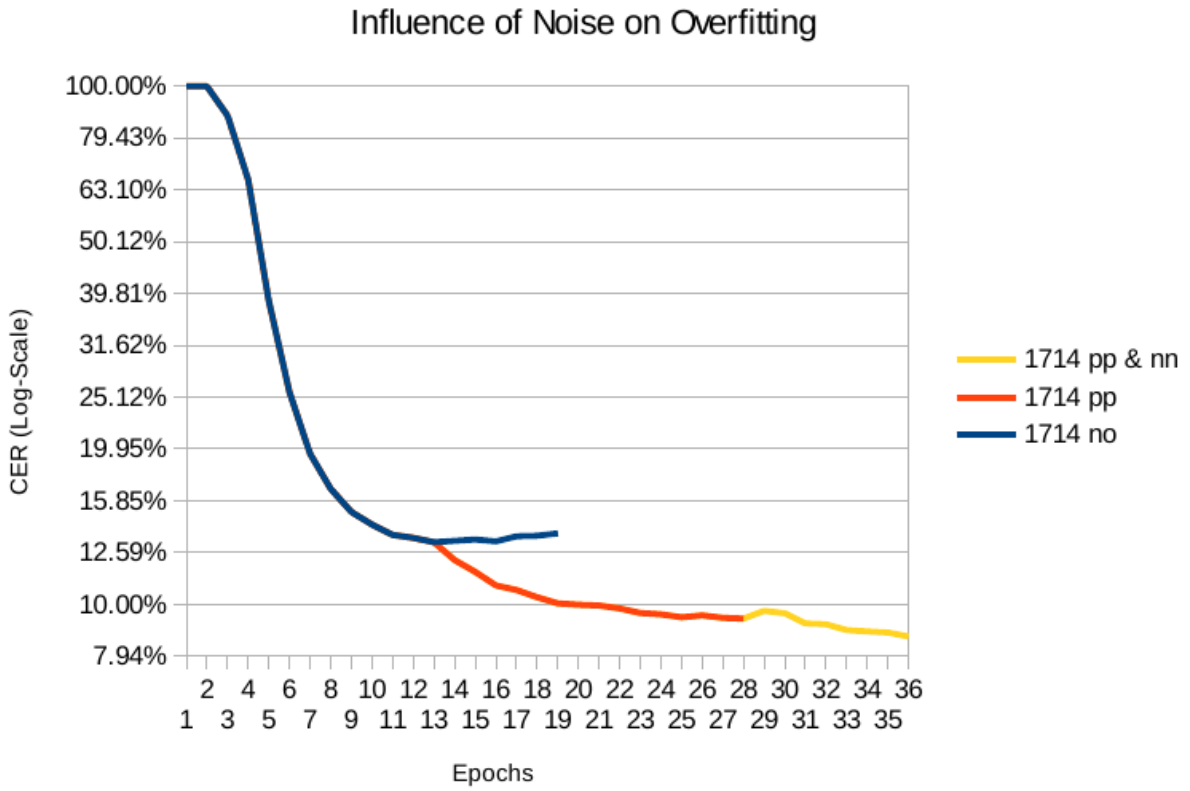


Figure 7: This figure shows the influence of noise to overfitting. The models were trained with 1714 training samples. With the classical training approach the error on the validation set decreases for the first 13 epochs. Afterwards the model starts to overfit (error on the validation set increases). Introducing noise in epoch 14 yields a further drop of the validation error.

A.2. RNN/HMM Hybrid System – Results

The following is taken from the read wiki.

Alejandro @ URO (14.05.2016 - 17.06.2016) [edit]

Participants [edit]

- Alejandro, Tobias S., Gundram, Tobias G., Roger

Mayor task [edit]

The task is to understand the systems from each other so that one can combine the systems at different levels. We will start at the bottom (image, PageXML with baseline) go through normalization and feature extraction to the "real" HTR system. URO and UPVLC will take a baseline system and will try to improve it by plugging in different components.

Dataset used for HTR Evaluation and Assessment Measure [edit]

HTR performance under the different component/module combination is evaluated on Bozen dataset used in the:ICFHR2016 Competition on Handwritten Text Recognition on the READ Dataset .

Basic statistics of this dataset are given in the below table. Running words, lexicon size and running OOVs figures are reported for the tokenized version of the transcripts.

	Training	**Validation**
#Pages	350	50
#Lines	8367	1043
Run-Words	35169	3994
Lex-Size	6993	1527
Run-OOV(%)		16.7

As assessment measures we employ:

- **WER**: Word Error Rate
- **CER**: Character Error Rate

Both are computed on tokenized reference transcripts and recognized hypotheses.

It was also discussed about using an external corpus to provide more text for training language model and for additional lexicon. We tried Alvermann Konzilsprotokolle Alvermann, but the new ROOV didn't decrease significantly (16.35%).

Components [edit]

Both HTR systems can be separated into different components. The input and output of theses components can be images or feature maps in lower levels and ConfMats and word graph in higher levels. The format to load/save a binary feature is described in Featurefile:Format. For each component we will describe the main goal, the method, the desired input and the resulting output of each component.

UPV HTR System: Component Description [edit]

UPV HTR system comprises the following components:

1. Line extraction: (shortcut **R**) Line images are extracted from original pages images employing the poly-rectangles computed using the provided baselines.
2. Line-level preprocessing: Extracted line images are preprocessed for improving contrast and correcting skew and slant.
3. Aachen feature extraction: Applied on preprocessed line images, generating for each of them a 256-dimensional feature vector sequence. PCA is applied in order to reduce the dimension of sequences to 20. 4 extra components (moments coefficients used in the normalization process) are added to the

previous 20 components. At the end, each line image preprocessed is represented by a 24-dimensional feature vector sequence. Size normalization is carried out in this process implicitly.

4. Training module: Using previously feature vector sequences and corresponding transcripts, Baum-Welch algorithm is employed to train character HMMs. In addition word-level 2-gram language model is trained using the transcripts of the training partition. Words in the lexicon are modeled by concatenating corresponding characters (HMMs).
5. Decoding module: for each input feature vector sequence of the test partition, the best word sequence is outputted (Viterbi Algorithm) by the HMM-based HTR recognizer employing trained character HMMs, lexicon model and word-level 2-gram model.

*Contrast enhancement*method for line-level preprocessing is similar to the Sauvola binarization approach, but in this case the original gray-level of foreground (i.e. handwritten strokes) is actually kept. On the other hand, *skew/slant*correction method is based on maximizing the corresponding horizontal/vertical projection profile for different rotation/shear angles.

(shortcut **A**) the component group: 1, 2 and 3 (without applying PCA).

Modeling Issues and Adopted Parameter Values [edit]

Aachen Feature Extraction parameters:

Sliding window width = 32 pixels

Sliding window step = 2 pixels

Scaled width = 8 pixels

Scaled height = 32 pixels

Outputted feature vectors: 256 components + 4 components based on the 1st and 2nd order moment coefficients computed

Dimension reduction (PCA): 256 --> 20 + 4

For HMM and Language Model training:

- For training n-gram each transcript of line image is enclosed between . These two meta-symbols have now associated morphological models: <BS> (begin sentence) and <ES> (end sentence) respectively modeled with HMMs of 3 states.
- In the dictionary, for each word in addition to specify the constituent HMM characters, we add the morphological symbols: <is> (initial space) and <fs> (final space) also modeled with HMMs of 3 states.

HMM Topology: For the 92 different characters, in general 6 states and 64 Gaussians per state were set up. The exceptions are:

Symbols	#HMM-States	
. , : ; | ! '	3	
( )	4	
i	5	
M	7	
<is> <fs>	3	word initial and final spaces
<BS> <ES>	3	Beginning and Ending Sentence

Language Model used:

Word-Level (WL):

2-gram model trained from the 8367 training transcripts

Smoothing method applied: Kneser-Ney

Lexicon based (6993 different words)

Character-Level (CL):

3-, 7- and 8-gram models trained from the 8367 training transcripts

Smoothing method applied: Kneser-Ney

Lexicon free (92 chars)

Baseline recognition result using the full pipeline [edit]

Baseline recognition result obtained for the full pipeline (components: 1-5) before described.

WER_WL	CER_WL
57.7%	27.6%

The CER_WL figure was optimized on the validation set tunning GSF and WIP language model parameters. The suffix **_WL**means that it was computes using *Word-Level*language model (i.e. 2-gram word model).

URO HTR System: Preprocessing Component Description [edit]

Seam Carving [edit]

(shortcut **S**) The input of this module is an image with a corresponding PAGE-Xml file. It collects the baselines from the xml file and separates the image between these baselines using the seam carving algorithm. So for each baseline it calculates one surrounding polygon (independent from the polygon saved in the Coords-tag). The output is a set of lines. For each line it saves

- the sub image (bounding box of the surrounding polygon)
- the corresponding text (from the Unicode-tag)
- additional information (file name, position in original image, surrounding polygon,...).

Parameters

- distcost: Attracts the surrounding polygon to the baseline
- recalc: Captures missed punctuations (hyphens, dots, diaeresis,...)

Contrast Enhancement [edit]

(shortcut **C**) This modules enhances the contrast of the image. It does not make binarize! It assumes that there is a minimal percentage of foreground and background pixels and that there are more background pixels than foreground pixels. It calculates a threshold for the foreground/background intensity. Intensities above and below these thresholds are set to white/black, between the threshold the intensity is scaled lineary.

Parameters

- foreground quantile (1-5%)
- background quantile (30-70%)
- slope (effects a dynamic background threshold calculation)

URO Writing Normalization [edit]

(shortcut **U**) This module assumes an image with white background and black foreground. It tries to minimize the variability of the writings. All image manipulations are done as "soft" as possible (small changes in the image should not have a great effect on the normalization). The size of the writing is normalized so that a globally specified inter quantile distance has a desired height. The skew is corrected by applying a horizontal triangular filter on local pixel intensities. A shift in y-direction is applied so that at each x-position the quantile of the smoothed pixel intensities is at the same height. The slant in corrected so that the average slant between -45° and 45° is zero.

Parameters

- quantile size (30-60%)
- target inter quantile distance
- size of triangular filter
- quantile for skew normalization (50%-60%)

Squashing [edit]

(shortcut **Q**) The input image for this module is of arbitrary height and width with a normalized writing (see Writing Normalization). The Aim of this module is to squash the height of the image to a specific value by changing the image as less as possible. A baseline is detected by using a p-quantile. From this baseline a specific number of pixels above and below are left unchanged. The other ranges (ascenders and descenders) are squashed to a specific height using the tanh as squashing function.

Parameters

- p-quantile (50%-60%)
- upper/lower mainbody height
- ascender/descender height

Feature Extraction [edit]

(shortcut **F**) As feature extraction a gabor-like feature extractor is applied to the image. For each frequency and angle one gets a feature map, which is smaller than the original image (dependent on the subsampling). In one experiment we change the subsampling rate in y-dimension from 4 to 2 which is denoted by the suffix '-ssy2'.

Parameters

- subsampling in x and y (2-4)
- number of frequencies (1-3)
- number of angles (3-4)

Network Features [edit]

(shortcut **N**) A trained MDRNN provides for given input features the ConfMat. This ConfMat contains the confidence of a character at a specific position. When the Confmat is normalized using softmax, the values can be interpreted as character posterior probability. The ConfMat can also be interpreted as feature. The number of features are then the number of characters and one feature for "Not a Character" (NaC). The the number of features is 60-150.

The network has a kind of shrink-factor in x-direction, so that the number of features is lower by a network-specific factor (4-8). To solve this problem one can "unsample" the trained network to get the same number of features as the input.

In addition the internal activations of the last hidden layers can be seen as features. The number of features varies dependent on the MDRNN layout. These features contain all information to produce the ConfMat - the ConfMat is a trained linear combination of these features. Typically they are bounded in [-1,1]. The number of features is 200-1000.

Parameters

- take ConfMat as feature (true)
- take activations in last hidden layer as features (true)

Preprocess Noise [edit]

(shortcut ***) To increase the variability of training samples it is possible to add noise to the training process. Instead of adding noise to the image (pixel noise, blur, degradation, bleedthrough,...) we add noise to the parameter of the preprocess. In most cases we add uniform distributed noise to quantiles. This noise is denoted by "parameter" After a specific number of (pre-)training epochs the noise is added - this improves the error rate on the validation set significantly when the training set is small. It is also possible to add additional gaussian noise to the feature **F** - this noise is denoted by "activation".

Baseline recognition result using the full pipeline [edit]

The baseline system is done using **SCUQF2**features. The MDRNN has two multidimensional multidirectional layers (MD layers) with LeakyLP cells. Between both MD layers we put a feed forward layer because of parameter dimension reduction and better generalization.

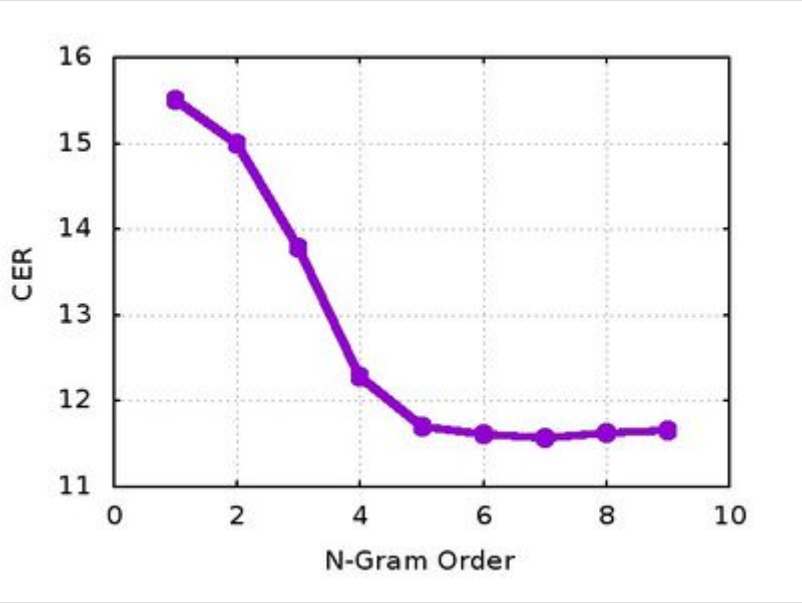
Combination of HTR Components and Evaluation [edit]

UPV HTR System: Testing URO preprocessing modules [edit]

1. **BSL**: Baseline recognition result at WL obtained for the full pipeline (components: 1-5) before described.
2. **SCU**:
   - Preprocess: Seam-carving line extraction, contrast enhancement and normalization.
   - Feature Extraction: Aachen FE (PCA): 24-dimensional features vectors
   - Optical Modelling: character HMMs, with BSL topology, trained from feature extraction data
   - Lex & LM Modelling:
     - **WL**: lexicon and word level 2-gram
3. **SCUQF**:
   - Preprocess: Seam-carving line extraction, contrast enhancement, normalization and squashing line height.
   - Feature Extraction: URO's Gabor-like FE (PCA): 24-dimensional features vectors
   - Optical Modelling: character HMMs, with BSL topology, trained from feature extraction data
   - Lex & LM Modelling:
     - **WL**: lexicon and word level 2-gram
4. **RCUQF**:
   - Preprocess: Poly-rectangle line extraction, contrast enhancement, normalization and squashing line height.
   - Feature Extraction: URO's Gabor-like FE (PCA): 24-dimensional features vectors
   - Optical Modelling: character HMMs, with BSL topology, trained from feature extraction data
   - Lex & LM Modelling:
     - **WL**: lexicon and word level 2-gram
     - **CL**: non-lexicon and character level 8-gram
5. **RCUQFN-H**:
   - Preprocess: Poly-rectangle line extraction, contrast enhancement, normalization and squashing line height.
   - Feature Extraction: output activations of the last hidden layer of the RNN (PCA): 24-dimensional features vectors
   - Optical Modelling: character HMMs, with BSL topology, trained from feature extraction data
   - Lex & LM Modelling:
     - **CL**: non-lexicon and character level 7-gram
6. **RCUQFN-M**:
   - Preprocess: Poly-rectangle line extraction, contrast enhancement, normalization and squashing line height.
   - Feature Extraction: n/a
   - Optical Modelling: 0ne-state character HMMs, whose state emission probabilities are taken directly from the RNN's confidence matrix.
   - Lex & LM Modelling:
     - **CL**: non-lexicon and character level 8-gram

	WER_WL	CER_WL	CER_CL	Frames	Length
BSL	57.7%	27.6%	--	494	260 / 24
SCU	59.0%	31.2%	--	539	260 / 24
SCUQF	63.5%	35.3%	--	528	128 / 24
RCUQF	58.3%	28.6%	28.0%	539	128 / 24
RCUQFN-H	--	--	13.3%	531	400 / 60
RCUQFN-M	--	--	11.6%	531	89 / --

The **CER_WL** and **CER_CL** figures were optimized on the validation set tunning GSF and WIP language model parameters. Column labelled with **Frames**reports average number of frames per sample. Column labeled with **Length**, shows dimension of feature extraction before / after applying PCA.



The plot of CER vs N-gram order corresponds to the confidence matrix obtained with the **RCUQF-M** process (CER ~ 18%). The table below shows **CER_CL** using char-level 6-gram for different number of states per each character HMM.

	#HMM-States	CER_CL
RCUQFN-M	1	11.6%
RCUQFN-M	4	11.1%

For the **RCUQFN-M**figure, the NaC (NoT a Character) State-Loop Probability was optimized (SLP=0.05).

URO HTR System: Testing UPV preprocessing and FE modules (A) [edit]

To evaluate the influence of the preprocess we train the described MDRNN using stochastic gradient decent. We take the character error rate (CER) after 5 and 10 epochs on the validation set. The name of the experiment is the order of preprocessing modules which we applied. A '-<description>' shows how we change the preprocess from the default implementation. The baseline system is **SCNQF**. To make the feature comparable we try to hold the properties constant

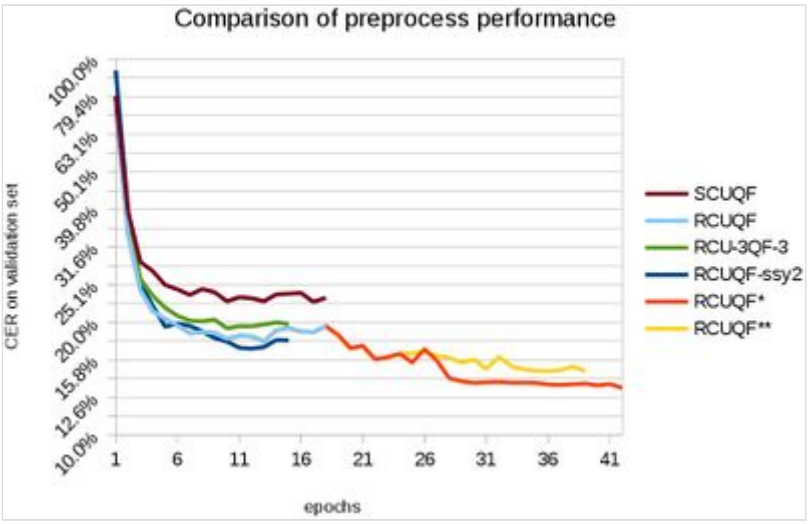
- **Frames**: Average number of frames for one sample/line
- **Length**: Dimension of the feature vector for one frame

To have nearly the same number of frames, the target height for size normalization of module U have to be 17px times larger than the subsampling in x-direction of module F. In the baseline system the height is 34 and the subsampling 2 accordingly. In one experiment we changed to height 51 and subsampling 3, which we denoted by '-3' as suffix to the process steps.

Setup	CER-5	CER-10	Frames	Length
SCUQF	25.1%	22.7%	520	128
RA	25.0%	22.7%	493	260
RCUQF	20.4%	18.0%	532	128
RCU-3QF-3	21.8%	19.2%	530	128
RCUQF-ssy2	19.4%	17.7%	532	256

Setup	Epochs	Learning rate	Noise	CER
RCUQF*	18	5e-3	no	19.5%
	9	5e-3	parameter	15.8%
	15	1e-3	parameter	13.4%
	18	5e-3	no	19.5%

RCUQF**	6	5e-3	parameter	16.5%
	15	5e-3	parameter and activation	14.8%



URO HTR System: Testing Different Decoding Strategies [edit]

The decoding should be directly comparable to **RCUQFN-M**.

- **no LM**:
  - The output is the most likely character sequence from the ConfMat
- **1-gram**:
  - Segmentation by simple regular expression, vocabulary lookup on word regions
  - Default system (automatically generated regular expression and vocabulary)
- **1-gram, beam**:
  - Segmentation by simple regular expression, vocabulary lookup on word regions
- **1-gram, optimal segmentation**
  - As **1-gram**but **forced segmentation based on the groundtruth**(lower bound for reachable error-rate)

LM	CER
no LM	18.0%
1-gram	14.3%
1-gram, beam	14.4%
1-gram, optimal segmentation	12.2%