CITlab ARGUS for Keyword Search in Historical Handwritten Documents

Description of CITlab's System for the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task

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

CITlab, Institute of Mathematics, University of Rostock, Germany {tobias.strauss, tobias.gruening, gundram.leifert, roger.labahn}@uni-rostock.de

Abstract. We describe CITlab's recognition system for the *Handwritten Scanned Document Retrieval Task 2016* attached to the CLEF 2016 hold in the city of Évora in Portugal, 5-8 September 2016 (see [9]). The task is to locate positions that match a given query – consisting of possibly more than one keyword – in a number of historical handwritten documents. The core algorithms of our system are based on multi-dimensional recurrent neural networks (MDRNN) trained by connectionist temporal classification (CTC). The software modules behind that as well as the basic utility technologies are essentially powered by PLANET's ARGUS framework for intelligent text recognition and image processing.

Keywords — MDRNN, LeakyLP cells, CTC, handwriting recognition, neural network, keyword spotting

1 Introduction

The Conference and Labs of the Evaluation Forum (CLEF) 2016 ([9]) hosts a variety of competitions. Among others, the *Handwritten Scanned Document Retrieval Task 2016* competition on the *tranScriptorium* Dataset (HTRtS) attracted our attention because we expected CITlab's handwriting recognition software to be able to successfully deal with the respective task.

Our neural networks have basically been used previously in the international handwriting competitions OpenHaRT 2013 attached to the ICDAR 2013 conference (see [4]), the HTRtS14 and ANWERSH14 attached to the ICFHR 2014 conference (see [8,3]) as well as in KWS15 and HTRtS15 attached to the ICDAR 2015 conference (see [6]).

Affiliated with the Institute of Mathematics at the University of Rostock, CITlab[1] hosts joint projects of the Mathematical Optimization Group and PLANET intelligent systems GmbH[2], a small/medium enterprise focusing on computational intelligence technology and applications. The work presented here is part

[1] http://www.citlab.uni-rostock.de

[2] http://www.planet.de

of our ongoing text recognition projects and is extensively based upon PLANET's ARGUS software modules and the respective framework for development, testing and training.

Task The *Handwritten Scanned Document Retrieval Task 2016* aims at an advanced keyword spotting. Besides ordinary keyword search, the competition comprises the detection of multiple word queries consisting of possibly hyphenated keywords within sections. The writings used for this task are unpublished manuscripts of Jeremy Bentham – an English philosopher and reformer of the 18th century.

The goal is to detect queries in a "segment" which is defined as six consecutive lines. A segment contains a query if all keywords appear in the correct order. Two consecutive segments overlap in 5 lines. This means, a match of a query possibly appears in up to 6 segments depending on the difference between the indices of the last and first index of matching lines. A detailed description of this task and their results can be found in [10].

2 System Description

2.1 Basic Scheme

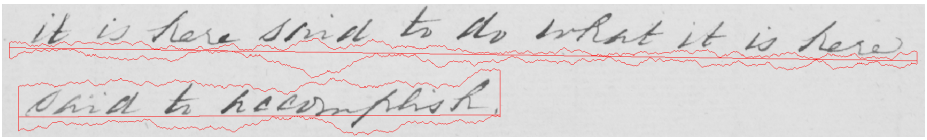
For the general approach, we may briefly refer to previous CITlab system descriptions [4,3,8,6] because the overall scheme has essentially not been changed.

2.2 From Baseline to Polygon

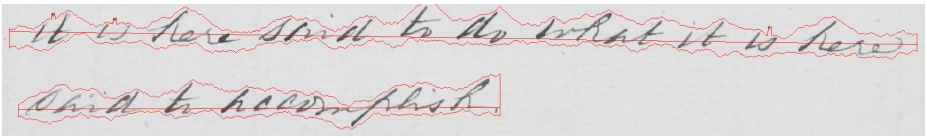
This section briefly describes an algorithm to calculate polygons surrounding the text lines given its baselines. Given that for the test set (see Table 1) only baselines are provided, such an algorithm is mandatory since the recognition system requires a cropped text line as input.

The baseline to polygon algorithm basically follows [1]. The idea is that given a medial seam (which is roughly spoken a polyline following the main body of the text line) separating seams are calculated by optimizing an appropriate cost function using dynamic programming (see [1]). Here, the cost function penalizes a separating seam for crossing regions with high Sobel values and for its distance to the medial seam. The Sobel values are calculated by convolving the input image with the Sobel operator.
Using the given baseline directly as medial seam leads to insufficient results, e.g. in Fig. 1a the provided baseline even does not touch the text – as a consequence the calculated separating seams even do not contain the text at all. Hence, an optimal shift is calculated for each baseline such that the sum of Sobel values on the shifted baseline is maximal. Fig. 1b depicts the effect of this approach.

There are surrounding polygons given for the training and development set (Table 1). Since they look quite different to the polygons calculated using the described algorithm, we did not train on the given surrounding polygons. These



(a) Medial seam (baseline) with resulting separating seams



(b) Medial seam (translated baseline) with resulting separating seams

Fig. 1: Comparison of the baseline to polygon approach for different medial seams

polygons were only used to calculate baselines, which were used as input for the baseline to polygon algorithm. This approach ensures homogeneity of training, development and test data.

2.3 Preprocessing

Given the line polygon, we apply certain standard preprocessing routines, i.e.

- image normalization: contrast enhancement (no binarization), size;
- writing normalization: line bends, line skew, script slant.

Then, images are further unified by CITlab's proprietary writing normalization: The writing's main body is placed in the center part of an image of fixed 96px height. While the length-height ratio of the main body stays untouched, the ascenders and descenders are squashed to focus the network's attention on the more informative main body. These images are the input for the feature generation.

2.4 Feature Generation

The feature generation works like a convolutional filter with complex coefficients. The input image is converted in a set of feature maps that contain local frequency information. Let $\boldsymbol{X} \in [0,1]^{u \times v}$ be an image of width u and height v. Let $\omega \in \mathbb{R}^+$ be a frequency, $\theta \in [0, 2\pi)$ be an angle and $r_0 \in \mathbb{R}^+$ be a window radius. The complex convolutional kernel is defined by

$$f(r) = \begin{cases} \frac{1}{2}\left(1 + \cos\left(\frac{\pi r}{2}\right)\right) & \text{if } r < 2 \\ 0 & \text{else} \end{cases}$$

$$g(x,y) := g(x,y)_{\omega,\theta,r_0} = f\left(\frac{\sqrt{x^2+y^2}}{r_0}\right) \exp\Big(i\omega\big(x\cos(\theta) + y\sin(\theta)\big)\Big)$$

around a centre $(0,0)$. The frequency feature $b(x,y)$ at point (x,y) is then calculated by

$$b\,(x,y) = \left| \sum_{(i,j)\in\mathbb{Z}^2} X_{i,j} g(i-x, j-y) \right|.$$

The advantage of this frequency features is the robustness against shifts and noise on the input image. In this applications we use parameters $\omega \in \left\{\frac{\pi}{4}, \frac{\pi}{2}\right\}$, $\theta \in \left\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\right\}$ and $r_0 \approx 4$. These 8 feature images (2 frequencies and 4 angles) are the input for the MDRNN.

2.5 Recurrent Neural Network

The resulting features were fed into so called Sequential Processing Recurrent Neural Network (SPRNN). The SPRNN has 3 hidden layers with 355210 trainable weights. The first and third layer are multidimensional and multidirectional recurrent layers. To reduce the computational complexity and increase the ability to generalize, these recurrent layers are connected through a feedforward layer. Instead of using LSTMs in the MDRNN the LeakyLP cells are used [5].

The SPRNN's output then consists of a certain number of vectors. This number is related to the line length because every vector contains information about a particular image position. More precisely, the entries are understood as the estimate of the probabilities of every alphabet character at the position under consideration. Hence, the vector lengths all equal the alphabet size, and putting all vectors together leads to the so-called confidence matrix "*ConfMat*". This is the intrinsic recognition result which will subsequently be used for the decoding.

Note further that, for *Handwritten Scanned Document Retrieval Task 2016*, we worked with an alphabet containing

- all digits, lowercase and uppercase letters of the ISO basic Latin alphabet
- special characters `/&£§+-\_.,:;!?'"=[]()` and ␣, whereby different types of quotation marks and hyphens were mapped to one of the respective symbols.

Finally, the above alphabet is augmented by an artificial, not-a-character symbol, CITlab's NaC[3]. In particular, it may be used to detect character boundaries because, generally speaking, our SPRNNs emit high NaC confidences in uncertain situations.

2.6 Training Data

The composition of the data set provided by the competition organizers is summarized in Table 1.

[3] In the literature it is also called *blank*, *no-symbol*, *no-label*.
[4] There exist polygons but they are not accurate enough for using.

Table 1: Composition of the data sets provided by the organizers

	#pages	#lines	#characters	polygons	baselines
training set	363	9645	65488	yes	no
development set	433	10589	80758	yes	no
test set	200	6355	—	no[4]	yes

2.7 Network Training

The network is trained using an extension of Nesterov's Accelerated Gradient Descent with learning rate $5e-4$ and momentum 0.9. For each training epoch, we choose a random subset of 10,000 lines from the training set. The first 19 epochs were trained using the original images with a fixed learning rate. For 3 epochs we added noise to the preprocess parameters and network activations. For 19 additional epochs we set the learning rate to $5e-5$ and added degradation (pixel noise, blur, cross outs,...) to the images.

2.8 Decoding Schemes

Word Matchings The neural networks output, the ConfMat, consists of confidences $y_{t,l}$ for any label l and position t where the labels are the characters and the NaC. The confidences are positive and sum to 1 for fixed position t. Thus, they can be interpreted as conditional probability for label l at position t given input image $\boldsymbol{X}$. The number of positions is typically greater than the length of the decoded words such that different label sequences decode the same word. Following the original notation of [2], let $\mathcal{F}$ be the function mapping a sequence of labels to a sequence of characters by merging consecutive identical labels and deleting NaCs. Instead of calculating the probability[5] of a string $\boldsymbol{s}$, we calculate the maximum probability $\mathrm{P}(\boldsymbol{\pi}^*(\boldsymbol{s})|\boldsymbol{X})$[6] of any path collapsing to $\boldsymbol{s}$ (this corresponds to the Viterbi approximation for HMMs):

$$\mathrm{P}(\boldsymbol{\pi}^*(\boldsymbol{s})|\boldsymbol{X}) = \max_{\boldsymbol{\pi}\in\mathcal{F}^{-1}(\boldsymbol{s})} \prod_{t=1}^{T} y_{t,\pi_t}.$$

In the following, $\boldsymbol{z}$ denotes a single keyword. Since the ConfMat could contain more than one word, the path probability of $\boldsymbol{z}$ must be calculated on a specific submatrix:

$$\mathrm{P}_{s:e}(\boldsymbol{\pi}^*(\boldsymbol{z})|\boldsymbol{X}) = \max_{\boldsymbol{\pi}\in\mathcal{F}^{-1}(\boldsymbol{z})} \prod_{t=s}^{e} y_{t,\pi_t},$$

[5] To get the (CTC) probability one replaces the maximum operator by the sum: $\mathrm{P}(\boldsymbol{s}|\boldsymbol{X}) = \sum_{\boldsymbol{\pi}\in\mathcal{F}^{-1}(\boldsymbol{z})} \prod_{t=1}^{T} y_{t,\pi_t}$.

[6] In the following, we call $\mathrm{P}(\boldsymbol{\pi}^*(\boldsymbol{s})|\boldsymbol{X})$ the path probability of $\boldsymbol{s}$.

where s and e are the start and end position of the submatrix within the ConfMat. Since $y_{t,l} < 1$ for any t, l, the path probability typically decreases if $e - s$ increases. Thus, we accept a keyword $\boldsymbol{z}$ ranging from position s to e of a certain ConfMat if path probability relative to the number of positions $\mathrm{P}_{s:e}(\boldsymbol{\pi}^*(\boldsymbol{z})|\boldsymbol{X})/(e - s + 1)$ is higher than a certain threshold.

To ensure that the match is not only part of a larger word, the word has to be separated by spaces, parentheses, hyphens etc. if it does not appear at the beginning or the end of a line. This pattern can be described by a regular expression: `(.*[␣(-])?` *keyword* `([␣)-].*)?` . This search is accomplished using the decoder described in [7]. The result is a path $\boldsymbol{\pi}$ of length T aligned to the ConfMat positions. The indices s and e are determined by the subpath $\boldsymbol{\pi}_{s:e}$ corresponding to the keyword (i.e. $\mathcal{F}(\boldsymbol{\pi}_{s:e}) = \boldsymbol{z}$). Because of $\mathcal{F}$, s and e are still not well defined since e.g. NaCs will be deleted. To avoid ambiguity, we use the most greedy subpath. Again, if the probability of the separators does not exceed a certain threshold, the match is assumed to be part of a larger word and it is rejected.

Multiple word queries are treated by searching the keywords individually.

Incorporating Hyphens The strategy above obviously does not work for hyphens since the keyword is spread over two ConfMats. To treat also hyphens, we search for pairs of consecutive ConfMats where the first ConfMat likely ends on a hyphen symbol and extract the submatrices containing the last word of the first matrix and the first word of second matrix. Both submatrices will be combined and the new ConfMat is added to the list of all ConfMats such that we can use the above strategy to search for hyphenated words in those combined matrices.

To search for ConfMats containing hyphenations, we simply search for hyphens at the end again using the RegEx-Decoder from [7]. The used regular expression is `(.*␣)?[A-Za-z]+␣?-␣?` which extracts the first part of the hyphenation and the hyphen at once.

Score Many of the word matches are false positives and will influence the result in a negative way. The four evaluation measures penalize false matches with a high score more than false positives with a relatively low score. Therefore, a "good" score is crucial for a good evaluation.

The path probability $\mathrm{P}_{s:e}(\boldsymbol{\pi}^*(\boldsymbol{z})|\boldsymbol{X})$ is an obvious score. This probability reflects the maximum confidence that the input contains the word $\boldsymbol{z}$.

To our experience, the ability to learn dependencies between characters depends highly on the training data. If many variations appear in the trained character sequences, the network's output will depend only weakly on character transitions. Thus, the network will not be able to predict word priors. It rather predicts the character sequence as accurate as possible. To incorporate also word priors, we borrow some basic ideas from domain adaptation. The source domain $\mathcal{S}$ is the domain learned by the neural network which includes those weak dependencies on character sequences. The target domain $\mathcal{T}$ reflects the correct word

statistics. For sake of simplicity, assume for the moment that all input images $\boldsymbol{X}$ reflect single word snippets. The only assumption of the below derivation is that $\mathrm{P}_{\mathcal{S}}(\boldsymbol{X}|\boldsymbol{z}) = \mathrm{P}_{\mathcal{T}}(\boldsymbol{X}|\boldsymbol{z})$ which basically means that the fonts are the same. The beauty of this approach is that the word distribution of training and test data may differ. By Bayes law, we know

$$\begin{aligned}\mathrm{P}_{\mathcal{T}}(\boldsymbol{z}|\boldsymbol{X}) &= \frac{\mathrm{P}_{\mathcal{S}}(\boldsymbol{z}|\boldsymbol{X})\,\mathrm{p}_{\mathcal{S}}(\boldsymbol{X})}{\mathrm{P}_{\mathcal{S}}(\boldsymbol{z})}\,\frac{\mathrm{P}_{\mathcal{T}}(\boldsymbol{z})}{\mathrm{p}_{\mathcal{T}}(\boldsymbol{X})} \\ &= \frac{1}{N}\quad\frac{\mathrm{P}_{\mathcal{T}}(\boldsymbol{z})}{\mathrm{P}_{\mathcal{S}}(\boldsymbol{z})}\,\mathrm{P}_{\mathcal{S}}(\boldsymbol{z}|\boldsymbol{X}) \qquad (1)\end{aligned}$$

where

$$N = \sum_{\boldsymbol{z}'} \frac{\mathrm{P}_{\mathcal{T}}(\boldsymbol{z}')}{\mathrm{P}_{\mathcal{S}}(\boldsymbol{z}')}\,\mathrm{P}_{\mathcal{S}}(\boldsymbol{z}'|\boldsymbol{X}).$$

In principle, $\mathrm{P}_{\mathcal{T}}(\boldsymbol{z})$ could be any language model. In this task, we simply use a word unigram. The source prior $\mathrm{P}_{\mathcal{S}}(\boldsymbol{z})$ is a character transition probability learned by the neural network. We estimate $\mathrm{P}_{\mathcal{S}}(\boldsymbol{z})$ in three different ways:

- $\mathrm{P}_{\mathcal{S}}(\boldsymbol{z}) = \mathrm{P}_{\mathcal{T}}(\boldsymbol{z})$ thus target posterior is equal to the source posterior times the normalization. We refer to this prior scheme as `abs`.
- $\mathrm{P}_{\mathcal{S}}(\boldsymbol{z}) \propto 1$, thus only the prior is used.[7] We refer to this prior scheme as `prior`.
- $\mathrm{P}_{\mathcal{S}}(\boldsymbol{z}) \propto \left(\prod_{i=1}^{|\boldsymbol{z}|} \mathrm{P}(z_i)\right)^c$ where z_i is the character prior and $0 < c \leq 1$.[8] We refer to this prior scheme as `da`.

In last both schemes, `prior` and `da`, the prior probabilities are estimated up to a constant factor $1/N'$ which is basically the reciprocal of the sum of the estimates of $\mathrm{P}_{\mathcal{S}}(\boldsymbol{z})$ over the finite set of all words $\boldsymbol{z}$. We integrate N' into N for both schemes.

If the written characters of one word do not influence those of another, it is reasonable to reestimate the word probability $\mathrm{P}_{s:e}(\boldsymbol{z}|\boldsymbol{X})$ within the positions from s to e of the ConfMat according to eq. (1). Then, the normalization for `prior` and `da` consists of the finite set of all word and part of word sequences fitting in this submatrix. In the same way, we reestimate the path probability $\mathrm{P}_{s:e}(\boldsymbol{\pi}^*(\boldsymbol{z})|\boldsymbol{X})$.

To sort a set of words according to their probability on a specific submatrix of a given ConfMat, there is no need to calculate the normalization $1/N$. The normalization is only crucial for comparing probabilities of different submatrices. To analyze the impact of the normalization, we submitted results with (`normed`) and without (`unnormed`) normalization. Typically, the vocabulary only represents a

[7] It would be statistically more reasonable to model the character/label probability by some constant such that $\mathrm{P}_{\mathcal{S}}(\boldsymbol{z}) = c^{|\boldsymbol{z}|}$ or $\mathrm{P}_{\mathcal{S}}(\boldsymbol{z}) = c^T$

[8] In the submitted system, this character priors are estimated on the training set and $c = 0.5$.

small part of all words of the considered language. Thus, it is impossible to sum all these feasible words. Words not contained in the vocabulary are called *out-of-vocabulary words (OOV words)*. Usually, the normalization constant N has to be approximated.[9]

Typically, our posterior probabilities are calculated using the path probability. To investigate the impact of using the path probability as an approximation of CTC we submitted comparable systems for both source posterior probabilities. Using the CTC scheme, we only use the path probability to calculate s and e. All the above equations we substitute the path probability $\mathrm{P}(\boldsymbol{\pi}^*(\boldsymbol{z})|\boldsymbol{X})$ by the CTC probability $\mathrm{P}(\boldsymbol{z}|\boldsymbol{X})$. We refer to these posterior scheme as `path` or `ctc`, respectively.

Combining Keywords for Multiple Word Queries For single word queries, we are already done. The matches can be saved for six consecutive segments. For multiple word queries, all keywords have to be detected in a certain segment and the order how they appear has to be the same as in the query.

There is one score for each query. So the scores of the matches of multiple word queries has to be merged to one score. We tested the minimum, the arithmetic and geometric mean of the scores of each match. In our tests, arithmetic and geometric yield almost the same error rate while the minimum of all scores yielded significantly higher error rates. We worked with the geometric mean.

3 Results

This section reports the results of the *Handwritten Scanned Document Retrieval Task 2016*. The experiments are designed to combine the described decoding components of Sect. 2.8:

- posterior probability (`path` and `ctc`)
- prior probability (`abs`, `prior` and `da`)
- normalization (`normed` and `unnormed`)

We were restricted to submit only 10 systems. Since we usually use the probability of the most likely path instead of CTC probabilities, we skipped two decoding schemes which use the CTC posterior and are not normalized at the same time.

We are especially interested in the scores at segment level. The scores at box level highly depend on the precise detection of a bounding box of a keyword. Since this is out of our scope we concentrate our investigations on the segment score. To get an impression of the differences, we refer to Table 4 in Appendix A showing the same results as Table 2 only on box level. Appendix A also contains the results of subsets of the keyword queries such as hyphenated or OOV words for both the development and the test set.

[9] In our experiments, we sum up the 10 most likely vocabulary matches plus an additional OOV term if the best string (also raw output) is not contained in those matches.

Additionally, we submitted search results obtained by a neural network trained with additional external data. Unfortunately, we accidentally also used data from the HTRtS15 training set with overlaps with the development data set. The resulting network yields improved results on the development set. Other additional training data seems to fit poorly to the test data and thus confused the network. Thus, the recognition rates on the test set decrease.

Table 2: Results on the development set on segment level for the MDRNN trained only on the training set

	source prior	abs		prior		da	
	source posterior	path	ctc	path	ctc	path	ctc
gAP	normed	94.81	94.89	95.36	**95.42**	94.99	95.04
	unnormed	94.77		91.73	91.87	92.58	
mAP	normed	89.71	**89.90**	89.58	89.76	89.63	89.82
	unnormed	89.42		88.59	88.89	89.13	
gNDCG	normed	96.72	96.78	96.78	**96.83**	96.73	96.77
	unnormed	96.69		96.34	96.41	96.46	
mNDCG	normed	90.77	**90.97**	90.66	90.85	90.70	90.89
	unnormed	90.61		89.96	90.25	90.36	

Table 3: Results on the test set on segment level for the MDRNN trained only on the training set

	source prior	abs		prior		da	
	source posterior	path	ctc	path	ctc	path	ctc
gAP	normed	33.89	33.98	43.20	43.72	46.74	**47.13**
	unnormed	42.21		36.17	36.30	39.65	
mAP	normed	38.62	39.46	39.10	**40.03**	39.19	39.89
	unnormed	38.18		36.50	37.13	37.91	
gNDCG	normed	59.39	60.03	61.40	62.14	61.98	**62.70**
	unnormed	61.14		59.70	60.37	60.62	
mNDCG	normed	40.57	41.40	40.96	**41.86**	41.05	41.73
	unnormed	40.22		38.85	39.51	40.04	

Normalization The Tables 2 and 3 show the impact of the normalization on the four different measures. Normalizing the probabilities typically improves the recognition score except for one configuration: If the source prior is equal to the target prior (i.e. `abs`). Then the normalization can be counterproductive if the data is different from the trained data. The network is trained to optimize the unnormalized CTC probability. So it is not surprising that the system works well if we use only the source posterior probability as score. If the network output gets blurry (because of e.g. untrained writing styles), alternative results become more likely compared to the proposed result. The normalization value will grow for keywords which have a small edit distance to many other vocabulary words. Thus, if the network is not able to make clear decisions, the normalization value will much more depend on the keyword position in the vocabulary. Even for the development set – where the data seems to fit the training data well – the gain from the normalization is not significant. Thus, the normalization can be omitted using the `abs` decoding scheme.

If source and target prior differ, the posterior scale changes depending on the word. Thus for different keywords, the scores are not comparable anymore. The normalization maps the scores into the same range. Therefore, normalization increases the recognition rate by around 7 `gAP` points for `prior` and `da` schemes at test set (Table 3).

All other tables show the same behavior. Therefore, we omit the row with the unnormalized decoding schemes in Table 4 - 8.

Path vs. CTC Probability The network is trained to optimize the CTC posterior likelihood. Thus, it is not surprising that the CTC probability is typically slightly better (less than 0.6 gAP points on the test set 3) than the path probability except for few experimental setups: The box level gAP on the development set (Table 4) and the gAP on the development set restricted to broken words (Table 5). A query match may contain additional false keyword matches although the query match on segment level is correct. These additional false matches are penalized by the gAP on box level. Since the CTC probability is typically higher than the path probability and the rejection thresholds stays constant, there are more additional false keyword matches within a query match. So the error increases for the CTC probability.

Finally, the gap between path and CTC posterior probability is small for all experiments. The path probability also preserves the relation between the source prior and normalization decoding schemes. Thus, the path probability is a good approximation.

Priors The evaluation is even less clear than the one above. The results do not only depend on different experimental setups (i.e. different tables) but also they highly depend on the measure. Considering the development set (Table 2), the `prior` scheme works slightly better (less than 0.6 gAP points) than `abs` and `da`. The mAP measure puts more weight on the infrequent words. Thus, the `abs`

decoding scheme works better than the `prior` decoding scheme which naturally favors frequent words.

Compared to the development set, the results on the test set gain more from including prior knowledge since the posterior probabilities are less reliable. Especially, if the gAP value is measured, the `da` scheme yields better results (greater than 3 gAP points) than the others. Measuring the mAP, the `prior` decoding scheme is slightly better (less than 0.2 mAP points compared to `da`).

In Table 8, the `da` scheme yields the lowest error rates independent of the measure. This may indicate that the OOV prior could be improved. The current estimation of an OOV prior is constant for all OOV words. For future research, we plan to investigate a more sophisticated OOV prior such as a character n-gram of small order.

4 Conclusion

In this paper we present the fundamental concepts of our systems submitted to the *Handwritten Scanned Document Retrieval Task 2016* attached to the CLEF in 2016. We submitted 10 systems comparing different rescoring strategies.

Unfortunately, there is no winning rescoring strategy. Normalization almost always improves the score and typically the score is slightly better when using the CTC posterior probability compared to the probability of the most likely path. Nevertheless, the probability of the most likely path is a good approximation to the CTC probability. Using domain adaptation to switch from the learned source domain to the target domain, we scale the posterior probability by the target prior - source prior ratio. Fixing the target prior (as unigram probability), we vary the source prior. For all three tested source priors there are setups where the way of calculating this specific prior is preferable. This might indicate that the estimated prior does not fit the prior learned by the neural network.

Acknowledgement

First of all, the CITlab team really wishes to express its great gratitude to our long-term technology & development partner *PLANET intelligent systems GmbH* (Raben Steinfeld & Rostock, Germany) for the extremely valuable, ongoing support in every aspect of this work. Participating in *Handwritten Scanned Document Retrieval Task 2016* would not have been possible without that! In particular, we continued using PLANET's software world which was developed and essentially improved in various common CITlab–PLANET projects over previous years.

From PLANET's side, our activities were essentially supported by Jesper Kleinjohann and Richard Schwark, whom we especially thank for ongoing very helpful discussions and his continuous development support.

Being part of our current research & development collaboration project, the development work was funded by grant no. KF2622304SS3 (Kooperationsprojekt) in *Zentrales Innovationsprogramm Mittelstand (ZIM)* by Bundesrepublik

Deutschland (BMWi). The contest application has been adapted while working in the EU Horizon 2020 project READ – Recognition and Enrichment of Archival Documents (official no. 674943).

Finally, we are indebted to the competition organizers from the PRHLT group at UPV – in particular Mauricio Villegas – for setting up this evaluation and the contest as well as the entire *tranScriptorium* project for providing all the data.

References

1. Arvanitopoulos, N., Susstrunk, S.: Seam carving for text line extraction on color and grayscale historical manuscripts. In: Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on. pp. 726–731. IEEE (2014)
2. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on Machine learning. pp. 369–376. ACM (2006)
3. Leifert, G., Grüning, T., Strauß, T., Labahn, R.: CITlab ARGUS for historical data tables: Description of CITlab's system for the ANWRESH-2014 Word Recognition task. Technical Report 2014/1, Universität Rostock (Apr 2014), `http://arXiv.org/abs/1412.6012`
4. Leifert, G., Labahn, R., Strauß, T.: CITlab ARGUS for arabic handwriting: Description of CITlab's system for the OpenHaRT 2013 Document Image Recognition task. In: Proceedings of the NIST 2013 OpenHaRT Workshop [Online] (Aug 2013), `http://arXiv.org/abs/1412.6061`, available: `http://www.nist.gov/itl/iad/mig/hart2013_wrkshp.cfm`
5. Leifert, G., Strauß, T., Grüning, T., Labahn, R.: Cells in multidimensional recurrent neural networks. arXiv preprint arXiv:1412.2620 (2014), submitted to Journal of Machine Learning Research
6. Leifert, G., Strauß, T., Grüning, T., Labahn, R.: CITlab ARGUS for historical handwritten documents - Description of CITlab's System for the HTRtS 2015 Task : Handwritten Text Recognition on the tranScriptorium Dataset . Technical report, Universität Rostock (Apr 2015)
7. Strauß, T., Leifert, G., Grüning, T., Labahn, R.: Regular expressions for decoding of neural network outputs. Neural Networks 79, 1 – 11 (2016), `http://www.sciencedirect.com/science/article/pii/S0893608016000447`
8. Strauß, T., Grüning, T., Leifert, G., Labahn, R.: CITlab ARGUS for historical handwritten documents: Description of CITlab's system for the HTRtS 2014 Handwritten Text Recognition task. Technical Report 2014/2, Universität Rostock (Apr 2014), `http://arXiv.org/abs/1412.3949`
9. Villegas, M., Müller, H., García Seco de Herrera, A., Schaer, R., Bromuri, S., Gilbert, A., Piras, L., Wang, J., Yan, F., Ramisa, A., Dellandrea, E., Gaizauskas, R., Mikolajczyk, K., Puigcerver, J., Toselli, A.H., Sánchez, J.A., Vidal, E.: General Overview of ImageCLEF at the CLEF 2016 Labs. Lecture Notes in Computer Science, Springer International Publishing (2016)
10. Villegas, M., Puigcerver, J., Toselli, A.H., Sánchez, J.A., Vidal, E.: Overview of the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task. In: CLEF2016 Working Notes. CEUR Workshop Proceedings, CEUR-WS.org <http://ceur-ws.org>, Évora, Portugal (September 5-8 2016)

A Additional Tables

Table 4: Results on the development set on box level for the MDRNN trained only on the training set

	source prior	abs		prior		da	
	source posterior	path	ctc	path	ctc	path	ctc
gAP	normed	72.73	72.31	**73.40**	72.92	70.87	70.54
mAP	normed	72.88	**72.94**	72.92	72.87	72.42	72.25
gNDCG	normed	75.42	75.20	**75.63**	75.38	74.73	74.43
mNDCG	normed	75.93	76.00	75.99	**76.02**	75.72	75.65

Table 5: Results on the development set only for *broken words* on segment level for the MDRNN trained only on the training set

	source prior	abs		prior		da	
	source posterior	path	ctc	path	ctc	path	ctc
gAP	normed	**60.86**	60.47	60.60	60.09	59.35	59.35
mAP	normed	48.96	**49.61**	48.28	48.64	47.85	48.44
gNDCG	normed	75.10	75.03	**76.55**	76.43	75.98	75.98
mNDCG	normed	50.07	**50.75**	49.36	49.76	49.02	49.52

Table 6: Results on the development set only for *OOV words* on segment level for the MDRNN trained only on the training set

	source prior	abs		prior		da	
	source posterior	path	ctc	path	ctc	path	ctc
gAP	normed	88.57	88.98	89.31	**89.66**	89.01	89.28
mAP	normed	88.43	88.71	87.83	88.36	88.31	**88.86**
gNDCG	normed	92.08	92.42	92.19	**92.52**	92.16	92.47
mNDCG	normed	89.18	89.48	88.65	89.20	88.98	**89.53**

Table 7: Results on the test set only for *broken words* on segment level for the MDRNN trained only on the training set

	source prior	abs		prior		da	
	source posterior	path	ctc	path	ctc	path	ctc
gAP	normed	16.69	16.98	19.64	20.21	23.56	**24.33**
mAP	normed	22.65	23.63	22.9	**23.83**	22.81	23.73
gNDCG	normed	33.46	34.29	34.56	35.52	35.81	**36.84**
mNDCG	normed	23.35	24.4	23.53	**24.51**	23.47	24.45

Table 8: Results on the test set only for *OOV words* on segment level for the MDRNN trained only on the training set

	source prior	abs		prior		da	
	source posterior	path	ctc	path	ctc	path	ctc
gAP	normed	28.55	28.78	37.97	38.61	41.92	**42.59**
mAP	normed	38.21	38.89	38.7	39.31	38.9	**39.53**
gNDCG	normed	52.69	53.07	55.27	55.79	56.2	**56.67**
mNDCG	normed	39.89	40.52	40.38	40.95	40.54	**41.12**