https://doi.org/10.18453/rosdok_id00005818

CITlab ARGUS for historical data tables

Description of CITlab's System for the ANWRESH-2014 Word Recognition Task

Gundram Leifert Tobias Grüning Tobias Strauß
Roger Labahn*

April 14, 2014
last revision: May 02, 2014

We describe CITlab's recognition system for the ANWRESH-2014 competition attached to the 14. International Conference on Frontiers in Handwriting Recognition, ICFHR 2014. The task comprises word recognition from segmented historical documents. The core components of our system are based on multi-dimensional recurrent neural networks (MDRNN) and 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, LSTM, CTC, handwriting recognition, neural network

1 Introduction

The International Conference on Frontiers in Handwriting Recognition ICFHR 2014 (`www.icfhr2014.org`) hosts a variety of competitions in that area. Among others, ANWRESH-2014 attracted our attention because we expected CITlab's handwriting recognition software to be able to successfully deal with the respective task. With a system very similar to the one presented here, the CITlab team also took part in 2014 ICFHR's HTRtS competition on historical handwritten document recognition, see [SGLL14] for the according system description.

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

ANWRESH-2014 comprises a task of word recognition for segmented historical documents, see `http://collections.ancestry.com/ANWRESH-2014` for further details. These data consist of page images taken from the 1930 US Census General Population Schedules. Each page mainly has a table with 50 rows containing various fields with personal data, out of which five are to be read here. Table 1 explains the essential data fields and shows the labels used throughout the paper to refer to the respective network type.

network type ...	reads data field ...
$\mathcal{N}$	`NAME`, which itself is composed of family name and given name
$\mathcal{R}$	`RELATION`, e.g. *Head, Lodger, Wife, Daughter, ...*
$\mathcal{A}$	`Age` (at last birthday), i.e. integers and possibly fractions counting month
$\mathcal{M}$	`Marital condition`
$\mathcal{B}$	`PLACE OF BIRTH`

Table 1: Network types: Labels and Data

Moreover, we apply committees of networks for mixture-of-experts evaluations for certain data. This variety of networks is referred to by indexing.

Our neural networks have basically been used previously in the international handwriting competition OpenHaRT 2013 attached to the ICDAR 2013 conference, see [LLS13]. Moreover, with a system very similar to the one presented here, the CITlab team also took part in ICFHR's HTRtS competition on a handwritten text recognition task, see [SGLL14] for the according system description.

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, a small/medium enterprise focusing on computational intelligence technology and applications. The work presented here was part of a common text recognition project 2010 – 2014 and is extensively based upon PLANET's ARGUS software modules.

2 Preprocessing

For ANWRESH-2014, additional field segment information was provided but the given polygons typically do not match the table field borders exactly. Therefore, we firstly tried to find the lines of the table more precisely: Starting from a certain enlargement of the given polygons as a rough region approximation, we select pixel density peaks in projection profiles to both page borders. Cutting-out along those table lines yields the single writings to be recognized.

Note that the neural networks only work properly as long as the input image properties are close to or even meet those of the images shown during training. In order to meet such requirements, and depending on the network type (see Table 1), we first normalize the

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

image heights to 96 or 128 pixels for networks of type $\mathcal{B}$ or all other types, respectively, and then the contrast, i.e. the gray values for black and white pixels. There is no further image preprocessing.

3 System specifications

The following part describes the general architecture of the entire workflow, and we explain details for the core neural network and the decoding procedure.

3.1 Input

The networks use entire writings as being prepared in the above preprocessing. In particular, there is no further segmentation. As developed in [GS08], every writing image is processed by reading its pixel data in four column-first "directions" that arise from combining top-down and bottom-up column traversals with left-to-right and right-to-left row traversals.

3.2 Neural Network

The neural networks used for CITlab systems are essentially based on preceding work presented in [GS08]. The basics are the same for all network types applied for ANWRESH-2014. However, as mentioned before, we use networks of various types and even several networks of the same type. All those may be different in

- the total (and per-layer) numbers of cells (neurons) they are built of, see 3.2.1;
- their output layer and data as these correspond to the respective task, see 3.2.2;
- the learning data and certain parameters used for train them, see 4.1.

3.2.1 Architecture

The architecture of the neural nets particularly follows [GS08], but we introduced essential modifications: Instead of traditional MDLSTM cells, we use two layers with MDLeaky cells ([LLS13]) which were shown to be more stable and thus yield better performance. They embrace one layer of classical tanh cells in between.

Furthermore, this network core is preceeded by a first layer accomplishing GABOR filtering with fixed parameters, thus essentially reducing the number of trainable weights and, consequently, training time. Finally, the concluding layer right before the output layer simply adds up preceding activations such that for every pixel column, just one value arrives here. Thus, this results in an overall size reduction (subsampling) from the standard image height (see Section 2) downto 1, but in fact, it is done step-wisely over the layers by factors 4 in y-direction or 3 in x-direction, respectively.

While the above details are common to all networks used here, for different networks, we tested various numbers of units. Those finally chosen are again based on [GS08]) and, particular, on [LLS13] and further own experimental experience and testing. Table 2 shows some network architecture parameters in detail.

	number of		
network	cells	trainable weights	output neurons
$\mathcal{N}_1$	1900	958387	56
$\mathcal{N}_2$	1166	363477	56
$\mathcal{N}_3$	1166	363477	56
$\mathcal{R}_1$	1900	1006387	56
$\mathcal{R}_2$	1900	1006387	56
$\mathcal{A}$	1869	933556	25
$\mathcal{M}$	1851	967138	7
$\mathcal{B}_1$	1899	1005586	55
$\mathcal{B}_2$	1532	655747	55

Table 2: Architecture parameters

3.2.2 Output & Type

According to its specific tasks, every network type has a particular alphabet for the data to be read. In the following table, we summarize the details. The shown number of output neurons also counts the standard, artificial "garbage" symbol that every alphabet additionally contains.

Note that one particularity in the `NAME` data field deserved special treatment: While in the images, family names were abbreviated by a solid line if they were just repeated for consecutive family members, the reference data contained the fully spelled family name. We reverted this by replacing the reference by an underscore _ as a special symbol.

network type	table data	♯output neurons	alphabet
$\mathcal{N}$	`NAME`	56	␣'_ A...Z a...z
$\mathcal{R}$	`RELATION`	56	␣-_ A...Z a...z
$\mathcal{A}$	`Age`	25	␣ 0...9 $\frac{0}{12}$...$\frac{12}{12}$
$\mathcal{M}$	`Marital condition`	7	letters S,M,W,D,C,V
$\mathcal{B}$	`PLACE OF BIRTH`	55	␣- A...Z a...z

Table 3: Network output

3.2.3 Training Algorithm

As usual, the activation of an output neuron at a particular time is trained to estimate the probability of the occurrence of its corresponding character at a specific position in

the original writing. The network is trained by Backpropagation-Through-Time (BPTT) using the Connectionist Temporal Classification (CTC) algorithm described in [GFGS06] to calculate the gradient.

3.3 Decoding

After applying the standard softmax normalization, at each time step, the neural net provides a vector of probability estimates, each component counting for one entry of the alphabet. Collecting those vectors over time, finally yields the network output matrix, $N(x)$, for a given input writing, x. Decoding algorithms then typically search for a most likely word w^* for the network output under consideration:

$$w^* = \operatorname*{argmax}_w p(w|N(x)) \,.$$

Since the garbage class typically has a high probability compared to other classes, the garbage "letter" is often cheap to insert which might mislead the decoding between shorter or longer word guesses. In order to correct for this, we prefer short words by "normalizing" the above word probability by some function depending on the word length. Furthermore, in order to incorporate prior probabilities, we multiply with relative frequencies, $p(w)$, computed from the ANWRESH dictionaries. All in all, we take

$$w^* = \operatorname*{argmin}_w \frac{-\ln p(w|N(x))}{|w|^\alpha} - \beta \ln p(w) \tag{1}$$

where α and β are some constants.

For some data fields, we apply several networks simultaneously, and choose the answer

$$w^* = \operatorname*{argmin}_{w, i \in I} \frac{-\ln p(w|N_i(x))}{|w|^\alpha} - \beta \ln p(w) \tag{2}$$

where I is the index set of all networks. The value used in (2), $\min_{i \in I} \frac{-\ln p(w|N_i(x))}{|w|^\alpha} - \beta \ln p(w)$, (resp. (1) if not applying committee's vote) will be called the *cost* of a certain dictionary entry w.

In a postprocessing stage, we delete inconsistent field results: If, e.g., the relation is *wife* and the given name is doubtlessly male, we add additional penalty costs onto the costs of either the relation or the name result depending on which field is more unlikely. Let's say the name is more unlikely and gets additional penalty costs, then, if there is a sufficiently probable second alternative being female, this second name will be preferred over the first name.

4 Application

We conclude with describing the working setup specifically used for this competition in training and testing our system.

4.1 Training

We divide the ANWRESH-2014 data set into a training and a validation set: For each data field type, these sets contain about 120k and 12k writings, respectively. The exact value depends on the image ground truth: Some of them are missing or for other reasons unusable such that some field images have to be omitted. In every epoch of the batch training, 20k of these writings were used.

Networks were trained with a fixed momentum of 0.9, and two different learning rates: we started the main training portion with 0.002 and then concluded by a rather short post-training with a learning rate of 0.001. Table 4 shows the various numbers of epochs that the used networks have been trained for. The intended maximum was 100 epochs but mainly due to time limitations, we eventually had to stop earlier.

	number of training epochs	
network	main learning	post-training
$\mathcal{N}_1$	31	8
$\mathcal{N}_2$	54	8
$\mathcal{N}_3$	60	9
$\mathcal{R}_1$	100	7
$\mathcal{R}_2$	100	10
$\mathcal{A}$	100	6
$\mathcal{M}$	100	9
$\mathcal{B}_1$	67	8
$\mathcal{B}_2$	99	8

Table 4: Training parameters

4.2 Testing

For the experiments, we finally used 9 neural nets altogether. For every network (resp. committee) reading one data field, we then have to choose the constants α, β for the cost calculation (1, 2), see 3.3. Starting from values by experience, these parameters where roughly optimized by rather small grid searches over some validation data taken from the ANWRESH training data, see 4.1.

Here, it turned out that family names, i.e. the first word of the table field `NAME`, and given names, i.e. the rest of that field, should better use different decoding weights

Table 5 shows those committee and decoding parameters details.

field data	network type	number of networks (size of committee)	decoding α	weights β
NAME – family name	$\mathcal{N}$	3	0.50	0.50
NAME – given name			0.25	0.25
RELATION	$\mathcal{R}$	2	1.00	0.00
Age	$\mathcal{A}$	1	0.75	0.25
Marital condition	$\mathcal{M}$	1	0.75	0.25
PLACE OF BIRTH	$\mathcal{B}$	2	1.00	0.00

Table 5: Committees and Decoding Parameters

Acknowledgement

The work presented in this paper was funded by research grant no. V220-630-08-TFMV-S/F-059 (Verbundvorhaben, Technologieförderung Land Mecklenburg-Vorpommern) in European Social / Regional Development Funds.

References

[GFGS06] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[GS08] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *NIPS*, pages 545–552, 2008.

[LLS13] Gundram Leifert, Roger Labahn, and Tobias Strauß. 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]*, August 2013. Available: `http://www.nist.gov/itl/iad/mig/hart2013_wrkshp.cfm`.

[SGLL14] Tobias Strauß, Tobias Grüning, Gundram Leifert, and Roger Labahn. 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, April 2014. Available: `http://ftp.math.uni-rostock.de/pub/preprint/2014/pre14_02.pdf`.