READ
RECOGNITION & ENRICHMENT OF ARCHIVAL DOCUMENTS

D7.20
Model for Semi- and Unsupervised HTR Training P2

How to get a good HTR without expensive ground truth production

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	31.12.2017
Actual date of delivery	30.12.2017
Date of last update	December 11, 2017
Deliverable number	D7.20
Deliverable title	Model for Semi- and Unsupervised HTR Training P2
Type	Demonstrator
Status & version	final
Contributing WP(s)	WP7
Responsible beneficiary	URO
Other contributors	UPVLC
Internal reviewers	Joan Andreu Sánchez (UPVLC), Tobias Hodel (StAZH)
Author(s)	Gundram Leifert, Tobias Strauß, Roger Labahn
EC project officer	Martin Majek
Keywords	semisupervised, unsupervised, assignment

Contents

Executive summary

The second year deliverable describes the task and generic sub-tasks of it. After first results in year one, we achieved very good results in scenarios with given transcripts, even with missing line breaks and hyphenations.

1 Introduction

As mentioned in D7.18, the accuracy of the Handwritten Text Recognition (HTR) increases when more training data is available. So, it is meaningful to provide as much training data as possible to the HTR. The classical way to produce training data still is to extract text lines and transcribe them manually. This is expensive, because a lot of user interaction is necessary: A human has to mark a text line by drawing a baseline under the text and transcribe the corresponding text line.

For many documents the transcripts are already available, for example as TEI-XML-file. However, these transcripts are usually only assigned on page level, almost never on line level. Therefore, it is highly needed to assign transcripts on line level to use the documents as training data – Section 2 targets this.

Even better would be a training process which does not need transcripts at all. How this works will be explained in Section 4.

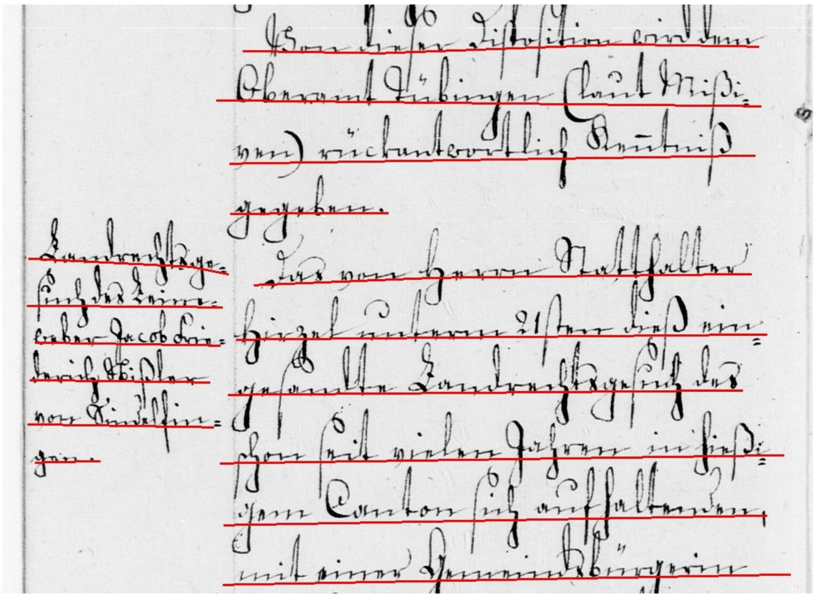
2 Images with corresponding transcripts

To train an HTR with the classical training one has to transcribe 40 to 200 pages, depending on the difficulty of the script. This requires a lot of manual effort and domain knowledge consequently it is expensive. As mentioned before, many editions as well as transcription projects can provide transcribed texts that could not be used for training of HTR models. The key idea is to use these data. Whereas in year one, we tackled the problem of assigning transcripts with correct line breaks, in year two we investigated an algorithm that only needs alignment on page level. In Fig. 1 an example image with text lines and the corresponding transcripts is shown.

2.1 Semi-supervised training workflow

To start the semi-supervised training workflow, one needs an initial loosely trained HTR for the first text2image process: Either one can take an already trained HTR (for example from previous projects) or train an HTR on only a few pages. Even if the HTR has a Character Error Rate of 50%, it reads sufficiently well for an initial text2image process. Due to more ground truth (GT) produced by text2image the training will result in a better HTR. If we use this HTR for text2image again, the contribution of the HTR is better and even more GT can be generated. This training-text2image-cycle can be done multiply times with allocating more GT and better HTR after each iteration.

The general workflow for semi-supervised training can be divided into two subprocesses **text2image** and **training** (see Fig. 2). The text2image process first applies an LA



Von dieser Disposition wird dem Oberamt Tübingen (laut Mißiven) rück-antwortlich Kenntniß gegeben. Landrechtsge-such des Leineweber Jacob Friederich Nißler von Sin-delfingen. Das von Herrn Statthalter Hirzel unterm 21sten dieß eingesandte Landrechtsgesuch des schon seit vielen Jahren in hießigem Canton sich aufhaltenden, mit einer Gemeindsbürgerin

(a) Image (b) Transcripts

Figure 1: **The task of the text2image process:** The text line (annotated by a human or automatically assigned by an LA) have to be assigned to the transcripts. The algorithm have to deal with distortions like missing/additional text lines or transcripts without hyphenations. A more challenging task is to assign the blue written transcripts in 1b to the marginalia in 1a, even if the even if the text line order of 1a does not correspond to 1b.

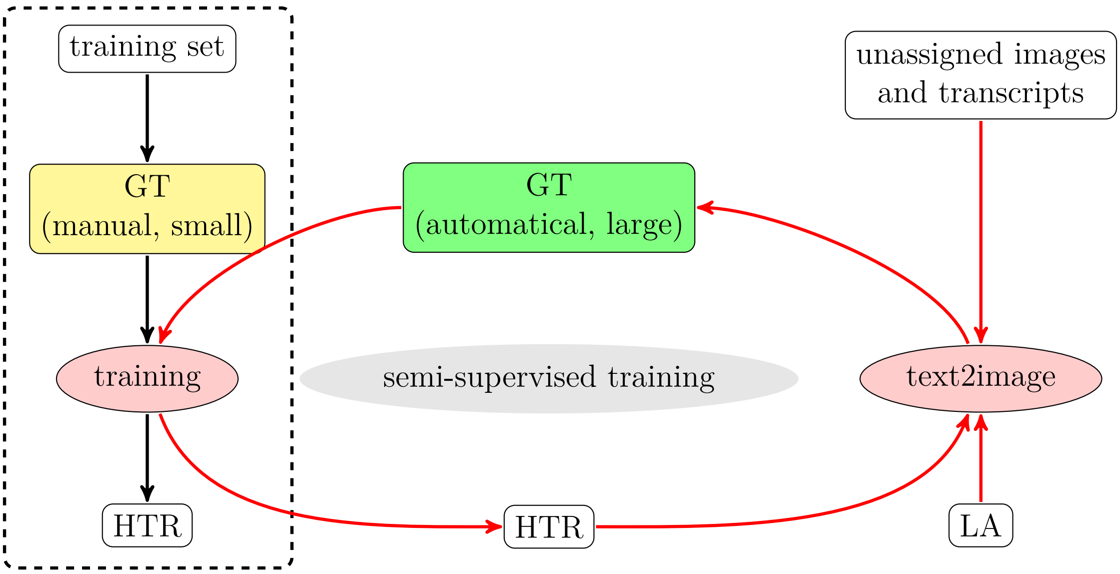


Figure 2: **Semi-supervised training workflow:** the classical training (dashed box) only uses manually produced GT (yellow box). The text2image process uses an LA and HTR to create GT automatically (green box) from the unassigned images and transcripts. With this additional GT a better HTR can be trained.

(see D6.11) on the images to find possible text lines for training. Note that the result of an LA can be incorrect (missing lines, split/concatenate lines/columns, ...) which have to be handled by the text2image process. The text2image process tries to assign the transcripts to the text lines with the help of the HTR. If there is a text line where a transcript assignment is good enough, the line is used as GT (see Fig. 2, green box). How the process works in detail is discussed in Section 2.2. The training process is similar to the classical HTR training (see Fig. 2, dashed box): a set of GT data is used to train an HTR (see D7.7 Section 3.3).

2.2 Text2Image

2.2.1 General process

We may assume to have a list of text line images – either manually annotated or automatically produced by an LA. The HTR transforms the text line image into a *Line-ConfMat* (see D7.7 Section 3.4). Next, we concatenate all Line-ConfMats adding a row in between which has the probability of 1 for the synthetic character "↵". Now, we have one long ConfMat containing all text lines in a page and one long transcription. We use the CTC-algorithm (see [2] for details) to calculate the probability of the transcript in this ConfMat. To handle missing line breaks in the transcripts, we also allow the space character in the transcripts to be interpreted as line breaks. If one only uses the most probable path (without summing up paths), one gets an exact assignment between the ConfMat and the transcripts. This path we call *BestPath*. If the BestPath is cut at the synthetic "↵" characters, one get one transcript for each Line-ConfMat. Now, we can validate if the assigned transcript fits to the Line-ConfMat using the CTC algorithm. If the confidence is sufficiently high, we can use the text line with the corresponding transcript as GT.
In realistic scenarios this mapping is not that easy. There are many distortion in text lines and transcripts, that have to be handled. We will explain the most important ones and show how we can handle them.

2.2.2 Skipping a Line-ConfMat

When the LA erroneously found a text line, i.e by incorrectly interpreting a distortion as line, the process should be able to ignore its corresponding Line-ConfMat. This can be achieved by adding an additional path in the CTC-algorithm. With a specific probability the whole Line-ConfMat can be skipped. Let $L_i \in \mathbb{N}$ be the length of Line-ConfMat i and $p_1 \in [0, 1]$ a hyperparameter to skip a text line. We add the additional path in the CTC-algorithm to skip the Line-ConfMat i with the probability $p = p_1^{L_i}$.
This solution also solves the problem if transcripts are missing.

2.2.3 Skipping transcripts

Like in Section 2.2.2, there are similar reasons to skip transcripts, for example if the LA did not find a text line. This also can be done by adding an additional path into the CTC-algorithm. Let $\boldsymbol{t} := (t_1, \dots, t_X)$ be the transcripts of length $X \in \mathbb{N}$ with Unicode characters t_i. Let $S := \{0, X+1\} \cup \{s \in \mathbb{N} | t_s = \text{"␣"}\}$ be indices of spaces and boundaries in the transcript and let $p_2 \in [0,1]$ be a hyperparameter to skip a word. For each $s_i, s_j \in S, s_i < s_j$, we add a path to skip all character between s_i and s_j (so the sequence $\left(t_{s_i+1}, t_{s_i+2}, \dots, t_{s_j}\right)$ will be skipped) with probability $p = p_2^{s_j - s_i}$. This solution also solves the problem if given transcripts are not on the image.

2.2.4 Unknown character

In many scenarios, there exist characters in the transcripts which cannot be recognized by the HTR, because they were not in the training set so far. Theoretically the probability to read such a character in the ConfMat is 0. This makes it impossible to expand the character set of the HTR. On the other hand, we know, if the HTR has to recognize an unknown character, it tries to read any of the known character. So for unknown characters we add the highest probability of *any character except spaces*. In this way so far unknown characters can come into the training set. Note that this only works, if unknown characters occur seldom (less than 2%).

2.2.5 Hyphenations

In many transcript, the hyphenations are not transcribed which causes that any text line with a hyphenated word at the beginning or end cannot be used for training. In some scenarios this hits over 25% of the lines, so it is crucial to handle hyphenations as well.
The rules for hyphenation differ a lot in documents so that we use following assumptions to tackle hyphenations.

- Hyphenation can only appear between character with the Unicode general category "LETTER"
- The text line ends either with or without a hyphenation sign.
- The text line begins either with or without a hyphenation sign.

Note that both, beginning and end hyphenation signs, are sets of characters, that means one can define more than one character as hyphenation sign (e.g. "-" and "¬"). Between any two characters, which can be divided by a hyphenation, we add a *hyphenation paths* in the CTC-algorithm.
In this deliverable, we concentrate on a simple and the most common configuration with one hyphenation sign "¬" at the end of a line and no hyphenation sign at the beginning of a line. Let t_i, t_{i+1} be two characters that can be divided by a hyphenation and let $p_3 \in [0,1]$ be a hyperparameter to add hyphenation. As alternative to the sequence (t_i, t_{i+1}), we add the sequence $(t_i, \text{"¬"}, \text{"↵"}, t_{i+1})$ with an additional probability penalty $p = p_3$. If it is more likely to hyphenate a word instead of forcing the assignment to squash the hyphenated word into one of the lines, the transcript is extended by two characters "¬" and "↵".

2.3 Experiments

In this section we want to show that semi-supervised training is a real alternative to the classical supervised training process.

2.3.1 Data set

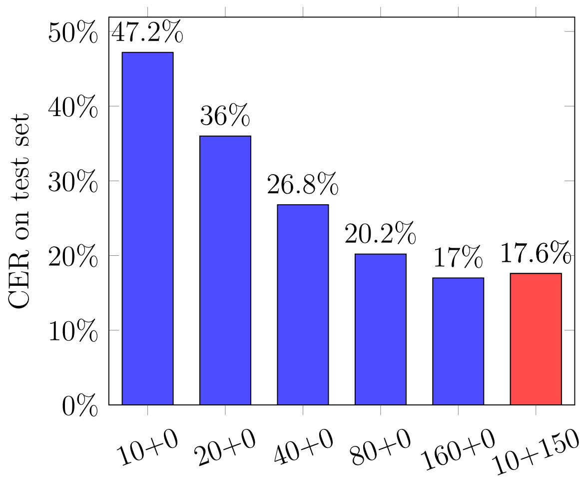
The data set for the experiments was chosen from 800 pages of the Staatsarchiv des Kantons Zürich[1] collection (see Fig. 1a). This set consist of *Regierungsratsbeschlüsse*, minutes of the highest executive of the canton of Zürich (Switzerland).
The texts consists of resolutions and enactments of the cabinet (starting in 1848 "canton"). The first documents were written in 1803, the last in 1882. The script is a very well-formed and highly trained German current. Different scribes wrote, from a paleographic point of view they are quite similar but still distinguishable.

A subset of this collection was used for training the neural network: 4 pages from every second year, i.e., 160 pages.
For validation of the models a test set with 1 page every second year is chosen so that the writer are different between training and test set.

2.3.2 Classical supervised training

The classical training process is already implemented in Transkribus. The HTR is trained on 10, 20, 40,80 or 160 pages. We used stochastic gradient decent training with learning rate $\lambda = 10^{-4}$, momentum $\mu = 0.9$ and mini-batch $b = 16$ over 400.000 training samples. We applied image augmentation and network noise while training to prevent overfitting. The resulting Character Error Rate (CER) on the test set can be seen in Fig. 3 (blue columns). We can easily see that the more training data are used the better the HTR works on the test set.

Figure 3: **Comparison of training size and method:** An HTR is trained on a specific number of pages and the CER is calculated on an independent test set (blue columns). If we train further the HTR "10+0" using the other 150 pages of the training set for the text2image process (red column), we can nearly reach the same performance as using all 160 pages for the classical training.



[1]The Staatsarchiv Zürich is a large scale demonstrator of the READ project.

2.3.3 Semi-supervised training

The HTR trained on 10 pages has a very bad performance (Fig. 3, column "10+0"). It cannot be used for transcription at all. But using it for the text2image process, it is sufficiently trained. Even the very bad HTR with 47.5% CER can assign 35% of the lines. The number of training sample increases and after only 20.000 additional training samples the CER of the HTR drops to 31.9% which again results in a better text2image result. Repeating this circle of text2image and training multiply times, more training data can be produced and a better HTR is trained. At the end 97% of the text lines can be used for training. A final training results in an HTR with 17.6% CER on the test set. This is only 0.6% more CER than the HTR trained on 160 pages and 2.4% less CER than the HTR trained on 80 pages.

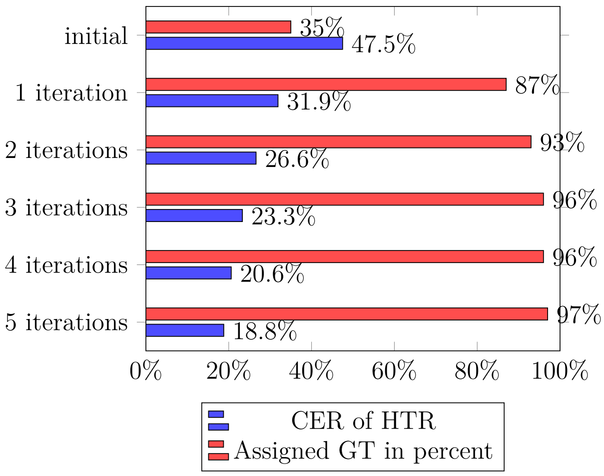


Figure 4: **The assignment quality of the text2image process depends on the HTR quality:** The text2image process in the semi-supervised training workflow (see Fig. 2) can assign 35% of the text lines using an HTR with 47.5% CER. After each iteration the number of automatically assigned GT samples increases and the HTR gets better. At the end the HTR with 18.8% CER can assign 97% of the text lines.

2.4 Integration in Transkribus X

After implementing the workflow in the middle of year two the workflow[2], the software was integrated into Transkribus X by UIBK (see D5.4; Section 3.6) and was provided for selected users. StAZH (Large Scale Demonstrator) applied this semi-supervised training workflow as implemented in Transkribus X on two own and three external data sets with very good and promising results (see D8.4; Section 2.1). This demonstrates that the semi-supervised training workflow in TranskribusX can be applied by non-expert users in practice. The constraints for the ingest process and the semi-supervised training thus needs only circumstantial knowledge and can be opened to power users (similar like HTR training).

2.5 Conclusion

We showed that using the semi-supervised training workflow can help training good HTRs even if the transcripts are not assigned to the text lines. An automatic assignment

[2]source code available on https://github.com/Transkribus/CITlabModule

done by the text2image process can create GT so that the classical training can be used.

3 Transcripts without usable breaks

Having, maybe, hundreds of pages and thousands of lines, the problem is to find an algorithm that scales properly.

As this problem will be tackled later in the project, we will report on it in later deliverables.

4 Images without transcripts

This section aims at generating training samples by transcribing text lines using a pre-trained HTR system. By scoring these transcripts, only reliable samples are used for training.

Remark 1. The work outlined in this section was done by UPVLC.

Current HTR technology requires transcribed lines together with the corresponding line images to train the HTR models. Producing this GT is expensive and time-consuming. In [3], we described the problem of semi-supervised HTR Training. In semi-supervised HTR training, the idea is to produce a small set of GT and then, to produce new GT automatically with the help of a language model as follows [1]:

1. Obtain initial models trained in a supervised way, with a small amount of lines.
2. Use the current models and a **very good Language Model** for obtaining transcripts automatically from a large amount of untranscribed lines together with confidence measures (at character level or at word level).
3. Use the more confident transcripts (and the corresponding images) from step 2 (and supervised samples from step 1) to train the models with forced alignment.
4. Repeat steps 2 and 3 until converge

Figure 5 shows a scheme of this process.

We applied this scheme to the same dataset defined in [4] and Table 1 shows a summary of the obtained results. Row “Tr-Sup 46K” shows the results when all training dataset was used to train the models in the usual way. Row “Tr-Sup 1K +Tr-Unsup 45K” shows the results when 1K words were used for initializing the models. We can see that the CER was very high, but note that the GT was very small. These are encouraging results.

For the following period we are going to apply these techniques to some of the datasets used in READ and to research in the following directions:

- To research if these WER/CER are enough for KWS/indexing.
- To research other methods for obtaining initial HMM.
- To research beyond HMM as optical models.

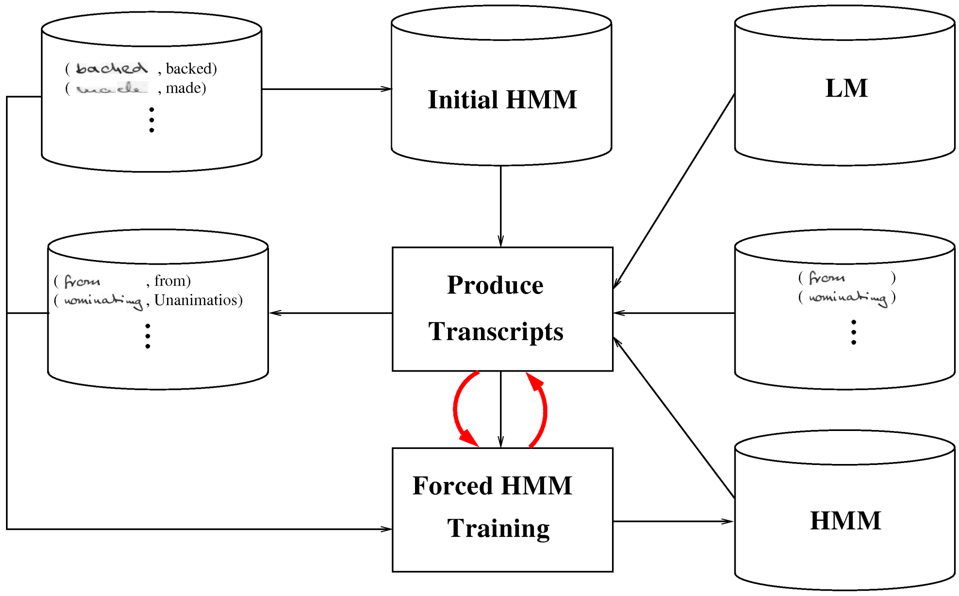


Figure 5: Scheme of the semi-supervised HTR training.

Table 1: Results obtained with the semi-supervised HTR training.

	CER	WER
1-gram word LM [4]	13.7	28.6
character-based LM		
Tr-Sup 46K	15.0	27.4
Tr-Sup 1K	55.5	76.0
Tr-Sup 1K +Tr-Unsup 45K	25.1	43.7

References

[1] V. Frinken, M. Baumgartner, A. Fischer, and H. Bunke. Semi-supervised learning for cursive handwriting recognition using keyword spotting. In *ICFHR*, pages 49–54, 2012.

[2] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural nets. In *ICML '06: Proceedings of the International Conference on Machine Learning*, 2006.

[3] T. Grüning, G. Leifert, T. Strauß, and R. Labahn. D7.19 model for semi- and unsupervised htr training p1. approaches, scenarios and first results. Technical report, READ EU project (674943), 2016.

[4] M. Kozielski, M. Nuhn, P. Doetsch, and H. Ney. Towards unsupervised learning for handwriting recognition. In *ICFHR*, pages 549–554, 2014.