READ

RECOGNITION & ENRICHMENT OF ARCHIVAL DOCUMENTS

D7.12
Language Models

Improving transcriptions by external language resources

Tobias Strauß, Max Weidemann, and 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	31.12.2018
Actual date of delivery	December 5, 2018
Date of last update	December 5, 2018
Deliverable number	D7.12
Deliverable title	Language Models
Type	report
Status & version	complete
Contributing WP(s)	WP7
Responsible beneficiary	URO
Other contributors	
Internal reviewers	ASV, UPVLC
Author(s)	Tobias Strauß, Max Weidemann, and Roger Labahn
EC project officer	Martin MAJEK
Keywords	n-grams, language models

Contents

Executive summary

In the last year, our experiments showed that the integration of language models into the decoding process decreases the error rates significantly. This year we focused on the integration such that it is now ready to be used from Transkribus.

1 Introduction

The key idea is to support the handwritten text recognition (HTR) system by external domain knowledge about the language. For any position t of the text line image, the neural network (as they are used in Transkribus) outputs a probability $y_{t,c}$ for any learned character c and also for a garbage label called NaC (compare D7.7 Section 3.4). The networks probability of any sequence of such labels (characters and NaCs) is simply the product of the individual probabilities at the specific positions. Thus, the network provides a probability for any possible transcription of the specific text line image. So-called *language models (LM)* estimate the probability of a specific word w given a history of words $w_1, w_2, \ldots$ using external language resources[1]. Assuming that these probabilities model the language of the current document well, we output the transcription which maximizes a combination of the HTR probability and the LM probability (as it was done in [Amodei et al., 2015], see Eq. (12)).

Task 7.4

The task is described in *Grant Agreement: 674943 — Recognition and Enrichment of Archival Documents (READ)*:

> This task will research in different ways how to prepare linguistic resources for the collections to be transcribed:
>
> 1. to use adaptation techniques for selecting the text from modern linguistic resources more closely related with the documents that is being transcribed;
> 2. to research how to obtain inflected forms of words for historical variants of a given language;
> 3. to research how to deal with hyphenated words and
> 4. dealing with Out-of-Vocabulary (OOV) words (i.e., words that the HTR engine has not seen during the training process).
>
> In this task we will research how to deal with this problem by using character-based language models. These models have to be combined efficiently with the word-based language models in the HTR system.

[1] Language resources can be extracted from existing transcripts from XML, PDFs or DOCX. See D5.2 for further details.

2 Character n-grams

As a result of the last year's investigation, we favor n-gram models over neural models. The classical n-gram yielded better results (lower character error rates). We assume that the reason is that neural models are better at word level since they can model the back-off probabilities depending on the meaning of the words. This seems not to transfer to character language models since characters itself do not have a meaning.

2.1 Character n-Gram

Recall definition of n-grams from the last year's deliverable: In statistical natural language processing, so-called n-grams (see [Manning et al., 1999]) are well-known and widely used. They make extensively use of the multiplication theorem of probability: Given the sequence $w_1, w_2, \ldots, w_N$ over some alphabet $\mathcal{A}$, then

$$\mathrm{P}(w_1, \ldots, w_N) = \mathrm{P}(w_1) \prod_{i=2}^{N} \mathrm{P}(w_i \mid w_{i-1}, \ldots, w_1) \tag{1}$$

assuming a Markov property of order n

$$= \mathrm{P}(w_1)\, \mathrm{P}(w_2 \mid w_1) \ldots \mathrm{P}(w_{n-1} \mid w_1, \ldots, w_{n-2}) \prod_{i=n}^{N} \mathrm{P}(w_i \mid w_{i-1}, \ldots, w_{i-n+1}). \tag{2}$$

Let $c(w_1, \ldots, w_k)$ denote the number of occurrences of the sequence $w_1, \ldots, w_k \in \mathcal{A}^k$ in an a-priori given text corpus. The n-grams model the conditional probabilities $\mathrm{P}(w_i \mid w_{i-1}, \ldots, w_{i-n+1})$ by counting the relative frequencies

$$\mathrm{P}(w_i \mid w_{i-1}, \ldots, w_{i-n+1}) = \begin{cases} \frac{c(w_{i-n+1}, \ldots, w_i)}{c(w_{i-n+1}, \ldots, w_{i-1})} & \text{if } c(w_{i-n+1}, \ldots, w_{i-1}) \neq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

Note that the alphabet is not limited to a character set. It may also consist of natural words, syllables etc. such that $w_1, \ldots, w_N$ may be a sequence of characters or words, equally. If not stated otherwise, we assume $\mathcal{A}$ as a character set since character n-grams are able to handle inflections, hyphenations and OOV words as required by Task 7.4.

The classical n-grams as defined by Eq. (3) can model only frequencies of sequences which appeared in the training corpus. Any other sequence gets zero probability although many words in the sequence might equally be substituted by synonyms. A sufficiently large training corpus which covers this problem is typically not available. Several smoothing techniques were proposed to also assign non-zero probabilities to unseen sequences (e.g., in [Kneser and Ney, 1995]).

We use Berkeley LM[2] since it is easy to use, open source and optimized in running time.

[2] `https://code.google.com/archive/p/berkeleylm/`

Table 1: Character n-gram vs. word n-gram – comparative results on the StAZH dataset

Model	Training data	CER in %
No language model	-	3.0
Character 7-gram	training set	2.6
Character 7-gram	training and validation set	2.2
Character 8-gram	training and validation set	2.1
Word 2-gram	training set	3.0
Word 2-gram	training and validation set	1.9
Word 3-gram	training and validation set	1.8

The advantages of n-grams are the well understood theory besides the fast training (only counting frequencies). The drawbacks are that the n-grams tables can become huge which slows down the lookup for great n. Furthermore, n-grams do not take the word meaning into account.

2.2 Comparison to Word-n-grams

An open point of the last year's deliverable was the comparison of character and word language models. Please note that a fair comparison is not possible since character and word language models always differ in the number of parameters they rely on.

Again we chose the StAZH dataset for the comparison like last year. This set consists of *Regierungsratsbeschlüsse*, minutes of the highest executive of the canton of Zürich (Switzerland).

The texts include 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 per year, i.e., 320 pages. For validation of the models, 1 page of every second year from the above mentioned 800 pages was used, i.e., in total 40 pages.

Compared to last year, the HTR error rates decreased drastically (from 14.9 % CER to 3% CER).

The results can be found in Table 1. The character 7-gram decreases the error rate by 10% if it is trained only on the training set while the word 2-gram yields almost no improvement under the same conditions. One could say that this is not a fair comparison since the word language models cannot decode unknown words. To estimate the possible improvement under perfect conditions, we trained both models on the training and validation set. This shows that word language model can work better if the true set of words is known. But also the character language models perform much better.

Therefore, we decided to use n-grams in the decoding process for now.

Table 2: Experimental Results

	CER (in %)	
Dataset	no LM	n-gram
StAZH (SPRNN)	14.9	8.8 ($n = 10$)
StAZH (HTR+)	3.0	2.6 ($n = 7$)
IAM (HTR+)	5.93	5.18 ($n = 6$)
Konzilsprotokolle	12.305	8.382 ($n = 6$)

2.3 Process

To enhance the decoding process by language models, we need texts to train the language models on. Since we also need training data for the optical model / neural network, in many cases the ground truth of the neural network training can be reused to train the language model.

We implemented a JAVA-package[3] which handles the tokenization, wraps the training process and provides the beamsearch for the decoding. The training is fast, i.e., in a few minutes an n-gram is trained on several hundred pages. The Training can be executed conveniently through CITlabModule[4].

The result is an n-gram file (either the standard "ARPA" - text file or a binary compressed file) which has to be passed to the decoding process. This beamsearch-decoding implements standard interfaces from CITlabModule. Best-path-decoding the dictionary-lookup and the n-gram beamsearch-decoding can be used interchangeably in CITlabModule.

3 Conclusion and future work

The comparison of word and character n-grams showed that word language models may decrease the error rate more than character language models if all words are known and the training set fits to the test scenario. This seems to be a too restrictive condition. For now character n-grams together with beamsearch are the preferred choice of decoding enhancement. We implemented a workflow which allows fast training and inference of (character and word) n-grams from Transkribus.

It remains to investigate how combinations of character and word language models can be used to combine the advantages of both methods. On the other hand, there is a working decoding with language models available in Transkribus such that the language model integration of WP7 can be considered complete within READ.

[3] `https://github.com/CITlabRostock/CITlabLanguagemodel`
[4] `https://github.com/CITlabRostock/CITlabModule`

References

[Amodei et al., 2015] Amodei, D., Anubhai, R., Battenberg, E., Carl, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J., Fan, L., Fougner, C., Han, T., Hannun, A., Jun, B., LeGresley, P., Lin, L., Narang, S., Ng, A., Ozair, S., Prenger, R., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Wang, Y., Wang, Z., Wang, C., Xiao, B., Yogatama, D., Zhan, J., and Zhu, Z. (2015). Deep-speech 2: End-to-end speech recognition in English and Mandarin. *Jmlr W&Cp*, 48:28.

[Kneser and Ney, 1995] Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

[Manning et al., 1999] Manning, C. D., Schütze, H., et al. (1999). *Foundations of statistical natural language processing*, volume 999. MIT Press.