READ

RECOGNITION & ENRICHMENT OF ARCHIVAL DOCUMENTS

D7.11
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	31.12.2017
Actual date of delivery	December 6, 2017
Date of last update	December 6, 2017
Deliverable number	D7.11
Deliverable title	Language Models
Type	report
Status & version	in process
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

1 Introduction **4**

2 Approaches **5**
2.1 Dictionary 5
2.2 Character n-Gram 5
2.3 Neural character language model 6

3 Tools **6**
3.1 Preprocessing 6
3.2 Beam search decoder 7

4 Experiments **7**
4.1 Data set 7
4.2 Size of training corpus 8

5 Conclusion and future work **9**

Executive summary

While last year the language models were applied to modify textual transcriptions to find and expand abbreviations (deliverable D7.10), this year we integrated language models into the decoding process. Besides the recognition engine, URO implements also the decoding of the neural network (compare deliverable D7.7). Thus, the direct integration of language models by URO is more efficient / less work than for example for ASV. This is also the reason why we moved the *responsible beneficiary* from ASV to URO.

First tests show a significant drop in the error rate.

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).

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

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.

2 Approaches

To enhance the transcription by external language resources, we implemented a dictionary support in year 1 (see Subsection 2.1). In year 2, we implemented n-gram language models (see 2.2) and started some research in neural language models (see Subsection 2.3).

2.1 Dictionary

Currently, external language resources are only used as dictionaries. Consider the network output: If consecutive positions contain most likely letters, the true transcription of these regions are probably words. We decode the most likely word from the dictionary within such word region and return the most likely character (e.g, spaces, punctuations, numbers etc.) of any other position. We also incorporate the relative frequencies of these dictionary words if they are available (this corresponds to a 1-gram, see Subsection 2.2). This method is fast, simple and needs only little training data. The drawback is that this method is prone to segmentation errors. To satisfy the requirements of Task 7.4 (especially OOV words), we also accept the "raw reading" of the neural network, i.e., the most likely character sequence, besides the dictionary words. This is only reliable if the character error rate is very low.

2.2 Character n-Gram

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 on 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.

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.

2.3 Neural character language model

Neural networks can also be trained to model the probabilities of Eq. (1). Recurrent neural networks model $\mathrm{P}(w_i \mid w_{i-1}, \ldots, w_1)$, directly. Thus, they can incorporate an arbitrarily long context in theory which results in "around 18% reduction of word error rate" in speech recognition experiments ([Mikolov et al., 2010]). However, the last $i-1$ sequence elements serve as input to the neural network such that the number of possible inputs grows exponentially in n. We avoid the problem of choosing the most promising sequences per position for further exploration by again assuming the Markov property such that the networks model $\mathrm{P}(w_i \mid w_{i-1}, \ldots, w_{i-n+1})$.

We use a rather simple (feed forward) network architecture to stay efficient compared to the n-grams: The input to the network is a learned character embedding followed by a hidden layer with rectifier activation function and a softmax output layer.

This approach might model unseen character sequences better (see [Arisoy et al., 2012, Bengio et al., 2003]) since the various articles show that networks are able to model meanings of words (e.g., see [Mikolov et al., 2013]). So in principle, the neural network can learn for example verbs and assign a high probability to an inflected form without ever seen it. Drawbacks are a longer training time and a dependency of hyperparameters such as network size, architecture, learning rate etc.

3 Tools

3.1 Preprocessing

To learn language models, textual resources have to be split into small atomic parts from $\mathcal{A}$ such as characters, syllables or words. A tool which splits texts in such parts is

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

called Tokenizer. We implemented a Tokenizer[3].

Our implementation relies on two main classes: The actual Tokenizer and a Categorizer. The Tokenizer splits the character sequence at isolated characters and delimiter classes. The Categorizer determines for a given character the category and properties such as isolation and delimitation. Different tasks require different Categorizers: for example, preparation of texts to train a character or word language model involve different Categorizers.

3.2 Beam search decoder

The language models calculate the probability of a token w_t given a history of tokens $w_{t-n+1}, \ldots, w_{t-1}$. To calculate the exact maximum, the probability of any possible history at any position has to be compared. The number of histories can be huge even for small n. For example: For a language model with alphabet size even a 5-gram has 100^5 (i.e., 10 billion) histories which have to be determined at each position. The typical way to deal with that problem is to explore only the most promising sequences. For this reason, we implemented a beam search decoder. This decoder is able to handle classical n-grams as well as neural language models of both types: word and character language models.

4 Experiments

4.1 Data set

The data set for the experiments was chosen from 800 pages of the Staatsarchiv des Kantons Zürich[4] collection. 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 per year, i.e., 320 pages. For validation of the models two, separate validation sets (see Fig. 1) were used:

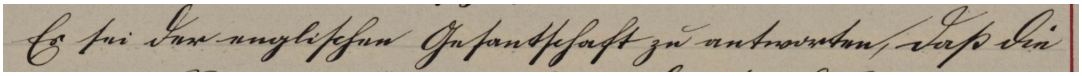
val_a contains 1 page of every second year from the above mentioned 800 pages, i.e., in total 40 pages.

val_b consists of 20 pages of minutes of the parliament of the state of Zurich from the year 1882 (Transkribus document id 13709). The writers are unknown to the neural network.

[3]See D5.2 for details. The code is available at `https://github.com/Transkribus/TranskribusTokenizer`.

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

a) sample from val_a



b) sample from val_b

Figure 1: Sample lines from the two different validation sets.

Table 1: Error rates for val_b. The n-grams and the neural LM take the shown minimum CER at $n = 13, 12$ and 12, respectively.

	raw reading	word 1-gram	"big" n-gram	"small" n-gram	neural LM
CER in %	22.8	20.4	17.7	18.4	18.6
n	-	-	13	12	12

4.2 Size of training corpus

Setup We used two corpora: a "big" corpus consisting of 2598 randomly selected paragraphs from the whole collection (504821 words, 3679928 characters, no texts from the validation pages included) was used to learn character n-grams (referred to as "big" n-grams, see Subsection 2.2) and (feed forward) neural language models (see Subsection 2.3) with n ranging from 3 to 13.

A "small" corpus consisting only of the 320 training pages (60548 words, 429743 characters) was used to train "small" n-grams. The dictionary for the word unigram (see Subsection 2.1) is created also from the small corpus.

The error measure is the character error rate (CER).

Results Figure 2 shows the results on the val_a set. The raw reading resulting directly from the neural network yields 14,9% CER. The currently implemented dictionary / word 1-gram method already decreases the error to 12,3% with the standard hyperparameters as used in Transkribus. For greater n, the grams and neural LMs yield far better error rates. The "big" n-gram even decreases the error below 8,8% ($n = 10$). The "small" n-gram yields 9.3% (any n from 8 to 13) and the neural LM 9,7% ($n = 10$). Surprisingly, the decoding time with Berkley LM on val_a increased significantly from 33 min ($n = 3$) to 123 min ($n = 13$). The decoding time with the neural LMs on the same data increased from 43 min ($n = 3$) to 77 min ($n = 13$).

The results on the validation set val_b (see Table 1) are similar although the relative CER drop is not that spectacular. Also here, the n-gram models beat the other methods.

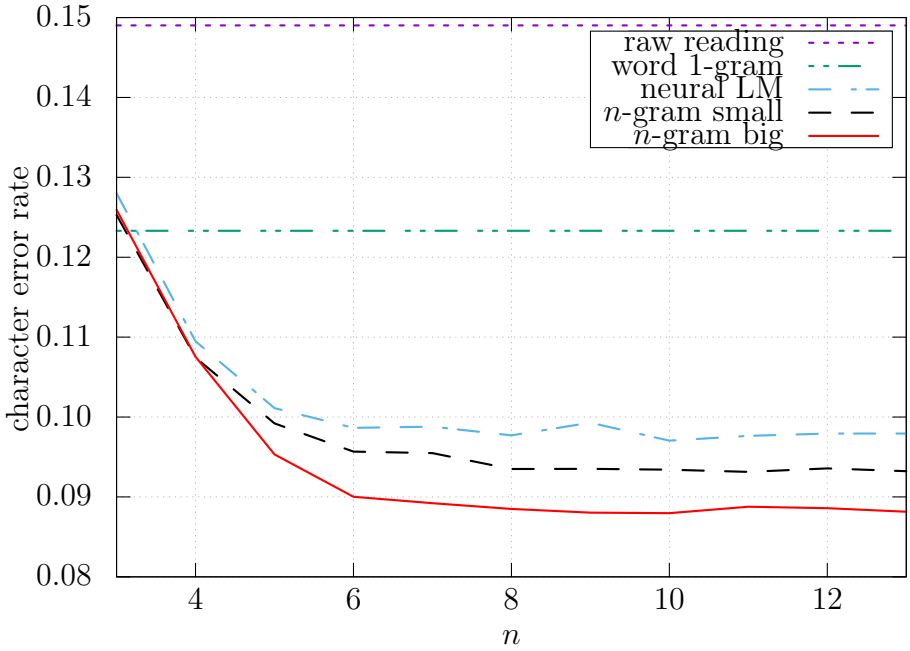


Figure 2: Error rates on validation set val_a for various language models.

Discussion The data from val_a clearly fits to the training data of both: the network and the language model. Thus, the good results are not surprising. But the general statements transfer also to data which do not fit directly to the training data as the results on val_b show. Currently, the n-gram models seem to be the method of choice. They are easy and fast to train and yield the best error rates in both experiments. Unfortunately, we could not confirm the abstraction capabilities of neural LMs reported in the literature. Either the architecture is too simple or these properties only hold for word LMs where a specific sequence is much more unlikely since there are much more words than characters.

5 Conclusion and future work

We showed that language models yield a huge performance boost. The CER on the validation set val_a was decreased from 14.9% to 9.3 % by using only the training data and even further if other external language resources are integrated. The generation of such language models is simple and will be implemented in year three into Transkribus.

In year three, a comparison to word language models has to be done for sake of completeness. We also plan to be investigate better neural language models to reproduce results from the literature.

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.

[Arisoy et al., 2012] Arisoy, E., Sainath, T. N., Kingsbury, B., and Ramabhadran, B. (2012). Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28. Association for Computational Linguistics.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

[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.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.