10.12871/CLICIT2014136

Correcting OCR errors for German in Fraktur font

Michel Généreux, Egon W. Stemle, Verena Lyding and Lionel Nicolas
EURAC Research
Viale Druso, 1 / Drususallee 1
39100 Bolzano / Bozen - Italy
{michel.genereux, egon.stemle}@eurac.edu
{verena.lyding, lionel.nicolas}@eurac.edu

Abstract

English. In this paper, we present ongoing experiments for correcting OCR errors on German newspapers in Fraktur font. Our approach borrows from techniques for spelling correction in context using a probabilistic edit-operation error model and lexical resources. We highlight conditions in which high error reduction rates can be obtained and where the approach currently stands with real data.

Italiano. *Il contributo presenta esperimenti attualmente in corso che mirano a correggere gli errori di riconoscimento ottico dei caratteri (OCR) in articoli di giornale scritti in lingua tedesca e nel carattere gotico Fraktur. L'approccio è basato su tecniche di controllo ortografico contestuale e utilizza un modello probabilistico di correzione degli errori assieme a delle risorse lessicali. Si descrivono le condizioni in cui è possibile ottenere un alto tasso di riduzione degli errori e si illustra infine lo stato di avanzamento attuale mediante dati reali.*

1 Introduction

The OPATCH project (Open Platform for Access to and Analysis of Textual Documents of Cultural Heritage) aims at creating an advanced online search infrastructure for research in an historical newspapers archive. The search experience is enhanced by allowing for dedicated searches on person and place names as well as in defined subsections of the newspapers. For implementing this, OPATCH builds on computational linguistic (CL) methods for structural parsing, word class tagging and named entity recognition (Poesio et al., 2011). The newspaper archive contains ten newspapers in German language from the South Tyrolean region for the time period around the First World War. Dating between 1910 and 1920, the newspapers are typed in the blackletter Fraktur font and paper quality is derogated due to age. Unfortunately, such material is challenging for optical character recognition (OCR), the process of transcribing printed text into computer readable text, which is the first necessary pre-processing step for any further CL processing. Hence, in OPATCH we are starting from majorly error-prone OCR-ed text, in quantities that cannot realistically be corrected manually. In this paper we present attempts to automate the procedure for correcting faulty OCR-ed text.

2 Previous work

Projects, scientific meetings[1] and studies like OPATCH dealing with historical texts (Piotrowski, 2012) are numerous and one recurring theme is the struggle for clean OCR-ed data.

Approaches to post-OCR correction include machine learning (with or without supervision) (Abdulkader and Casey, 2009; Tong and Evans, 1996), merging of more than one system outputs (Volk et al., 2011) or high frequency words (Reynaert, 2008). The approach in Niklas (2010) combines several methods for retrieving the best correction proposal for a misspelled word: A general spelling correction (Anagram Hash), a new OCR adapted method based on the shape of characters (OCR-Key) and context information (bigrams). A manual evaluation of the approach has been performed on The Times Archive of London, a collection of English newspaper articles spanning from 1785 to 1985. Error reduction rates up to 75% and F-Scores up to 88% could be achieved.

For German, an approach akin to ours is Hauser (2007). This approach shares a number of features

[1]For example, DATeCH 2014: Digital Access to Textual Cultural Heritage, May 19-20 2014, Madrid, Spain.

with ours, such as a reference lexicon with similar coverage (90%) and fuzzy lookup matching of potential candidates in the lexicon for correction, based on the Levenshtein distance. However, while our weighting scheme for edit operations is based on an annotated corpus, Hauser (2007) uses a weighting model based on Brill (2000). Our approach also includes contextual information based on bigrams.

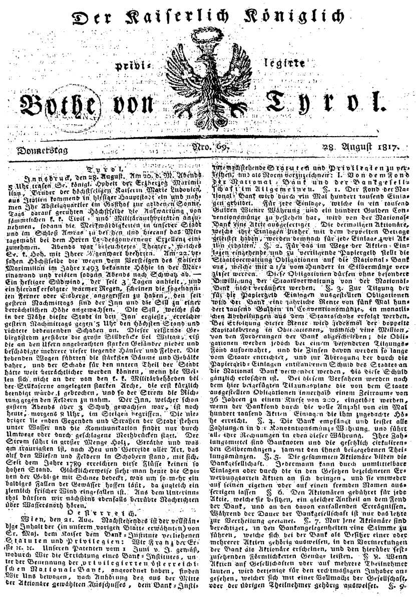
Hauser (2007) provides an evaluation of their approach on similar OCR-ed documents, that is from the same period (19th century) and font (Blackletter). Their evaluation on four collections shows error reduction rates from 1.9% to 4.8%, rates quite similar to those we report in tables 2 and 3. However, our results show error reduction rate can go up to 93%, depending on a number of idealized conditions which we will spell out further.

3 Corpora

We used two types of data sources: ten OCR-ed newspaper pages along with their manually corrected version, our Gold Standard (GS), and an independent reference corpus of German.

3.1 OCR-ed pages

Each of the ten pages has been OCR-ed[2] and revised manually[3] , so that for each page we have a scanned image in format TIF, an OCR-ed version in the format METS[4]-ALTO[5] and a manually revised version of the text. The Fraktur font and the decreased paper quality make the translation into text particularly challenging, so the OCR-ed documents are extremely noisy. On average, more than one out of two tokens is misrecognized (see table 3), let alone a substantial number of fragmented and missing tokens. Almost half (48%) of tokens need a minimum of three edit operations for correction (see section 4.1). In total, the OCR-ed documents are made up of 10,468 tokens and 3,621 types. Eight pages (8,324/2,487) are used as training data (section 4) and two pages (2,144/1,134) for testing. One such page is shown in figure 1.


Der Kaiſerlich Königlich privi- legirte Bothe von Tyrol.

Donnerſtag Nro. 69. 28. Auguſt 1817.

[illegible]

Figure 1: A typical OCR-ed page

3.2 Reference corpus

The reference corpus is used as a basis for constructing a frequency list of unigrams (a dictionary) and a frequency list of bigrams. We used the SdeWaC corpus (Faaß and Eckart, 2013), a German corpus harvested from the web. We enriched the corpus with texts closer in time to the OCR-ed documents, that is texts in the categories of novels and stories (*Romane und Erzählungen*) from the period 1910-20, a total of 1.3 M tokens.[6] Our final reference corpus is made of 745M tokens from which we derived a dictionary of size 5M and a list of bigrams of 5M.[7] The 5M entries in the dictionary cover 91% of all tokens from the manually corrected OCR-ed files.

4 Approach

The approach consists of three steps: first we build a probabilistic model of edit-operations needed for correction, then we define a procedure to generate candidates for correction based on the model and finally we apply a scoring system to evaluate the most suitable candidate.

[2]Using ABBYY: `http://www.abbyy.com/`

[3]The GSs have not been aligned with the originals, so that there is no trace of where words added, subtracted or simply corrected.

[4]Metadata Encoding & Transmission Standard: `http://www.loc.gov/standards/mets/`

[5]Analyzed Layout and Text Object: `http://www.loc.gov/standards/alto/`

[6]Project Gutenberg: `http://www.gutenberg.org/`

[7]The size of the dictionaries was optimized for running times and computer memory.

4.1 Constitution of the edit-operations probability model

A correction is deemed necessary when a token has no entry in the dictionary. The OCR error correction system uses a probabilistic model built on typical edit errors to generate candidates when a correction is required. To build such a model, our first task is to collate and tally all edit-operations (*delete*, *insert* and *replace*) needed to transform all unrecognized tokens from the training OCR-ed texts to its corrected form in the GS. For example, to transform the token *Veranstaltnngstage* to *Veranstaltungstage* 'days of the event', we must replace the second 'n' with a 'u'. This edit-operation, replacing an 'n' with a 'u', is therefore recorded and tallied. From these counts we build a probability distribution. This part of the system finds its inspiration from Segaran and Hammerbacher (2009). This model defines our alphabet, which includes all the graphemes for German and all spurious symbols generated by the OCR process.[8] All edit-operations recorded constitute the constrained model. The unconstrained model also includes all edit-operations unseen during training, which are assigned a low residual probability.

4.2 Candidate generation

Candidate generation is achieved by finding the closest entries in the dictionary by applying the minimum number of edit-operations to an unrecognized OCR-ed token. The number of candidates is a function of the maximum number of edit-operations allowed and the model used (constrained or not) and may often be by the hundreds, and the sheer number of possibilities makes finding the closest entry more difficults. For example, if presented with the token *wundestc* and asked to generate all candidates within two edit-operations and using the constrained model, the system generates eight candidates, among which: *wundesten* 'sorest', by replacing a 'c' with an 'e' and then inserting an 'n' after the 'e'. When asked to use the unconstrained model, the number of candidates raises to fifteen.

4.3 Selection of the most suitable candidate

We consider the following four features to select the best possible candidate:

- The probability of all edit-operations multiplied together. The more number of operations involved, the lower the probability. In the example we presented in section 4.2, the edit cost to go from *wundestc* to *wundesten* would be the probability of replacing 'c' for 'e' multiplied by the probability of inserting an 'n' after an 'e'.
- The probability of the candidate drawn from the reference corpus (the relative frequency). This would be the frequency of *wundesten* (24) divided by the size of the reference corpus (745M).
- The two probabilities of co-occurrence of the candidate with the immediate left (and also right) neighbour of the token to correct. Probabilities are drawn from the frequency list of bigrams in the reference corpus and processing is carried out left-to-right.

Each candidate is then given a score by simply adding together the values for each of the four features above. In order for each of the features to contribute fairly to the overall score, we normalized the three distributions (*edit*, *unigram* and *bigram*) so that their mean is the same. We also stretched each distribution so that least probable values for a feature tend to zero and most probable to one. The scoring formula for candidate 'c' is:

$$\begin{aligned}&\prod_i prob(edit\_op_i) + prob(c)\\&\quad + prob(left\_word + c)\\&\quad + prob(c + right\_word)\end{aligned} \qquad (1)$$

5 Experiments

In the following experiments we first used a small list of OCR errors to test our model and then we applied the system on the two remaining pages from the ten OCR-ed pages.

5.1 Artificially generated errors

In order to have a comparable set of conditions to evaluate how the system performs, we generated a list of 2,363 errors somewhat artificially. To achieve this we extracted random trigrams from the GS (left context, target, right context) and applied, in reverse, the edit error model. Errors were introduced up to a limit of two per target and contexts. At the end of this process, we have two context words and five candidates, including the target. Table 1 shows the results. When given

[8] Alphabet: üÜöÖäÄß»«èà,,ì^()-/016 and [a-z][A-Z]

Five cand.	On the fly, open list of candidates					
	Constr. model			Unconstr. model		
	E1	E2	E3	E1	E2	E3
93%	86%	61%	40%	83%	28%	9%

Table 1: Error reduction rate on 2,363 artificially created errors

five candidates, the system picked the target 93% of the time. The En labels indicate the maximum edit-operations performed[9] to generate candidates. When candidates are generated 'on-the-fly' in variable quantity, we can see a drop in error reduction which was best when we limited the number of candidates (small value for n) and used the constrained model of edit errors. This is hardly surprising, given how the errors were generated in the first place.

5.2 Real errors

We now turn our attention to real errors, those coming from the two remaining pages of our OCR corpus. Table 2 shows the result. The set of 233

Constrained model			Unconstrained model		
E1	E2	E3	E1	E2	E3
16%	18%	15%	20%	16%	9%

Table 2: Error reduction rate on 233 real errors

errors is the result from aligning the two OCR-ed texts with their corresponding GS. We kept only tokens for which we had a clear alignment (see footnote 3 on page 2) and a target which was part of the dictionary. Accuracies dropped drastically for all types of configuration, due to a high proportion of tokens heavily modified by the OCR process (edit distance above 2). Finally, we applied our system to the whole of the two test pages. Evaluating the performance was made difficult because the OCR process may have deleted tokens and fragmented some. Table 3 shows counts of how many tokens have been preserved from the GS to the OCR-ed files as well as to the files corrected by the system (AC). To obtain counts, we compared files line by line as bag of words. Therefore, word order was not taken into account, but the line based comparison mitigated this effect for the text as a whole. Not surprisingly, accuracies were on average 10% lower than those from table

[9]One caveat: n is increased by 1 when the candidate's length is above four. The longer the word we are trying to correct, the more edit-operations necessary.

	Constr. model		Unc. model	
	E1	E2	E1	E2
$\|$GS$\|$	2153	2153	2153	2153
$\|$OCR$\|$	2322	2322	2322	2322
$\|$GS ∩ OCR$\|$	1185	1185	1185	1185
$\|$GS ∩ AC$\|$	1268	1263	1279	1234
Improvement	7%	7%	8%	4%

Table 3: Error reduction rate. | | = size of

2, which can be explained by the fact that not all targets from the test set can be found in the dictionary.

Two final remarks about the evaluation presented. That a token is part of the dictionary does not mean that it is correct. In fact, wrong substitutions constitute a very hard problem with OCR-ed texts and a source of contamination difficult to trace and fix. There is also the problem of misleading tokenization by missing or falsely inserting space characters, producing disrupted and continued tokens which cannot be corrected by comparing words one by one. Work such as Furrer (2013) is an attempt to improve post-correction of OCR-ed texts by using the internal structure of tokens to produce a tokenization scheme less sensitive to segmentation errors produced by the recognition system.

6 Conclusion

The approach we presented to correct OCR errors considered four features of two types: edit-distance and n-grams frequencies. Results show that a simple scoring system can correct OCR-ed texts with very high accuracy under idealized conditions: no more than two edit operations and a perfect dictionary. Obviously, these conditions do not always hold in practice, thus an observed error reduction rate drops to 10%. Nevertheless, we can expect to improve our dictionary coverage so that very noisy OCR-ed texts (i.e. 48% error with distance of at least three to target) can be corrected with accuracies up to 20%. OCR-ed texts with less challenging error patterns can be corrected with accuracies up to 61% (distance 2) and 86% (distance 1).

Acknowledgments

We are grateful to Katalin Szabò from Teßmann for producing the Gold Standard (GS).

References

Ahmad Abdulkader and Mathew R. Casey. 2009. *Low Cost Correction of OCR Errors Using Learning in a Multi-Engine Environment*. In Proceedings of the 2009 10th International Conference on Document Analysis and Recognition, pages 576–580, Washington, DC, USA.

Andreas W. Hauser. 2007. *OCR Postcorrection of Historical Texts*. Master Thesis, Ludwig-Maximilians-Universität München.

Gertrud Faaß and Kerstin Eckart. 2013. *Sdewac - a corpus of parsable sentences from the web*. In Gurevych, Biemann and Zesch, editors, GSCL, volume 8105 of Lecture Notes in Computer Science, pages 61–68. Springer.

Eric Brill and Robert C. Moore. 2000. *An improved error model for noisy channel spelling correction*. In ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pages 286–293, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

Kai Niklas. 2010. *Unsupervised Post-Correction of OCR Errors*. PhD Thesis, Leibniz Universität Hannover.

Lenz Furrer. 2013. *Unsupervised Text Segmentation for Correcting OCR Errors*. Master Thesis, Universität Zürich, July 2013.

Martin Reynaert. 2008. *Non-interactive ocr post-correction for giga-scale digitization projects*. In Proceedings of the 9th international conference on Computational linguistics and intelligent text processing, CICLing'08, pages 617–630.

Martin Volk, Lenz Furrer and Rico Sennrich. 2011. *Strategies for reducing and correcting OCR error*. In Language Technology for Cultural Heritage, pages 3–22, Sporleder, Caroline and Bosch, Antal van den and Zervanou, Kalliopi, ISBN 978-3-642-20226-1.

Massimo Poesio, Eduard Barbu, Egon W. Stemle and Christian Girardi. 2011. *Natural Language Processing for Historical Texts*. In Proc. 5th ACL-HLT Work. Lang. Technol. Cult. Heritage, Soc. Sci. Humanit. (LaTeCH 2011), pages 54–62, Portland, OR, USA. Association for Computational Linguistics.

Michael Piotrowski. 2012. *Natural Language Processing for Historical Texts*. Leibniz Institute of European History, Synthesis Lectures on Human Language Technologies 17, Morgan and Claypool Publishers.

Toby Segaran and Jeff Hammerbacher. 2009. *Beautiful Data: The Stories Behind Elegant Data Solutions*. Theory in practice. O'Reilly Media.

Xion Tong and David A. Evans. 1996. *A Statistical Approach to Automatic OCR Error Correction In Context*. In Proceedings of the Fourth Workshop on Very Large Corpora (WVLC-4), pages 88–100.