Multi-modular domain-tailored OCR post-correction

Sarah Schulz and **Jonas Kuhn**
Institute for Natural Language Processing (IMS)
University of Stuttgart
firstname.lastname@ims.uni-stuttgart.de

Abstract

One of the main obstacles for many Digital Humanities projects is the low data availability. Texts have to be digitized in an expensive and time consuming process whereas Optical Character Recognition (OCR) post-correction is one of the time-critical factors. At the example of OCR post-correction, we show the adaptation of a generic system to solve a specific problem with little data. The system accounts for a diversity of errors encountered in OCRed texts coming from different time periods in the domain of literature. We show that the combination of different approaches, such as e.g. Statistical Machine Translation and spell checking, with the help of a ranking mechanism tremendously improves over single-handed approaches. Since we consider the accessibility of the resulting tool as a crucial part of Digital Humanities collaborations, we describe the workflow we suggest for efficient text recognition and subsequent automatic and manual post-correction.

1 Introduction

Humanities are no longer just the realm of scholars turning pages of thick books. As the worlds of humanists and computer scientists begin to intertwine, new methods to revisit known ground emerge and options to widen the scope of research questions are available. Moreover, the nature of language encountered in such research attracts the attention of the NLP community (Kao and Jurafsky (2015), Milli and Bamman (2016)). Yet, the basic requirement for the successful implementation of such projects often poses a stumbling block: large digital corpora comprising the textual material of interest are rare. Archives and individual scholars are in the process of improving this situation by applying Optical Character Recognition (OCR) to the physical resources. In the *Google Books*[1] project books are being digitized on a large scale. But even though collections of literary texts like *Project Gutenberg*[2] exist, these collections often lack the texts of interest to a specific question. As an example, we describe the compilation of a corpus of adaptations of Goethe's *Sorrows of the young Werther* which allows for the analysis of character networks throughout the publishing history of this work.

The success of OCR is highly dependent on the quality of the printed source text. Recognition errors, in turn, impact results of computer-aided research (Strange et al., 2014). Especially for older books set in hard-to-read fonts and with stained paper the output of OCR systems is not good enough to serve as a basis for Digital Humanities (DH) research. It needs to be post-corrected in a time-consuming and cost-intensive process.

We describe how we support and facilitate the manual post-correction process with the help of informed automatic post-correction. To account for the problem of relative data sparsity, we illustrate how a generic architecture agnostic to a specific domain can be adjusted to text specificities such as genre and font characteristics by including just small amounts of domain specific data. We suggest a system architecture (cf. Figure 1) with trainable modules which joins general and specific problem solving as required in many applications. We show that the combination of modules via a ranking algorithm yields results far above the performance of single approaches.

[1] https://books.google.de/, 02.04.2017.
[2] http://www.gutenberg.org, 14.04.2017.

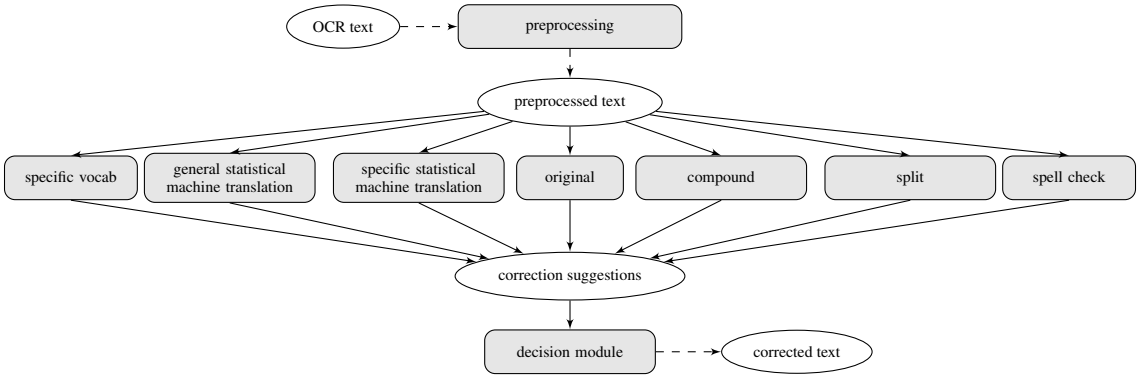


Figure 1: Multi-modular OCR post-correction system.

We discuss the point of departure for our research in Section 2 and introduce the data we base our system on in Section 4. In Section 5, we illustrate the most common errors and motivate our multi-modular, partly customized architecture. Section 6 gives an overview of techniques included in our system and the ranking algorithm. In Section 7, we discuss results, the limitations of automatic post-correction and the influence the amount of training data takes on the performance of such a system. Finally, Section 8 describes a way to efficiently integrate the results of our research into a digitization work-flow as we see the easy accessibility of computer aid as a central point in Digital Humanities collaborations.

2 Related work

There are two obvious ways to automatically improve quality of digitized text: optimization of OCR systems or automatic post-correction. Commonly, OCR utilizes just basic linguistic knowledge like character set of a language or reading direction. The focus lies on the image recognition aspect which is often done with artificial neural networks (cf. Graves et al. (2009), Desai (2010)). Post-correction is focused on the correction of errors in the linguistic context. It thus allows for the purposeful inclusion of knowledge about the text at hand, e.g. genre-specific vocabulary. Nevertheless, post-correction has predominantly been tackled OCR system agnostic as outlined below. As an advantage, post-correction can also be applied when no scan or physical resource is available.

There have been attempts towards shared datasets for evaluation. Mihov et al. (2005) released a corpus covering four different kinds of OCRed text comprising German and Bulgarian. However, in 2017 the corpus was untraceable and no recent research relating to the data could be found.

OCR post-correction is applied in a diversity of fields in order to compile high-quality datasets. This is not merely reflected in the homogeneity of techniques but in the metric of evaluation as well. While accuracy has been widely used as evaluation measure in OCR post-correction research, Reynaert (2008a) advocates the use of precision and recall in order to improve transparency in evaluations. Dependent on the paradigm of the applied technique even evaluation measures like BLEU score can be found (cf. Afli et al. (2016)).

Since shared tasks are a good opportunity to establish certain standards and facilitate the comparability of techniques, the Competition on Post-OCR Text Correction[3] organized in the context of ICDAR 2017 could mark a milestone for more unified OCR post-correction research efforts.

Regarding techniques used for OCR post-correction, there are two main trends to be mentioned: statistical approaches utilizing error distributions inferred from training data and lexical approaches oriented towards the comparison of source words to a canonical form. Combinations of the two approaches are also available.

Techniques residing in this **statistical** domain have the advantage that they can model specific distributions of the target domain if training data is available. Tong and Evans (1996) approach post-correction as a statistical language modeling problem, taking context into account. Pérez-Cortes et al. (2000) employ stochastic finite-state automaton along with a modified version of the Viterbi Algorithm to perform a stochastic error correcting parsing. Extending the simpler stochastic context-sensitive models, Kolak and

[3] https://sites.google.com/view/icdar2017-postcorrectionocr/home, 3.07.2017.

Resnik (2002) apply the first noisy channel model, using edit distance from noisy to corrected text on character level. In order to train such a model, manually generated training data is required. Reynaert (2008b) suggests a corpus-based correction method, taking spelling variation (especially in historical text) into account. Abdulkader and Casey (2009) introduce an error estimator neural network that learns to assess error probabilities from ground truth data which in turn is then suggested for manual correction. This decreases the time needed for manual post-correction since correct words do not have to be considered as candidates for correction by the human corrector. Llobet et al. (2010) combine information from the OCR system output, the error distribution and the language as weighted finite-state transducers. Reffle and Ringlstetter (2013) use global as well as local error information to be able to fine-tune post-correction systems to historical documents. Related to the approach introduced by Pérez-Cortes et al. (2000), Afli et al. (2016) use statistical machine translation for error correction using the Moses toolkit on character level. Volk et al. (2010) merge the output of two OCR systems with the help of a language model to increase the quality of OCR text. The corpus of yearbooks of the Swiss Alpine Club which has been manually corrected via crowdsourcing (cf. Clematide et al. (2016)) is available from their website.

Lexical approaches often use rather generic distance measures between an erroneous word and a potential canonical lexical item. Strohmaier et al. (2003) investigate the influence of the coverage of a lexicon on the post-correction task. Considering the fact that writing in historical documents is often not standardized, the success of such approaches is limited. Moreover, systems based on lexicons rely on the availability of such resources. Historical stages of a language – which constitute the majority of texts in need for OCR post-correction – often lack such resources or provide incomplete lexicons which would drastically decrease performance of spell-checking-based systems. Ringlstetter et al. (2007) address this problem by suggesting a way to dynamically collect specialized lexicons for this task. Takahashi et al. (1990) apply spelling correction with preceding candidate word detection. Bassil and Alwani (2012) use Google's online spelling suggestions for as they draw on a huge lexicon based on contents gathered from all over the web.

The **human component** as final authority has been mentioned in some of these projects. Visual support of the post-correction process has been emphasized by e.g. Vobl et al. (2014) who describe a system of iterative post-correction of OCRed historical text which is evaluated in an application-oriented way. They present the human corrector with an alignment of image and OCRed text and make batch correction of the same error in the entire document possible. They can show that the time needed by human correctors considerably decreases.

3 Evaluation metrics

We describe and evaluate our data by means of word error rate (WER) and character error rate (CER). The error rates are a commonly used metric in speech recognition and machine translation evaluation and can also be referred to as length normalized edit distance. They quantify the number of operations, namely the number of insertions, deletions and substitutions, that are needed to transform the suggested string into the manually corrected string and are computed as follows:

$$\text{WER} = \frac{\text{word insertions} + \text{word substitutions} + \text{word deletions}}{\text{\# words in the reference}}$$

$$\text{CER} = \frac{\text{char insertions} + \text{char substitutions} + \text{char deletions}}{\text{\# characters in the reference}}$$

4 Data

As mentioned in the introduction, errors found in OCRed texts are specific to time of origin, quality of scan and even the characteristics of a specific text. Our multi-modular architecture paves the way for a solution taking this into account by including general as well as specific modules. Thus, we suggest to include domain specific data as well as larger, more generic data sets in order to enhance coverage of vocabulary and possible error classes. The data described hereafter constitutes parallel corpora with OCR output and manually corrected text which we utilize for training statistical models.

4.1 The Werther corpus

Since our system is developed to help in the process of compiling a corpus comprising adaptations of Goethe's *The Sorrows Of Young Werther* throughout different text types and centuries, we

1	Berichtigung der Geschichte des jungen Werthers	H. von Breitenbach	1775
2	Schwacher jedoch wohlgemeynter Tritt vor dem Riss, neben oder hinter Herren Pastor Goeze, gegen die Leiden des jungen Werthers und dessen ruchlose Anhänger	anonymous	1775
3	Lorenz Konau	David Iversen	1776
4	Werther der Jude	Ludwig Jacobowski	1910
5	Eine rührende Erzählung aus geheimen Nachrichten von Venedig und Cadir (first letter)	Joseph Codardo und Rosaura Bianki	1778
6	Afterwerther oder Folgen jugendlicher Eifersucht	A. Henselt	1784
7	Der neue Werther oder Gefühl und Liebe	Karl P. Bonafont	1804
8	Leiden des modernen Werther	Max Kaufmann	1901

Table 1: Werther texts included in our corpus from different authors and times of origin.

collected texts from this target domain. To be able to train a specialized system, we manually corrected a small corpus of relevant texts (cf. Table 2). We use the output of Abbyy Fine Reader 7 for several Werther adaptations (Table 1) all based on scans of books with German Gothic lettering.

4.2 The Deutsches Textarchive (DTA) corpus

Even though manual OCR post-correction is a vital part of many projects, only very little detailed documentation of this process exists. *Das Deutsche Textarchiv* (The German Text Archive) (DTA) is one of the few projects providing detailed correction guidelines along with the scans and the text corrected within the project (Geyken et al., 2012). This allows the compilation of a comprehensive parallel corpus of OCR output and corrected text spanning a period of four centuries (17th to 20th) in German Gothic lettering. For OCR, we use the open source software *tesseract*[4] (Smith and Inc, 2007) which comes with recognition models for Gothic font.

4.3 Gutenberg data for language modeling

Since the output of our system is supposed to consist of well-formed German sentences, we need a method to assess the quality of the output language. This task is generally tackled by language modeling. We compiled a collection of 500 randomly chosen texts from Project Gutenberg[5] comprising 28,528,078 tokens. With its relative closeness to our target domain it constitutes the best approximation of a target language. The language model is trained with the KenLM toolkit (Heafield, 2011) with an order of 5 on token level and 10 on character level following De Clercq et al. (2013).

[4]Considering the open source aspect of our resulting system, we decided to use the open source OCR software tesseract and move away from Abbyy some time after our project started: https://github.com/tesseract-ocr.

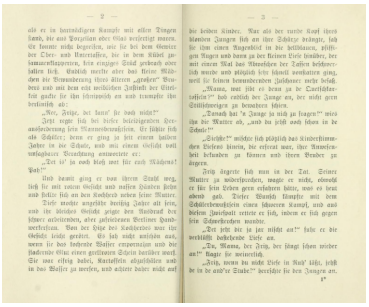
[5]Project Gutenberg. Retrieved January 21, 2017, from www.gutenberg.org.

5 Why OCR post-correction is hard

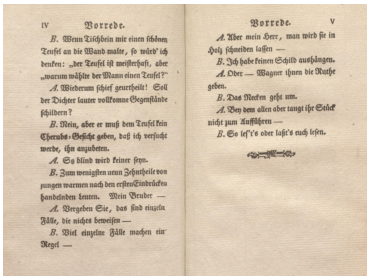
In tasks like the normalization of historical text (Bollmann et al., 2012) or social media, one can take advantage of regularities in the deviations from the standard form that appear throughout an entire genre or in case of social media e.g. dialect region (Eisenstein, 2013). Errors in OCR, however, depend on the font and quality of the scan as well as the time of origin which makes each text unique in its composition of features and errors.

In order to exemplify this claim, we analyzed three different samples: *Lorenz Konau* (1776), *Werther der Jude* (1910) and a sample from the DTA data. Figure 2 (a-c) illustrate the point that the quality of scan is crucial for the OCR success. Figure 2a shows a text from the 20th century where the type setting is rather regular and the distances between letters is uniform as opposed to Figure 2b. Figure 2c shows how the writing from the back of the page shines through and makes the script less readable. Thus, we observe a divergence in the frequency of certain character operations between those texts: the percentage of substitutions range between 74% for *Lorenz Konau* and 60% for *Werther der Jude* and 18% and 30% of insertions, respectively. The varying percentage of insertions might be due to the fact that some scans are more "washed out" than others. Successful insertion of missing characters, however, relies on the precondition that a system knows a lot of actual words and sentences in the respective language and cannot be resolved via e.g. character similarity like in the substitution from *l* to *t*.

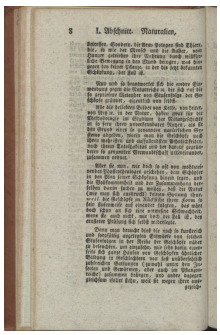
Another factor that complicates the correction of a specific text is the number of errors per word. Words with an edit distance of one to the correct version are easier to correct those with more than one necessary operation. With respect to errors per word our corpus shows significant differences in error distributions. Especially in our DTA corpus the number of words with two or more character-level errors per word is considerably higher than those with one error. For *Werther*



(a) Werther der Jude (1910)



(b) Lorenz Konau (1776)



(c) DTA: Blumenbach (1791): Handbuch der Naturgeschichte

Figure 2: Scans of three different texts from our corpora. Emphasizes differences in quality of scan and differences in type setting, font and genre (e.g. drama).

der Jude (WER 10.0, CER 2.4) the number of errors in general is much lower than for *Konau* (WER: 34.7, CER: 10.9). These characteristics indicate that subcorpus-specific training of a system is promising.

6 Specialized multi-modular post-correction

In order to account for the nature of errors that can occur in OCR text, we apply a variety of modules for post-correction. The system proceeds in two stages and is largely based on an architecture suggested by Schulz et al. (2016) for normalization of user-generated contents. In the first stage, a set of specialized modules (Section 6.1) suggest corrected versions for the tokenized[6] OCR text lines. Those modules can be context-independent (work on just one word at a time) or context-dependent (an entire text line is processed at a time). The second stage is the decision phase. After the collection of various suggestions per input token, these have to be ranked to enable a decision for the most probable output token given the context. We achieve this by assigning weights the different modules with the help of Minimal Error Rate Training (MERT) (Och, 2003).

6.1 Suggestion modules

In the following, we give an outline of techniques included into our system.

6.1.1 Word level suggestion modules

- **Original**: the majority of words do not contain any kind of error, thus we want to have the initial token available in our suggestion pool
- **Spell checker**: spelling correction suggestion for misspelled words with hunspell[7]
- **Compounder**: merges two tokens into one token if it is evaluated as an existing word by hunspell
- **Word splitter**: splits two tokens into two words using compound-splitter module from the Moses toolkit (Koehn et al., 2007)
- **Text-Internal Vocabulary**: extracts high-frequent words from the input texts and suggests them as correction of words with small adjusted Levenshtein distance[8]

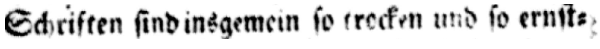


Figure 3: Irregular type setting in German Gothic lettering. *sind* and *insgemein* are two separate words but yet written closely together.

The compound and word split techniques react to the variance in manual typesetting, where the distances between letters vary. This means that the word boundary recognition becomes difficult (cf. Figure 3).

A problem related to the spell-checking approach is the limited coverage of the dictionary since it uses a modern German lexicon. Related to this is the difficulty of out-of-vocabulary words above average for literature text. Archaic words from e.g. the 17th century or named entities cannot be found in a dictionary and can therefore not be covered with any of the approaches mentioned above.

[6] Tokenizer of TreeTagger (Schmid, 1997).

[7] https://github.com/hunspell/hunspell.

[8] OCR-adjusted Levenshtein distance taking frequent substitution, insertion and deletion patterns learned from training data into account.

However, especially named entities are crucial for the automatic or semi-automatic analysis of narratives e.g. with the help of network analysis. Our Text-Internal Vocabulary technique is designed to find frequent words in the input text, following the assumption that errors would not be regular enough to distort those frequencies. We compile a list from those high-frequency words. Subsequently, erroneous words can be corrected calculating an OCR-adjusted Levenshtein distance. In this way misspelled words like *Loveuzo* could be resolved to *Lorenzo* if this name appears frequently. Since the ranking algorithm relies on a language model which will most probable not contain those suggestions, we insert the high-frequency words into the language modeling step.

6.1.2 Sentence level suggestion modules

As has been suggested by Afli et al. (2016), we include Phrase-based **Statistical Machine Translation** (SMT) into our system. We treat the post-correction as a translation problem translating from erroneous to correct text. Like in standard SMT, we train our models on a parallel corpus, the source language being the OCRed text and the target language being manually corrected text. We train models on token level as well as on character-level (unigram). This way, we aim at correcting frequently mis-recognized words along with frequent character-level errors. We train four different systems:

- token level
  - domain specific data (cf. Section 4.1)
  - general data (cf. Section 4.2)
- character level
  - domain specific data (cf. Section 4.1)
  - general data (cf. Section 4.2)

The models are trained with the Moses toolkit (Koehn et al., 2007). Moreover, we use a subsequent approach by forwarding the output of the character-based SMT model to the token-based SMT.

6.1.3 Additional feature

The information whether a word contains an error can help to avoid the incorrect alternation of an initially correct word (overcorrection). In order to deliver this information to the decision module without making a hard choice for each word, we include the information whether a word has been found either in combination with the word before or after in a corpus (cf. Section 4.3) into the decision process in form of a feature that will be weighted along with the other modules. This naive language modeling approach allows for a context-relevant decision of the correctness of a word.

6.2 Decision modules: the ranking mechanism

Since the recognition errors appearing in a text are hard to pre-classify by nature, we run all modules on each sentence of the input, returning suggestions for each word. Since the output of some of our modules are entire sentences, input sentence and output sentence have to be word-aligned in order to be able to make suggestions on word level. The word alignment between input and output sentence is done with the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970), an algorithm originally developed in bioinformatics.
From all corrected suggestions the most probable well-formed combination has to be chosen. To solve the combinatorial problem of deciding which suggestion is the most probable candidate for a word, the decision module makes use of the Moses decoder.
As in general SMT, the decoder makes use of a language model (cf. Section 4.3) and a phrase table. The phrase table is compiled from all input words along with all possible correction suggestions. In order to assign weights to the single modules and the language model, we tune on the phrase tables collected from a run on our $\text{dev}_{overall}$ set, following the assumption that suggestions of certain modules are more reliable than others and expect their feature weights to be higher after tuning.

7 Experiments

7.1 Experimental Setup

To guarantee diversity, we split each of texts 1-4 (cf. Table 1) into three parts and combined the respective parts: 80% train (train), 10% development (dev_{SMT}) and 10% test (test_{init}).

Test setup We introduce two different test scenarios. Even though both test sets are naturally compiled from unseen data, the first test set consists of a self-contained Werther adaptation introducing new named entities, originating from a different source and thus showing a different error

set	# tokens (OCR)	# tokens (corr)	WER	CER
train	70,159	68,608	15.7	5.5
train$_{ext}$	133,457	131,901	12.9	4.0
dev$_{SMT}$	12,464	12,304	13.9	3.5
dev$_{overall}$	13,663	13,396	16.75	4.6
test$_{init}$	17,443	17,367	9.4	2.5
test$_{unk}$	13,286	13,304	31.2	9.2

Table 2: Werther specific parallel corpus of OCR text and corrected text showing the number of tokens before and after post-correction along with WER and CER

set	# tokens (OCR)	# tokens (corr)	WER	CER
train	3,452,922	3,718,712	41.6	13.2
dev	663,376	836,974	30.4	9.1

Table 3: DTA parallel corpus of OCR text and corrected text showing the number of tokens before and after post-correction along with WER and CER

constitution. It constitutes an evaluation in which no initial manual correction as support for the automatic correction is included in the workflow. We henceforth call this unknown set *test$_{unk}$* (text 6).

In contrast, the second set contains parts of the same texts as the training, thus specific vocabulary might have been introduced already. The results for this test set give a first indication of the extent to which pre-informing the system with manually correcting parts of a text could assist the automatic correction process. Since this scenario can be described as a text-specific initiated post-correction, we henceforth refer to this test set as *test$_{init}$*.

We further on experiment with an extended training set train$_{ext}$ (train with texts 7 and 8) to assess the influence of the size of the specific training set on the overall performance. The sizes of the datasets before and after correction along with WER and CER are summarized in Table 2. The sizes for the general dataset before and after correction along with WER and CER are summarized in Table 3.

7.2 Evaluation

In the following we concentrate on the comparison of WER and CER before and after automatic post-correction. As a baseline for our system we chose the strongest single-handed module (SMT on character-level trained on Werther data).

training set	system	test$_{init}$ WER	test$_{init}$ CER	test$_{unk}$ WER	test$_{unk}$ CER
	original text	23.5	15.1	36.7	30.0
train	baseline	22.0	13.2	26.6	26.3
	overall system	**4.7**	**8.0**	**15.4**	**19.6**
train$_{ext}$	baseline	21.1	11.7	24.0	20.4
	overall system	**4.4**	**7.2**	**15.2**	**16.4**

Table 4: WER and CER for both test sets before and after automatic post-correction for the system trained with the small training set (train) and the larger training set (train$_{ext}$). Baselines: the original text coming from the OCR system and the character-level SMT system trained on the Werther data.

Overall performance As indicated previously, our test sets differ with respect to their similarity to the training set. The results for both test scenarios for systems trained on our two training sets are summarized in Table 4. The results from test$_{init}$ and test$_{unk}$ show that our system performs considerably better than the baseline and can improve quality of the OCR output considerably.

For test$_{unk}$, the system improves the quality by almost 20 points of WER from 36.7 to 15.4 and over 10 points in CER from 30.0 to 19.6. For test$_{init}$, our system improves the quality of the text with a reduction of approximately 20 points of WER from 23.5 to 4.7 and 7 points in CER from 15.1 to 8.0. It is not surprising that the decrease in WER is stronger than the decrease in CER. This is due to the fact that many words contain more than one error and require more than one character level operation to get from the incorrect to the correct string.

Just slight improvement can be shown by adding training material to the Werther-specific parts of the system (cf. train$_{ext}$ row of Table 4). Merely the CER can be improved whereas the WER stays about the same. The improvement in test$_{unk}$ is higher than for test$_{init}$.

Module specific analysis Since a WER and CER evaluation is not expedient for all modules as they were designed to correct specific problems and not the entirety of them, we look into the specialized modules in terms of correct suggestions contributed to the suggestion pool and correct suggestions only suggested by one module (unique suggestions). As the system including the extended training set train$_{ext}$ delivered slightly better results, in the following we will describe the contribution of the single modules

module	test$_{init}$ # overcorrected	# corrected	# unique correct	test$_{unk}$ # overcorrected	# corrected	# unique correct
SMT Werther token	128	364	10	209	1,089	0
SMT Werther character	235	684	0	700	1,919	0
SMT Werther cascaded	273	697	2	728	1,933	4
SMT DTA token	2,179	229	8	1,627	893	19
SMT DTA character	4121	372	22	3,143	1,530	115
text-internal vocab	3,317	131	16	4,142	244	60
word split	594	3	0	720	45	2
spell check	1,329	219	15	2,819	731	40
compound	222	0	0	169	2	2
overall system	238	2171	-	675	2,642	-

Table 5: Number of overcorrected, corrected and uniquely corrected words per module out of 17,367 tokens in test$_{init}$ (2,726 erroneous words) and 13,304 tokens in test$_{unk}$ (4,141 erroneous words)

to the overall performance of this system (cf. Table 5). For test$_{unk}$ the number of corrected tokens along with the number of overcorrections is higher than for test$_{init}$ throughout all modules. Clearly, for test$_{init}$ the Werther-specific modules are strongest. The more general modules prove useful for test$_{unk}$. The number of corrected words increases for the SMT module trained on DTA data on character-level. The usefulness of the module extracting specific words (text-internal vocab) as well as the general SMT model and the spell checker becomes evident in terms of unique suggestions contributed by those modules.

The analysis of the output of the individual modules and their contribution to the overall system uncovers an issue: those modules that produce a high number of incorrect suggestions, thus overcorrecting actually correct input tokens, are at the same time those modules that are the only ones producing correct suggestions for some of the incorrect input words. Consequently, those uniquely suggested corrections are not chosen in the decision modules due to an overall weak performance of this module. These suggestions are often crucial to the texts like the suggestions by the special vocabulary module which contain named entities or words specific to the time period. For our test$_{unk}$ set, the text-internal vocabulary module yields around 60 unique suggestions, out of which 15 are names (Friedrich, Amalia) or words really specific to the text (*Auftrit* spelled with one t instead of two).

Challenges In the context of literature OCR post-correction is a challenging problem since the texts themselves can be considered *non-standard text*. The aim is not to bring the text at hand to an agreed upon standard form but to digitize exactly what was contained in the print version. This can be far from the standard form of a language. In one of our texts, we find a character speaking German with a strong dialect. Her speech contains a lot of words that are incorrect in standard German, however, the goal is it to preserve this "errors" in the digital version. Thus, correction merely on the basis of the OCR text without consulting the printed version or an image-digitized facsimile, can essentially never be perfect. It follows, that the integration of automatic post-correction techniques into the character recognition process could lead to further improvements.

7.3 Adaptability

Reusability as a key concept in NLP for DH originates in the time limitations given in such projects. Since DH projects do not evolve around the development of tools but the analysis performed with the help this tools in order to answer a specific question, the tools are expected to be delivered in an early phase of collaborative projects. From-scratch development easily exceeds this time limits. We show that our OCR post-correction system is modular enough to be adjusted to correct texts from other languages by training it for two other languages, English and French, with data released in the OCR post-correction competition organized in the context of ICDAR 2017[9]. The texts originate from the the last four centuries and come from different collections and therefore have been digitized using different OCR systems. The data is summarized in Table 6[10].

We adjust our system to the language by retraining the SMT models and including spell-checkers for the respective languages. Due to the modular architecture these adjustments can be made eas-

[9] https://sites.google.com/view/icdar2017-postcorrectionocr/home, 3.07.2017.

[10] The test set does not comply with the official shared task set since the manually corrected data is not yet available for the test set. We test on a combination of periodicals and monographs.

language	$train_{ocr}$	$train_{gold}$	$dev1_{ocr}$	$dev1_{gold}$	$dev2_{ocr}$	$dev2_{gold}$	$test_{ocr}$	$test_{gold}$
English	309,080	282,738	71,049	65,480	13,000	11,966	14,302	12,859
French	805,438	783,371	167,473	163,373	9,566	9,216	12,289	11,780

Table 6: Number of tokens in the English and French corpus provided by the competition on OCR-postcorrection.

ily and with a low expenditure of time. Since the datasets are compilation of a variety of texts, we use all modules but the domain-specific SMT models. We solely include one token-level and character-level SMT module for each language.

language	system	WER	CER
English	original	29.4	28.4
	SMT Cascaded	22.7	23.6
	overall system	22.1	24.5
French	original text	13.3	25.0
	SMT Cascaded	9.9	20.0
	overall system	8.7	21.5

Table 7: The results reported in word error rate (WER) and character error rate (CER) for the English and French test set.

The strongest unique module for these two languages is the subsequent combination of the character-level SMT and the token-level SMT models (Cascaded). For English it performs just slightly worse on WER and even outperforms the overall system on the CER. For French, the overall system is clearly stronger than the Cascaded SMT system with more than 1 percent improvement of WER but also performs worse in terms of CER by 1.5 percent. Generally, the OCR post-correction system achieves about 25% reduction of WER for English and over 30% reduction in WER for French.

8 Digitization workflow

We integrate the automatic OCR process with tesseract and our automatic post-correction system into a workflow which results in an hocr file, an XML format which is readable by PoCoTo (Vobl et al., 2014) a tool for supporting manual post-correction of OCRed text through alignment of image and digitized text. The upload of scans or images is provided online via a webapplication[11]. This shields the user from the technicalities of the correction process and provides them with the input for the PoCoTo tool.

The implementation of an easy-to-handle workflow is an often underemphasized aspect of DH. It needs to be intuitive enough to not absorb the time ion has been saved via automation. Since the final post-correction step requires that the human corrector compares the digitized version with the scan, presenting both next to each other is an ideal scenario. This functionality is one of the main strengths of PoCoTo, a visual correction tool, supporting manually initiated correction operations and batch correction of the same error.

9 Conclusion

We can show that the enhancement of a general, adaptable architecture by including small but specific data sets can improve results within a specific domain. Moreover, the combination of different techniques for of OCR post-correction is significantly superior to single techniques. Especially the integration of SMT models on token level and character level contributes to the overall success of the system. Due to the complexity of OCR post-correction, there cannot be a general solution. Even though the ranking algorithm achieves large improvement, further potential lies in the inclusion of fine-tuned language models since the decision process highly depends upon it. The intrinsic characteristic of literature as being *non-standard* complicates the task. However, techniques that focus on these features like our module that is specialized on extracting text-specific vocabulary show promising results for e.g. named entity correction.

10 Acknowledgements

We thank the German Ministry of Education and Research (BMBF) for supporting this research completed within the Center for Reflected Text Analytics (CRETA). Furthermore, we acknowledge the support of our colleagues at Deutsches Textarchiv.

[11] http://clarin05.ims.uni-stuttgart.de/ocr/, for access please contact the author.

References

Ahmad Abdulkader and Mathew R. Casey. 2009. Low Cost Correction of OCR Errors Using Learning in a Multi-Engine Environment. In *10th International Conference on Document Analysis and Recognition, ICDAR 2009, Barcelona, Spain, 26-29 July 2009*, pages 576–580.

Haithem Afli, Zhengwei Qiu, Andy Way, and Praic Sheridan. 2016. Using SMT for OCR Error Correction of Historical Texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Youssef Bassil and Mohammad Alwani. 2012. OCR Post-Processing Error Correction Algorithm Using Google's Online Spelling Suggestion. *Journal of Emerging Trends in Computing and Information Sciences*, 3(1):90–99.

Marcel Bollmann, Julia Krasselt, and Florian Petran. 2012. Manual and semi-automatic normalization of historical spelling Case studies from Early New High German. In *Proceedings of KONVENS 2012 (LThist 2012 workshop*, pages 342–350.

Simon Clematide, Lenz Furrer, and Martin Volk. 2016. Crowdsourcing an OCR Gold Standard for a German and French Heritage Corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Orphée De Clercq, Bart Desmet, Sarah Schulz, Els Lefever, and Veronique Hoste. 2013. Normalization of Dutch user-generated content. In *Proceedings of Recent Advances in Natural Language Processing*, pages 179–188. INCOMA.

Apurva A. Desai. 2010. Gujarati Handwritten Numeral Optical Character Reorganization Through Neural Network. *Pattern Recogn.*, 43(7):2582–2589.

Jacob Eisenstein. 2013. What to do about bad language on the Internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

Alexander Geyken, Susanne Haaf, Bryan Jurish, Matthias Schulz, Christian Thomas, and Frank Wiegand. 2012. TEI und Textkorpora: Fehlerklassifikation und Qualittskontrolle vor, whrend und nach der Texterfassung im Deutschen Textarchiv. In *Jahrbuch für Computerphilologie*, page online.

Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2009. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Justine T. Kao and Dan Jurafsky. 2015. A computational analysis of poetic style: Imagism and its influence on modern professional and amateur poetry. *Linguistic Issues in Language Technology*, 12(3):1–31.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Okan Kolak and Philip Resnik. 2002. OCR Error Correction Using a Noisy Channel Model. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 257–262, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Rafael Llobet, Jose-Ramon Cerdan-Navarro, Juan-Carlos Perez-Cortes, and Joaquim Arlandis. 2010. OCR Post-processing Using Weighted Finite-State Transducers. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pages 2021–2024, Washington, DC, USA. IEEE Computer Society.

Stoyan Mihov, Klaus U. Schulz, Christoph Ringlstetter, Veselka Dojchinova, and Vanja Nakova. 2005. A Corpus for Comparative Evaluation of OCR Software and Postcorrection Techniques. In *Eighth International Conference on Document Analysis and Recognition (ICDAR 2005), 29 August - 1 September 2005, Seoul, Korea*, pages 162–166.

Smitha Milli and David Bamman. 2016. Beyond Canonical Texts: A Computational Analysis of Fanfiction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2048–2053.

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Juan Carlos Pérez-Cortes, Juan-Carlos Amengual, Joaquim Arlandis, and Rafael Llobet. 2000.

Stochastic Error-Correcting Parsing for OCR Post-Processing. In *15th International Conference on Pattern Recognition, ICPR'00, Barcelona, Spain, September 3-8, 2000*, pages 4405–4408.

Ulrich Reffle and Christoph Ringlstetter. 2013. Unsupervised Profiling of OCRed Historical Documents. *Pattern Recogn.*, 46(5):1346–1357.

Martin Reynaert. 2008a. All, and only, the Errors: more Complete and Consistent Spelling and OCR-Error Correction Evaluation. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Martin Reynaert. 2008b. 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, Berlin, Heidelberg. Springer-Verlag.

Christoph Ringlstetter, Klaus U. Schulz, and Stoyan Mihov. 2007. Adaptive Text Correction with Web-crawled Domain-dependent Dictionaries. *ACM Trans. Speech Lang. Process.*, 4(4).

Helmut Schmid. 1997. Probabilistic Part-of-Speech Tagging Using Decision Trees. In Daniel Jones and Harold Somers, editors, *New Methods in Language Processing*, Studies in Computational Linguistics, pages 154–164. UCL Press, London, GB.

Sarah Schulz, Guy De Pauw, Orphée De Clercq, Bart Desmet, Véronique Hoste, Walter Daelemans, and Lieve Macken. 2016. Multimodular Text Normalization of Dutch User-Generated Content. *ACM Trans. Intell. Syst. Technol.*, 7(4):61:1–61:22.

Ray Smith and Google Inc. 2007. An overview of the Tesseract OCR Engine. In *Proc. 9th IEEE Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pages 629–633.

Carolyn Strange, Daniel McNamara, Josh Wodak, and Ian Wood. 2014. Mining for the Meanings of a Murder: The Impact of OCR Quality on the Use of Digitized Historical Newspapers. *Digital Humanities Quarterly*, 8(1).

Christian M. Strohmaier, Christoph Ringlstetter, Klaus U. Schulz, and Stoyan Mihov. 2003. Lexical Postcorrection of OCR-Results: The Web as a Dynamic Secondary Dictionary? In *7th International Conference on Document Analysis and Recognition (ICDAR 2003), 2-Volume Set, 3-6 August 2003, Edinburgh, Scotland, UK*, pages 1133–1137.

Hiroyasu Takahashi, Nobuyasu Itoh, Tomio Amano, and Akio Yamashita. 1990. A spelling correction method and its application to an OCR system. *Pattern Recognition*, 23(3-4):363–377.

Xiang Tong and David A. Evans. 1996. A Statistical Approach to Automatic OCR Error Correction in Context. In *Fourth Workshop on Very Large Corpora*, pages 88–100.

Thorsten Vobl, Annette Gotscharek, Uli Reffle, Christoph Ringlstetter, and Klaus U. Schulz. 2014. PoCoTo - an Open Source System for Efficient Interactive Postcorrection of OCRed Historical Texts. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, pages 57–61, New York, NY, USA. ACM.

Martin Volk, Torsten Marek, and Rico Sennrich. 2010. Reducing OCR errors by combining two OCR systems. In *Proceedings of the ECAI 2010 Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH 2010)*, pages 61–65.