Chantal Amrhein, Simon Clematide

# Supervised OCR Error Detection and Correction Using Statistical and Neural Machine Translation Methods

**Abstract**

For indexing the content of digitized historical texts, optical character recognition (OCR) errors are a hampering problem. To explore the effectivity of new strategies for OCR post-correction, this article focuses on methods of character-based machine translation, specifically neural machine translation and statistical machine translation.

Using the ICDAR 2017 data set on OCR post-correction for English and French, we experiment with different strategies for error detection and error correction. We analyze how OCR post-correction with NMT can profit from using additional information and show that SMT and NMT can benefit from each other for these tasks. An ensemble of our models reached best performance in ICDAR's 2017 error correction subtask and performed competitively in error detection.

However, our experimental results also suggest that tuning supervised learning for OCR post-correction of texts from different sources, text types (periodicals and monographs), time periods and languages is a difficult task: the data on which the MT systems are trained have a large influence on which methods and features work best. Conclusive and generally applicable insights are hard to achieve.

## 1 Introduction

Optical character recognition (OCR) is an important processing step for text digitization, especially with the growing interest in digital humanities. Unfortunately, the output of OCR systems for historical documents is often faulty due to both orthographic and typographic variation as well as due to poor condition of the source material (especially for newspapers). Therefore, it is necessary to take measures to improve the quality of existing OCR-generated text if re-OCRing of the image material is not an option. Recently natural language processing (NLP) tasks have profited from neural methods. Neural machine translation (NMT) outperformed statistical machine translation (SMT), thus, NMT now supersedes SMT in research and practice. Since string-to-string translation methods have been used to correct OCR errors for a long time, it is interesting to explore how character-based NMT can be employed for this task.

This paper focuses on two questions: How does character-based NMT perform compared to character-based SMT in the case of OCR post-correction? How can these approaches be improved by using more information during training and translation?

Note that our historical data we are working with represents the task of correcting OCR output into the spelling found in the original documents and does not include any modernization of the texts. The rest of this article is structured as follows: In the next section, we discuss previous work relevant to OCR post-correction, neural modeling of related NLP tasks and character-based MT. Section 3 describes the data used in our experiments and Section 4 presents the methods. Section 5 gives details on the results and discusses the major findings. In Section 6, we offer our ideas for future work and finally, we conclude in Section 7.

## 2 Related Work

### 2.1 OCR Post-Correction

Volk et al. (2011) discuss three possible ways to tackle the problem of OCR errors: Modifying the input images, altering the OCR system, or post-processing the output text. Since it does not involve re-OCRing, a considerable number of approaches have focused on the last option. Kukich (1992) discusses various automatic methods to find and correct errors, such as n-gram or dictionary-based techniques. Eger et al. (2016) compare traditional spelling error correction techniques with general string-to-string translation methods, and evaluate on an OCR data set. They show that the latter methods achieve significantly better results than (weighted) edit distance or the noisy channel model (Brill and Moore, 2000).

Based on the idea that OCR post-correction can be cast as a translation task, Afli et al. (2016) have successfully trained an SMT system to translate historical French texts with OCR errors into corrected text. Their model outperforms language model based techniques. However, they use a large training set of more than 60 million tokens. This exceeds by far the size of the ICDAR training sets used in our experiments. In previous experiments, Afli et al. concluded that word-level SMT systems perform slightly better than character-level systems for OCR post-correction (Afli et al., 2015). The size of the training set there was over 90 million tokens.

### 2.2 OCR Correction of Historical Texts

The introductory book by Piotrowski (2012) resumes the problems of OCR, historical spelling and correction. The TICCL system by Reynaert (2016) is an unsupervised corpus-based approach to OCR post-correction that has been developed over many years and works for several languages. However, it requires high-quality lexicons in order to be effective. Silfverberg et al. (2016) present an interesting supervised approach for historical Finnish OCR correction of isolated words that formulates the problem as a sequence labeling task and uses weighted finite-state transducer techniques to implement it. The official ICDAR OCR post-correction paper (Chiron et al., 2017) contains concise descriptions of the different approaches that the shared task participants used for their solutions.

The recent paper of Schulz and Kuhn (2017) presents a complex architecture for OCR post-correction of historical texts that includes token and character-level SMT as well as specialized tools for merging and splitting of erroneous tokens. Their related work section gives a good overview on older and newer approaches. Due to space restrictions and our focus on experimenting and evaluation, we refer the interested reader to their discussion.

### 2.3 Neural Networks for NLP Tasks

Recently, neural networks have been given much attention in the field of computational linguistics. Especially the work of Sutskever et al. (2014) has prompted much research which focused on employing sequence-to-sequence (seq2seq) neural networks in various NLP tasks. Examples include effectively using neural networks for transliteration (Rosca and Breuel, 2016), for grapheme-to-phoneme conversion (Yao and Zweig, 2015), for historical spelling normalisation (Bollmann and Søgaard, 2016) and language correction of second language learners (Xie et al., 2016).

All of these tasks can (at least partly) be viewed as string-to-string translation problems and are, thus, closely related to OCR post-correction. The models were all successful, performing at least equally but mostly better than traditional NLP methods such as methods based on (weighted) edit distance (Xie et al., 2016).

However, Schnober et al. (2016) express their doubt whether neural networks are already at the point where they can entirely replace traditional approaches. They evaluated the performance of encoder-decoder neural networks against established methods for spelling correction, OCR post-correction, grapheme-to-phoneme conversion and lemmatisation. Some of the string-to-string translation systems they used as baselines were also evaluated by Eger et al. (2016). It is particularly interesting to see how neural network approaches performed compared to these models which were previously shown to exceed (weighted) edit distance and other techniques.

In the experiments of Schnober et al. on OCR post-correction, the neural network systems did not manage to outperform a system built with Pruned Conditional Random Fields on a training set of 72,000 misrecognized words. Despite these negative results, the work gives valuable insight into model selection for neural network approaches. Attention-based models (Bahdanau et al., 2015) show significant improvements over plain seq2seq models. On a smaller training set, the attention-based models also performed better than the Pruned Conditional Random Fields. In contrast to the experiments conducted by Schnober et al., our data does not only contain misrecognized words. Therefore, the task becomes harder since the systems also need to detect erroneous words before trying to correct them.

#### 2.3.1 Character-based NMT

NMT (Koehn, 2017) needs a fixed vocabulary to generate fixed-size vectors as input to the neural network. The larger this vocabulary is, the less unknown words occur, but the longer it takes to train the model and to apply it. To address this drawback,

| | Monograph | | Periodical | |
|---|---|---|---|---|
| **Language** | **fr** | **en** | **fr** | **en** |
| **Total # of Tokens in Ground Truth** | 597967 | 624678 | 255527 | 249483 |
| **Total # of Tokens in Original OCR Output** | 604444 | 649604 | 268742 | 274413 |
| **Erroneous Tokens** | **6.34%** | **6.22%** | **11.89%** | **12.11%** |

**Table 1:** Overview of the training set: Percentage of erroneous tokens (relative to the number of tokens in the OCR output using whitespace tokenization).

many approaches have been proposed to design open vocabulary NMT systems. These range from simple dictionary lookup techniques (Luong et al., 2015) over integrating SMT features (He et al., 2016) to specific design choices for the NMT architecture itself. Luong and Manning (2016) use a hybrid-system which translates primarily on word-level and falls back to a character-based representation for OOV words. Similarly, Sennrich et al. (2016) propose to translate on subword-level. They use byte pair encoding to create a vocabulary that is built bottom up, starting with characters and then adding larger subwords made from already known entries. The resulting vocabulary will contain characters, subwords as well as whole words.

Chung et al. (2016) designed a character-based decoder without explicitly segmenting the character sequences to match words. They motivate their approach with the following arguments: There is always a risk of introducing artificial errors when sentences are explicitly segmented into words. Furthermore, a character-based model will not suffer as much from data sparsity since stem forms and affixes can be treated separately. Finally, the system has a much better chance at generalization for unseen word forms. Extending this idea, Lee et al. (2017) aimed to design a fully character-based NMT system without explicit segmentation. They observe that their model is capable of locating spelling errors and still producing the correct translation. This finding is shared by Zhao and Zhang (2016).

Motivated by the mentioned approaches and their outcomes, we employ character-based NMT for post-correcting OCR errors. Since character-based NMT proved to handle spelling errors we assume that it will also perform well on OCR errors.

## 3 ICDAR 2017 OCR Post-correction Data

Our experimental data comes from the OCR post-correction Shared Task[1] of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR 2017) (see (Chiron et al., 2017) for more information). The data set is a subpart from a corpus

[1] https://sites.google.com/view/icdar2017-postcorrectionocr/home

| | Monograph | | Periodical | |
|---|---|---|---|---|
| **Language** | **fr** | **en** | **fr** | **en** |
| **Total # of Chars in Ground Truth** | 3560500 | 3592543 | 1637087 | 1574796 |
| **Total # of Chars in Original OCR Output** | 3569285 | 3592763 | 1640237 | 1575613 |
| **Insertions** | 0.32% | 0.51% | 0.68% | 0.63% |
| **Deletions** | 0.57% | 0.52% | 0.87% | 0.68% |
| **Substitutions** | 0.75% | 0.78% | 1.88% | 2.46% |
| **Errors Total** | **1.64%** | **1.81%** | **3.43%** | **3.77%** |
| **Unrecognizable** | 0.42% | 3.32% | 5.27% | 9.33% |

**Table 2:** Edit operations needed to correct training set: Percentage of total characters (relative to the characters in the OCR output) that need to be inserted, deleted, or substituted.

that was built in the AmeliOCR project[2]. The documents in the corpus originate from different digital collections and vary in terms of their condition and date of origin as well as the OCR engine and the post-correction initiative used to create the corpus (e.g. project-internal correction or external projects such as Gutenberg or Wikisource).

The equally sized English and French data consists of 12M OCRed characters and their alignments to a ground truth (GT)[3]. The training set has 10M characters (83%) and the official test set 2M (17%). The data is distributed as raw text files (one paragraph per file) where the first line contains the OCR output. On the second line, the OCRed text is vertically aligned with the GT. Wherever a character has to be inserted to match the length of the GT (which means there is no possible alignment) an "@" is inserted. On the third line that contains the GT, an "@" is inserted for all characters that appear in the OCR output but cannot be aligned with a GT character. The following example from 1860 illustrates the format:

(1) `ATRAVELLER STOPPED AT A WIDOW'S GATE.` [OCR]
`A@TRAVELLER STOPPED AT A WIDOW'S GATE.` [OCR aligned]
`A TRAVELLER STOPPED AT A WIDOW@S GATE.` [GT aligned]

Additionally, "unrecognizable" character sequences that cannot be identified with certainty in the original image are aligned with the "#" character in the GT. Probably due to different processing of soft hyphens, the GT for hyphens is very inconsistent. Following the decision of the shared task organizers, we ignore all hyphens in evaluations and error analyses, however, we keep them in the training data.

[2] `https://bit.ly/2BLsN7B`
[3] We use terms "gold standard" and "ground truth" interchangeably.

| | Monograph | | Periodical | | fr | | en | | both | |
|---|---|---|---|---|---|---|---|---|---|---|
| | fr | en | fr | en | form | freq | form | freq | form | freq |
| 1. | f → s | 1 → l | t → l | fi → ﬁ | f → s | 2346 | 1 → l | 2279 | f → s | 2653 |
| 2. | u → v | é → e | , → . | b → h | é → e | 870 | b → h | 1250 | 1 → l | 2310 |
| 3. | é → e | U → ll | e → é | u → n | e → é | 790 | fi → ﬁ | 993 | é → e | 1647 |
| 4. | i → j | e → é | o → e | - → - | u → v | 740 | u → n | 898 | b → h | 1338 |
| 5. | v → u | h → b | . → , | ff → ﬀ | t → l | 677 | é → e | 777 | u → n | 1320 |
| 6. | e → é | b → h | é → e | f → ſ | , → . | 639 | - → - | 720 | . → , | 1225 |
| 7. | l → t | d → ll | i → l | o → e | l → t | 562 | ff → ﬀ | 712 | , → . | 1214 |
| 8. | c → e | e → c | a → e | li → h | . → , | 535 | c → e | 701 | e → é | 1209 |
| 9. | 1 → ! | f → s | u → n | c → e | o → e | 521 | . → , | 690 | c → e | 1157 |
| 10. | è → e | ' → e | l → t | . → , | i → j | 504 | li → h | 666 | o → e | 1110 |

**Table 3:** 10 most common substitutions of characters ngrams (excluding hyphens). x → y means x was recognized instead of y.

The data set is difficult since the documents (a) origin from different collections (the BnF and the British Library), (b) were published either in periodicals or monographs, (c) cover a large time span (1654 to 2000). 92% of the documents with a known publication date are from the 19th century.

Diachronic document collections pose several challenges for OCR post-correction. OCR quality is generally lower for historical documents, typically worse for documents typeset in black letter fonts than in antiqua fonts. Additionally, old and modern spellings coexist, for instance, *compleated* (1744) vs *completed* (1894). Some characters as long s "ſ" disappear over time.

Tables 1 and 2 summarize the size of our training data, the proportion of "unrecognizable" characters, and the edit operations needed for error correction. Overall, periodical texts need more correction operations, as OCR quality for these texts is worse than for monographs. The better the OCR quality is, the harder it is to improve it with post-correction. Consequently, it will be easier to correct the periodical texts than the monographs. Note that English texts contain many more unrecognizable characters than the French texts.

**Statistics on character level** Table 3 shows the ten most frequent substitutions over character n-grams aggregated according to different text criteria. The substitutions are derived from the precomputed alignments of the data, and most of them can be explained due to the visual similarity of the glyphs. Note that the frequencies of the substitutions are not only specific to language but also to text type. Additionally,

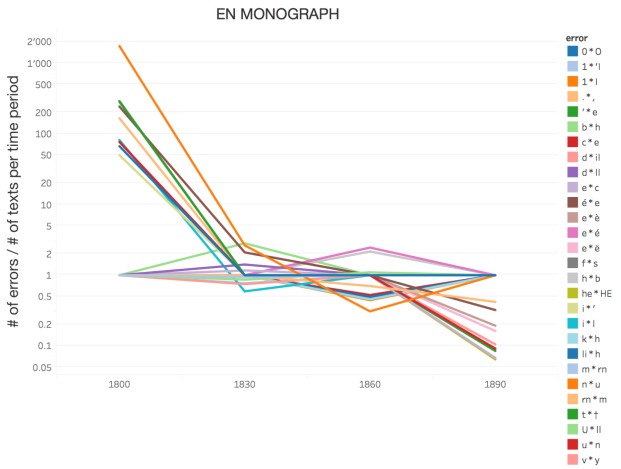


**Figure 1:** Changes of frequencies of the most frequent error types over time in English monograph data sampled in buckets of 30 years.

error frequencies also differ due to font type, paper quality and other document-related characteristics. However, since the source images of the data set were not provided, the influence of these factors on error frequencies cannot be assessed here. The confusion pair "e/é" in English is related to code-switching.

How do error types develop over time? Figure 1 is a screenshot from an interactive visualization of the frequency of the ten most frequent errors per 30 years in English monographs. The y axis is logarithmic and shows the number of errors normalized by the number of files for the specific time period. It is not essential to know which error belongs to which line. Instead, it is interesting to see the error development, for instance, the most frequent error from 1800 appears 1000 times less in 1830. Many errors in early documents occur rarely in later periods. Even though not all data sets show such extreme differences between the periods, similar observations can be made. Thus, it can be concluded that the different time periods all offer distinct challenges for OCR post-correction.

**Statistics on word level** 87.2% (en) and 90.2% (fr) of the wrong tokens only appear once in the training set (see Table 4). These numbers are computed by excluding any punctuation characters, e.g. a comma at the end of a word. The large percentage of

| | fr | | | en | | |
|---|---|---|---|---|---|---|
| | OCR | GT | Freq | OCR | GT | Freq |
| 1. | font | sont | 470 | 1 | I | 2245 |
| 2. | a | à | 331 | tbe | the | 513 |
| 3. | 1 | ! | 286 | thé | the | 510 |
| 4. | d | de | 167 | tlie | the | 303 |
| 5. | !. | !... | 164 | aud | and | 211 |
| 6. | do | de | 145 | 1 | 'I | 195 |
| 7. | ta | la | 139 | tho | the | 188 |
| 8. | ie | je | 129 | he | be | 172 |
| 9. | vn | un | 116 | hut | but | 165 |
| 10. | dé | de | 101 | ail | all | 165 |
| all | | | 51699 | | | 47583 |
| hapax legomena | | | 46644 | | | 41487 |

**Table 4:** Distribution of most frequent word errors

hapax legomena suggests that supervised post-correction on character level will be more beneficial than on word level due to data sparsity issues.

Table 4 also shows the ten most frequently misspelt words. Both datasets contain frequently occurring real word errors that cannot be corrected in isolation. Considering the Zipfian distribution of words, it is probably not surprising that four OCR error types of "the" are among the most frequently misrecognized English words. The mean of spelling versions per misrecognized GT type is 1.7 for English and 1.77 for French. The English word "the" has the highest number of spelling variants (275 unique types), for French it is the word "de" (214 types). Again, these findings suggest that a character-based approach is suitable for our task.

## 4 Methods

### 4.1 Experimental Setup

In order to keep the official test data untouched while exploring many different experimental settings, we randomly split the official training data in an internal training (80%), development (10%) and test set (10%). Unrecognizable characters are excluded from our data. Table 5 characterizes the resulting data sets.

Our baseline approach for character-level OCR correction preprocesses the data into a verticalized text format as follows: Whenever two space characters are aligned between the OCR output and the GT, a newline is inserted. After each actual text character

| | Periodical | | Monograph | | Combined | | |
|---|---|---|---|---|---|---|---|
| | fr | en | fr | en | fr | en | both |
| Train | 1320728 | 1268194 | 2862351 | 2887538 | 4183079 | 4155732 | 8338811 |
| Dev | 165449 | 158748 | 360090 | 362068 | 525539 | 520816 | 1046355 |
| Test | 165032 | 158492 | 356580 | 360228 | 521612 | 518720 | 1040332 |

**Table 5:** Character counts of internal train, dev and test sets (including whitespace and hyphens) for all four document types. The combined data sets contain periodicals and monographs.

(@ can be ignored), we add a space to force the MT systems to translate on character level. Example 2 illustrates the conversion into the common line-based training format of MT systems. Having one word per line makes the training of the MT model simple, however, it results in a context-insensitive approach.

(2) `This@couise@was agretd to,` [...] Aligned OCR output
`This course was agreed to,` [...] Aligned GT

| OCR (Source) | GT (Target) |
|---|---|
| T h i s c o u i s e w a s<br>a g r e t d<br>t o , | T h i s ␣ c o u r s e ␣ w a s<br>a g r e e d<br>t o , |

**Character-based SMT** We train a baseline character-based SMT model analogous to (Pettersson et al., 2013) using the Moses toolkit (Koehn et al., 2007), GIZA++ character alignment (Och and Ney, 2003), and MERT optimization (Och, 2003). We use a 10-gram language model in all experiments. For a more extensive introduction to SMT, please refer to Koehn (2009).

**Character-based NMT** For a more in-depth and general introduction to NMT, we refer to Koehn (2017).

We first tested the convolutional framework described in Lee et al. (2017), which is a character-level NMT system by design. However, as early experiments did not show satisfactory results we discarded this option.

Nematus[4] is a high-performing NMT framework (Sennrich et al., 2017). Character-level MT can be enforced by using the same whitespace insertion "trick" in the data format as for Moses. Initial experiments showed positive results and we relied on this framework for all subsequent experiments.

[4] `https://github.com/EdinburghNLP/nematus`

Nematus – as most neural translation frameworks – has many hyperparameters that influence its performance. Given training times of several hours even on fast GPUs, it is not feasible to tune these parameters systematically for each data set. We decided to explore them on the French periodical data for two reasons: First, as one of the smaller data sets it is faster to train, and, second, prior SMT experiments had shown that this set is harder to correct than others. In the following, we quickly introduce our base hyperparameter settings that we determined by experiments on the French periodicals.

Input or output embeddings are low-dimensional dense continuous vector representations of one-hot encoded categorial data with typically much larger dimensionality. The dimensionality of characters is orders of magnitude lower than that of words. We choose an embedding size of 256, tested against 32 and 512. We configure them as tied embeddings, meaning that the embedding of a character on the source side is the same as on the target side. This seems reasonable given that both sides are the same language apart from OCR errors. According to our experiments, not using tied embeddings achieves better results in error detection but worse results in error correction. Eventually, we decided to use tied embeddings as they speed up training.

Dropout is a useful regularization method for neural architectures. During training, some units in the network are switched off. Consequently, the model does not have access to all its information and is less prone to overfit. The dropout rate specifies how many units are switched off at the same time. We set it to 0.2.

Batch size refers to the number of training examples which are used to compute the gradient and update the weights in the network. A larger batch size leads to a speed-up in training, but it can also have a negative impact on learning. We set it to 100, tested against 50 and 200.

For a few hyperparameters, we use the default values from Nematus: hidden layer size (1000), optimizer (adadelta), gradient clipping threshold (1) and learning rate (0.0001). Furthermore, we set the maximum sequence length to 23 for all context-insensitive experiments and to 53 for context-sensitive experiments. These limits cover 99.99% of all training examples.

### 4.2 Using More Training Material

In MT, the most straightforward method to increase translation quality is by adding more training material. Since OCR errors mostly occur due to the visual similarity of characters, the same errors occur in periodicals as in monographs and also in both languages. Therefore, it makes sense to combine the training sets of the individual text types (henceforth, "medium" data set size), maybe also combining all the available data across languages (henceforth, "large" data set). The latter especially because we noticed code switching effects in Section 3.

### 4.3 Using More Context

Some OCR errors introduce wrong tokens that are valid words of the document language. For example, in Table 4 the most frequent French OCR error on word-level is "sont"

wrongly recognized as "font" (probably because of the long s glyph). However,"font" cannot be corrected in isolation since it is a frequent French word. We need some context in order to change it appropriately. Our context-sensitive format with two preceding and one succeeding words with respect to a focus word inbetween is shown in Example 4.3.

(3) `Si ces principes font fondés sur le goût` Aligned OCR
`Si ces principes sont fondés sur le goût` Aligned GT

| OCR (Source) |
|---|
| S i # c e s # p r i n c i p e s # f o n t |
| c e s # p r i n c i p e s # f o n t # f o n d é s |
| p r i n c i p e s # f o n t # f o n d é s # s u r |
| GT (Target) |
| S i # c e s # p r i n c i p e s # f o n t |
| c e s # p r i n c i p e s # s o n t # f o n d é s |
| p r i n c i p e s # s o n t # f o n d é s # s u r |

An artificial word boundary marker which otherwise does not occur in the training data needs to be chosen.[5] Alternatively, we could have chosen to translate only the focus word, however, forcing the MT system to produce the context on the target side as well, produced better results. This representation increases the training material, which is good, but unfortunately also extends training time.

### 4.4 Factored Character-based NMT

Nematus supports "factored" NMT models (Sennrich and Haddow, 2016) where structured information can be included on the source side: for instance, the time span from which the text originates, or its text type, or even its language if we want to merge both languages in order to have more training material.[6] In Nematus, factors are implemented as embedding vectors that are concatenated with the input character embeddings.[7]

Factors can be conveniently expressed in the input format. Example 4 with a text snippet from English periodicals between 1800 and 1849 shows the context-insensitive input format with factors. The symbol | marks the beginning of a factor. The first factor refers to the text type, and the second stands for the time span[8] when the text was written. This "feature" can be helpful to model time-dependent OCR errors which occur due to orthographic or typographical changes as illustrated in Figure 1.

(4) `who may wish` Aligned OCR
`who may wish` Aligned GT

[5] If the boundary marker does not appear 3 times on the translated target side, we select the word with the smallest edit distance to the input focus word as its translation.
[6] OCR errors for French and English can be similar due to their common Latin script system.
[7] Due to a problem in Nematus, we could not use tied embeddings in combination with factors.
[8] Split in bins of 50 years and represented by YYYY//50 (integer division).

| OCR (Source) | GT (Target) |
|---|---|
| w\|peri\|36 h\|peri\|36 o\|peri\|36 | w h o |
| m\|peri\|36 a\|peri\|36 y\|peri\|36 | m a y |
| w\|peri\|36 i\|peri\|36 s\|peri\|36 h\|peri\|36 | w i s h |

### 4.5 Glyph Embeddings

State-of-the-art NMT often uses pre-trained distributional word embeddings, which mainly put words closer together in the vector space if they share the same contexts. For OCR post-correction, characters do not necessarily need to be substituted with characters that often share the same context, but rather share similar character shapes. Therefore, we generate pre-trained embeddings that express some visual similarity between characters. As no source images are available for our data, we simply use 16 by 16 grey-scale pixel values of every character from the Helvetica font as a proxy.[9]

Of course, there are better ways how visual information can be included in an NMT system. However, this is a straightforward and time-saving technique and is therefore used in this experiment.

### 4.6 Error-focused Models

Most of the OCR output is simple to process for an MT system since it is already correct and does not need to be translated. This means that for the most part the NMT system just learns how to copy characters from the source to the target side. If the system is trained on a data set which contains a larger proportion of errors, it will become better at detecting errors and will do this more aggressively at the cost of over-correction. However, these systems might over-correct. Therefore, it is essential to determine a reasonable amount of non-error tokens in the training data to boost the error detection recall but not affecting precision too much. For our experiments, we first tried a split of 75% error and 25% non-error tokens. Since there are fewer errors in the monographs, we adjust to 50% errors for French monograph and 37.5% for the English. We randomly filter out examples without errors in this subsampling process.

### 4.7 Ensemble Decoding

Ensemble decoding is a common technique in NMT where the individual probability distributions of different models are averaged. The rationale is that the biases of single models' outputs even out by the combination of several models. With Nematus it is possible to do ensemble decoding by using either models from the same training run at different time steps or multiple models. For our experiments, we try both. In order to distinguish them, the former will be called "single ensemble" (using different time stamp models of the same system) and the latter "multi ensemble" (using the best model of different systems). All experiments with ensemble decoding are conducted on

[9] The values are normalized into a range between -1 and 1 and then scaled by 0.01 as done in Nematus for randomly initialized values.

the context-sensitive input formats. Single ensembles are tested for the base context-sensitive NMT system and the error-focused NMT system. For single ensembles, we combine the best model with the models from two previous time stamps. Multi ensembles use the base context-sensitive NMT system, the one with glyph embeddings and the error-focused system.

### 4.8 System Combinations

The final method explored in our experiments is combining the outputs of multiple MT systems. The output from such post-translation combination of systems was submitted to the ICDAR shared task.

Different strategies are used for error detection and error correction. Keep in mind that for the error correction test set, the shared task organizers published the positions of erroneous tokens. Therefore, we translate all data for error detection, however, for the error correction we only translate the erroneous tokens.

A flow diagram of our decision procedure is shown in Figure 2. The error detection algorithm uses the output of five MT systems for a specific data set which worked best in combination as tested on the dev set. We have four conditions which trigger the error detection. First, if the model with the smallest Levenshtein distance proposes a change, we assume there is an error. Second, we check the five best systems, and if the most frequently proposed token is different from the OCR output, we also assume an error. Third, if the original OCR token does not occur in the GT training and dev sets for both text types, we assume an error. Finally, if the current token and one of its neighbors (both can be translated or untranslated) do not occur in the training set, but their concatenation does, we assume that the tokens have to be merged.

The algorithm for error correction works slightly different. The five models with the smallest Levenshtein distance on the dev set are used to generate correction candidates. Since the evaluation script for the shared task evaluates two scenarios, a "fully-automated" one where only one correction candidate is given and a "semi-automated" one where a ranked list of candidates with confidence scores is given, we pay special attention to our choice of candidates. If the system with the smallest Levenshtein distance on the dev set suggests a correction, we take this as an exclusive correction candidate. Otherwise, we look at the candidates of all five systems and model the weights according to the frequency distribution of their suggestions which are different from the OCR output.

## 5 Results and Discussion

### 5.1 Evaluation Setup

All configurations are evaluated on our internal test set by macro-averaging the scores of three individual models using the same configuration. This reduces effects caused by system variance. Error detection is measured on word level using the following measures:

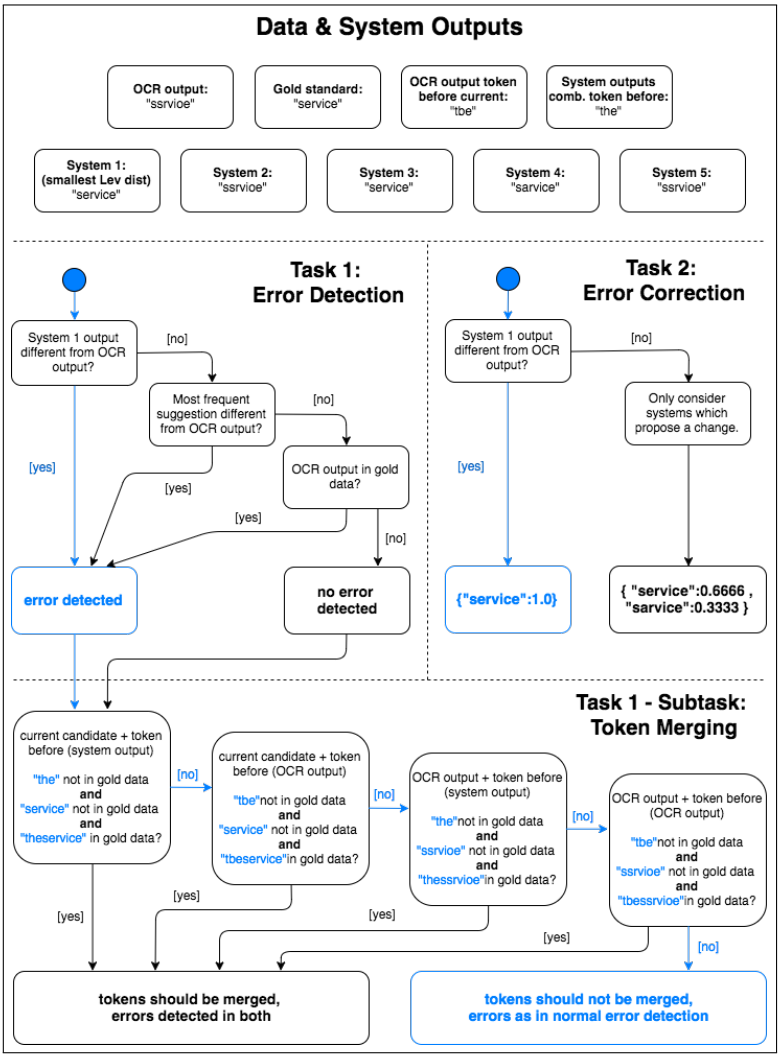


**Figure 2:** Algorithms for error detection and correction explained with an example.

| | Error Detection | | | Error Correction | | |
|---|---|---|---|---|---|---|
| | P ↑ | R ↑ | F1 ↑ | Lev. ↓ | % Rel. Imp. ↑ | % Correct ↑ |
| English Periodicals | | | | | | |
| OCR baseline | Char/Tok ER: 3.47% / 11.28% | | | 0.1898 | 0% | - |
| SMT baseline * | 83.82 | 61.84 | 71.17 | 0.1347 | 40.96 | 53.42 |
| SMT medium | 91.33 | 46.87 | 61.94 | 0.1461 | 29.96 | 57.28 |
| SMT large | **93.26** | *40.51* | *56.47* | 0.1510 | 25.71 | 56.78 |
| NMT baseline | 87.33 | 59.27 | 70.61 | 0.1472 | 28.95 | 49.00 |
| NMT medium | 90.60 | 46.60 | 61.53 | 0.1584 | 19.85 | 47.52 |
| NMT large | 91.47 | 42.51 | 58.04 | *0.1593* | *19.13* | 47.42 |
| SMT context * | 85.19 | 58.87 | 69.62 | **0.1339** | **41.83** | **58.98** |
| NMT context * | 89.31 | 59.56 | 71.46 | 0.1406 | 34.95 | 51.34 |
| NMT time factor no context | 87.38 | 60.27 | 71.33 | 0.1475 | 28.68 | 48.24 |
| NMT time factor context | 89.99 | 57.77 | 70.37 | 0.1420 | 33.65 | 51.38 |
| NMT factor medium | 88.82 | 58.94 | 70.85 | 0.1448 | 31.12 | 49.94 |
| NMT factor large | 89.32 | 57.28 | 69.79 | 0.1415 | 34.10 | 50.95 |
| NMT glyph embeddings * | 89.62 | 59.16 | 71.27 | 0.1422 | 33.44 | 51.17 |
| NMT error-focused | *76.95* | 68.01 | 72.20 | 0.1591 | 19.28 | *41.09* |
| NMT single ensemble * | 89.92 | 59.54 | 71.64 | 0.1388 | 36.73 | 52.22 |
| NMT single ensemble error | 78.41 | **68.28** | **72.99** | 0.1533 | 23.80 | 43.35 |
| NMT multi ensemble * | 89.20 | 60.79 | 72.30 | 0.1369 | 38.68 | 52.37 |
| English Monographs | | | | | | |
| OCR baseline | Char/Tok ER: 1.79% / 6.38% | | | 0.0830 | 0% | - |
| SMT baseline * | **92.84** | 33.92 | 49.68 | 0.0631 | 31.50 | **72.03** |
| SMT medium | 86.32 | 36.58 | 51.37 | 0.0649 | 27.84 | 64.52 |
| SMT large | 89.41 | *32.84* | *48.03* | 0.0652 | 27.30 | 68.56 |
| NMT baseline * | 91.77 | 35.50 | 51.19 | 0.0652 | 27.33 | 65.94 |
| NMT medium | 87.56 | 36.74 | 51.76 | 0.0668 | 24.18 | 61.54 |
| NMT large | 87.82 | 33.92 | 48.93 | 0.0678 | 22.45 | 62.52 |
| SMT context * | 88.85 | 39.07 | 54.27 | 0.0628 | 32.25 | 70.34 |
| NMT context * | 87.97 | 39.93 | 54.92 | 0.0632 | 31.27 | 65.38 |
| NMT time factor no context | 90.19 | 40.64 | 56.03 | 0.0668 | 24.26 | 58.39 |
| NMT time factor context * | 85.31 | 45.04 | **58.94** | 0.0644 | 28.77 | 61.06 |
| NMT factor medium | 92.33 | 39.66 | 55.48 | 0.0652 | 27.27 | 61.22 |
| NMT factor large | 91.62 | 39.24 | 54.95 | 0.0654 | 26.83 | 61.91 |
| NMT glyph embeddings * | 87.75 | 40.27 | 55.20 | 0.0631 | 31.46 | 65.73 |
| NMT error-focused | 50.51 | 50.81 | 50.60 | 0.0884 | -6.01 | 35.80 |
| NMT single ensemble | 89.95 | 39.51 | 54.90 | **0.0625** | **32.87** | 67.39 |
| NMT single ensemble error | *49.38* | **51.70** | 50.43 | *0.0888* | *-6.31* | *35.58* |
| NMT multi ensemble * | 86.85 | 41.14 | 55.82 | 0.0627 | 32.38 | 65.78 |

**Table 6:** English macro-averaged experiments results. Best results are marked in blue, worst results in red. Asterisks mark indicates systems used for ICDAR submission. (ER = error rate)

| | Error Detection | | | Error Correction | | |
|---|---|---|---|---|---|---|
| | P ↑ | R ↑ | F1 ↑ | Lev. ↓ | % Rel. Imp. ↑ | % Correct ↑ |
| French Periodicals | | | | | | |
| OCR baseline | Char/Tok ER: 3.41% / 11.85% | | | 0.1930 | 0% | - |
| SMT baseline * | 83.48 | *35.31* | *49.62* | 0.1656 | 16.54 | 47.39 |
| SMT medium | 86.93 | 38.00 | 52.87 | 0.1663 | 16.06 | 46.97 |
| SMT large | 87.18 | 35.37 | 50.31 | 0.1675 | 15.22 | 44.43 |
| NMT baseline * | 87.87 | 43.52 | 58.21 | 0.1700 | 13.53 | 39.25 |
| NMT medium | 88.49 | 41.75 | 56.72 | 0.1695 | 13.88 | 38.93 |
| NMT large | 88.01 | 40.07 | 55.06 | 0.1735 | 11.21 | 35.84 |
| SMT context | 81.75 | 40.00 | 53.71 | **0.1614** | **19.64** | **53.66** |
| NMT context | **88.53** | 44.26 | 59.01 | 0.1645 | 17.32 | 42.53 |
| NMT time factor no context | 85.81 | 44.58 | 58.66 | 0.1755 | 10.02 | 37.77 |
| NMT time factor context * | 87.78 | 46.74 | 61.00 | 0.1655 | 16.68 | 40.88 |
| NMT factor medium | 87.51 | 44.20 | 58.72 | 0.1699 | 13.64 | 39.09 |
| NMT factor large | 86.28 | 43.83 | 58.11 | 0.1704 | 13.30 | 39.24 |
| NMT glyph embeddings * | 88.30 | 43.22 | 58.03 | 0.1657 | 16.45 | 41.81 |
| NMT error-focused * | *67.47* | 60.72 | 63.91 | *0.1956* | *-1.32* | *28.99* |
| NMT single ensemble | 88.30 | 44.26 | 58.96 | 0.1631 | 18.34 | 43.45 |
| NMT single ensemble error | 69.54 | **61.55** | **65.30** | 0.1903 | 1.40 | 30.99 |
| NMT multi ensemble * | 86.79 | 47.84 | 61.68 | 0.1621 | 19.08 | 42.17 |
| French Monographs | | | | | | |
| OCR baseline | Char/Tok ER: 1.61% / 6.18% | | | 0.0692 | 0% | - |
| SMT baseline * | 82.69 | 38.55 | 52.52 | **0.0558** | **24.00** | 55.04 |
| SMT medium | 80.81 | 36.30 | *50.10* | 0.0574 | 20.59 | 51.95 |
| SMT large | 82.44 | *36.24* | 50.18 | 0.0587 | 18.07 | 48.84 |
| NMT baseline | 84.22 | 40.10 | 54.38 | 0.0602 | 15.00 | 44.13 |
| NMT medium | 83.59 | 38.23 | 52.37 | 0.0593 | 16.71 | 43.89 |
| NMT large | 82.81 | 37.15 | 51.05 | 0.0610 | 13.66 | 40.66 |
| SMT context * | 82.49 | 40.51 | 54.18 | 0.0568 | 22.15 | **58.47** |
| NMT context * | 84.17 | 42.95 | 56.85 | 0.0586 | 18.26 | 46.10 |
| NMT time factor no context | 84.37 | 41.53 | 55.75 | 0.0591 | 17.28 | 45.38 |
| NMT time factor context | 81.60 | 43.04 | 56.45 | 0.0583 | 18.85 | 45.68 |
| NMT factor medium | **85.94** | 40.97 | 55.46 | 0.0580 | 19.31 | 47.32 |
| NMT factor large | 85.63 | 39.36 | 53.86 | 0.0577 | 20.11 | 47.48 |
| NMT glyph embeddings * | 83.55 | 43.30 | 57.00 | 0.0583 | 18.73 | 45.43 |
| NMT error-focused | *58.53* | 54.56 | 56.48 | *0.0756* | *-8.28* | *30.23* |
| NMT single ensemble * | 85.72 | 42.98 | 57.29 | 0.0575 | 20.43 | 48.10 |
| NMT single ensemble error | 59.71 | **55.15** | 57.37 | 0.0744 | -6.99 | 31.87 |
| NMT multi ensemble * | 82.27 | 45.73 | **58.91** | 0.0581 | 19.09 | 46.30 |

**Table 7:** French macro-averaged experiments results. Best results are marked in blue, worst results in red. Asterisks mark indicates systems used for ICDAR submission. (ER = error rate)

**Precision** $P$ How many of the tokens predicted as incorrect actually needed a correction?

**Recall** $R$ How many of the incorrect tokens were actually predicted as incorrect?

**F1-Measure** Harmonic mean of precision and recall: $F1 = \frac{2*P*R}{P+R}$

Error correction is measured on character level using Damerau-Levenshtein distance averaged over all tokens.[10] Levenshtein distance between tokens $t_1$ and $t_2$ measures the minimal amount character substitutions, deletions or inserts needed to turn $t_1$ into $t_2$ (Levenshtein, 1966). Damerau distance (Damerau, 1964) additionally allows to swap two adjacent characters.[11] Note that for the official submissions in task 2, a list of candidate corrections combined with their probability was evaluated. In this setup, the evaluation reports the weighted sum of edit distances for all candidates (weight equals probability). For the results shown on our internal test set, we always sticked to only one correction candidate.

For task 2, i.e. the error correction, the positions of erroneous tokens were given and only those were evaluated. Of course, evaluating error correction only on erroneous tokens is an artificial scenario. In practice, it is not known if a token is correct or not. Any potential corrections may also introduce new errors to correct tokens. This evaluation is therefore not ideal but the evaluation setup was given by the competition. Note that we used the same training data for both tasks.

In the following, we present and discuss the results for different experimental settings in the same order as they were introduced in the preceding section. Table 6 summarizes all results on English data sets, Table 7 on French. In addition to the F1-score and the relative Levenshtein distance improvement, these tables also report character- and token-level error rates as well as the Levenshtein distance for the original OCR output, precision and recall scores for all experiments as well as the percentage of translated words that are correct.

### 5.2 More Training Material

What is the effect of combining the different data sets for training? Against our expectations, error detection and correction performance mostly decreases with medium training sets, especially in the case of English periodicals (see Tables 6 and 7). Only SMT can sometimes profit from more data. This suggests that OCR errors are particular to a text type and that NMT approaches specifically adapt to the specific OCR error distribution (frequency and types) of a training set. This trend is even stronger when cross-linguistically combining larger training sets. There, performance drops even more.

More general observations about context-insensitive correction can be made: First, SMT models perform better than NMT in error correction and worse in error detection.

[10] The official evaluation script was released via `https://git.univ-lr.fr/gchiro01/icdar2017`. Although the official ICDAR report refers to Levenshtein distance, the implementation actually uses Damerau-Levenshtein distance.

[11] Damerau-Levenshtein actually is more useful for measuring human spelling distance than OCR spelling distance.

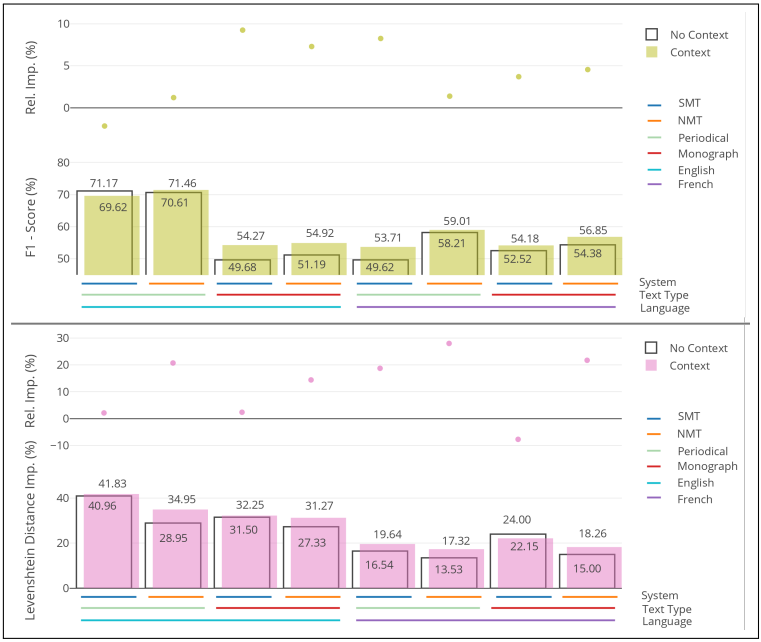


**Figure 3:** Effect of the inclusion of context (two preceding and one succeeding token) on performance

Second, error correction works better on English than on French. Third, error detection works better on the periodical data, probably due to the higher a priori error rate of this text type.

### 5.3 Using More Context

The results of the experiments described in Section 4.3 are shown in Figure 3. As expected, more context for translation effectively improves results in all but one data set. The transparent columns with black borders show the context-sensitive baseline trained separately on language-specific text types. The colored columns show the results using the context-insensitive sets. The scatter plots above the bar plots give the relative improvement (scales are not comparable between figures).

The overall performance pattern as discussed before still applies to the individual data sets, and SMT still performs better for error correction and NMT for error detection.

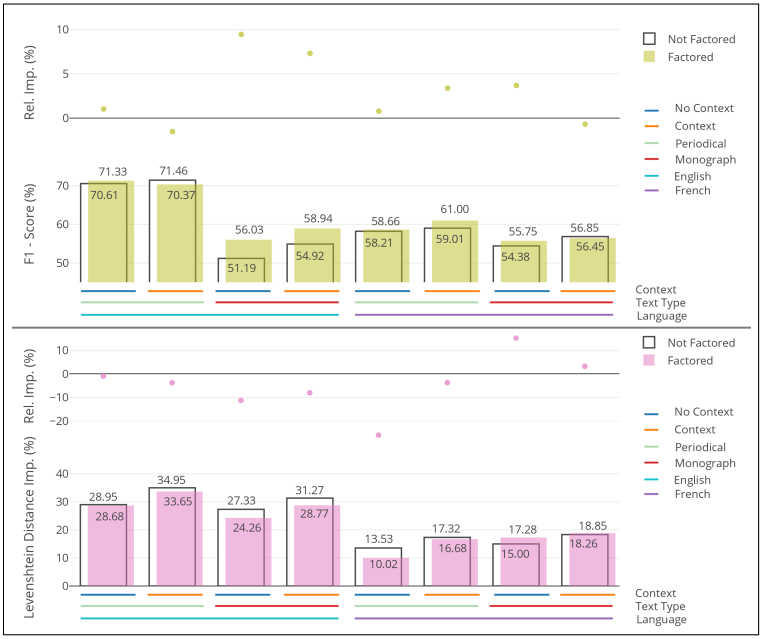


**Figure 4:** Effect of time-factored training material

### 5.4 Factored Character-based NMT

Using factors as described in Section 4.4 is another strategy to include more information when training an NMT system. Figure 4 shows the influence of the factor on the publication time. Overall, time factors result in a better performance in error detection for most data sets, especially for English monographs. This phenomenon can be explained with a quick look at Figure 1. Most English monograph texts do not need many corrections. However, this is problematic for the earlier files (around 1800), which contain many more errors than the later ones. Time factors allow the model to detect errors more aggressively on texts that were published around 1800. Therefore, time factors control the aggressiveness of an NMT model's error detection. In contrast to error detection, there is a decrease in the relative Levenshtein distance improvement in error correction for all but two data sets. This development could be simply because there is an improvement in error detection. Detecting more errors does not automatically

mean providing accurate corrections. Trying to correct more tokens can lead to an overall higher distance to the GT. Another explanation could be that, especially, with smaller data sets there might be data sparsity issues.

Unfortunately, also adding factors for languages and combining all data folds into a large data set still does not generally improve the performance (results not shown).

### 5.5 Glyph Embeddings

The use of glyph images as pre-trained embeddings only shows a small improvement for monographs in both F1-score and relative Levenshtein distance (results not shown). For periodicals, the results are even slightly worse than with randomized embeddings. Adding the intuitively promising information about glyph similarity therefore needs another modeling approach than ours for more general applicability. Maybe using glyph images with closer resemblance to the actual fonts helps, maybe we should concatenate the glyph vector representation with the character embeddings instead of using them as initialization weights.

### 5.6 Error-Focused Models

We have seen in Table 2 that periodicals contain many more errors in need of correction than monographs. Therefore, we subsample error-focused training sets as described in Section 4.6. As expected, error correction suffers a lot from such unrepresentative training material and over-corrects the texts. As intended, error detection F1 score improves for periodicals as recall and precision get more balanced. Without subsampling precision is typically 20 to 40 percentage points higher than recall. Unfortunately, F1 decreases for monographs (especially for English). Maybe an improved tuning of the optimal error proportion for subsampling can remedy these rather unexpected differences.

### 5.7 Ensemble Decoding

The results confirm that in almost all cases error detection and correction improve when single model ensembles are used. Relative improvement for error correction is much higher than for error detection. This suggests that at different training moments the models provide different correction candidates which in combination converge to better suggestions.

Context-sensitive neural multi ensembling as described in Section 4.7 is strong for error detection and competitive for NMT error correction. However, SMT generally outperforms neural multi ensembles on error correction by a large margin. The performance pattern of multi ensembles across languages and text types is still similar to the patterns seen for the context-insensitive baseline, suggesting that these patterns are not due to specific idiosyncrasies of one of our models.

| Data Set | Error Detection (Task 1) | | Error Correction (Task 2) | |
|---|---|---|---|---|
| en Periodical | 73.0 | NMT single ensemble error | 41.8 | SMT context |
| en Monograph | 58.9 | NMT time factor context | 32.8 | SMT baseline |
| fr Periodical | 65.3 | NMT single ensemble error | 19.6 | SMT context |
| fr Monograph | 58.9 | NMT multi ensemble | 24.0 | SMT context |

**Table 8:** The best systems for each data set. For error detection, macro-averaged F1 score is considered and for error correction macro-averaged relative Levenshtein distance improvement.

### 5.8 Summary on Systems

Generally, the performances across data sets vary a lot, except for error detection on monographs. Table 8 collects all best-performing systems for error detection (macro-averaged F1-measure) and error correction (macro-averaged relative Levenshtein distance improvement). All best systems for error detection use NMT, while all best systems for error correction use SMT. Context-sensitive SMT consistently works best in all but one case. For error detection, all but one system are ensembles. We explain the exception of English monographs by its high error variation over time, something that time-factored models capture effectively. Our results indicate that the optimal choice of an MT systems for OCR post-correction strongly depends on the task (detection or correction), language, text type, time span and error distribution. Ensemble decoding in NMT models almost always boosts performance. Ensembling of SMT output could also help, but we did not conduct such experiments.

### 5.9 Our ICDAR 2017 OCR Post-Correction Competition Submission and Results

As described in Section 4.8, our submission combines the outputs from different systems. Unfortunately not all system types presented in this article were available at submission time of the shared task, for instance, the well-performing ensembles of the error-focused models. The systems for submission were selected such that their combination performs best on an internal tuning set (see Table 9 for a complete listing for each data set). Note that the selected systems are not necessarily the top five systems on that data set as their combined performance is considered.

Table 10 summarizes the results of the best 6 teams (out of 11) from the official paper on the ICDAR 2017 OCR post-correction shared task (Chiron et al., 2017).[12] Our approach (Char-SMT/NMT) achieved the best results of all submitted methods in error correction (task 2). The relative improvement columns report the relative

[12] Probably due to the rather complex output format defined by the shared task organizers, many teams seem to have produced inconsistent data.

| Error Detection (Task 1) | | Error Correction (Task 2) | |
|---|---|---|---|
| **en Periodical** | **en Monograph** | **en Periodical** | **en Monograph** |
| NMT multi ensemble | NMT baseline | SMT context * | SMT baseline * |
| SMT baseline * | SMT context * | SMT baseline | SMT context |
| NMT single ensemble | NMT multi ensemble | NMT multi ensemble | NMT baseline |
| NMT glyph embeddings | NMT context | NMT single ensemble | NMT glyph embeddings |
| NMT context | NMT time factor context | NMT context | NMT multi ensemble |
| **fr Periodical** | **fr Monograph** | **fr Periodical** | **fr Monograph** |
| NMT multi ensemble | NMT multi ensemble | NMT multi ensemble * | SMT context * |
| NMT baseline * | SMT context * | NMT time factor context | SMT baseline |
| NMT time factor context | NMT single ensemble | NMT error-focused | NMT multi ensemble |
| NMT error-focused | NMT context | NMT glyph embeddings | NMT single ensemble |
| NMT glyph embeddings | NMT glyph embeddings | SMT baseline | NMT context |

**Table 9:** Systems which were included in our system combination for task 1 and 2. Systems marked with an asterisk have the smallest Levenshtein distance of the systems in that combination.

token-level improvement if the top-ranked correction candidate of a system is applied. Compared to the other approaches, we consistently deliver best results across all data sets. A hyphen in the cells indicates that a system actually deteriorated more tokens than it improved. In contrast to our submission, quite a lot of system failed to improve the texts, which is an indication of the difficulty of the task.

In error detection (task 1), we performed comparatively to other submissions. The best approach for task 1 applies a noisy channel model to the OCR post-correction, and uses the Google Books Ngrams for their vocabulary and language model. Therefore, error detection probably profits from external data. In contrast, our approach does not need additional resources to train the MT systems. Other models that achieve similar results as our method use character-based NMT with context, SMT on character and token level, spell checkers, error frequency patterns, or a 2-pass RNN architecture where the first RNN works on character level and the second on token level. The official shared task report (Chiron et al., 2017) contains short summaries of the chosen approach from all participating teams.

Table 11 shows the results of task 1 in more detail. Accuracy and recall can be compared directly. Our method achieved by far the highest accuracy. However, even though we tried to increase the recall for error detection via system combination, our recall is not as high as it is for other methods. Still, the amount of documents that our system actually improves (102 documents) is much higher than for all other systems. The best system for error detection is only able to improve about 60% of documents

| Teams/Data Sets | Task 1 (F-measure) | | | | | Task 2 (%Improvement) (top-ranked correction candidate) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | en mono | en peri | fr mono | fr peri | mean | en mono | en peri | fr mono | fr peri |
| # tokens (error rate) | 63371 (10%) | 33176 (15%) | 32274 (5%) | 48356 (7%) | | 63371 (10%) | 33176 (15%) | 32274 (5%) | 48356 (7%) |
| **WFST-PostOCR** | **0.73** | **0.68** | **0.55** | **0.69** | **0.66** | 28% | - | - | - |
| 2-pass-RNN ‡ | 0.66 | 0.66 | 0.43 | 0.60 | 0.59 | x | - | x | x |
| EFP | 0.69 | 0.54 | 0.40 | 0.54 | 0.54 | 13% | - | 23% | 5% |
| **Char-SMT/NMT** | 0.67 | 0.64 | 0.31 | 0.50 | 0.53 | **43%** | **37%** | **44%** | **29%** |
| CLAM | 0.67 | x | 0.36 | 0.54 | 0.52 | 29% | 22% | 1% | 5% |
| MMDT ‡ | 0.66 | 0.44 | 0.36 | 0.41 | 0.47 | 20% | - | 3% | 2% |
| Post-Submission | 0.62 | 0.59 | 0.35 | 0.51 | 0.52 | 18% | 21% | 40% | 24% |

**Table 10:** Official ICDAR OCR post-correction results (only 6 best teams out of 11 shown). Systems marked with "‡" had partially malformed submission format. Cell entries with "x" indicate missing or malformed submissions for this data set. Cell entries with "-" indicate that no improvement was achieved, meaning there was a deterioration of the texts. The last row shows the performance of our high-performing single system post-submission.

compared to our submission.

Table 11 provides valuable information on what could be improved with our method and in what cases it makes sense to use it. Our approach is optimal for an automatic OCR correction scenario with high precision requirements for error detection and high correction quality. On the other hand, if recall is more important, for instance, if manual verification is applied to the correction candidates, our method might need to be adapted further, or another approach should be used for the error detection.

There is a practical issue with our approach if we want to apply it to new datasets. Building all the necessary systems for our system combination is a bit complex and laborious. In a post-submission experiment, we therefore tried to build the most promising single system according to our experimental experience.

Our post-submission system is a combination of the strategies with the greatest potential to profit from each other[13]: For each language, we combined the context-sensitive (two preceding and one following token) training material for periodicals and monographs and used factors to encode information on the time period and text type of the documents. The last row in Table 10 shows that such a single system performs reasonably across all data sets, however, there is a substantial drop in comparison to the best reported result.

[13]We did not empirically test all combinations systematically, though.

| | en | | fr | | en | | fr | | all |
|---|---|---|---|---|---|---|---|---|---|
| | mono | peri | mono | peri | mo | pe | mo | pe | all |
| System | Acc./Rec. | Acc./Rec. | Acc./Rec. | Acc./Rec. | # | # | # | # | # |
| Char-SMT/NMT | **0.98**/0.51 | **0.88**/0.50 | **0.74**/0.19 | **0.93**/0.34 | **40** | **4** | **46** | **12** | **102** |
| CLAM | 0.93/0.52 | -/- | 0.48/0.28 | 0.71/0.44 | 36 | **4** | 31 | 7 | 78 |
| MMDT | 0.84/0.55 | 0.72/0.32 | 0.62/0.25 | 0.71/0.28 | 37 | 3 | 28 | 7 | 75 |
| EFP | 0.62/0.77 | 0.54/0.55 | 0.29/**0.60** | 0.49/0.58 | 31 | 1 | 24 | 11 | 67 |
| WFST-PostOCR | 0.67/**0.82** | 0.68/**0.68** | 0.51/0.59 | 0.72/**0.66** | 36 | 0 | 20 | 3 | 59 |
| 2-pass RNN | 0.58/0.77 | 0.64/**0.68** | 0.33/0.60 | 0.09/0.04 | × | 0 | × | × | 0 |
| | | | Total # of Documents in Test Set ⇒ | | 41 | 4 | 54 | 12 | 110 |

**Table 11:** Official ICDAR results break down (only 6 best performing teams out of 11 shown) on accuracy, recall, and number of documents where a net improvement resulted from the corrections of task 2.

## 6 Future Work

The comparison of the best ICDAR systems in error detection with our results indicates that our methods do not yet spot the optimal amount of OCR issues that need correction. The resources that our models use are strictly limited to the official training material. The best ICDAR error detection system uses additional material from Google Books n-grams. For a practically oriented OCR error detection, we should think about ways to integrate external data for error detection. However, for solving the full OCR post-correction task one still has to be able to actually generate the correct spelling.

An important question for practical applicability is the amount of training data needed for our approach. Ablation experiments where the training material is systematically reduced would offer some insights about the correlation of training data size and the post-correction improvement, which is probably also dependent on the original error rate.

A better tuning of the subsampling for error-focused models is worth trying. For an optimal F1 score in error detection, recall and precision should be perfectly balanced. However, this goal is difficult to optimize directly and needs further experimentation.

In our work, we experimented with only one way (two preceding and one succeeding token) for introducing textual context into the translation model. Context is needed for high-quality OCR post-correction with character-based MT, but other ways of integrating it might work as well.

A more sophisticated model representation of glyph shapes that are more similar to the historical fonts used for typesetting is worth exploring. Furthermore, our experiments

suggest that it would be interesting to train models on data from smaller time periods, e.g. only documents from 1800 to 1850.

Finally, one should better analyze how the training material size, relative error frequency, language, text type, time span and typeface influence the OCR post-correction quality. Our experiments with different data sets with different properties often showed strongly varying or inconsistent effects. Conclusive and generally applicable insights are hard to achieve, but they are strongly needed for a practical application of supervised OCR post-correction.

## 7 Conclusion

This article presented a broad overview as well as extensive experiments and evaluations on how character-based neural and statistical machine translation techniques can be used for OCR post-correction. We showed that SMT systems perform better in error correction, while NMT systems achieve higher results in error detection. This is important to know for anyone who plans to employ character-based MT for any of these subtasks.

We tested both state-of-the-art and novel strategies to include more information in the training and translation process of NMT systems. Giving enough context to the systems for a correction candidate allows better detection of real word errors and increases the training material. We found that no improvement in OCR post-correction can be achieved when data sets with different error characteristics are combined. However, when the individual training examples are labelled with their data set characteristics as factors, neural systems are able to generalize from the increased training sets and produce better results, especially for error correction. Specifically, data sets with considerably varying error rates profit from time factors. Another insight is that error-focused models can boost error detection, but have a rather negative influence on error correction. Correctly calibrating the error threshold for subsampling the training material is essential for these models. We showed that the decrease in error correction with error-focused models can be mitigated by using ensemble decoding. In fact, normal ensembling of single systems' output is useful for OCR correction, even more so, if different systems are combined for ensemble decoding. Finally, we experimented with a novel, straightforward approach how visual information on glyphs can be included in the training process of character-based NMT systems.

However, we also saw that for the given training sizes and the highly varied training material it is hard to achieve conclusive and generally applicable insights about the best settings and hyperparameters. We often observed some improvements on some data sets and at the same time performance deterioration or no effect on others. Thus, it is hard to reuse models for out-of-domain data sets.

A carefully compiled ensemble of our models reached best performance in ICDAR's 2017 error correction subtask and performed competitively in error detection. Due to the individual systems' strengths and weaknesses, we proposed an algorithm that combines different system outputs. The results of the shared task show that our

approach is competitive in error detection and strongly outperforms other approaches in error correction, even though we do not use any external resources. We also presented post-submission results on the ICDAR data set for the best single-model configuration, which is more practical to adapt and apply on new data sets than complex ensemble solutions. Our evaluation suggests that future work on NMT for OCR post-correction should focus on improving error detection.

## Acknowledgements

SC has been supported by the Swiss National Science Foundation under grant CRSII5_173719.

## References

Afli, H., Barrault, L., and Schwenk, H. (2015). OCR error correction using statistical machine translation. In *16th International Conference on Intelligent Text Processing and Computational Linguistics, Cairo, Egypt*.

Afli, H., Qiu, Z., Way, A., and Sheridan, P. (2016). Using SMT for OCR error correction of historical texts. In *Proceedings of LREC-2016, Portorož, Slovenia*, pages 962–965.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Bollmann, M. and Søgaard, A. (2016). Improving historical spelling normalization with bidirectional LSTMs and multi-task learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 131–139, Osaka, Japan.

Brill, E. and Moore, R. C. (2000). An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 286–293.

Chiron, G., Doucet, A., Coustaty, M., and Moreux, J.-P. (2017). ICDAR2017 competition on post-OCR text correction. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1423 –1428.

Chung, J., Cho, K., and Bengio, Y. (2016). A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176.

Eger, S., Mehler, A., et al. (2016). A comparison of four character-level string-to-string translation models for (ocr) spelling error correction. *The Prague Bulletin of Mathematical Linguistics*, 105(1):77–99.

He, W., He, Z., Wu, H., and Wang, H. (2016). Improved neural machine translation with SMT features. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 151–157. AAAI Press.

Koehn, P. (2009). *Statistical machine translation*. Cambridge University Press.

Koehn, P. (2017). Neural machine translation. *CoRR*, abs/1709.07809.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439.

Lee, J., Cho, K., and Hofmann, T. (2017). Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8).

Luong, M.-T. and Manning, C. D. (2016). Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015). Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Pettersson, E., Megyesi, B., and Tiedemann, J. (2013). An SMT approach to automatic annotation of historical text. In *Proceedings of the workshop on computational historical linguistics at NODALIDA 2013; May 22-24; 2013; Oslo; Norway. NEALT Proceedings Series 18 / Linköping Electronic Conference Proceedings 87*, pages 54–69. Linköping University Electronic Press.

Piotrowski, M. (2012). Natural language processing for historical texts. *Synthesis Lectures on Human Language Technologies*, 5(2).

Reynaert, M. (2016). OCR post-correction evaluation of early dutch books online - revisited. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA).

Rosca, M. and Breuel, T. (2016). Sequence-to-sequence neural network models for transliteration. *CoRR*, abs/1610.09565.

Schnober, C., Eger, S., Do Dinh, E.-L., and Gurevych, I. (2016). Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714, Osaka, Japan.

Schulz, S. and Kuhn, J. (2017). Multi-modular domain-tailored ocr post-correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2716–2726, Copenhagen, Denmark. Association for Computational Linguistics.

Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Miceli Barone, A. V., Mokry, J., and Nadejde, M. (2017). Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain.

Sennrich, R. and Haddow, B. (2016). Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany.

Silfverberg, M., Kauppinen, P., and Lindén, K. (2016). Data-driven spelling correction using weighted finite-state methods. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, pages 51–59, Berlin, Germany. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Volk, M., Furrer, L., and Sennrich, R. (2011). *Strategies for Reducing and Correcting OCR Errors*, pages 3–22. Springer Berlin Heidelberg, Berlin, Heidelberg.

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., and Ng, A. Y. (2016). Neural language correction with character-based attention. *CoRR*, abs/1603.09727.

Yao, K. and Zweig, G. (2015). Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. In *16th Annual Conference of the International Speech Communication Association (INTERSPEECH 2015)*, pages 3330–3334.

Zhao, S. and Zhang, Z. (2016). An efficient character-level neural machine translation. *CoRR*, abs/1608.04738.