Universität Zürich[UZH]

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Arts
der Philosophischen Fakultät der Universität Zürich

Post-Correcting OCR Errors Using Neural Machine Translation

Verfasserin: Chantal Amrhein
Matrikel-Nr: 14-703-656

Referent: Prof. Dr. Martin Volk

Betreuer: Dr. Simon Clematide

Institut für Computerlinguistik

Abgabedatum: 10.12.2017

Abstract

OCR errors are an enormous problem for the digitisation of, especially, historical texts. To explore new strategies to fix this problem, this Bachelor thesis focuses on post-correcting OCR errors using character-based neural machine translation (NMT). First, I compare the performance of NMT to previously tested approaches with character-based statistical machine translation (SMT). Furthermore, I analyse how OCR post-correction with NMT can profit from using additional information. This is investigated in different ways: both by employing well-known NMT approaches and testing novel ideas. My findings show that SMT and NMT can benefit from each other for OCR post-correction. Moreover, I discover that the data on which the MT systems are trained has a large influence on which methods work best. Finally, I show that a combination of my experiments can outperform other OCR post-correction approaches in the task of error correction and perform comparatively in error detection.

Zusammenfassung

OCR-Fehler stellen ein grosses Problem für die Digitalisierung vor allem historischer Texte dar. Um neue Erkenntnisse zu sammeln, wie dieses Problem gelöst werden kann, beschäftigt sich diese Bachelorarbeit mit der Post-Korrektur von OCR-Fehlern mit zeichenbasierter neuronaler maschineller Übersetzung (NMÜ). Zuerst vergleiche ich die Qualität von NMÜ mit früheren Experimenten mit zeichenbasierter statistischer maschineller Übersetzung (SMÜ). Zusätzlich analysiere ich, wie sich OCR Post-Korrektur mit NMÜ durch das Hinzuführen von zusätzlichen Informationen verbessern lässt. Im Besonderen wird das auf zwei Arten untersucht: Indem ich bereits bekannte Ansätze für NMÜ einsetze und indem ich neue Ideen ausprobiere. Die Ergebnisse zeigen, dass NMÜ und SMÜ voneinander profitieren können bei der Post-Korrektur von OCR-Fehlern. Weiterhin stellt sich heraus, dass die Art von Daten, auf welchen die Systeme trainiert werden, einen grossen Einfluss darauf hat, welche Methoden am besten funktionieren. Zum Schluss zeige ich, dass eine Kombination meiner Experimente andere OCR Post-Korrektur-Ansätze übertrifft, wenn es darum geht Fehler zu korrigieren, und ähnliche Resultate liefert, wenn es darum geht Fehler zu erkennen.

Acknowledgement

First of all, I would like to thank Martin Volk for his never-ending enthusiasm. You woke my fascination for computational linguistics at the beginning of my studies and always encouraged me to explore my interests.

Many thanks go to my supervisor Simon Clematide. It was always a pleasure to work with you, and I appreciated your valuable feedback, our interesting discussions and your continuous support throughout my work on this thesis.

Additionally, I am extremely grateful to Natalia Korchagina for her helpful insights into her work with character-based machine translation and to Tilia Ellendorff for the initial idea to experiment with images in this Bachelor thesis.

A heartfelt thank you to my best friends who shared my joy in moments of success and, likewise, never failed to build me up when the struggle was real.

Last but not least, I want to thank my family who believes in me no matter what. Thank you for all you taught me and for making me the person I am today.

Without any of you, this work would never have been written.

Contents

List of Figures

List of Tables

List of Acronyms

BnF	Bibliothèque nationale de France
IAPR	International Association of Pattern Recognition
ICDAR	International Conference on Document Analysis and Recognition
MERT	Minimum Error Rate Training
MT	Machine Translation
NLP	Natural Language Processing
NMT	Neural Machine Translation
OCR	Optical Character Recognition
RNN	Recurrent Neural Network
SMT	Statistical Machine Translation

1 Introduction

In the digital age, optical character recognition (OCR) is an important processing step for text digitisation, and it becomes more and more important with the growing interest in digital humanities. Unfortunately, the output of OCR systems is often faulty. Especially historical documents pose a challenge due to both orthographic and typographic variation and sometimes also poor condition of the source material (e.g. stains on the documents). Therefore, it is often necessary to take measures to improve the quality of OCR-generated text.

Recently, more and more natural language processing (NLP) tasks have profited from the use of neural networks. Especially in machine translation (MT), neural machine translation (NMT) managed to outperform statistical machine translation (SMT) with much better results. 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.

Therefore, this thesis focuses on answering the following two research questions:

1. How does character-based NMT perform compared to character-based SMT in the case of OCR post-correction?

2. How can these approaches be improved by using more information during training and translation?

In the following chapter, I discuss previous works relevant to OCR post-correction, neural networks used for NLP tasks and character-based NMT. In Chapter 3, I give an overview of the data used in this thesis and describe its properties. Chapter 4 presents the methodology used for my experiments, which are then evaluated in Chapter 5. Finally, I summarise the results in Chapter 6. More detailed graphs and tables are presented in Appendix A.

2 Related Work

2.1 OCR post-correction

OCR errors are a severe problem for digitised texts, and it is often necessary to avoid or correct them. As described in Volk et al. [2011], there are three possible ways to achieve this: 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 has focused on the last option. Kukich [1992] discusses various automatic methods to find and correct errors, such as n-gram or dictionary-based techniques. These methods have frequently been used for correcting OCR output.

Generally, OCR post-correction can be seen as a special case of spelling error correction. Eger et al. [2016] give an overview of common error correction techniques and compare them to general NLP string-to-string translation models for an OCR data set. They show that the latter models can achieve significantly better results than for example (weighted) edit distance or the noisy channel model [Brill and Moore, 2000].

Based on the idea that OCR post-correction can be perceived as a translation task, Afli et al. [2016] have successfully trained an SMT system to translate historical OCR-generated text with errors into corrected text. Their system managed to outperform language model based techniques for OCR post-correction. However, for the training of their models, they used a data set of more than 60 million OCR output tokens. This exceeds the data set used in this Bachelor thesis by far. For their experiments, a word-level SMT system was used. In previous experiments, Afli et al. concluded that word-level SMT systems perform better than character-level systems for OCR post-correction [Afli et al., 2015]. The size of the training set there was over 90 million OCR output words. It is therefore interesting to see, how character-level MT performs if the available training data is much smaller.

Motivated by the above work, this Bachelor thesis shall give an insight into how NMT systems perform in the correction of noisy OCR output. A similar approach has been tested in a Master project by Valette [2017]. However, in contrast to his

work, I train a supervised end-to-end system without separating error detection and correction and without external lexical resources.

2.2 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., my work will work with data that does not only contain misrecognized words. Therefore, the task becomes much harder since the systems also need to detect erroneous words to correct them accurately.

2.3 Character-based NMT

Machine Translation with neural networks has been very popular over the last few years. Even though such systems achieve much better results than SMT systems, there is always a trade-off between OOV words and computation time. NMT systems need 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 use it in practice for translation.

To address this drawback, 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. This approach mainly focused on using smaller units than words for translation. Luong and Manning [2016] use a hybrid-system which translates primarily on word-level and falls back to a character-based representation for out-of-vocabulary 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.

Going one level deeper, 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 generalisation for unseen word forms.

Extending this idea, Lee et al. [2017] aimed to design a fully character-based NMT system without explicit segmentation. They show that such an architecture is also beneficial for a multilingual many-to-one translation environment. More importantly for this Bachelor thesis, they also 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, this Bachelor thesis will employ character-based NMT in post-correcting OCR errors. Since character-based NMT proved to handle spelling errors well it is not far-fetched to believe that it will also perform well on OCR errors.

3 Materials

3.1 ICDAR2017 OCR Post-correction Data

The data for the experiments of this Bachelor thesis comes from the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR 2017). Every second year, competitions in various subfields concerning document analysis problems are organised. One of these competitions in 2017 was dedicated to OCR post-correction[1].

The full data set for this competition consists of around 12M OCRed characters and their alignments to a gold standard[2]. It consists of an equal split of English and French data. The set is a subpart from a corpus that was built in the AmeliOCR project[3] realised by the L3i laboratory (University of La Rochelle, France) and the National Library of France (BnF).

The data is distributed as a training set consisting of 10M characters (83% of the full data set) and an evaluation set of 2M characters (17%). The data is split into multiple files, most likely each referring to an individual paragraph. Every file is distributed with three lines of text. The following example shows an excerpt from the training data:

(3.1) "[OCR_toInput] `ATRAVELLER STOPPED AT A WIDOW'S GATE.`"
"[OCR_aligned] `A@TRAVELLER STOPPED AT A WIDOW'S GATE.`"
"[GS_aligned] `A TRAVELLER STOPPED AT A WIDOW@S GATE.`"
Example from a document written in 1860.

The first line contains the original output of the OCR system. On the second line, the OCRed text is aligned with the gold standard. This means that for every OCRed character the aligned gold standard character can be found directly below. Wherever

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

[2] The term "gold standard" is used with "ground truth" interchangeably.

[3] `http://actions-recherche.bnf.fr/BnF/anirw3.nsf/IX01/A2016000030_post-correction-d-ocr-pour-les-ouvrages-anciens-en-exploitant-les-associations-lexicales-de-l-ocr-bruite`

a character has to be inserted to match the length of the gold standard (which means there is no possible alignment) an "@" is inserted. The last line contains the gold standard. Here, the "@" is added for all characters that appear in the OCRed text but cannot be aligned with a character in the gold standard. Any characters or longer segments that could not be identified with certainty in the original image are aligned with the "#" character in the gold standard.

It is a challenging data set since the documents origin from different collections, e.g. the BnF and the British Library. The texts were all published either in periodicals or monographs. Additionally, the data covers a considerable time span. In the training set, the oldest document is from the year 1654 and the most recent from the year 2000. Unfortunately, for some documents the publication year is unknown. Of those where this information is known, about 92% origin from the 19th century. Despite this substantial majority, there is still much variance in spelling both of typographic as well as orthographic nature. For example, consider the difference in the spelling of "completed" between the middle of the 18th century and the end of the 19th century:

(3.2) "[...] all her Forces to be *compleated* without delay."
Example from a document written in 1744.

(3.3) "[...] other parts of the kingdom, where railroads are *completed* [...]."
Example from a document written in 1894.

With this example, it becomes clear that diachronic documents pose a great challenge to OCR post-correction. First of all, the OCR quality will already be worse since such texts are often harder to recognise correctly for OCR systems. For lexicon-based approaches, it is essential that as many historical spelling variants as possible are included. This is also an issue for learning algorithms. Especially for supervised methods, having many distinct spellings can create data sparsity. Furthermore, models have to deal with special characters, e.g. " ſ " (long s). The data contains many hyphens for which the gold standard is often very inconsistent. Following the decision of the organisers of the OCR post-correction competition, all hyphens are ignored in both the error analyses in this chapter as well as in all evaluations. They are not removed from the data in order to use the original training setting, but they are simply ignored when producing any statistics or evaluating translation outputs.

Table 1 gives an overview of the operations needed to correct the OCR errors in the training data. Insertions and deletions are measured counting the "@" characters in the aligned OCRed text and the aligned gold standard respectively. Other pairs of aligned characters that do not match are counted as substitutions. A character is

	Monograph		Periodical	
Language	**fr**	**en**	**fr**	**en**
Total # of Chars in Gold Standard	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%**
Unrecognisable	0.42%	3.32%	5.27%	9.33%

Table 1: Edit operations needed to correct OCRed training data (excluding hyphens): Percentage of total characters that need to be inserted, deleted, substituted in the OCR output or are unrecognisable in the gold standard. Relative numbers are measured over the total number of characters in the original OCR output.

classified as unrecognisable if it is aligned with an "#" in the gold standard. Overall, the texts from the periodicals need more correction operations, possibly due to worse quality of their images. Interestingly, the English texts seem to contain many more unrecognisable characters than the French texts for both document types.

Table 2 takes a closer look at the ten most frequent substitutions over characters ngrams. The substitutions were counted as illustrated in the following artificial example:

(3.4) "[...] `the rnassive bouse` [...]" (original OCR output)
"[...] `the @massive house` [...]" (gold standard)
Substitutions found: rn → m and b → h

The table suggests that the frequency of certain substitutions is very text type specific. Most of the substitutions can be easily explained due to the visual similarity of the characters, e.g. mistaking "v" for "u". However, it is perplexing to see that "é" is recognised instead of "e" and the other way around in the English monograph texts. A quick look at the data shows that sometimes there are French citations in the English texts. An example of code-switching can be seen below. Finally, there seem to be many substitutions for punctuation. A comma is often recognised instead of a full stop and vice versa.

	Monograph		Periodical		fr		en		both	
Language	**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 2: 10 most common substitutions of characters ngrams (excluding hyphens). → means x was recognised "instead of" y.

(3.5) "[...] his astonishment over Martine's de-scription of the apparel of Sganarelle in *Le médecin malgré lui. Un habit jaune et vart ! C'est done le médecin des perroquets.* We light-minded [...]."
Gold standard example from a document written in 1888.

From the corrections needed on word-level, it can be observed that most wrong tokens only appear once in the whole data set. Table 3 shows the percentage of hapax legomena over all tokens that are wrong in the OCR output. I exclude any punctuation characters to compute which tokens need to be corrected for this analysis, because there are many errors where only a punctuation character needs to be added or removed. If these errors were considered, there would be many hapax legomena tokens where, for example, only a comma needs to be removed from the end. With this measure, only non-punctuation errors are counted like recognising "rnessage" instead of "message" but not a comma instead of a full stop. The large percentage of hapax legomena suggests that performing post-correction on character-level will be more beneficial than on word-level.

Table 4 shows the ten most common tokens that need to be corrected for both languages. It is interesting to see that for the English texts there are four misspelt versions of "the" among the ten most frequent misrecognized words. The average number of spelling versions per misrecognized gold standard type is 1.7 for the English data set and 1.77 for the French data set. The English word with the most

	fr	en
Total Mistakes	51699	47583
Total Hapax Legomena Mistakes	46644	41487
Percentage of Hapax Legomena in original OCR output	90.22%	87.19%

Table 3: Percentage of hapax legomena considering all tokens that need to be corrected in the OCR output (excluding hyphens).

error diversity is "the" with 275 distinct variants, for French, it is "de" with 214 distinct variants. All of these findings suggest that a character-based approach is more suitable for this task and data set.

	fr			en		
	OCR	GS	Freq	OCR	GS	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

Table 4: Ten most frequent errors on word-level (excluding hyphens).

Finally, it is interesting to see how errors develop over time. Figure 1 is a screenshot from an interactive visualisation of the frequency development of the ten most frequent errors per 30 years in the English monograph data. The numbers are divided by the number of files for the specific time span to normalise the amount of data. Unfortunately, the available colour palettes did either not cover the many data tracks or could not be visually distinguished from each other. In the interactive visualisation, the error would be shown by hovering over a line, or other errors could be faded out such that the development of one error can be seen better.

However, for the sake of this argument, it is not essential to know which error belongs to which line. Instead, it is astonishing to see the error development. Notice that the scale is logarithmic. For the texts around 1800, the most frequent errors occur by far more often than the most frequent errors in later years. The most frequent error in 1800 was observed around 2000 times, while for example around 1860 the most frequent error lies around a value of 2. Also, most of the errors in early documents rarely happen in files that were published later.

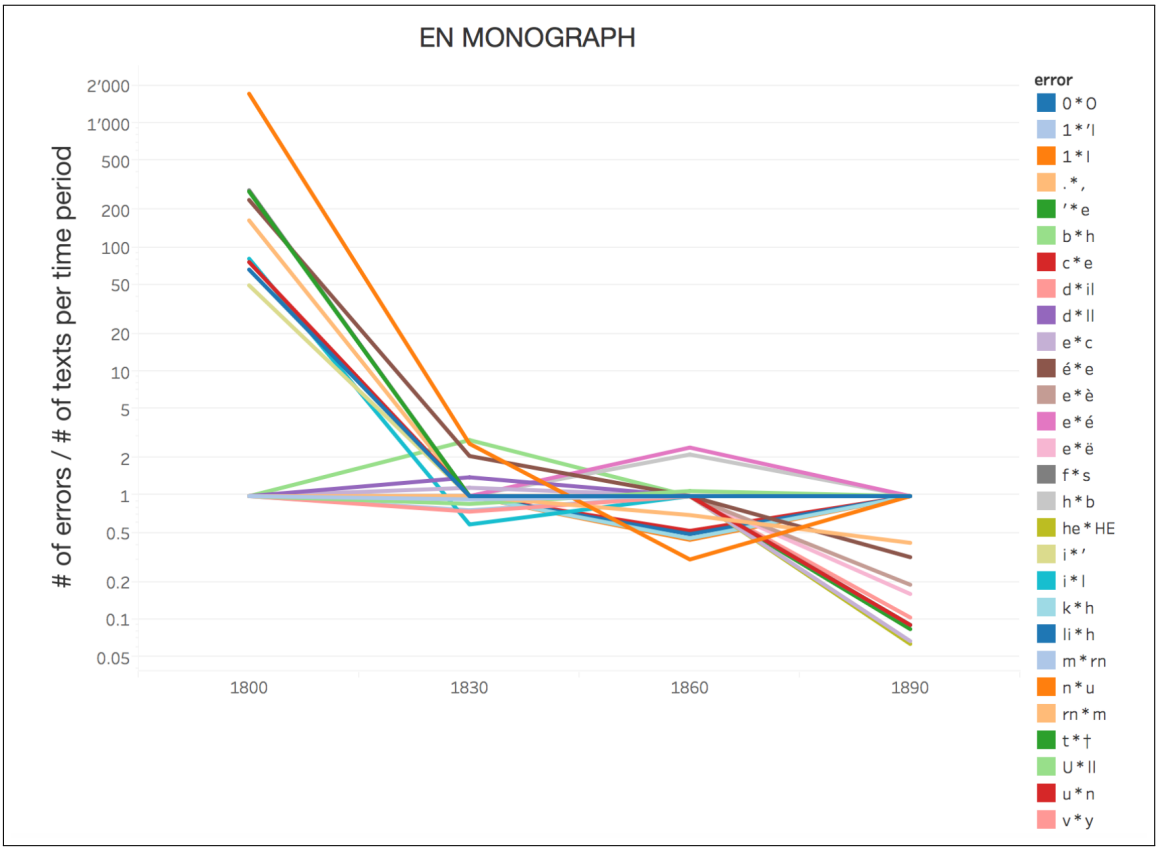

Figure 1: Errors over time in English monograph data. Frequency of ten most frequent errors per 30 years divided by number of files on logarithmic scale.

Due to spatial reasons, the visualisations for the other data sets are included in Appendix A. The graph for the English monograph was chosen in this section because it is the most interesting to discuss since the drop from a frequency of 2000 to 2 is striking. Even though the pictures for the individual data do not show such apparent differences between the periods, similar observations can be made about the frequency span of an error over time. Thus, it can be concluded that the different time periods all offer distinct challenges for OCR post-correction.

4 Methods

In the following sections, I present the approaches used in my experiments. First, I introduce the experimental setup and the preparation of the data. Then I describe my baseline systems and show how additional information can be included in these systems. In particular, I use techniques from SMT and NMT for my experiments. I do not go into great detail about how SMT and NMT work but I refer to the most important works which give an excellent introduction to the techniques I use. Furthermore, I describe how approaches like factored NMT and ensemble decoding can be applied in the context of OCR post-correction. Additionally, I propose a new technique on how glyphs can be used as embeddings for NMT systems, and I explore different strategies of transforming the data set, e.g. adding more translation context, building error-focused training sets and combining different data sets. Finally, I present our submission to the ICDAR 2017 competition on OCR post-correction.

4.1 Experimental Setup

Unfortunately, it was not possible to obtain the original test set gold standard in time for my experiments. Therefore, the training data released for the ICDAR 2017 OCR post-correction shared task was split again into a new training, development (dev) and test set. For each file, a random split of 10% was used for testing, another 10% as the dev set and the rest for training. This design choice was made due to the various orthographic and typographic changes and different error frequencies over time. If the training set overrepresented a particular time span, the MT systems would overfit to correcting OCR errors from this time span. Furthermore, characters that were aligned with a # in the gold standard, meaning they were unrecognisable when creating the gold standard, were ignored while building the train and dev set and excluded in the evaluation of the test set.

Table 5 shows the resulting data sets. These were the starting point for all further experiments, and whenever the data needed to be in a specific format, these data sets were transformed to fit it.

	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: Size of train, dev and test set in characters (including whitespace and hyphens) for all four document types. The combined French and English data sets were constructed by concatenating all available data (periodical & monograph). The last column shows the data set consisting of all, both French and English, data.

Chapter 3 illustrated that an enormous percentage of errors are hapax legomena errors, meaning they only occur once. This motivates my decision to use character-based approaches in my experiments. In contrast to word-based machine translation, one word or a sequence of more than one words are translated instead of a whole sentence. This is due to limited memory. The length of the text that should be translated is much longer since in character-based MT every character is treated as a separate word.

Tokenisation is not as important for a character-based approach as for a word-based approach. Therefore, the data sets described in Table 5 were processed further using only whitespace tokenisation. Whenever two space characters were aligned in the OCR output and the gold standard, a newline character was introduced to create a verticalised format. Having one word per line makes the training of the MT systems less complex. However, this results in a context-insensitive approach. Between each character, a space is inserted to force the MT systems to work on character-basis. All @ characters occurring in the original OCR text as well as in the gold standard were deleted.

(4.1) “This@couise@was agretd to, [...]” Original OCR text.
“This course was agreed to, [...]” Aligned gold standard.

(4.2)

T h i s c o u i s e w a s	T h i s c o u r s e w a s
a g r e t d	a g r e e d
t o ,	t o ,
(OCR text)	(Gold standard)

The resulting data format can be seen above. Example 4.1 shows an extract from the English periodical training data set. Example 4.2 shows the same extract in the format given to the MT systems. Note that all @-characters are removed, and the first OCRed word needs to be split into three separate words in the gold standard.

4.2 Character-based SMT

To have a baseline for the character-based NMT models, I trained character-based SMT systems on the ICDAR 2017 OCR post-correction data sets analogous to the experiments of Pettersson et al. [2013]. The core components of SMT systems are a translation model to translate words from a source to a target language, a language model to focus on translations that are as natural as possible and a reordering model which can change the word order. For a more extensive introduction to SMT, please refer to [Koehn, 2009].

All SMT systems were trained with the Moses[1] toolkit using GIZA++[2] for character alignment and MERT[3] for optimisation. Since training an SMT system on character-level does not generate a large vocabulary, it is possible to train a higher order language model. I used a 10-gram language model in all experiments.

4.3 Character-based NMT

There are different NMT frameworks which can be used for OCR post-correction. First, I chose to test the NMT system described in Lee et al. [2017]. This is an implementation which works on character-level by design. The architecture which they describe in their paper is different from other NMT systems. Instead of the bi-directional recurrent input layer, they use a set of convolutional filters to capture local dependencies of characters. Early experiments with this framework did not produce satisfactory results. In most cases, more errors were introduced than corrected.

Since Lee et al.'s framework did not produce acceptable results for OCR post-correction, I tested another framework which does not translate on character-level by design. Nematus[4] is a well-know NMT framework described in Sennrich et al.

[1] [Koehn et al., 2007]

[2] [Och and Ney, 2003]

[3] [Och, 2003]

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

[2017]. Using the same strategy as for SMT with Moses, it can also translate on character-level. Therefore, the input format for the NMT systems is the same as for the SMT systems. Initial experiments with Nematus convinced me to use this framework in all subsequent NMT experiments.

As discussed in Section 2.3, a fixed-sized vocabulary is used for translation with NMT systems. In order to have the option to combine individual systems in later experiments, I decided to build one large vocabulary over all data sets and use it for all the experiments.

Neural networks have many hyperparameters that can influence their performance. I decided to test different hyperparameters before performing my experiments, to find the best conditions for correcting OCR errors with NMT systems. In particular, I investigated how embedding size, the amount of dropout, the batch size and the use of tied embeddings affect the OCR error detection and correction quality. For a more extensive introduction to NMT, please refer to [Koehn, 2017].

Embeddings are encoded one-hot vectors which are projected into a dense, continuous vector space with a much lower dimension. The embedding size refers to the number of dimensions in the dense representation. Dropout is a state of the art technique in NMT. During training, some units in the network are switched off. Consequently, the model does not have access to all the information and is less prone to overfit on the training data. The amount of dropout regulates how many units are switched off at the same time. Batch size refers to the number of training examples which are used to compute the gradient and update the weights in the network. Larger batch size leads to a speed-up in training on GPUs. Lastly, I tested the usefulness of tied embeddings. This means that the embeddings of the target side are tied to the embeddings of the source side. In the case of OCR post-correction, the embedding for the character “f” would be the same for the input as for the translated output since they are always updated together.

To save some time with these configuration experiments, I only tested them on the French periodical data. There are two main reasons for this decision: First, it is one of the smaller data sets and, thus, it takes less time to train a system and, second, my experiments with SMT had already shown that this data set is harder to correct than others. Therefore, it is a suitable choice to make some assumptions about all the data. Of course, it would be better to evaluate this more systematically but it is not feasible to do this within the scope of this thesis.

Embedding Size	Dropout	Batch Size	Tied Embeddings
32	0	50	yes *
256 *	0.2 *	100 *	no
512		200	

Table 6: All possible options for each hyper-parameter. Parameter used in the base configuration are marked with an asterisk.

Table 6 shows all possible parameter options. The parameters which I used in the base configuration are marked with an asterisk. This means that whenever I tested the options for one parameter, the others were set to the asterisk option. These are the options which initially I assumed to work best for OCR post-correction. For the first three parameters, the base configuration did indeed work better than the other options in my experiments. For the last option, not using tied embeddings achieved better results in error detection but worse results in error correction. Therefore, both options could be have been chosen. Eventually, I decided to use tied embeddings since I also experienced a training time speed-up when using them.

Additionally, there are a few hyperparameter for which I did not evaluate which value is best. I used the default values from Nematus for the hidden layer size (1000), the optimizer (adadelta), the gradient clipping threshold (1) and the learning rate (0.0001). Furthermore, I 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.

4.4 Increased Training Material for SMT and NMT

In the first experiments, the context was ignored when translating a token. Hence, the verticalized one-token-per-line format described in example 4.1 could directly be used for training and testing. However, this also means that there is little information that the MT system can work with. Therefore, it makes sense to think about how more information can be fed into an MT system.

Since OCR errors mostly occur due to the visual similarity of characters, it is possible that the same errors occur in the OCRed periodicals as in the monographs and also in both languages. Moreover, in Section 3.1, I also found that there is some code-switching within the individual data sets. With these assumptions, it would be

sensible to combine the individual training sets and thus increase the available training material. Therefore, the traditional method of adding more training material in MT can be simulated in the scenario of the ICDAR competition by combining training sets from the different languages and text types: one with both French periodical and monograph data, one with both English periodical and monograph data and, finally, one combining all the available data.

4.5 Using More Context for SMT and NMT

Besides increasing the training material, another simple method to give more information to an MT system is using more context for every training example instead of using more training examples. By adding context to the word that should be corrected, I quadruple the training material since the model sees every word three more times as context words.

In addition to this benefit, there is another reason why context should be used for OCR post-correction. Some OCR-errors generate wrong tokens which are valid words of the document language. Unfortunately, these real word errors cannot be corrected in isolation, not even with the help of a dictionary. For example, in Table 4 the most frequent French OCR-error on word-level was the word “sont” which was wrongly recognised by the OCR system as “font”. The characters “s” and “f” themselves do not look very similar. However, it is likely that in these documents a font was used which used ligature s, “ ſ ”, which looks indeed very similar to an “f”. The misrecognized word “font” cannot be detected in isolation since it is part of the French language. However, with some context, it may be possible to produce the correct word. Therefore, the base experiment with SMT and NMT were conducted again, but this time with more context for the word that should be corrected. This means that the input data format for the MT systems had to be adapted slightly:

(4.3) “Si ces principes font fondés sur le goût [...]” Original OCR text.
“Si ces principes sont fondés sur le goût [...]” Aligned gold standard.

(4.4)

S i || c e s || p r i n c i p e s || f o n t	S i || c e s || p r i n c i p e s || s 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	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 || f o n t || f o n d é s || s u r	p r i n c i p e s || s o n t || f o n d é s || s u r
f o n t || f o n d é s || s u r || l e	s o n t || f o n d é s || s u r || l e
(OCR text)	(Gold standard)

The resulting data format can be seen in example 4.4. In the actual training data the word boundary, in this example marked with ||, needed to be a character which

otherwise could not occur in any of the training data. Therefore, an emoji character was chosen as an artificial word boundary marker.

Interestingly, forcing the MT system also to produce the context on the target side produced better results than only producing the word that was in focus at that moment. However, this means that (especially in the case of NMT) it is possible that not all of the three word-boundary emojis are produced in a translation. Whenever this happened, I chose the word from the translation which had the smallest Levenshtein distance to the actual word in the OCR output to be the final translation of the word in focus.

4.6 Factored Character-based NMT

In the sections above, I experimented with giving the MT systems more information by adding more training material from the combined data sets and by providing more context. However, since Nematus allows the training of "factored" NMT models, even more information can be included: the time span from which the text originates, whether it is a periodical or a monograph and finally what language it is.

The number of factors depends on what training set was used. For the four original, type- and language-specific sets only the information on when a text was published was used. This experiment was tested both with context-sensitive and context-insensitive data. Motivated by Figure 1 on Page 10, it is clear why a time factor makes sense. If the publication period is known, errors from a specific period can more easily be detected and, thus, better be corrected. To create this factor, I divided the files into bins of 50 years. This is a rather large time span, but, with smaller bins there is the risk of data sparsity problems. The text in example 4.6 was written between 1800 and 1849. Thus, divided by 50, its time span factor is 36. This information could be useful to help detect OCR errors which occur due to orthographical or typographical changes.

For the medium-sized training set, where the two text types were combined for each language, I used an additional factor. The factors consist of the publication time and the text type for each training example. The largest combined training set consisted of all training data that was available. Hence, another factor was added which specified to which language the training example belongs to. In this way, factors can be used to label the individual training examples such that it is known which training set they belong to. Due to time restrictions, these experiments were only conducted for the data sets without context.

(4.5) “[...] who may wish [...]” Original OCR text.
“[...] who may wish [...]” Aligned gold standard.

(4.6)

w|en|peri|36 h|en|peri|36 o|en|peri|36	w h o
m|en|peri|36 a|en|peri|36 y|en|peri|36	m a y
w|en|peri|36 i|en|peri|36 s|en|peri|36 h|en|peri|36	w i s h
(OCR text)	(Gold standard)

Example 4.6 shows the input format with all possible factors. The character | marks the beginning of a new factor. The first factor here is the language code, the second refers to the text type, and the third stands for the time span when the text was written. In contrast to factored SMT factored NMT does not translate on different levels. Rather, the embeddings are extended by concatenating the word embeddings (in this thesis character embeddings) and embeddings for each factor. Due to a bug in Nematus, I could not use tied-embeddings for the factored experiments. Detailed factored NMT with Nematus is described in Sennrich and Haddow [2016]. I did not test translating with factors for SMT, since it would not be directly comparable and is, thus, out of the scope of this Bachelor thesis.

4.7 Glyph Embeddings

Another interesting case for increased information is the use of glyph images as pre-trained embeddings. State of the art NMT often uses pre-trained embeddings instead of standard normal distribution initialisation. However, these pre-trained embeddings are mainly trained to put words in the vector space closer together if they share the same context. For OCR post-correction, characters do not necessarily need to be substituted with characters that often occur in the same context but rather visually similar characters. Therefore, it makes sense to generate non-randomised embeddings that have some intuition about the visual similarity of two characters.

Since the embedding size in the experiments so far was set to 256, the initialised embeddings can be replaced by pixel vectors, which were generated by reading the grey-scale values of a 16 by 16 pixel image of every character in the vocabulary. However, the source images were not released for the competition, and it was not possible to know the correct font to generate these images. Instead, I simply used the standard font Helvetica. An example can be seen below in Figure 2.

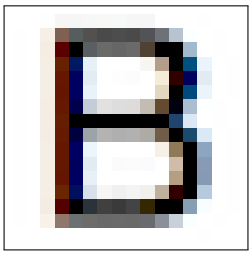

Figure 2: Character “B” as a 16 by 16 pixel image.

To exchange the randomised embeddings with the pixel vectors, I started training a Nematus model with the desired parameters, while I set the “max_epochs” parameter to zero. This way the training process stopped automatically as soon as the initialised model was generated. Then I retrieved the embedding lookup table. For each character in the vocabulary, I read an image of the character which I converted to grayscale. Then I read the pixels into a vector and normalised the values such that they were fit into a range between -1 and 1. Finally, I scaled them by 0.01 as it is done in the Nematus toolkit for the randomly initialised values as well.

After having generated the new glyph embeddings, I inserted them into the Nematus model and restarted training with this model. Of course, there are other ways how visual information could be included in an NMT system. However, this is a straightforward and time-saving technique and was therefore used in this experiment. I only tested the impact of using glyph embeddings in the context-sensitive experiments.

4.8 Error-focused Models

Most of the OCR generated data is very 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. However, these systems might over-correct while translating. Therefore, it is essential to find a reasonable threshold of the percentage of non-error tokens in the training data, to boost the error detection recall but not affecting precision too much.

For my experiments, I first tried a split of 75% error and 25% non-error tokens. Since there are fewer errors in the monograph training data, these values did not work well for these data sets. Consequently, I set the threshold to 50% errors for

French monograph and for the English monograph to 37.5% errors. I filtered out random examples which did not contain errors until the error threshold mirrored the number of errors in the training data. All experiments were done only with the context-aware data.

4.9 Ensemble Decoding

The idea of adding more information to a single model can be taken even further by using ensemble decoding. This is a common technique in NMT. For this, multiple models are combined in decoding. This is done by averaging the individual probability distributions of all the models. The reasoning behind ensemble decoding is that combining more models evens out the variance between single models' outputs.

With NMT it is possible to do ensemble decoding by using either models from the same training run at different time steps or multiple models. For my experiments, I tried both of these approaches. 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 ensembling were conducted on the context-aware data sets. Single ensembles were tested for the base context-aware NMT system and the error-focused NMT system. For single ensembles, I combined the best model with the models at the two previous time stamps. Multi ensembles were tested by using the base context-aware NMT system, the one with glyph embeddings and the error-focused system for decoding.

4.10 System Combination

The final method that is explored in this Bachelor thesis is combining multiple MT systems' outputs. The output from this post-translation combination of systems is what was submitted to the ICDAR shared task on OCR post-correction.

Different strategies were used for error detection and error correction. It has to be kept in mind that, for the error correction test set, the shared task organisers published the positions of erroneous tokens. This means that for the error detection set we translated all data and for the error correction set we only translated the erroneous tokens. A visual representation of our algorithms can be seen in Figure 3.

The error detection algorithm uses the output of five MT systems for a specific data

set which worked best in combination 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 there is an error. Third, if the original OCR token does not occur in the combined gold standard training and dev set for both text types, we assume there is an error. Finally, if the current token and one of its neighbours (both can be translated = current candidate or untranslated = OCR output) do not occur in the training set, but their concatenation does, we assume that they should be concatenated and there is an error in both tokens. The algorithm checks these steps in the order they are presented.

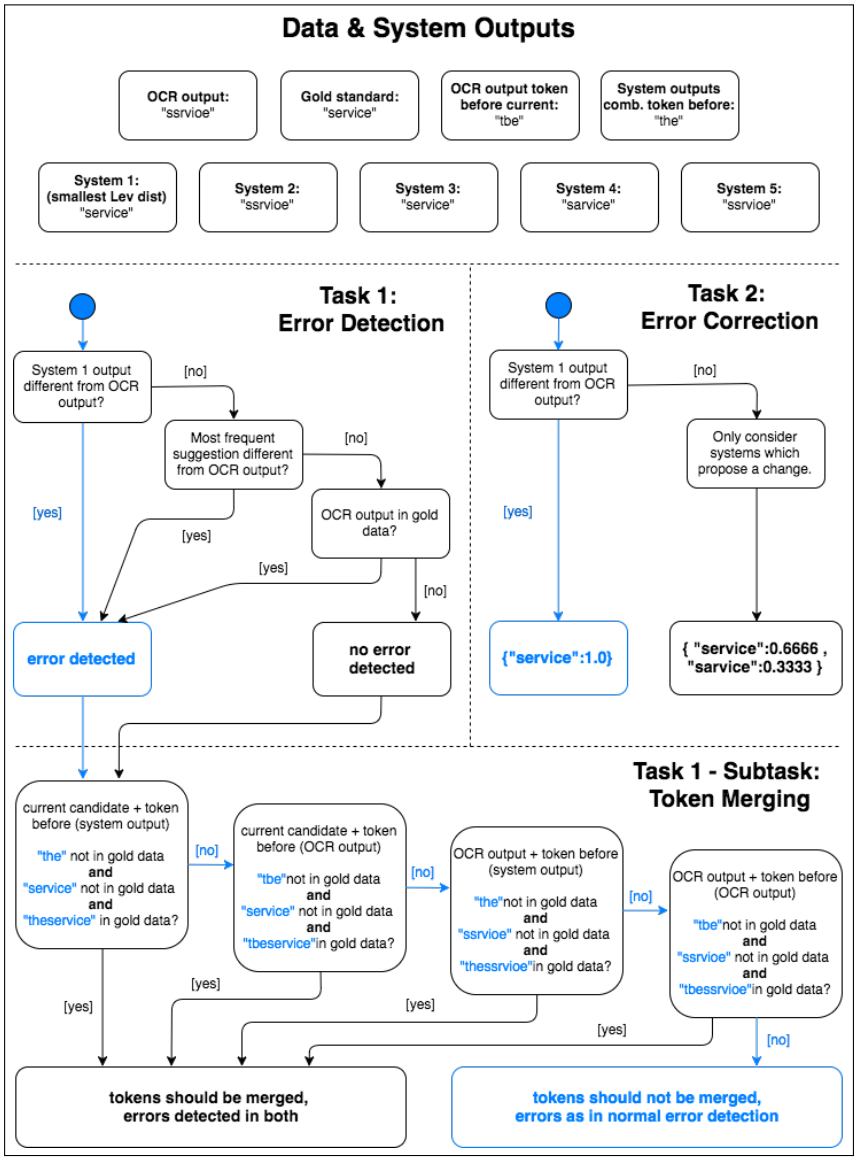

Figure 3: Algorithms for error detection and correction explained with an example.

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. An example of the OCR input and the format for the evaluation can be seen in Figure 4 and 5 respectively. The figures are taken from [Chiron et al., 2017]

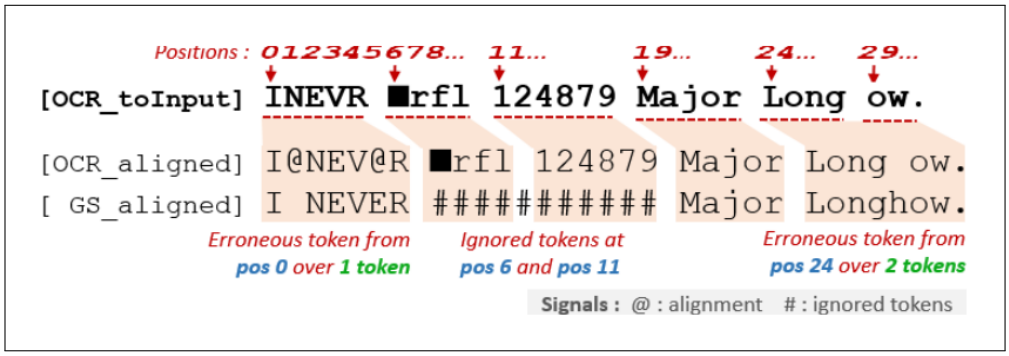

Figure 4: Sample of the training set for the competition

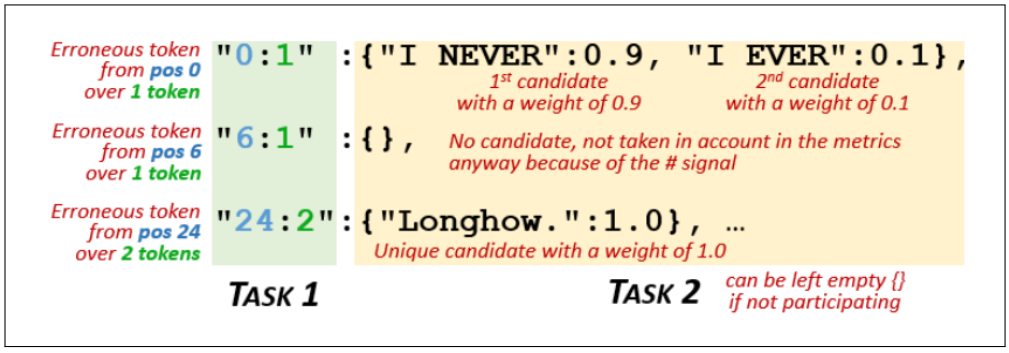

Figure 5: Format expected for submissions to both Task 1 and 2.

5 Results and Discussion

5.1 Evaluation Setup

All systems were evaluated on the test set which was generated from a subpart of the ICDAR 2017 training set (see Section 4.1). For comparison, the evaluation was done separately for error detection and error correction. Error detection is measured on word-level using precision, recall and the F1-score. Error correction is measured on character-level using Levenshtein distance between the translation candidate and the gold standard as well as the percentage of corrected tokens that match the gold standard. However, due to spatial constraints, only the F1-score and the relative Levenshtein distance improvement are presented and discussed in this section. All other results can be found in Appendix A. Please note that there are different scales in the visualisations.

As in the data overviews in Section 3.1, hyphens were ignored for both evaluation metrics since the gold standard is very inconsistent and, thus, hyphen errors occur frequently. It would take too long to train systems for cross-validation. Therefore, all results are computed by averaging the scores of three individual runs per system using the same data and the same configuration. This measure is taken to reduce the differences that occur due to systems' variance.

5.2 Increased Training Material for SMT and NMT

Traditionally, in machine translation, the most straightforward method to increase the translation quality is adding more training material by training the model on more samples. As described in Section 4.4, this was simulated by combining training sets from the different languages and text types. Figure 6 shows the results for the error detection and Figure 7 for the error correction when the medium-sized data set is used; meaning all data of one language is combined for training but tested on the individual test sets.

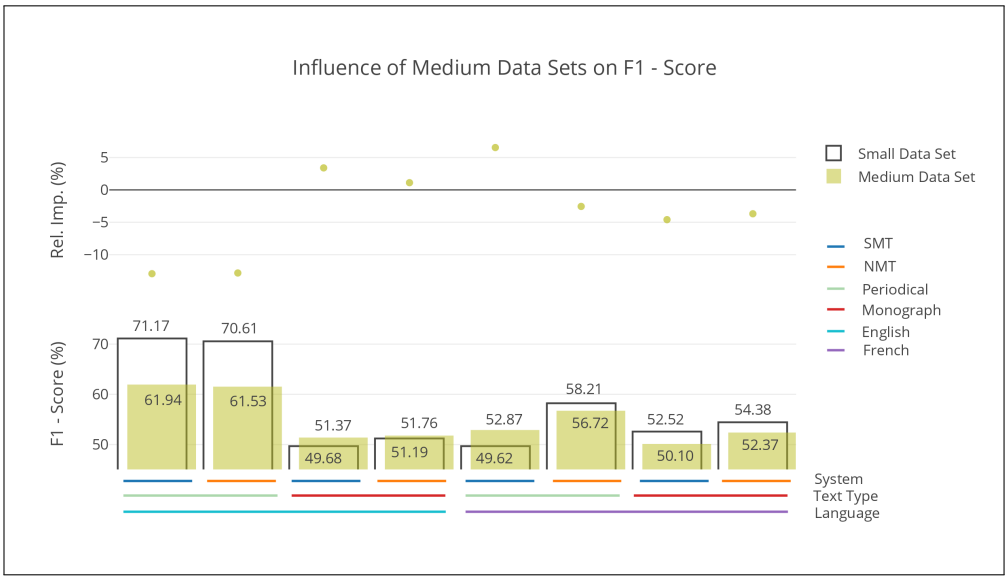

Figure 6: F1-score for the text type & language specific data set and the data sets with text type combined. Comparing SMT vs. NMT and different text types and languages.

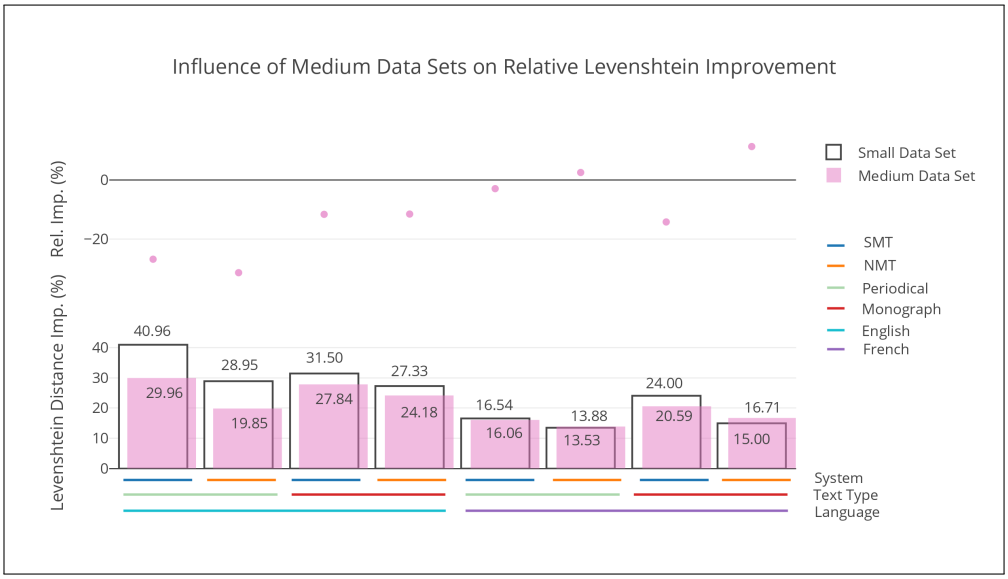

Figure 7: Relative Levenshtein distance improvement for the text type & language specific data set and the data sets with text type combined. Comparing SMT vs. NMT and different text types and languages.

The transparent columns with dark borders are the F1-scores of the context-insensitive baseline trained only on the text type and language-specific training set. The green and pink columns show the results of the combined data set for error detection and error correction respectively. The scatter plots above the bar plots show the relative

improvement from the baseline systems to those trained on the medium-sized data set. With the labels at the bottom, the results for SMT, NMT, the two languages and the different text types can be compared.

Before analysing the influence of using more training data, a few general observations can be made from these figures. First of all, it can be noticed that the SMT models perform better than NMT in error correction and worse in error detection. For error detection, the results trained only on the English periodical data set are much higher than the rest of the models trained on the individual sets. It can also be observed that error detection worked better on the periodical data. In comparison, error correction worked better on the English data sets than on French.

For the increased training data, there are also a few interesting patterns showing. In most cases, both error detection and error correction are performing worse when the two text types are combined for training. Especially for English periodicals, there is a great loss in performance both for error detection and for error correction. Since the MT models perform better on the periodical data than on the monographs adding the almost twice as large training data for the monographs will push the trained model to fit this data better. This finding is supported by the improvement on the English monograph data when the medium-sized data set is used for training. Because there are more errors in the English periodical data set, this has a positive influence on the monograph data set because errors will be detected more aggressively. In the best case, there had been an improvement on both test sets. However, the lack thereof shows that the errors occurring in the individual data sets are particular to a text type and that error frequency is an influential factor when training MT systems for OCR post-correction.

The same observations can be made for the large data set which consists of all the training material of each of the data sets. The results are shown in Figure 8 for error detection and in Figure 9 for error correction. Potentially, the correction of OCR texts on character-level could benefit from data from other languages since most of the errors are due to the visual similarity of characters and there are cases of code-switching. Both figures show that this is not the case for the given data set. The achieved scores are even lower here than for the combined text types discussed above.

Both experiments with increased training data indicate that character-based MT systems obtain a language and text-type specific model of the data.

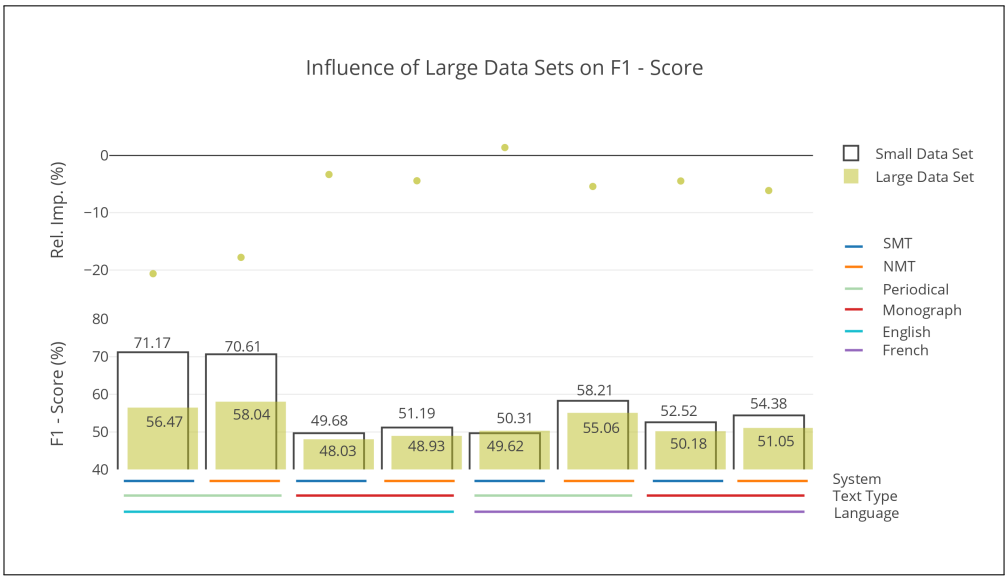

Figure 8: F1-score for the text type & language specific data set and all data sets combined. Comparing SMT vs. NMT and different text types and languages.

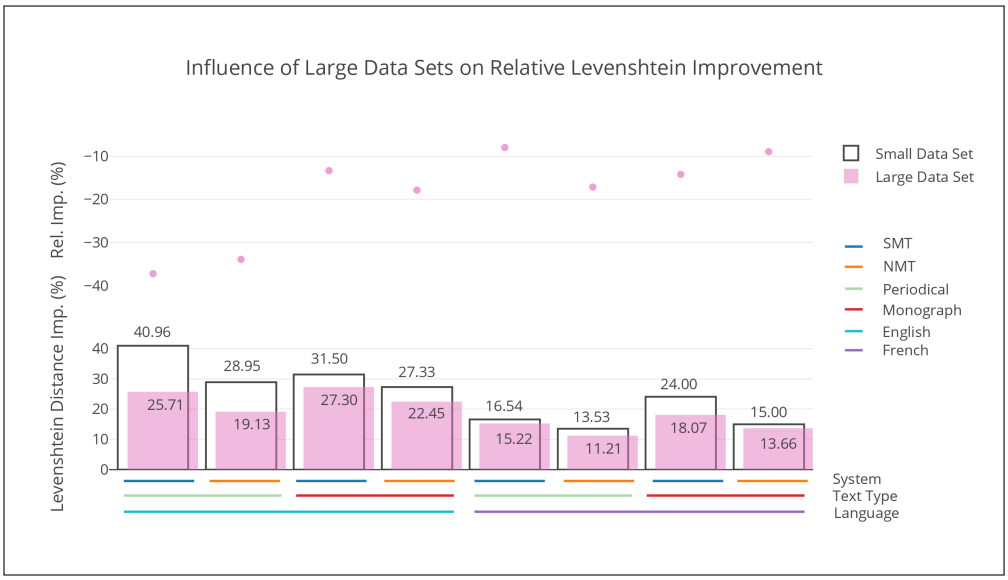

Figure 9: Relative Levenshtein distance improvement for the text type & language specific data set and all data sets combined. Comparing SMT vs. NMT and different text types and languages.

5.3 Using More Context for SMT and NMT

Since adding more training material did not prove to be very successful, it is interesting to see how other methods of adding more information to the NMT system perform. Using more context for translation turned out to be one of the most effective improvements. The results of the experiments from Section 4.5 are shown in Figure 10 and Figure 11.

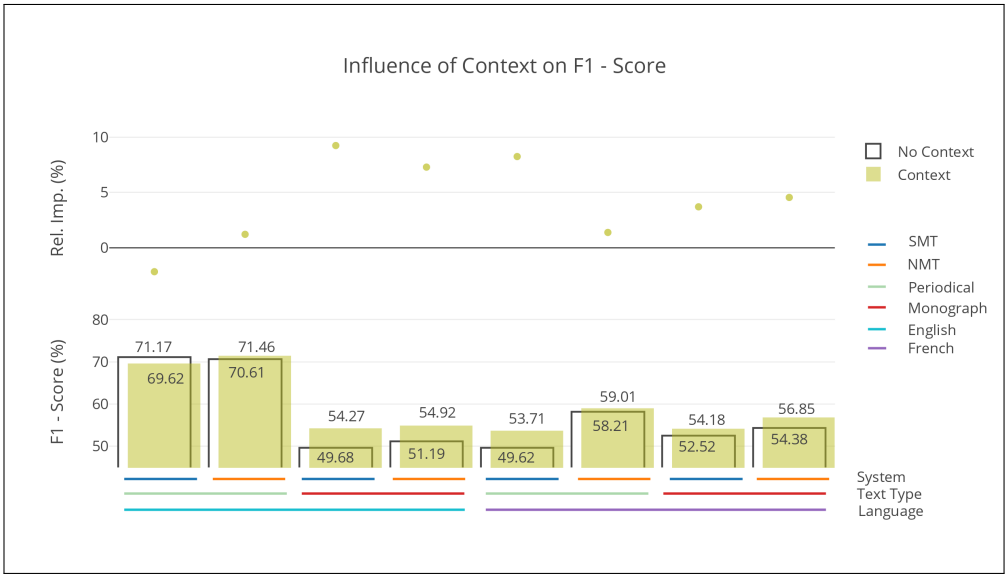

Figure 10: F1-score for context-insensitive and context-sensitive data sets. Comparing SMT vs. NMT and different text types and languages.

The differences between SMT and NMT are still the same as in the experiment above with the increased training set: Even though using more context has a positive influence on both, SMT still performs better for error correction and NMT for error detection. It is also interesting to notice that, again, the systems for the periodical data sets achieve higher F1-scores than for the monographs and the systems for English achieve higher relative Levenshtein distance improvements than for French.

Why using more context improves OCR post-correction is straightforward. Many errors cannot be detected or corrected accurately without context. The example discussed in Section 4.5 shows this quite well. The fact that all but one systems show an improvement with context confirms that there are many errors which cannot be corrected in isolation.

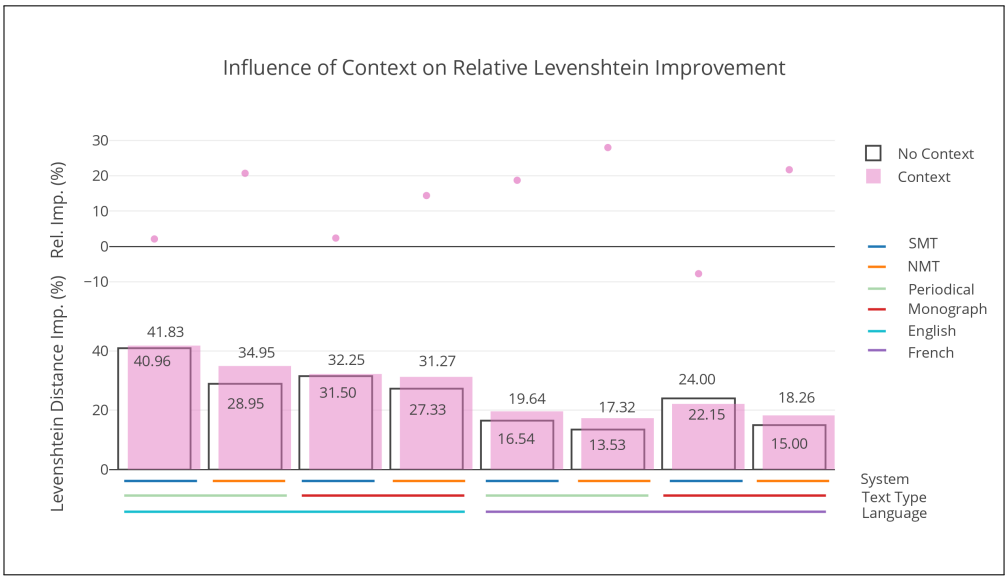

Figure 11: Relative Levenshtein distance improvement for context-insensitive and context-sensitive data sets. Comparing SMT vs. NMT and different text types and languages.

5.4 Factored Character-based NMT

Using factors as described in Section 4.6 is another strategy to include more information when training an NMT system. Figure 12 and Figure 13 show the influence of using a factor which holds information about the publication time of a text.

Overall, time factors result in a better performance in error detection for most data sets. It can be spotted immediately that using the time factor has a greater effect on the English monograph data than on the others. This phenomenon can be explained with a quick look at Figure 1 on Page 10. Most of the files belonging to the English monograph set do not need many corrections which results in a rather passive NMT model. This is problematic for the earlier files (around 1800), which contain, comparatively, much more errors than the later published files in this set. Consequently, in this case, using time factors allows the model to detect errors more aggressively on text that was published around 1800. Therefore, time factors are capable of controlling an NMT model's detection aggressiveness.

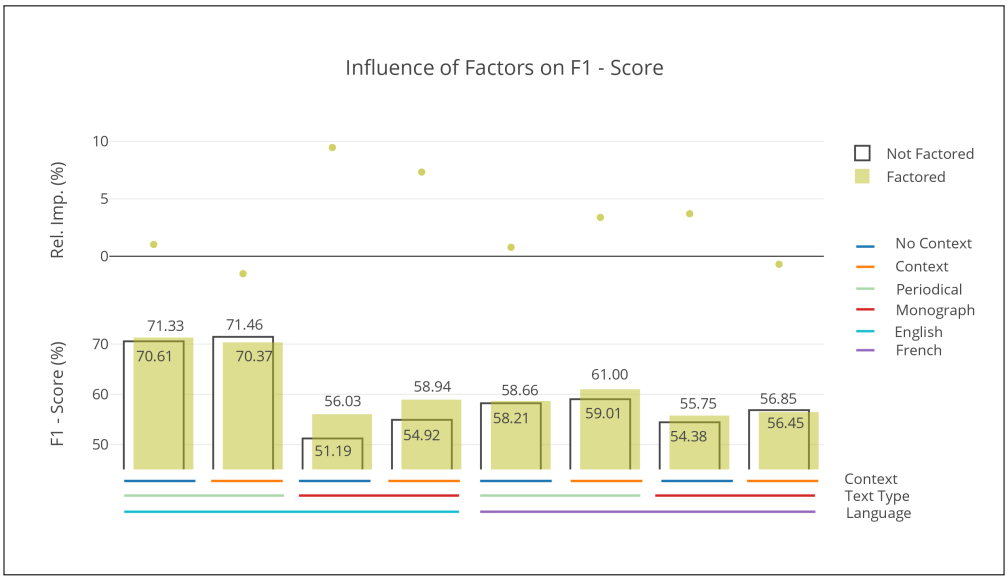

Figure 12: F1-score with and without time factor. Comparing context-insensitive vs. context-sensitive data and different text types and languages.

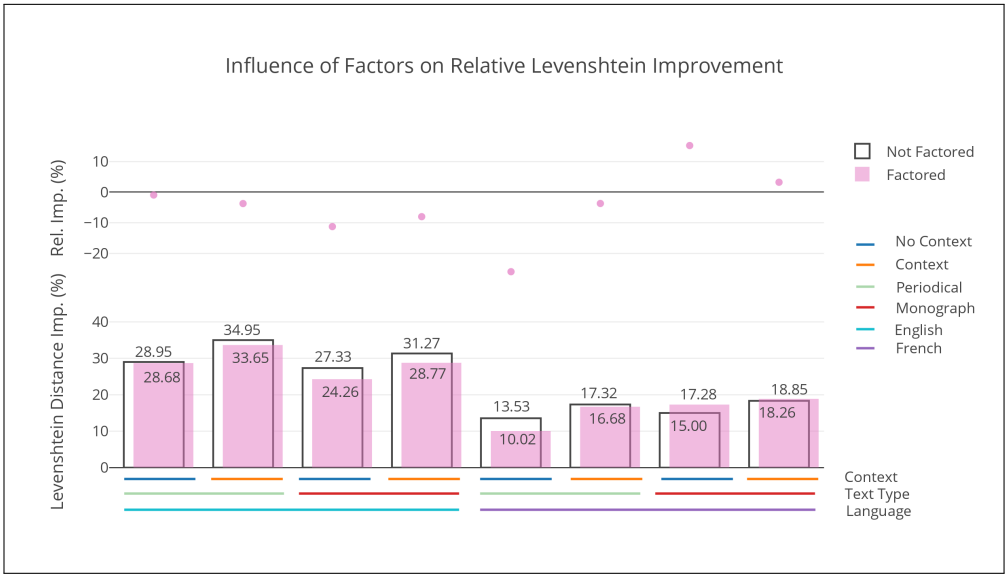

Figure 13: Relative Levenshtein distance improvement with and without time factor. Comparing context-insensitive vs. context-sensitive data and different text types and languages.

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. If a system detects more errors, it does not automatically mean it will also provide an accurate correction. Trying to correct more tokens can lead to an overall higher distance to the

gold standard than the original OCR output. Another explanation could be that, especially, with smaller data sets there might be data sparsity issues. The additional dimensions which are added to the embeddings for the time factor project the same character into different positions if the text is from a different period. This might then lead to problems in error correction since the character "f" from around 1800 cannot directly benefit from the character "f" from around 1950.

5.5 Factored Character-based NMT with Increased Training Set

Following the motivation of the section above, I repeated the experiments of Section 5.2 with factored data. In those experiments, using more training material from different data sets did not prove to be much useful for OCR post-correction. However, this was most likely due to the fact that the individual training sets contain specific errors and have different sizes. Despite these drawbacks, there might still be some valuable information to be gained from a larger, combined training set. This was tested in the following experiments by using a factor for the publication time period, the text type (for medium data sets) and an additional factor for the language (for large data sets). Unfortunately, due to time constraints, I could not test this method's influence when used with context-aware data.

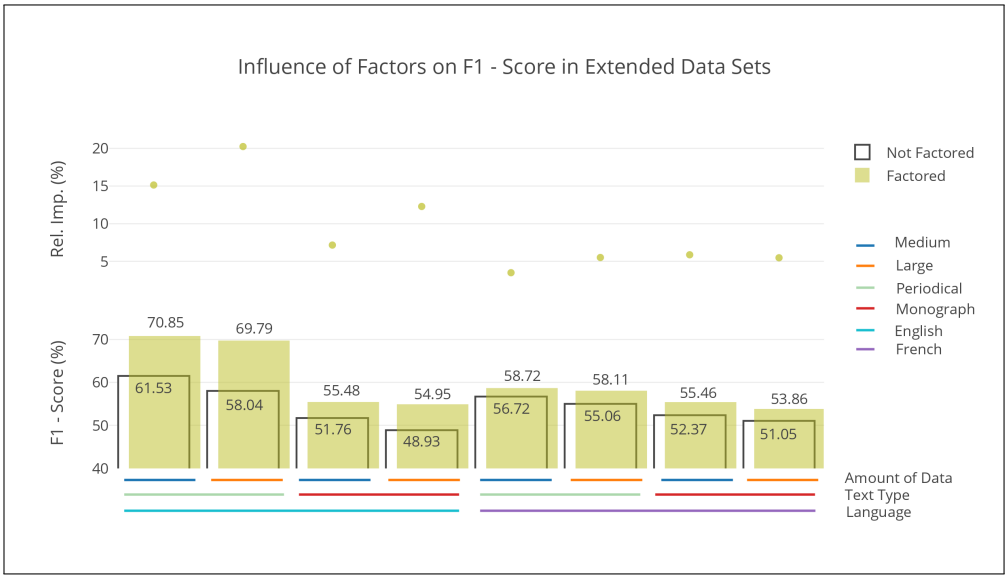

Figure 14: F1-score when using no factors or time, text type and language factors. Comparing medium vs. large data sets and different text types and languages.

Figure 14 and 15 show how using factors influences the F1-score and the relative Levenshtein distance improvement with increased data sets. As can be seen, the NMT system is powerful enough to make use of the factors and produces much better results than the previous experiments with combined training data. Still, the medium-sized data set has higher results than the large data set where all data is combined. This indicates that combining different languages for OCR post-correction with character-based NMT is not very useful, despite the code-switching discussed in Section 3.1.

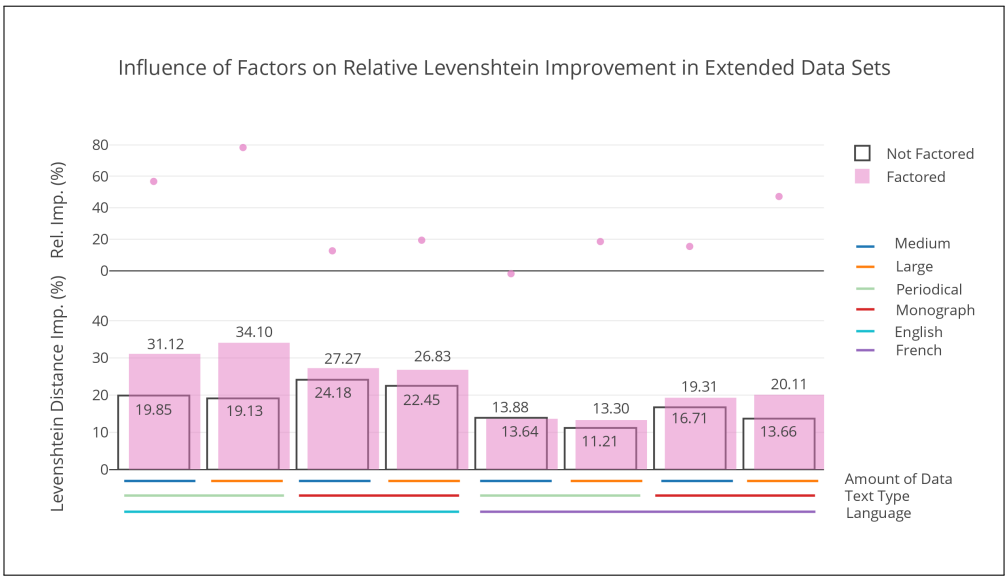

Figure 15: Relative Levenshtein distance improvement when using no factors or time, text type and language factors. Comparing data sets medium vs. large data sets and different text types and languages.

More interestingly, if these values are compared to the results for the individual data sets without factors and without context in Figure 12 and Figure 13, a performance improvement in error detection and even more so in error correction can be noticed. For error detection, the factored NMT models without context behave similarly to the same models trained on the medium and large data sets. In Section 5.4 using factors led to a decrease in performance in error correction. Surprisingly, when combining factors with the increased data sets, there is an improvement in error correction over both the factored and not factored data without context (except for English monograph). This finding supports the claim stated in Section 5.4 that using factors on small data sets might lead to data sparsity. In this experiment, the combined data sets are much larger which does not cause as much data sparsity when using factors for translation.

5.6 Glyph Embeddings

Another interesting case is the use of glyph images as pre-trained embeddings as described in Section 4.7. I expected this to have a positive effect on the post-correction, since OCR errors mostly occur due to visual and not due to contextual similarity. However, the results of the experiment look a bit different than expected: As can be seen in Figure 16 and 17, there was only a slight improvement in both F1-score and relative Levenshtein distance for the monograph texts. For the periodical texts, the results with glyph embeddings are even slightly worse than with randomised embeddings.

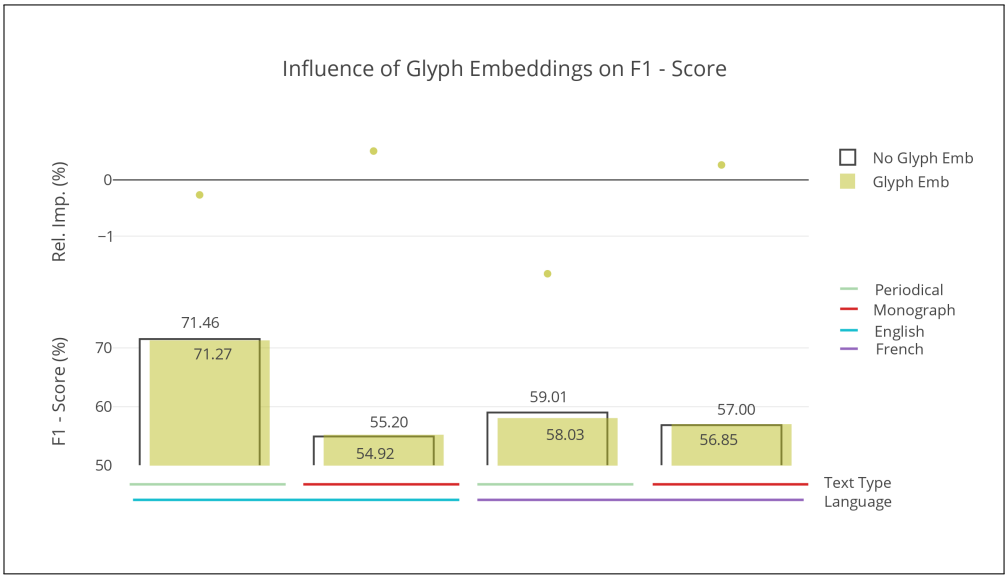

Figure 16: F1-score when using glyph embeddings. Comparing different text types and languages.

A possible explanation why this experiment did not work as expected is that the images did not adequately reflect what the OCR characters looked like. It has to be kept in mind that the images that were used to generate the glyph embeddings were created manually. Unfortunately, the original images that were OCRed were not made publicly available for this competition. This means that there was no way of recreating the actual font in which the texts were written. It is possible that the created glyph images match the font of the monograph texts more closely than those of the periodical texts. This would explain why glyph embeddings have a positive influence on the monograph systems and a negative influence on the periodical systems.

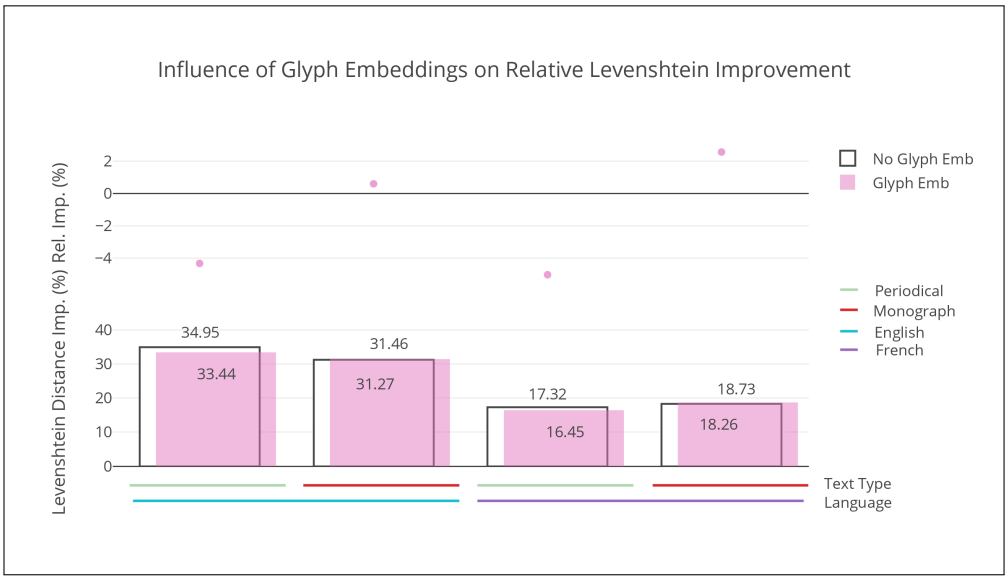

Figure 17: Relative Levenshtein distance improvement when using glyph embeddings. Comparing different text types and languages.

It might also be a coincidence that glyph embeddings worked better for the monograph data than the periodical data. Since the differences are so small and could be due to simple variance between systems, no satisfying conclusion can be drawn about glyph embeddings. To find out what the actual impact of glyph embeddings is, one would have to study their use more extensively and in different experiments. Also, it might be useful to give them as independent input to the NMT systems and not update them throughout training like embeddings. Still, this Bachelor thesis has shown that it is possible to use glyph images as pre-trained embeddings and that NMT systems trained with these embeddings perform more or less comparable to systems with randomly initialised embeddings.

5.7 Error-Focused Models

In Section 5.2, it was discovered that combining the training data of English monographs and periodicals led to a great loss in performance on the English periodical test set. Table 1 showed that the periodical data set contains much more errors that need to be corrected than the monograph data set. Following these findings, I created error-focused training sets which force the NMT systems to detect errors more aggressively as described in Section 4.8.

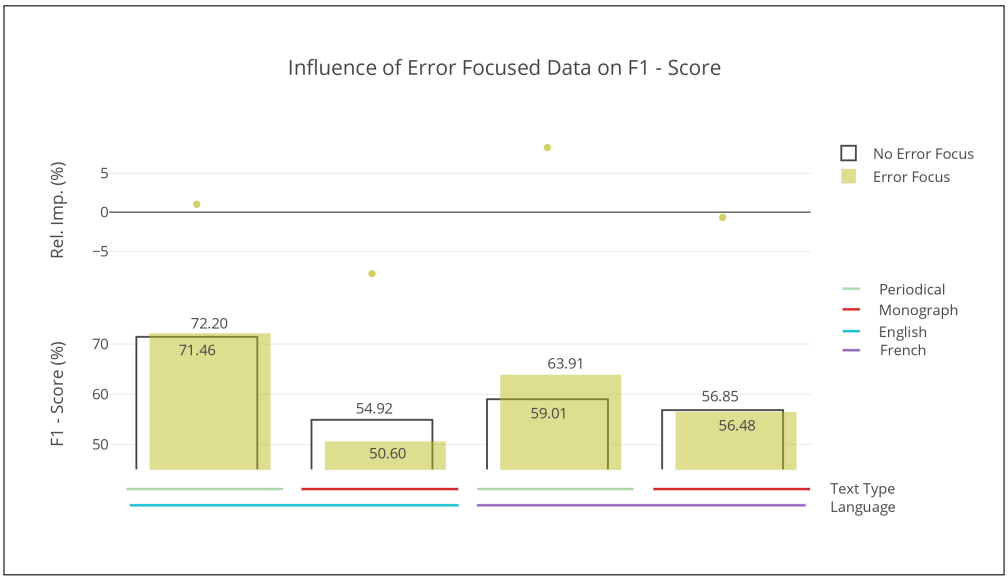

Figure 18: F1-score when using error-focused training sets. Comparing different text types and languages.

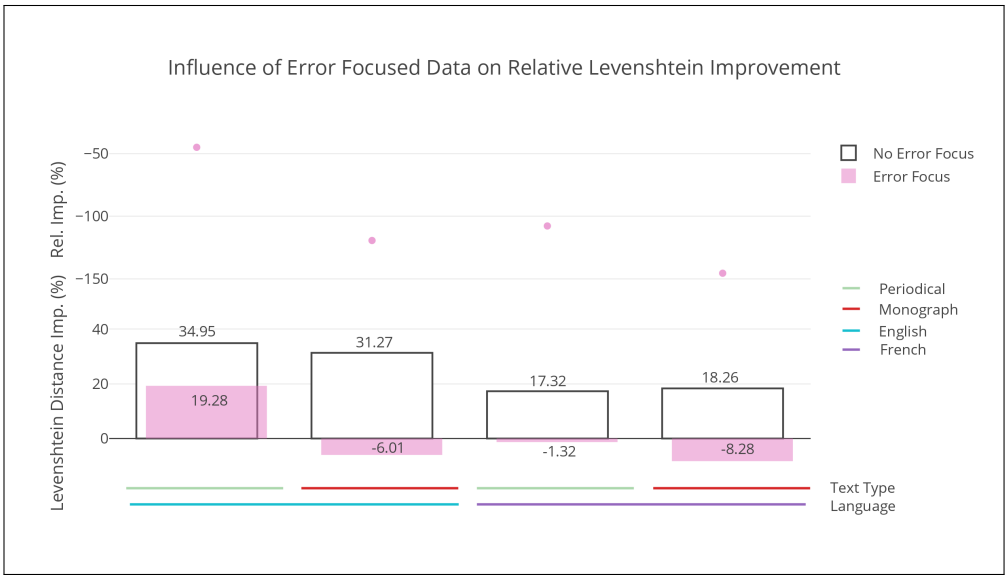

Figure 19: Relative Levenshtein distance improvement when using error-focused training sets. Comparing different text types and languages.

Figure 18 and 19 show the results of this experiment. As was to be expected, the performance in error correction sank drastically for all data sets. In case of error detection, the experiments showed a performance increase for the periodical data sets and a decrease for the monograph data sets. The reason for this is most likely that the error threshold in the monograph experiments was not ideal. For

the periodical experiments, the error-focused models produce better results in error detection. This proves that the concept works. More work would need to be done on finding a suitable threshold for all data sets. However, due to time and scope limits of this thesis no further experiments were conducted with error-focused models.

5.8 Ensemble Decoding

After an NMT system is trained, it is still possible to increase the amount of information for translation by ensembling different models as described in Section 4.9. Figure 20 and 21 show the results of the experiments with single ensembles. Both error detection and error correction improve when single embeddings are used in almost all cases. This finding shows that single embeddings are a simple method to increase the performance of a single NMT model for OCR post-correction. Again, the error-focused monograph data cannot be analysed reliably, since the error threshold was most likely not set ideally. Except for English monograph, the error-focused models show a higher improvement than the baseline context-aware models.

It is interesting to see that the relative improvement for error correction is much higher than for error detection. This suggests that at individual time stamps the models provide different correction candidates which in combination produce more correct translations. For error detection, the improvement is not as apparent. However, the models still benefit from each other and detect errors more accurately.

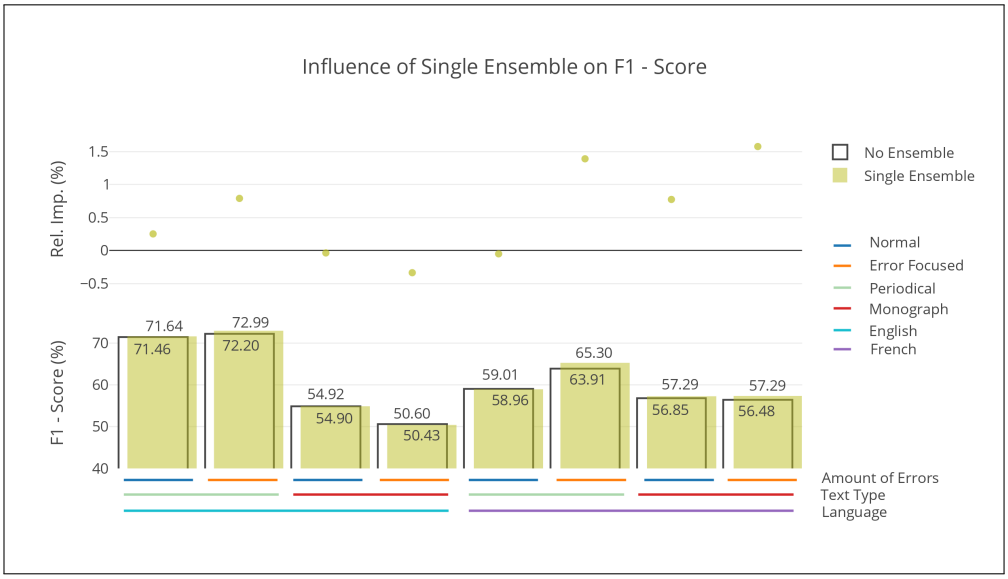

Figure 20: F1-score when using single ensembles. Comparing standard vs. error-focused models and different text types and languages.

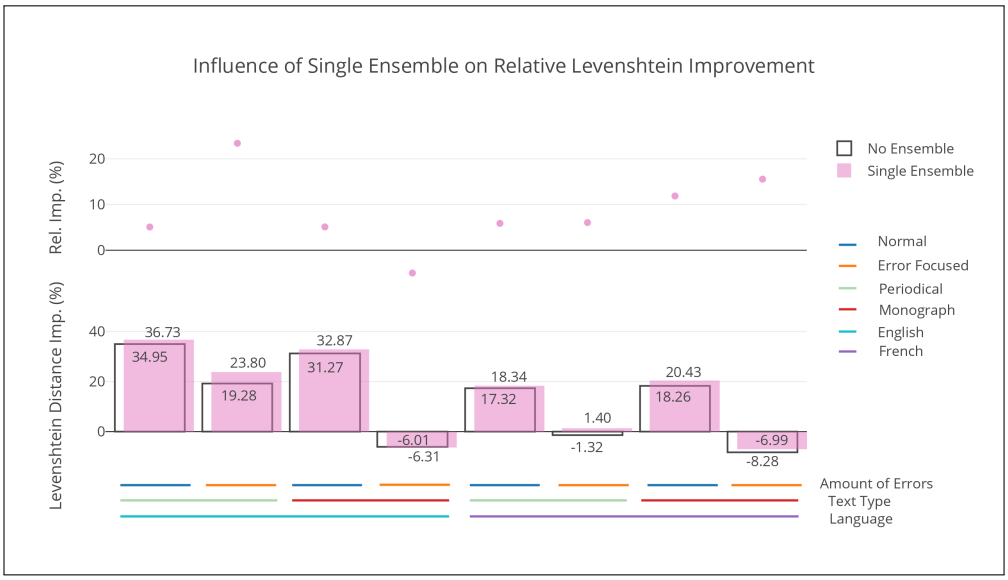

Figure 21: Relative Levenshtein distance improvement when using single ensembles. Comparing standard vs. error-focused models and different text types and languages.

Since single ensembles showed an improvement for both error detection and correction, I also wanted to find out what could be gained by using multi ensembles. Figure 22 and 23 show the results of this experiment. There is an improvement for all models, both for error detection and error correction.

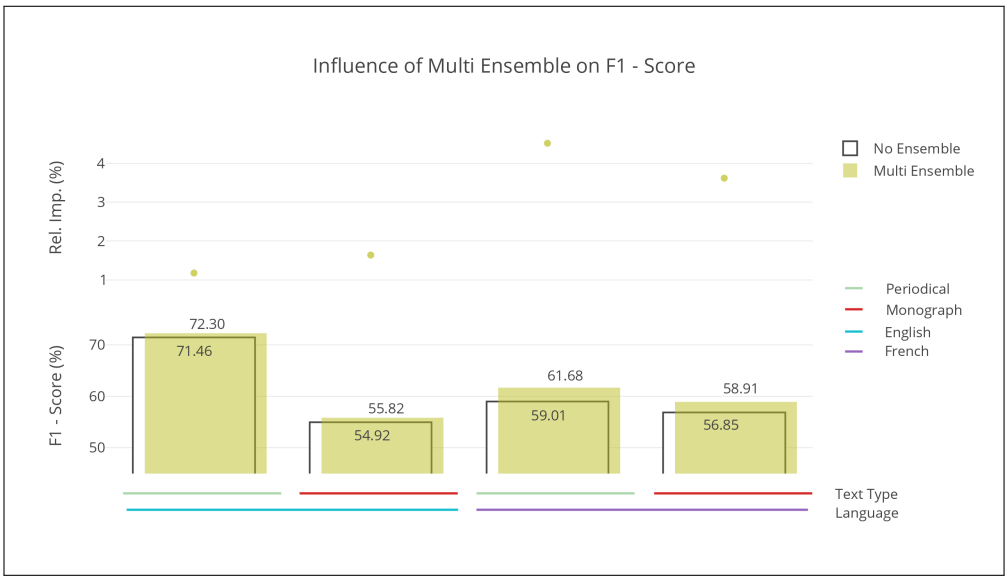

Figure 22: F1-score when using multi ensembles. Comparing different text types and languages.

Compared to the baseline models with context and the single ensembles built from them, the results for multi-ensembles are much better for error detection. For error correction on periodical data, the multi ensembles perform better than the single ensembles on the context-aware models. For the monographs, the results with multi ensembles are slightly worse. Again, this might be due to the error-focused monograph models which are included in the multi ensemble and the error threshold chosen for them.

In contrast to the error-focused single ensembles, multi ensembling was worse in error detection for both periodical data sets. However, in error correction, the multi ensemble models proved to be much more robust. This shows that some of the benefits from the error-focused models can also be found in the multi ensembles but without the considerable loss of performance in error correction.

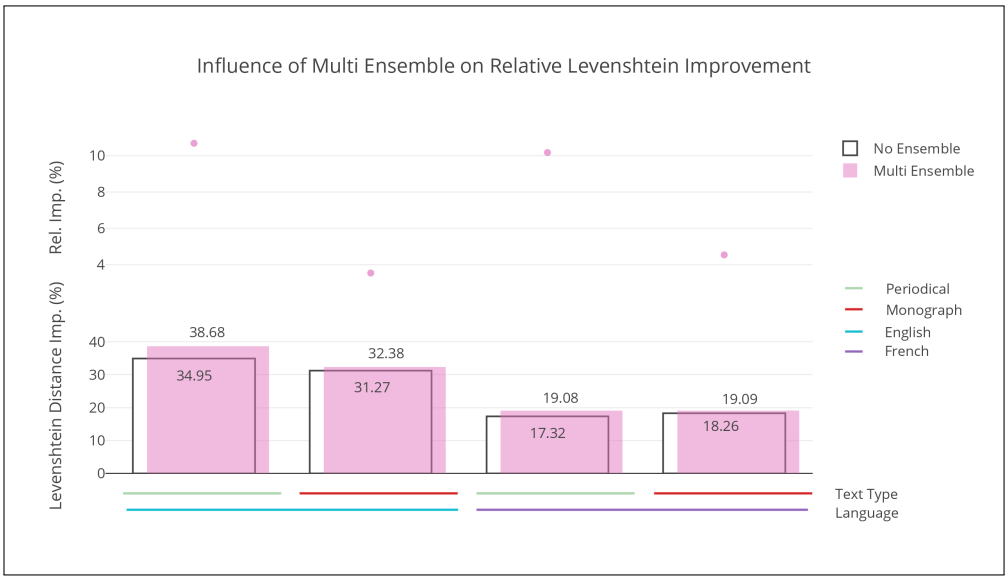

Figure 23: Relative Levenshtein distance improvement when using multi ensembles. Comparing different text types and languages.

Finally, it is surprising to see that the finding of the first experiment with the baseline systems without context has not changed. Error detection still works better on the periodical data, and error correction works better on the English data sets than on French. From these findings, it becomes clear that the performance in error detection is closely connected to how many errors occur in the data. Systems trained on data with more errors, see more examples and, thus, become better at detecting them. Moreover, the language of the data influences the performance in error correction. A possible explanation for this is that French has more inflexion than English. Therefore, it is harder to correct errors in the French data sets.

5.9 Overview of Best Systems

In the sections above, I discussed the results of my experiments. However, it is hard to see how all results relate to each other and which systems should be chosen for which data set. Therefore, I give an overview of the best systems for each data set. The full tables with all results can be found in Appendix A. The best results are marked in those tables as well, but Table 7 contains all best-performing systems. As can be seen, all best systems for error detection are NMT systems, while all best systems for error correction are SMT systems. Thus, this reinforces the finding that NMT systems perform better for error detection than SMT systems while SMT works better for error correction than NMT. It is interesting to see that, for error correction, SMT systems with context work best in all but one case.

Data Set	Error Detection	Error Correction
en Periodical	NMT single ensemble error	SMT context
en Monograph	NMT time factor context	SMT baseline
fr Periodical	NMT single ensemble error	SMT context
fr Monograph	NMT multi ensemble	SMT context

Table 7: The best.performing systems for each data set. For error detection, the F1 score is considered and for error correction the relative Levenshtein distance improvement.

In contrast, the best choice is less clear for error detection. For the periodical data sets, the single ensemble models trained on the error-focused data sets performed best. The best models for the monograph data sets, however, are not error-focused. This might again be due to the error threshold which was not chosen ideally. One thing that almost all of the best error detection models have in common is that they are ensembles. Only for the English monograph data set, the best system is not an ensemble but the factored model with context. This can easily be explained since the error rates in this data set vary largely over time, and this model captured these characteristics the best.

This overview of the best systems shows that the choice of MT system for OCR post-correction should depend on the data set and whether error detection or error correction is more important. NMT systems should be used if the focus is on error detection and SMT systems for error correction. If the data set contains different error frequencies in different time spans, a factored model will better capture these irregularities for error detection. In general, all NMT systems should be used

as ensembles because this reduces the variance of the system and produces better translations. Moreover, if the threshold for the error-focused models is set well, this can give an enormous boost to the recall in error detection and also to the F1-score. However, the results for error correction with the same system will suffer. Therefore, a combination of multiple systems might be the best solution if the resources and time for training are available. Our submission for the ICDAR 2017 competition was such a system combination for which the results are presented in the next section.

5.10 ICDAR 2017 Competition Submission and Results

Since the participation in the ICDAR 2017 shared task on OCR post-correction was part of this Bachelor thesis, the results from the competition are summarised in this section. It has to be kept in mind, that the systems used in the shared task were not trained on the same split of the data set. The gold standard of the test data was not published before the conference in November 2017, and the experiments discussed above were conducted before that. Therefore, the available training data had to be split into training, development and test sets again. However, for the models used in the competition, the full training set could be used to train the models (10% were used as a development set). Even though the results are also evaluated with precision, recall, F1-score for error detection and the Levenshtein distance for error correction, the evaluation script used in the shared task is slightly different from the one used in this Bachelor thesis. Consequently, the results from the competition cannot be directly compared to the results in the sections above. Still, seeing how the experiments from this Bachelor thesis perform compared to other approaches gives valuable insight into the usefulness of my approach.

As described in Section 4.10 our submission was a combination of different system outputs. We chose the systems such that their combination performed best on the development set. Table 8 shows which systems fed into our submission. Again, these systems cannot be directly compared to the experiments of this Bachelor thesis. They were trained on a different data split, and some system types discussed in the sections above were built after the submission, for example, the factored models of the increased data sets and the ensembles of the error-focused models. In the detailed result tables in Appendix A, the systems in our combinations are marked with an asterisk. Note that the systems used for the combination are not necessarily the top five systems on that data set but the five systems which worked best in combination. Therefore, a system which does not perform very well on its own can be useful in combination with other systems because it may discover

different types of errors.

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 normal	NMT multi ensemble	NMT multi ensemble	NMT baseline
NMT glyph embeddings	NMT context	NMT single ensemble normal	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 normal	NMT error-focused	NMT multi ensemble
NMT error-focused	NMT context	NMT glyph embeddings	NMT single ensemble normal
NMT glyph embeddings	NMT glyph embeddings	SMT baseline	NMT context

Table 8: Systems which were included in our system combination for task 1 and task 2. Systems marked with an asterisk have the smallest Levenshtein distance of the systems in that combination.

Figure 24 is a table taken from the competition paper of the ICDAR 2017 shared task on OCR post-correction [Chiron et al., 2017]. Our approach (Char-SMT/NMT) is marked in blue. As can be seen, we achieved the best results of all submitted methods in task 2, the error correction. In task 1, we performed comparatively to other submissions. The best approach for task 1 applies a noisy channel model to the OCR post-correction. They also use the Google Books Ngram Corpus as a source for their vocabulary and language model. Therefore, error detection probably profits a lot from external additional 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 (they use OpenNMT[1]), SMT on character- and token-level, spell checkers, error frequency patterns, 2-pass-RNN-architectures where the first RNN works on character-level and the second on token-level. The paper contains further information on the individual participants and their approaches, but since everyone was asked only to submit a summary of their work the texts are not very detailed.

[1]http://opennmt.net/

TABLE II
SUMMARIZED RESULTS FOR TASKS 1 AND 2 ACCORDING TO CORPUS SPECIFICITY: FRENCH, ENGLISH, PERIODICALS AND MONOGRAPHS

	Task 1 (F-mesure)				Task 2 (%Improvement) Auto (top1) / Semi (weighted mean on top5)				
Corpus part >	ENG-mono.	ENG-period.	FR-mono.	FR-period.	ENG-mono.	ENG-period.	FR-mono.	FR-period.	
NbTokens (E.R.) >	63371 (10%)	33176 (15%)	32274 (5%)	48356 (7%)	63371 (10%)	33176 (15%)	32274 (5%)	48356 (7%)	
5gram-KN-LV	0.05	0.51	0.25	0.35	x	x	x	x	E
LSTM Monochar	x	x	0.17	x	- / -	x	- / -	x	
Seq2Seq	0.45	0.39	x	x	- / -	- / -	x	x	E
BiLSTM	0.09	0.06	0.05	0.05	- / -	x	x	x	E
2-pass-RNN	0.66	0.66	0.43	0.60	x	- / -	x	x	E
Anavec	x	x	0.24	0.42	5% / -	- / -	- / -	- / -	
WFST-PostOCR	**0.73**	**0.68**	**0.55**	**0.69**	28% / =	- / -	- / -	- / -	
CLAM	0.67	x	0.36	0.54	29% / =	22% / =	1% / =	5% / =	
Char-SMT/NMT	0.67	0.64	0.31	0.50	**43% / =**	**37% / =**	**44% / =**	**29% / =**	
EFP	0.69	0.54	0.40	0.54	13% / 11%	- / -	23% / =	5% / 4%	
MMDT	0.66	0.44	0.36	0.41	20% / =	- / -	3% / =	2% / =	E

Figure 24: Table from [Chiron et al., 2017]. Results for error detection (task 1) and error correction (task 2).

Figure 25 shows the results of task 1 in more detail. Precision and recall can be compared with each other directly. As can be seen, our method achieved by far the highest precision. However, even though we focused on increasing the recall for error detection in multiple experiments, our recall is not as high as other methods. This comparison provides valuable information on what can still be improved with our method and in what cases it makes sense to use it. In a scenario where having a good precision for error detection and a high correction quality is of utmost importance, it would make sense to use our approach. On the other hand, if recall is more important, our method might need to be adapted further, or another approach should be used.

TABLE III
COMPARED PERFORMANCES (ACCURACY / RECALL) FOR TASK 1

Corpus part Nb tokens (Err.Rate.)	ENG-mono. 63371 (10%)	ENG-period. 33176 (15%)	FR-mono. 32274 (5%)	FR-period. 48356 (7%)
5gram-KN-LV	0.20 / 0.03	0.50 / 0.53	0.17 / 0.46	0.26 / 0.52
LSTM Monochar	x	x	0.26 / 0.12	x
Seq2Seq	0.36 / 0.59	0.35 / 0.44	x	x
BiLSTM	0.21 / 0.06	0.25 / 0.03	0.06 / 0.05	0.09 / 0.04
2-pass-RNN	0.58 / 0.77	0.64 / 0.68	0.33 / 0.60	0.54 / 0.67
Anavec	x	x	0.18 / 0.37	0.40 / 0.43
WFST-PostOCR	**0.67 / 0.82**	**0.68 / 0.68**	**0.51 / 0.59**	**0.72 / 0.66**
CLAM	0.93 / 0.52	x	0.48 / 0.28	0.71 / 0.44
Char-SMT/NMT	0.98 / 0.51	0.88 / 0.50	0.74 / 0.19	0.93 / 0.34
EFP	0.62 / 0.77	0.54 / 0.55	0.29 / 0.60	0.49 / 0.58
MMDT	0.84 / 0.55	0.72 / 0.32	0.62 / 0.25	0.71 / 0.28

Figure 25: Table from [Chiron et al., 2017]. More detailed results of task 1.

5.11 Future Work

Due to the many experiments in this Bachelor thesis, the findings are rather broad and could not be discussed in depth. In future research, it is important to gain more details about every approach and evaluate them more extensively. Especially for the context-aware systems, it needs to be investigated how much context should be given to the MT systems. Of course, having more context means that the time for training the systems increases. But the results of my experiments have shown that context is beneficial to OCR post-correction with character-based NMT. Therefore, more experiments should be conducted to find the ideal balance between gained benefit and increased training time.

For the error-focused models, more research should be done on what the best threshold between erroneous and correct examples is. This thesis showed that this threshold is dependent on the general frequency of errors in the training data. Finding a good threshold is important since it will boost the recall for error detection while not harming error correction too much.

Concerning the amount of training material for OCR post-correction, a more extensive study should be conducted. In my experiments, I compared monographs to periodicals for which I had about 2'870'000 and 1'280'000 characters for training respectively. My results showed, for example, that error detection and correction work better for English periodical than for English monograph. Since the English periodical data set contains a larger percentage of errors, this suggests that the total number of characters is not as important for increasing performance as the error frequency in the data sets. Therefore, it should be investigated how both, the total size of the data set and the numbers of errors, influence OCR post-correction with character-based NMT.

Finally, for the glyph embeddings, it would be interesting to see how models behave when the correct font from the actual OCR images is used to generate the embeddings. For this competition, the glyph images belonging to the text were not released. But for other data sets, this might be possible. There might also be better ways to integrate visual information about glyphs into character-based NMT which would be worth exploring.

6 Conclusion

This Bachelor thesis gave a broad overview of how character-based NMT techniques can be used for OCR post-correction. I make all scripts and configurations used in my thesis publicly available on my GitHub[1] account. My work built on previous work with SMT and explored how NMT compares to this approach. The findings 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 OCR post-correction. If it is more important to detect as many errors as possible, an NMT system should be trained. However, if it is more important that the system produces the correct candidates, an SMT system should be used. This answers my first research question: How does character-based NMT perform compared to character-based SMT in the case of OCR post-correction?

Furthermore, I tested both state-of-the-art and novel strategies to include more information in the training and translation process of NMT systems. One of the most successful approaches was giving more context to systems. This measure allows better detection of real word errors and increases the training material. Additionally, I found out that no improvement in OCR post-correction can be achieved when data sets are combined that have different error characteristics. However, when the individual training examples are labelled with their original data set as factors, the systems are able to generalise from the increased training sets and produce better results, especially for error correction. Moreover, data sets with error rates that vary considerably across different time spans can profit from time factors. Another fundamental insight is that error-focused models can boost error detection but have a rather negative influence on error correction. Finding a well-balanced error threshold for these models is essential. I showed that the decrease in error correction with error-focused models can be cushioned by using ensemble decoding. In fact, ensembling proved to be quite useful for single systems and, even more so, if different systems were combined for ensemble decoding. Finally, I presented a novel, straightforward approach how visual information on glyphs can be included in the

[1] https://github.com/chanberg/charmender

training process of character-based NMT systems. Through all of these experiments, I explored answers to my second research question: How can OCR post-correction with character-based MT be improved by using more information during training and translation?

With the approach described in this Bachelor thesis, we participated in a shared task on OCR post-correction which was organised in the context of ICDAR 2017. 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 external resources. This evaluation suggests that future work should focus more on improving error detection. However, our approach is undoubtedly a useful solution for error OCR post-correction.

References

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

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

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

M. Bollmann and A. Søgaard. Improving historical spelling normalization with bi-directional lstms and multi-task learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 131–139, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL `http://aclweb.org/anthology/C16-1013`.

E. Brill and R. C. Moore. 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, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi: 10.3115/1075218.1075255. URL `http://dx.doi.org/10.3115/1075218.1075255`.

G. Chiron, A. Doucet, M. Coustaty, and J.-P. Moreux. Icdar2017 competition on post-ocr text correction. In *2017 14th International Conference on Document Analysis and Recognition (ICDAR), forthcoming*, 2017.

J. Chung, K. Cho, and Y. Bengio. 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, August 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P16-1160`.

S. Eger, A. Mehler, et al. 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, 2016.

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

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

P. Koehn. Neural machine translation. *CoRR*, abs/1709.07809, 2017. URL `http://arxiv.org/abs/1709.07809`.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 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, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P07-2045`.

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

J. Lee, K. Cho, and T. Hofmann. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378, 2017. ISSN 2307-387X. URL `https://www.transacl.org/ojs/index.php/tacl/article/view/1051`.

M.-T. Luong and C. D. Manning. 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, August 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P16-1100`.

T. Luong, I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba. 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, July 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P15-1002`.

F. J. Och. 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, July 2003. Association for Computational Linguistics. doi: 10.3115/1075096.1075117. URL http://www.aclweb.org/anthology/P03-1021.

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

E. Pettersson, B. Megyesi, and J. Tiedemann. 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*, number 087, pages 54–69. Linköping University Electronic Press, 2013.

M. Rosca and T. Breuel. Sequence-to-sequence neural network models for transliteration. *CoRR*, abs/1610.09565, 2016. URL http://arxiv.org/abs/1610.09565.

C. Schnober, S. Eger, E.-L. Do Dinh, and I. Gurevych. 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, December 2016. The COLING 2016 Organizing Committee. URL http://aclweb.org/anthology/C16-1160.

R. Sennrich and B. Haddow. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W16-2209.

R. Sennrich, B. Haddow, and A. Birch. 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, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1162.

R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A. V. Miceli Barone, J. Mokry, and M. Nadejde. 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,

April 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/E17-3017.

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

L. Valette. Ocr correction of le temps. Master project, EPFL, January 2017.

M. Volk, L. Furrer, and R. Sennrich. *Strategies for Reducing and Correcting OCR Errors*, pages 3–22. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-20227-8. doi: 10.1007/978-3-642-20227-8_1. URL http://dx.doi.org/10.1007/978-3-642-20227-8_1.

Z. Xie, A. Avati, N. Arivazhagan, D. Jurafsky, and A. Y. Ng. Neural language correction with character-based attention. *CoRR*, abs/1603.09727, 2016. URL http://arxiv.org/abs/1603.09727.

K. Yao and G. Zweig. 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, 2015.

S. Zhao and Z. Zhang. An efficient character-level neural machine translation. *CoRR*, abs/1608.04738, 2016. URL http://arxiv.org/abs/1608.04738.

A Tables and Graphs

English Periodical

	Error Detection			Error Correction		
	Precision ↑	Recall ↑	F1-Score ↑	Lev. ↓	% Rel. Imp. ↑	% Correct ↑
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 normal *	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

Table 9: English periodical results of all experiments. Best results are marked in blue, worst results in red. Asterisks mark systems in system combination.

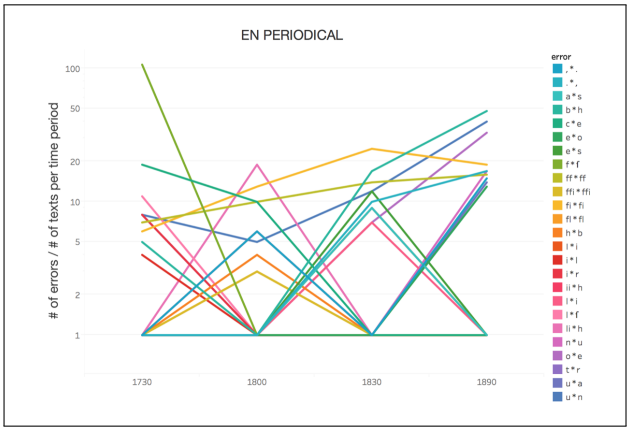

Figure 26: Errors over time in English periodical data. Frequency of ten most frequent errors per 30 years divided by number of files on logarithmic scale.

English Monograph

	Error Detection			Error Correction		
	Precision ↑	Recall ↑	F1-Score ↑	Lev. ↓	% Rel. Imp. ↑	% Correct ↑
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 normal	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 10: English monograph results of all experiments. Best results are marked in blue, worst results in red. Asterisks mark systems in system combination.

French Periodical

	Error Detection			Error Correction		
	Precision ↑	Recall ↑	F1-Score ↑	Lev. ↓	% Rel. Imp. ↑	% Correct ↑
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 normal	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

Table 11: French periodical results of all experiments. Best results are marked in blue, worst results in red. Asterisks mark systems in system combination.

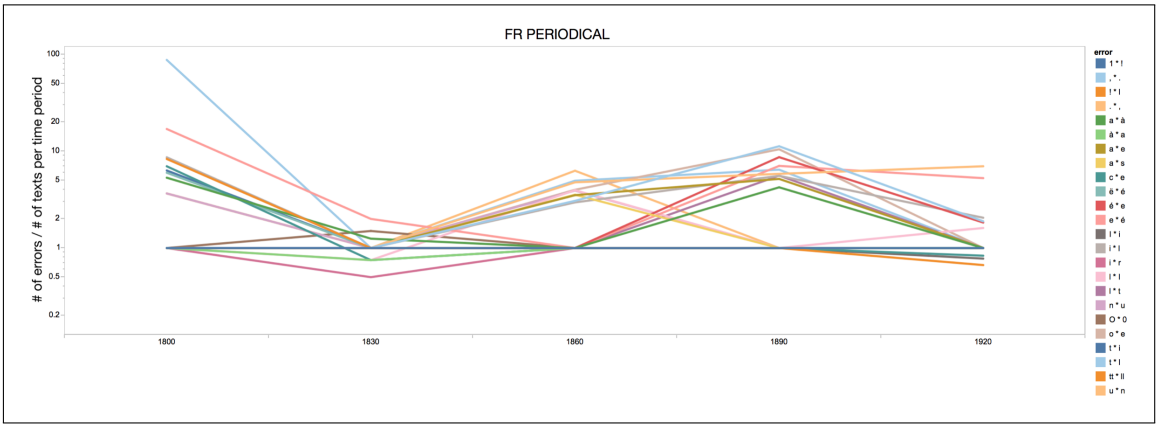

Figure 27: Errors over time in French periodical data. Frequency of ten most frequent errors per 30 years divided by number of files on logarithmic scale.

French Monograph

	Error Detection			Error Correction		
	Precision ↑	Recall ↑	F1-Score ↑	Lev. ↓	% Rel. Imp. ↑	% Correct ↑
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 normal *	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 12: French monograph results of all experiments. Best results are marked in blue, worst results in red. Asterisks mark systems in system combination.

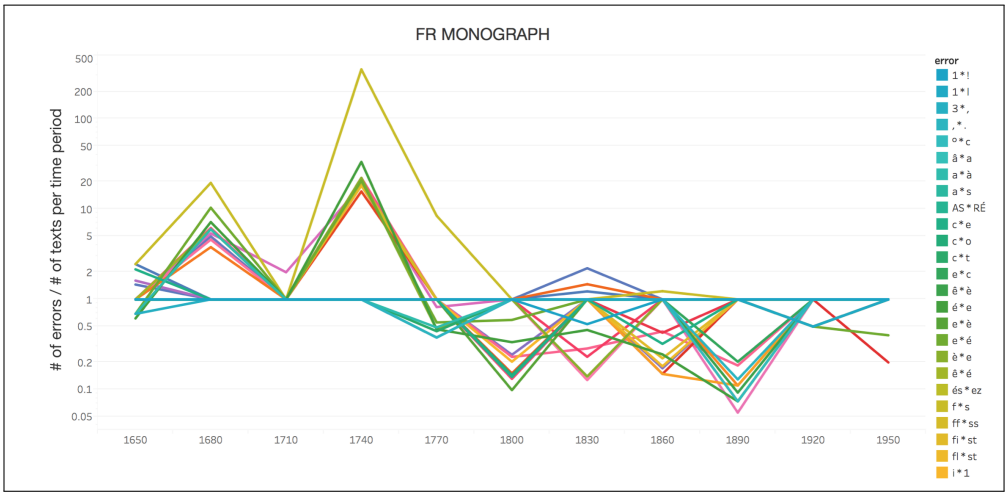

Figure 28: Errors over time in French monograph data. Frequency of ten most frequent errors per 30 years divided by number of files on logarithmic scale. (Legend for individual errors had to be shortened to fit image)