An Unsupervised method for OCR Post-Correction and Spelling Normalisation for Finnish

Quan Duong,♣ Mika Hämäläinen,♣,♢ Simon Hengchen♠
firstname.lastname@{helsinki.fi;gu.se}
♣University of Helsinki, ♢Rootroo Ltd, ♠University of Gothenburg

Abstract

Historical corpora are known to contain errors introduced by OCR (optical character recognition) methods used in the digitization process, often said to be degrading the performance of NLP systems. Correcting these errors manually is a time-consuming process and a great part of the automatic approaches have been relying on rules or supervised machine learning. We build on previous work on fully automatic unsupervised extraction of parallel data to train a character-based sequence-to-sequence NMT (neural machine translation) model to conduct OCR error correction designed for English, and adapt it to Finnish by proposing solutions that take the rich morphology of the language into account. Our new method shows increased performance while remaining fully unsupervised, with the added benefit of spelling normalisation. The source code and models are available on GitHub[1] and Zenodo[2].

1 Introduction

Nature language processing (NLP) is arguably tremendously difficult to tackle in Finnish, due to an extremely rich morphology. This difficulty is reinforced by the limited availability of NLP tools for Finnish in general, and perhaps even more so for historical data by the fact that morphology has evolved through time – some older inflections either do not exist anymore, or are hardly used in modern Finnish. As historical data comes with its own challenges, the presence of OCR errors makes the data even more burdensome to modern NLP methods.

Obviously, this problematic situation is not unique to Finnish. There are several other languages in the world with rich morphologies and relatively poor support for both historical and modern NLP. Such is the case with most of the languages that are related to Finnish like Erzya, Sami and Komi, these Uralic languages are severely endangered but have valuable historical resources in books that are not yet available in a digital format. OCR remains a problem especially for endangered languages (Partanen, 2017), although OCR quality for such languages can be improved by limiting the domain in which the OCR models are trained and used (Partanen and Rießler, 2019).

Automated OCR post-correction is usually modelled as a supervised machine learning problem where a model is trained with parallel data consisting of OCR erroneous text and manually corrected text. However, we want to develop a method that can be used even in contexts where no manually annotated data is available. The most viable recent method for such a task is the one presented by Hämäläinen and Hengchen (2019). However, their model works only on correcting individual words without considering the context in sentences, and as it focuses on English, it completely ignores the issues rising from a rich morphology. Extending their approach, we introduce a self-supervised model to automatically generate parallel data which is learned from the real OCRed text. Later, we train sequence-to-sequence (seq2seq) NMT models on character level with context information to correct OCR errors. The NMT models are based on the Transformer algorithm (Vaswani et al., 2017), whose detailed comparison is demonstrated in this article.

[1]Source Code, `https://github.com/ruathudo/post-ocr-correction`
[2]Trained models, `https://doi.org/10.5281/zenodo.4242890`

2 Related work

As more and more digital humanities (DH) work start to use the large-scale, digitised and OCRed collections made available by national libraries and other digitisation projects, the quality of OCR is a central point for text-based humanities research. Can one trust the output of complex NLP systems, if these are fed with bad OCR? Beyond the common pitfalls inherent to historical data (see Piotrowski (2012) for a very thorough overview), some works have tried to answer the question stated above: Hill and Hengchen (2019) use a subset of 18th-century corpus, ECCO[3] as well as its keyed-in counterpart ECCO-TCP to compare the output of common NLP tasks used in DH and conclude that OCR noise does not seem to be a large factor in quantitative analyses. A conclusion similar to previous work by Rodriquez et al. (2012) in the case of NER and to Franzini et al. (2018) for authorship attribution, but in opposition to Mutuvi et al. (2018) who focus on topic modelling for historical newspapers and conclude that OCR does play a role. More recently and still on historical newspapers, van Strien et al. (2020) conclude that while OCR noise does have an impact, its effect widely differs between downstream tasks.

It has become apparent that OCR quality for historical texts has become central for funding bodies and collection-holding institutions alike. Reports such as the one put forward by Smith and Cordell (2019) rise OCR initiatives, while the Library-of-Congress-commissioned report by Cordell (2020) underlines the importance of OCR for culturage heritage collections. These reports echo earlier work by, among others, Tanner et al. (2009) who tackle the digitisation of British newspapers, the EU-wide IMPACT project[4] that gathers 26 national libraries, or Adesam et al. (2019) who set out to analyse the quality of OCR made available by the Swedish language bank.

OCR post-correction has been tackled in previous work. Specifically for Finnish, Drobac et al. (2017) correct the OCR of newspapers using weighted finite-state methods, accordance with, Silfverberg and Rueter (2015) do the same for Finnish (and Erzya). Most recent approaches rely on the machine translation (MT) of "dirty" text into "clean" texts. These MT approaches are quickly moving from statistical MT (SMT) – as previously used for historical text normalisation, e.g. the work by Pettersson et al. (2013) – to NMT: Dong and Smith (2018) use a word-level seq2seq NMT approach for OCR post-correction, while Hämäläinen and Hengchen (2019), on which we base our work, mobilised character-level NMT. Very recently, Nguyen et al. (2020) use BERT embeddings to improve an NMT-based OCR post-correction system on English.

[3]Eighteenth Century Collections Online, `https://www.gale.com/primary-sources/eighteenth-century-collections-online`

[4]`http://www.impact-project.eu`

3 Experiment

In this section, we describe our methods for automatically generating parallel data that can be used in a character-level NMT model to conduct OCR post-correction. In short, our method requires only a corpus with OCRed text that we want to automatically correct, a word list, a morphological analyzer and any corpus of error free text. Since we focus on Finnish only, it is important to note that such resources exist for many endangered Uralic languages as well as they have extensive XML dictionaries and FSTs available (see (Hämäläinen and Rueter, 2018)) together with a growing number of Universal Dependencies (Nivre et al., 2016) treebanks such as Komi-Zyrian (Lim et al., 2018), Erzya (Rueter and Tyers, 2018), Komi-Permyak (Rueter et al., 2020) and North Sami (Sheyanova and Tyers, 2017).

3.1 Baseline

We design the first experiment based on the previous work (Hämäläinen and Hengchen, 2019), who train a character-level NMT system. Their research indicates that there is a strong semantic relationship between the correct word to its erroneous forms and we can generate OCR error candidates using semantic similarity. To be able to train the NMT model, we need to extract the parallel data of correct words and their OCR errors. Accordingly, we trained the Word2Vec model (Mikolov et al., 2013) on the Historical Newspaper of Finland from 1771 to 1929 using the Gensim library (Řehůřek and Sojka, 2010). After obtaining the Word2Vec model and its trained vocabulary, we extract the parallel data by using the Finnish morphological FST, Omorfi (Pirinen, 2015), provided in the UralicNLP library (Hämäläinen, 2019) and – following Hämäläinen and Hengchen (2019) – Levenshtein edit distance

(Levenshtein, 1965). The original approach used a lemma list for English for the data extraction, but we use an FST so that we can distinguish morphological forms from OCR errors. Without the FST, different inflectional forms would also be considered to be OCR errors, which is particularly counterproductive with a highly-inflected language.

We build a list of correct Finnish words by lematisating all words in the Word2Vec model's vocabulary: if the lemma is present in the Finnish Wiktionary lemma list,[5] it is considered as correct and saved as such. Next, for each word in this "correct" list, we retrieve the most similar words from the Word2Vec model. Those similar words are checked to see whether they exist in the correct list or not and separated into two different groups of correct words and OCR errors. Notice that not all the words in the error list are the wrong OCR format of the given correct word, and thus need to be filtered out. Following Hämäläinen and Hengchen (2019), we calculate the Levenshtein edit distance scores of the OCR errors to the correct word and empirically set a threshold of 4 as the maximum distance to accept as the true error form of that given word. As a result, for each given correct word, we have a set of similar correct words including the given one and a set of error words. From the two extracted groups, we do pairwise mapping to have one error word as training input and one correct word as the target output. Finally, the parallel data is converted into a character level format before feeding it to the NMT model for training. For example: $j\ o\ l\ e\ e\ n \rightarrow j\ o\ k\ e\ e\ n$ ("into a river") pair has the first word is incorrect and the second one is the right form. We follow Hämäläinen and Hengchen (2019) and use OpenNMT (Klein et al., 2017) with default settings, i.e. bi-directional LSTM with global attention (Luong et al., 2015). We train for 10,000 steps and keep the last checkpoint as a baseline, which will be referred to as "NATAS" in the remainder of this paper.

3.2 Methods

In the following subsections we introduce a different method to create a parallel dataset and apply a new sequence to the sequence model to train the data. The baseline approach presented above might introduce noise when we are unable to confidently know that the error word is mapped correctly to the given correct word, especially in the case of semantically similar words that have similar lengths. Another limitation of the baseline approach is that NMT model usually requires more variants to achieve better performance – something limited by the vocabulary of the Word2Vec model, which is trained with a frequency threshold so as to provide semantically similar words. To solve these problems we artificially introduce OCR-like errors in a modern corpus, and thus obtain more variants of the training word pairs and less noise in the data. We further specialise our approach by applying the Transformer model with context and non-context words experiments instead of the default OpenNMT algorithms for training. In the next section, we detail our implementation.

3.2.1 Dataset Construction

For the artificial dataset, we use the Yle News corpus[6] which contains more than 700 thousand articles written in Finnish from 2011 to 2018. All the articles are stored in a text file. Punctuation and characters not present in the Finnish alphabet are removed before tokenisation. After cleaning, we generate an artificial dataset by two different methods: random generator and a trained OCR error generator model.

Random Generator As previously stated, we will use a random generator to sample an OCR error word. In OCR text, an error normally happens when a character is misrecognized or ignored. This behavior causes some characters in the word to be missed, altered or introduced. The wrong characters will take a small ratio in the text. Thus, we design algorithm 1 to produce similar errors in the modern corpus.

For each word in the dataset, we will introduce errors to that word by deleting, replacing and adding characters randomly with a threshold of noise rate 0.07. The valid characters to be changed, added or removed must be in the Finnish alphabet, we do not introduce special characters as errors. The idea is that we select a random character position in the string with a probability smaller than noise rate multiplied with length of the string to restrict the percentage of errors in the word. This mean with the long word (eg. 15 characters), there will be always an error proposed. This process is repeated for each action of deleting, replac-

[5]https://fi.wiktionary.org/wiki/Wikisanakirja:Etusivu

[6]http://urn.fi/urn:nbn:fi:lb-2019030701

Algorithm 1 Random errors generator

```
1: procedure RANDOMERROR($Word, NoiseRate$)
2:     $Alphas$ = "abcdefghijklmnopqrstuvwxyzäåö"
3:     for $Action$ in [delete, add, replace] do
4:         generate $Rand$ is a random number between 0 and 1
5:         if $Rand < NoiseRate \times WordLength$ then
6:             Select a random character position $P$ in $Word$
7:             if character $P$ is in $Alphas$ then
8:                 Do the $Action$ on $P$ with $Alphas$
9:             end if
10:        end if
11:    end for
12: end procedure
```

ing, adding, thus a word could either have all kinds of errors or none if the random rate is bigger than threshold. A longer word is likely to have more errors than a shorter one.

Trained Generator Similarly to the random generator, we will modify the correct word into an erroneous form, but with a different approach. Instead of pure randomness, we build a model to better simulate OCR erroneous forms. The hypothesis is that if the artificial errors introduced to words have the same pattern as found in the real OCRed text, it would be more effective when applying the resulting model back to the real dataset. For example, the letter “i” and “l” are more likely to be misrecognized than “i” and “g” by the OCR engine.
To build the error generation model, we use the extracted parallel dataset from the NATAS experiment. However, the source and target for the NMT model are reversed to have correctly spelled words as the input and erroneous words as the output from the training. By trying to predict an OCR erroneous form for a given correct spelling, the model can learn an error pattern that mimics the real OCRed text. OpenNMT uses cross entropy loss by default, which causes an issue when applied to solve this problem. In our experiments, the model eventually predicted an output identical to the source because it is the most optimal way to reduce the loss. If we want to generate different output for the input, there is a need to penalize the model when having the same prediction as the input. To solve the problem, we built a simple RNN translation model with GRU (gated recurrent unit) layers and a custom loss function as shown in Equation 2. The loss function is built up from cross entropy cost function in Equation 1, where $H = \{h^{(1)}, ..., h^{(n)}\}$ is a set of predicted outcomes from the model and $T = \{t^{(1)}, ..., t^{(n)}\}$ is the set of targets. We calculate normal cross entropy of predicted output $\hat{Y}$ and the labels Y for finding an optimal way to mimic the target Y, on the other hand, the inverted cross entropy between $\hat{Y}$ and the inputs X is to punish the model if the outcomes are identical to the inputs.

The model's encoder and decoder each have one embedding layer with 128 dimensions and one GRU layer of 512 hidden units. The input sequences are encoded to have the source's context, this context is then passed through the decoder. For each next character of the output, the decoder concatenates the source's context, hidden context and character's embedded vector. The merged vectors then are passed through a linear layer to give the prediction. The model is trained by teacher enforcing technique with the rate 0.5. This means for the next input character, we either select the top one from the previous output or use the already known next one from the target label.

3.2.2 Models

Parallelisation and long memorisation are weakness characteristic of RNNs in NMT (Bai et al., 2018). Fortunately, Transformer proved to be much faster (mainly due to the absence of recursion), and since they process sequences as a whole they are shown to “remember" information better through their multi-head attention mechanism and positional embedding (Vaswani et al., 2017). Transformer has been shown to be extremely efficient in various tasks (see e.g. BERT (Devlin et al., 2018)), which is why we apply this model to our problem. Our implementation of the Trans-

$$\boxed{\begin{array}{c} cross\_entropy(H,T) = -\frac{1}{n}\sum_{i=1}^{n} t^{(i)} \ln h^{(i)} + (1-t^{(i)}) \ln(1-h^{(i)}) \quad (1) \\ loss = cross\_entropy(\hat{Y},Y) + 1 \div cross\_entropy(\hat{Y},X) \quad (2) \end{array}}$$

former model is based on (Vaswani et al., 2017) and uses the Pytorch framework.[7] The model contains 3 encoder and decoder layers, each of which has 8 heads of self-attention. We also implement a learned positional encoding and use Adam (Kingma and Ba, 2014) as the optimizer with a static learning rate of $5 \cdot 10^{-4}$ which gave a better convergence compared to the default value of 0.001 based on our experiment. Following prior work, cross entropy was again used as the loss function.

Our baseline NATAS only has fixed training samples extracted from the Word2Vec model. In this experiment, we design a dynamic data loader which generates new erroneous words for every mini-batch while training, allowing the model to learn from more variants at every iteration. As was mentioned in the introduction, we train contextualized sequence-to-sequence character-based models. Instead of feeding a single error word to the model as the input, we combine it with the context words before and after it in sequence. We only consider the correct form of that error word as the label, and are not predicting the context words. The input includes the error (target) word in the middle and its two sides context make up a window of odd number of words. Hence, a valid window sliding over the corpus must have an odd size, for instance 3, 5, etc. The way we construct the input and gold label is presented as follows:

- The window size of n words is selected. The middle word is considered the target word
- The words on left and right of the target are context words
- The input sequence is converted in proper format, for example with window n=5: `<sos> l e f t <sep> c o n t e x t <ctx> f a r g e t <ctx> r i g h t <sep> c o n t e x t <eos> <pad>`, where:
  - `<sos>` indicates the start of a sequence;
  - `<sep>` is the separator for the context words;
  - `<ctx>` separates left and right context with the target;
  - `<eos>` indicates the end of a sequence;
  - `<pad>` indicates the padding if needed for mini-batch training.

Following the previous section, the "target" word is generated by creating artificial errors in two different ways: using random generator, and a trained generator. For instance, the word "target" in the example above is modified to "farget", and the model is trained to predict the output "target". The gold label is also formatted in the same format, but without any context words. In this case, the label should have this form: `<sos> t a r g e t <eos>`. After having the pairs of input and label formatted properly, we feed them into the Transformer model with a batch size of 256 – a balance between the speed and accuracy in our case. In this experiment, we evaluate our model with 3 different window sizes: 1, 3, and 5, with the window size of 1 as a special case: there are no context words, and the input is `<sos> f a r g e t <eos>`. For every window size we train with two different error generators (Random and Trained), and have thus 6 models in total. These models are named hereafter **TFRandW1**, **TFRandW3**, **TFRandW5**, **TFTrainW1**, **TFTrainW3**, and **TFTrainW5**, where TF stands for Transformer, $Rand$ is for the random generator, $Train$ is for the trained generator and Wn for a window of n words. We proceeded with the training until the loss converged. All models converged after around 20 epochs. The losses for the $Train$ models are ~ 0.064 and those for $Rand$ are slightly lower, with ~ 0.059.

4 Evaluation

We evaluate all proposed models and the NATAS baseline on the Ground Truth Finnish Fraktur dataset[8] made available by the National Library of Finland, a collection of 479 journal and newspaper pages from the time period 1836 - 1918 (Kettunen

[7]https://pytorch.org/

[8]"OCR Ground Truth Pages (Finnish Fraktur) [v1](4.8 GB)", available at `https://digi.kansalliskirjasto.fi/opendata`

et al., 2018). The data format is constructed as a csv table with 471,903 lines of words or characters and there are four columns of ground truth (GT) aligned with the output coming from 3 different OCR methods TESSERACT, OLD and FR11 (Kettunen et al., 2018).

Despite the existence of character-level benchmarks for OCR post-correction (e.g. Drobac et al. (2017)), we elect to evaluate models on the more realistic setting of whole words. We would like to note that Finnish has very long words, and as a result this metric is actually tougher. In the previous section, our models are trained without non-alphabet characters, so all the tokens which have non-alphabet will be removed. We also removed the blank lines which have no result from OCR. After having the ground truth and OCR text cleaned, the number of tokens for each OCR method (TESSERACT, OLD, FR11) are 458,799, 464,543 and 470,905 with accuracies of 88.29%, 75.34% and 79.79% respectively. The OCR words will be used as input data for the evaluation of our post-correction systems. The translation processes apply for each OCR method separately with the input tokens formatted based on the model's requirement. In NATAS, we used OpenNMT to translate with the default settings. In Transformer models with context, we created a sliding window over the rows of the OCRed text. For the non-context model, we only need a single token for source input. These models do the translation with beam search $k = 3$ and the highest probability sequence is chosen as the output. The result is shown in Table 1.

Models	TESSERACT (88.29)	OLD (75.34)	FR11 (79.79)
NATAS	63.35	61.63	64.95
TFRandW1	69.78	67.33	71.64
TFRandW3	70.02	67.45	71.69
TFRandW5	**71.24**	68.35	72.56
TFTrainW1	70.22	68.30	72.22
TFTrainW3	71.19	69.25	73.14
TFTrainW5	71.24	**69.30**	**73.21**

Table 1: Models accuracy on word level for all three OCR methods (%)

4.1 Error Analysis

From the result in Table 1, we can see all the models could not make any improvement on OCR text. However, there is clearly an advantage of using an artificial dataset and Transformer model for training, which has a 7 percentage points higher accuracy compared to NATAS. After analyzing the result, we found that there are many interesting cases where the output words are considered as errors when compared to the ground truth directly but they are still correct. The difference is that the ground truth has been corrected by maintaining the historical spelling, but as our model has been trained to correct words to a modern spelling, these forms will appear as incorrect when compared directly with the ground truth. However, our models still corrected many of them right, but just happened to normalize the spelling to modern Finnish at the same time. As examples, the word *lukuwuoden* (“academic year") is normalized to *lukuvuoden*, and the word *kortt* (“card") is normalized to *korrti*, which are the correct spellings in modern Finnish. So, the problem here is that many words have acquired a new spelling in modern Finnish but are seen as the wrong result if compared to the ground truth, which affects the real accuracy of our models. In the 19th century Finnish text, the most obvious difference compared to modern Finnish is the variation of *w/v*, where most of the words containing *v* are written as *w* in old text, whereas in modern Finnish *w* is not used in any regular word. Kettunen and Pääkkönen (2016) showed in their experiments that the number of tokens containing letter *w* contribute to 3.3% of all tokens and 97.5% of those tokens is missrecognized by FINTWOL – a morphological analyzer. They also tried to replace the *w* with *v* and the unrecognized tokens decreased to 30.6%. These numbers are significant which give us an idea to apply it on our results to get a better evaluation. Furthermore, there is another issue for our models when they try to make up the new words which do not exist in Finnish vocabulary. For example the word *samppaajaa* is likely created from the word *samppanjaa* (“of Champagne") which must be the correct one. To solve these issues, we suggested a fixing pipeline for our result:

1. Check if the words exist in Finnish vocabulary using Omorfi with UralicNLP, if not then keep the OCRed words as the output.

2. Find all words containing letter *v*, replace by letter *w*.

After the processing with the strategy above, we get updated results which can be found in Tables 2, 3, and 4.

Models	Post processed accuracy	Error words accuracy	Correct words accuracy
NATAS	74.71	16.54	82.43
TFRandW1	80.49	16.13	89.03
TFRandW3	80.79	16.94	89.26
TFRandW5	81.89	17.02	90.49
TFTrainW1	83.05	17.11	91.79
TFTrainW3	83.96	**18.15**	92.68
TFTrainW5	**84.00**	18.02	**92.75**

Table 2: Models accuracy post-processing for Tesseract (88.29%)

Models	Post processed accuracy	Error words accuracy	Correct words accuracy
NATAS	71.19	30.66	84.45
TFRandW1	75.10	28.14	90.47
TFRandW3	75.40	28.26	90.83
TFRandW5	76.26	28.63	91.85
TFTrainW1	78.19	35.07	92.30
TFTrainW3	**79.26**	**36.03**	93.41
TFTrainW5	79.17	35.41	**93.50**

Table 3: Models accuracy post-processing for OLD (75.34%)

Models	Post processed accuracy	Error words accuracy	Correct words accuracy
NATAS	75.06	36.52	84.81
TFRandW1	79.66	36.04	90.71
TFRandW3	80.06	37.00	90.96
TFRandW5	81.09	38.04	91.99
TFTrainW1	82.39	43.39	92.26
TFTrainW3	**83.50**	**45.17**	93.21
TFTrainW5	83.34	44.01	**93.30**

Table 4: Models accuracy post-processing for FR11 (79.79%)

The results in Tables 2, 3 and 4 show a vast improvement for all models with the accuracy increased by 10-12 percentage points. In Tesseract, where the original OCR already has a very high quality with an accuracy of about 88%, there is no gain for all models. The best model in this case is TFTrainW5 with 84% accuracy. The reason for the models' worse performance is that they introduced more errors on the already correct words by OCR than fixing actual error words. While the ratio of fixing the error words (18.02%) is much higher than the ratio of confounding the correct words (7.25%), however, due to the number of correct words taking a much larger part in the corpus, the overall accuracy is decreased. In the OLD setting with accuracy of about 75%, 5 out of 7 models have successfully improved the accuracy of the original text. The highest number comes to TFTrainW3 which outperforms OLD by 3.92 percentage points, and following closely is TFTrainW5 with an accuracy of 79.17%. In OLD, we see better error words corrected (36.03%) compared to Tesseract. The accuracy of the TFTrainW5 model for the already corrected words is also slightly higher with 93.5% versus Tesseract 92.75%. The last OCR method for evaluation is FR11 (79%), where – just like in OLD – 5 out of 7 models surpass the OCR result. Again, the TFTrainW3 gives the highest number with 3.71 percentage points improvement on the OCRed text. While the TFTrainW3 shows surprisingly good results on fixing the wrong words with 45.17% accuracy, the TFTrainW5 performs slightly better at handling the right words. Common to all our proposed models, the window size of 1 somewhat unsurprisingly performs worse within both the $Rand$ and $Train$ variants.

5 Conclusion and Future work

In this paper, we have shown that creating and using an artificial error dataset clearly outperforms the NATAS baseline (Hämäläinen and Hengchen, 2019), with a clear advantage for the $Train$ over the $Rand$ configuration. Another clear conclusion is that a larger context window results in increasing the accuracy of the models. Comparing the new results for all three OCR methods, we see the models are most effective with FR11 when the ratio of fixing wrong words (45.17%) is high enough to overcome the issue of breaking the right words (6.7%). Our methods also work very well on OLD with ability to fix 36.03% of wrong words and handle more than 93% of right words correctly. However, our models are not compelling enough to beat the accuracy achieved by Tesseract, a conclusion we see as further work.

In spite of the effectiveness of the post-correction strategy, it does not guarantee that all the words with *w/v* replaced are correct, nor that

UralicNLP manages to recognize all the existing Finnish words. For example: the wrong OCR word *mcntoistamuotiscn* was fixed to *metoistavuotisen* which is the correct one according to the gold standard, but UralicNLP has filtered it out due to not considering that is the valid Finnish word. This is true, as the first syllable *kol* was dropped out due to a line break in the data, and without the line break, the word would be *kolmetoistavuotisen* (“13 years old"). This means that in the future, we need to develop better strategies more suitable to OCR contexts for telling correct and incorrect words apart.

This implies that in reality the corrected cases can be higher if we don’t revert the already normalized *w/v* words. In addition, if there is a better method to ensure a word is valid in Finnish, the result could be improved. Thus, our evaluation provides an overall view of how the Transformer and Trained Error Generator models with context words could improve the post OCR correction notably. Our methods also show that using artificial dataset from a modern corpus is very beneficial to normalize the historical text.

Importantly, we would like to underline that our method does not rely on huge amounts of hand annotated gold data, but can rather be applied for as long as one has access to an OCRed text, a vocabulary list, a morphological FST and error-free data. There are several endangered languages related to Finnish that already have these aforementioned resources in place. In the future, we are interested in trying our method out in those contexts as well.

References

Yvonne Adesam, Dana Dannélls, and Nina Tahmasebi. 2019. Exploring the quality of the digital historical newspaper archive KubHist. *Proceedings of DHN.*

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.

Ryan Cordell. 2020. Machine learning and libraries: a report on the state of the field. Technical report, Library of Congress.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding.

Rui Dong and David Smith. 2018. Multi-input attention for unsupervised OCR correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).*

Senka Drobac, Pekka Sakari Kauppinen, and Bo Krister Johan Linden. 2017. OCR and post-correction of historical Finnish texts. In *Proceedings of the 21st Nordic Conference on Computational Linguistics, NoDaLiDa, 22-24 May 2017, Gothenburg, Sweden.* Linköping University Electronic Press.

Greta Franzini, Mike Kestemont, Gabriela Rotari, Melina Jander, Jeremi K Ochab, Emily Franzini, Joanna Byszuk, and Jan Rybicki. 2018. Attributing authorship in the noisy digitized correspondence of Jacob and Wilhelm Grimm. *Frontiers in Digital Humanities*, 5:4.

Mika Hämäläinen and Simon Hengchen. 2019. From the paft to the fiiture: a fully automatic NMT and word embeddings method for OCR post-correction. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 431–436.

Mika Hämäläinen and Jack Rueter. 2018. Advances in synchronized xml-media wiki dictionary development in the context of endangered uralic languages. In *Proceedings of the XVIII EURALEX International Congress: Lexicography in Global Contexts.*

Mika Hämäläinen. 2019. UralicNLP: An NLP library for Uralic languages. *Journal of Open Source Software*, 4(37):1345.

Mark J Hill and Simon Hengchen. 2019. Quantifying the impact of dirty OCR on historical text analysis: Eighteenth century collections online as a case study. *Digital Scholarship in the Humanities*, 34(4):825–843.

Kimmo Tapio Kettunen, Jukka Kervinen, and Jani Mika Olavi Koistinen. 2018. Creating and using ground truth ocr sample data for finnish historical newspapers and journals. In *Proceedings of the Digital Humanities in the Nordic Countries 3rd Conference*, CEUR Workshop proceedings, pages 162–169. Technical University of Aachen. Digital Humanities in the Nordic Countries ; Conference date: 07-03-2018 Through 09-03-2018.

Kimmo Tapio Kettunen and Tuula Anneli Pääkkönen. 2016. Measuring lexical quality of a historical finnish newspaper collection – analysis of garbled ocr data with basic language technology tools and means. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016).*

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proc. ACL.*

Vladimir I. Levenshtein. 1965. Двоичные коды с исправлением выпадений, вставок и замещений символов. Доклады Академий Наук СССР, 63(4):845–848.

KyungTae Lim, Niko Partanen, and Thierry Poibeau. 2018. Multilingual dependency parsing for low-resource languages: Case studies on North Saami and Komi-Zyrian. In *Proceedings of LREC 2018*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Stephen Mutuvi, Antoine Doucet, Moses Odeo, and Adam Jatowt. 2018. Evaluating the impact of OCR errors on topic modeling. In *International Conference on Asian Digital Libraries*, pages 3–14. Springer.

Thi Tuyet Hai Nguyen, Adam Jatowt, Nhu-Van Nguyen, Mickael Coustaty, and Antoine Doucet. 2020. Neural machine translation with bert for post-ocr error detection and correction. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, pages 333–336.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Niko Partanen. 2017. Challenges in ocr today: Report on experiences from INEL. In Электронная Письменность Народов Российской Федерации: Опыт, Проблемы И Перспективы, pages 263–273.

Niko Partanen and Michael Rießler. 2019. An OCR system for the unified northern alphabet. In *The fifth International Workshop on Computational Linguistics for Uralic Languages*.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. 2013. An SMT approach to automatic annotation of historical text. In *Proceedings of the workshop on computational historical linguistics at NODALIDA 2013*.

Michael Piotrowski. 2012. Natural language processing for historical texts. *Synthesis lectures on human language technologies*, 5(2):1–157.

Tommi A Pirinen. 2015. Development and use of computational morphology of Finnish in the open source and open science era: Notes on experiences with Omorfi development. *SKY Journal of Linguistics*, 28:381–393.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Kepa Joseba Rodriquez, Mike Bryant, Tobias Blanke, and Magdalena Luszczynska. 2012. Comparison of named entity recognition tools for raw OCR text. In *KONVENS*, pages 410–414.

Jack Rueter, Niko Partanen, and Larisa Ponomareva. 2020. On the questions in developing computational infrastructure for komi-permyak. In *Proceedings of the Sixth International Workshop on Computational Linguistics of Uralic Languages*, pages 15–25.

Jack Rueter and Francis Tyers. 2018. Towards an Open-Source Universal-Dependency Treebank for Erzya. In *Proceedings of the Fourth International Workshop on Computatinal Linguistics of Uralic Languages*.

Mariya Sheyanova and Francis M. Tyers. 2017. Annotation schemes in North Sámi dependency parsing. In *Proceedings of the 3rd International Workshop for Computational Linguistics of Uralic Languages*.

Miikka Silfverberg and Jack Rueter. 2015. Can morphological analyzers improve the quality of optical character recognition? In *Septentrio Conference Series*, 2, pages 45–56.

David A. Smith and Ryan Cordell. 2019. A research agenda for historical and multilingual optical character recognition. Technical report, Northeastern University.

Daniel van Strien, Kaspar Beelen, Mariona Coll Ardanuy, Kasra Hosseini, Barbara McGillivray, and Giovanni Colavizza. 2020. Assessing the impact of ocr quality on downstream nlp tasks. In *ICAART (1)*, pages 484–496.

Simon Tanner, Trevor Muñoz, and Pich Hemy Ros. 2009. Measuring mass text digitization quality and usefulness. *D-lib Magazine*, 15(7/8):1082–9873.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.