CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

New tools for old news

Correcting OCR errors in digitized Swedish newspapers with transformers

Master's thesis in Computer science and engineering

Viktoria Löfgren

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

MASTER'S THESIS 2023

New tools for old news

Correcting OCR errors in digitized Swedish newspapers with transformers

Viktoria Löfgren

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

New tools for old news: Correcting OCR errors in digitized Swedish newspapers with transformers
VIKTORIA LÖFGREN

Supervisor: Dana Dannélls, Språkbanken Text,
University of Gothenburg
Examiner: Marina Axelson-Fisk, Applied Mathematics and Statistics, Chalmers University of Technology

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

New tools for old news: Correcting OCR errors in digitized Swedish newspapers with transformers

Viktoria Löfgren
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Many collections of digitized newspapers suffer from poor OCR quality, which impacts readability, information retrieval, and analysis of the material. Errors in OCR output can be reduced by applying machine translation models to "translate" it into a corrected version. Although transformer models show promising results in post-OCR correction and related tasks in other languages, they have not yet been used for correcting OCR errors in Swedish texts. This thesis presents a post-OCR correction model for Swedish 19th and 20th century newspapers based on the pre-trained transformer model ByT5. Three versions of the model were trained on different mixes of training data. The best model, which achieved a 37% reduction in CER, will be integrated in Språkbanken Text's annotation pipeline Sparv.

Keywords: Post-OCR correction, ByT5, newspaper digitization

Acknowledgements

I would like to thank my supervisor Dana Dannélls for introducing me to this topic and providing excellent guidance throughout the process of writing this thesis. This project would not have been possible without the data provided by Språkbanken Text and Litteraturbanken.

Viktoria Löfgren, Gothenburg, December 2023

Contents

List of Figures

List of Tables

1. *Introduction*

Historical newspaper archives are valuable resources for researchers across many disciplines. For long, access to these archives has been limited, but in recent years, cultural institutions have begun to digitize their collections. One of them is the National Library of Sweden, *Kungliga biblioteket*, whose digitized newspaper archive covers 34 million pages. Like many other archives it is publicly available online[1], allowing users to browse the collections remotely without risking damage to the fragile original documents.

Digitization brings benefits beyond improved accessibility [1]. Most archives are processed with optical character recognition (OCR), a technique that extracts machine-readable text from scanned documents. This allows users to search in the archive, and in turn ask questions that previously would require substantial work to answer: When was a subject first mentioned? How has the use of a specific word changed over time? Did my great-great-grandmother ever make the news?

Unfortunately, the answers to these questions may not be fully exhaustive. Modern OCR software struggles with historical documents, and as a result, many digitized historical collections contain errors [2]. In Kungliga biblioteket's collection, it is estimated that about 15% of all words are not recognized correctly [3]. These errors affect readability, information retrieval, and further analysis of the material [4]. Consider the following OCR transcription from Kungliga biblioteket's collection, with errors highlighted in bold:

> — Storartad gåfva till Göteborgs Museum. Den i HandelstidniDgens g&rdagsnnmmer omtalade hvalfisken, sorn fångats i Frölnndaviken, har i dag af hr brukspatronen James Dickson blifvit inköpt för 1,500 rdr och skänkt till härvarande Museum.

In this OCR transcription, ⟨u⟩ has been recognized as ⟨n⟩ in both *Frölnndaviken* (*Frölundaviken*) and *g&rdagsnnmmer* (*gårdagsnummer*). As a re-

[1] Kungliga biblioteket's newspapers are available at tidningar.kb.se. Other European digitized newspaper collections can be found at europeana.eu. (Links accessed November 2023).

sult, a user who searches for *Frölundaviken* would not find this news story. Another error is interpreting ⟨m⟩ as ⟨rn⟩ in *sorn* (*som*).

Minimizing OCR errors like these is necessary to enjoy the full potential of newspaper digitization. One way to achieve higher accuracy is to correct the OCR output, a task known as post-OCR correction. Various approaches to this task have been proposed [5], and among the more promising of these is to use machine translation methods to "translate" the OCR output to the correct version. These methods traditionally require massive amounts of training data. A previous study suggested that the available data for Swedish newspapers may be insufficient to train a long short-time memory (LSTM) based translation model from scratch [6].

Since the transformer model was introduced in 2017 [7], it has pushed the state-of-the-art in many natural language processing tasks, including machine translation. It has also led to the emergence of several pre-trained models, which require less training data and perform better than traditional methods. These have shown promising results in post-OCR correction and related tasks in several languages but have not yet been used for Swedish post-OCR correction.

1.1 *Aim*

This project aims to train a transformer model and evaluate its suitability for correcting OCR errors in Swedish newspapers. The model we use is ByT5, a transformer model suitable for text with character-level noise [8]. The questions we aim to answer are:

- How effective is a fine-tuned ByT5 in the task of correcting OCR errors in 19th and 20th century Swedish newspapers?
- With regard to transformer models' quadratic complexity in terms of sequence length, how should long texts be processed efficiently?
- What effect does further training on data from books and other domains have on the model's performance on newspapers?

1.2 *Contribution*

The fine-tuned model will be integrated in Språkbanken Text's annotation pipeline *Sparv* [9]. It is also available for download through HuggingFace[2].

[2] https://huggingface.co/viklofg/swedish-ocr-correction

2. *Post-OCR correction*

In the introduction we saw an example of a noisy OCR transcription. That particular transcription was produced by the commercial software Abbyy Finereader from the page shown in Figure 2.1. This chapter discusses how such software works and why errors occur with a focus on historical documents. This discussion is followed by a section on previous work in post-OCR correction and a section on common evaluation metrics.

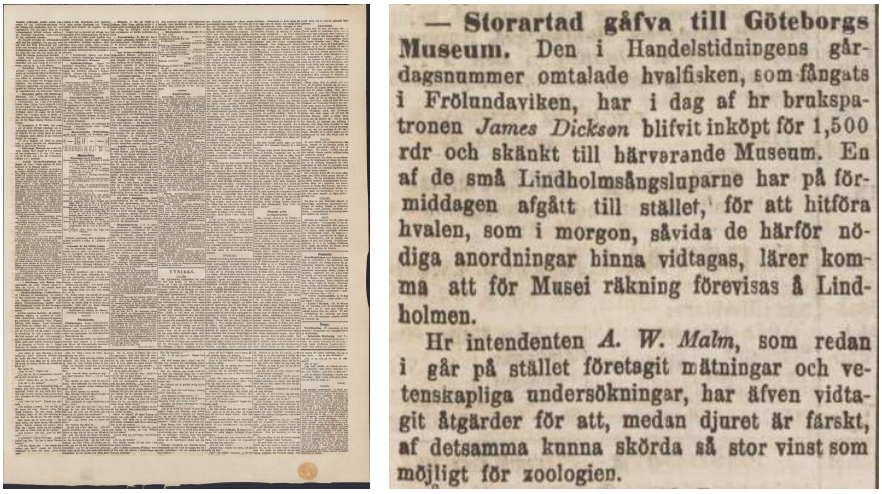
2.1 *The OCR process*

Today, various OCR software options are available. Aside from the already mentioned Abbyy Finereader, the open source Tesseract and OCRopus are popular options. They all use a multi-step process to extract text from images. The main steps in this process are layout analysis, binarization, and text recognition.

Layout analysis is necessary for newspaper pages and other documents with complex layouts. In this step, the page content is divided into text blocks. For example, the page shown to the left in Figure 2.1 would be divided into its seven columns, and each column would if necessary be further divided into paragraphs. The layout analysis also identifies images, ornaments, and other elements that should not be processed by OCR.

The aim of the next step, binarization, is to separate the text from the background. This is done by classifying each pixel as either text (black) or background (white). A simple binarization technique is to determine a suitable threshold value, and let all pixels that are darker than the threshold be classified as text and all lighter pixels be classified as background. Historical documents, however, may require more sophisticated techniques due to faded ink or yellowed paper [10].

Next, the binary image is segmented into individual lines, and often further into individual words or characters. The level of segmentation depends on the OCR software. These small text segments are then passed to the text recognition algorithm, which often is based on neural networks.



— **Storartad gåfva till Göteborgs Museum.** Den i Handelstidningens gårdagsnummer omtalade hvalfisken, som fångats i Frölundaviken, har i dag af hr brukspatronen *James Dickson* blifvit inköpt för 1,500 rdr och skänkt till härvarande Museum. En af de små Lindholmsångslupame har på förmiddagen afgått till stället, för att hitföra hvalen, som i morgon, såvida de härför nödiga anordningar hinna vidtagas, lärer komma att för Musei räkning förevisas å Lindholmen.

Hr intendenten *A. W. Malm*, som redan i går på stället företagit mätningar och vetenskapliga undersökningar, har äfven vidtagit åtgärder för att, medan djuret är färskt, af detsamma kunna skörda så stor vinst som möjligt för zoologien.

Figure 2.1: A page and a close-up of an article from Göteborgs Handels- och Sjöfartstidning, 1 November 1865, via Kungliga biblioteket [11]. OCR software applies several image processing steps before it is able to recognize text from an image like this.

2.2 *OCR of historical documents*

A major contributing factor to OCR quality is the physical condition of the source document. Old documents are seldom in perfect condition: stains, faded print, and ink bleed-through from other pages are examples of common issues. Documents with these issues are difficult to binarize properly [10], which may lead to broken or merged characters. Exposure to moisture may lead to warping of the paper, which affects both layout analysis and segmentation [12].

Another challenge is old typography, which may cause OCR errors even in well-preserved documents. The typical example is the character ⟨ſ⟩ (long s), an archaic form of ⟨s⟩, which is often understandably mistaken for ⟨f⟩. In early Swedish texts the umlaut above ⟨ä⟩ and ⟨ö⟩ was sometimes written as a tiny superscripted ⟨e⟩: ⟨a͛⟩ and ⟨o͛⟩, as seen in the headline of the advertisement shown in Figure 2.2. These glyphs are mainly found in blackletter typefaces. The blackletter typeface Fraktur was the standard font in Swedish texts until it was phased out in favour of modern Antiqua typefaces during the 1800s [13]. Fraktur is characterized by its thick vertical strokes, fancy capital letters and many ligatures, as shown in Figure 2.2. Although several OCR engines now support Fraktur, the results are often less accurate than on Antiqua texts [14].

Finally, OCR software often relies on dictionaries to resolve ambiguous cases. If it considers several different interpretations of a word, it will generally favour words found in the dictionary [14]. Constructing

Läro=Curs uti Tyſka Språket.

En ung man, ſom i flera år wiſtats uti Tyſkland, ämnar härſtädes gifwa en Curs uti Tyſka Språkets läſande, ſkriſwande och conwerſation. Priſet för en Curs, beräknad till Tre månader, är 20 R:dr Banco.

Underwisning meddelas ſå wäl Herrar ſom Damer och äfwen timwis efter öfwerenskommelſe med Läraren, ſom bor uti Hufet N:o 147 på Söder, en tr. upp.

Obs. Som Lärarens wiſtande härſtädes beror på ett wiſſt antal Elewer, torde anmälan med det ſnaraſte göras.

Figure 2.2: An advertisement printed in the blackletter typeface Fraktur. The title reads *Läro-Curs uti Tyſka Språket*. Blackletter fonts may lead to confusion of characters that are easy to distinguish in modern typefaces, for example *k* and *t*, and *ä* and *å*.

such dictionaries for historical language is difficult due to variations in spelling and vocabulary.

2.3 *Previous work*

Many different approaches to post-OCR processing have been proposed. Nguyen et al. list no less than eleven categories of approaches in their 2021 survey of the field [5]. They group automatic post-OCR correction methods into two main categories: isolated word and context-dependent approaches. Both kinds have been studied in the context of post-OCR correction of historical Swedish texts.

Isolated word approaches consider each word in the OCR output individually. An example is Persson [15] who used a support vector machine (SVM) classifier to detect incorrect words in OCR transcriptions of 17th to 19th century Swedish texts. For each detected word, a word list is consulted to find the best replacement word based on similarity and frequency. Another approach that falls under the isolated word category is to merge OCR transcriptions produced by different software. This method has been tested by Kungliga biblioteket to improve OCR accuracy in its digitized newspapers. The method, which was developed in collaboration with the company Zissor, merges outputs from the engines Tesseract and Abbyy Findereader [3].

Context dependent approaches consider the context around each word in the correction process. This way, a context dependent approach can handle *real word* errors: cases where the OCR system recognizes a word as another valid word (e.g., if *eat* is recognized as *cat*). This advantage comes at the cost of higher complexity.

A post-OCR correction approach that falls under the context dependent category is to use machine translation techniques to "translate" the OCR output to correct text. Thanks to the recent rapid development in neural machine translation, this approach has shown promising performance in post-OCR correction in various languages. Examples include the winner of the 2019 ICDAR competition [16], which was trained on ten European languages (but not Swedish). More recent machine translation based post-OCR correction models have been published for Finnish [17], Icelandic [18], and English [19], [20].

Nguyen et al. [5] note that while machine translation models outperform other techniques, they need a lot of training data to be successful. Lundberg and Torstensson [6] found that the amount of available training data for post-OCR correction of Swedish newspapers was not enough to train a neural machine translation model from scratch.

2.4 *Accuracy evaluation*

The accuracy of an OCR transcription is measured by comparing it to the correct transcription, the ground truth. A common metric in OCR evaluation is the character error rate (CER). A CER of 1% means that one out of 100 characters is not recognized correctly by the OCR system. The CER is defined as

$$\text{CER} = \frac{S_c + D_c + I_c}{N_c},$$

where S_c, D_c and I_c denote the minimum number of character substitutions, deletions, and insertions needed to correct the OCR transcription, and N_c is the number of characters in the ground truth. The numerator $S_c + D_c + I_c$ is an established string similarity metric known as the Levenshtein distance.

The word-level equivalent to CER, word error rate (WER), is also a common evaluation metric. It is computed in the same way:

$$\text{WER} = \frac{S_w + D_w + I_w}{N_w}.$$

Here, S_w, D_w and I_w denote the minimum number of word substitutions, deletions and insertions needed to correct the OCR transcription. N_w is the number of words in the ground truth. An example calculation of CER and WER of the introductory example is shown in Figure 2.1.

While seemingly similar, CER and WER measure different aspects of OCR quality. The CER measures to what extent the OCR transcription differs from the ground truth, and by extension, how much of it needs to be corrected, which makes it a relevant metric in the context

of post-OCR correction. The WER, however, is a good indicator of how these errors affect the practical use of the material since searching and indexing are done on the word level.

A natural way of evaluating post-OCR correction methods is to compare the error rates before and after processing. This is indeed an established practice [5]. When evaluating the detection task – a binary classification task – the standard precision and recall metrics are commonly used (e.g. in the ICDAR competitions [21], [16]). With some modification, these metrics can be used for the combined detection and correction task as well. They are defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

where TP (true positive) is the number of corrected errors, FP (false positive) is the number of introduced errors and FN (false negative) is the number of errors that remain after correction [5]. Examples of the different cases are given in Table 2.2.

Table 2.1: Calculation of the error rates of the example in the introduction. In total, four word edits (all substitutions) and six character edits (five substitutions and one deletion) are needed to correct it. With 33 words and 236 characters, this gives a WER of 12% and a CER of 2.5%.

		Word-level	Character-level
Edits	HandelstidniDgens → Handelstidningens		D → n
	g&rdagsnnmmer → gårdagsnummer		& → å
			n → u
	sorn → som		delete r
			n → m
	Frölnndaviken → Frölundaviken		n → u
Total edits		4	6
N		33	236
Error rate		12%	2.5%

Table 2.2: The possible post-OCR correction cases from which precision and recall metrics are derived.

Class	Example	Comment
True positive (TP)	sorn → som	Corrected an error
False negative (FN)	sorn → soru	Changed an error
	sorn → sorn	Ignored an error
False positive (FP)	hvalfisk → hva1fisk	Introduced an error
True negative (TN)	hvalfisk → hvalfisk	No errors, no changes

3. *Transformer models*

Machine translation methods are popular tools for post-OCR correction, as discussed in Section 2.3. A contributing factor to their popularity is the past years' significant progress in the field of neural machine translation. Driving this progress is the transformer architecture, which was introduced in 2017 [7].

This chapter provides a background on transformer models. The first two sections discuss the transformer architecture. These are followed by an introduction to transfer learning, which is a common approach to using transformer models. Finally, the particular transformer model used in this project, ByT5, is introduced.

3.1 *Encoder-decoder models*

The transformer belongs to a family of models called encoder-decoder models. These models are the standard choice for sequence-to-sequence (seq2seq) tasks such as machine translation because of their ability to handle input and output sequences of different lengths. This ability is illustrated in Figure 3.1, where the five-word phrase *She always reads the newspaper* is translated into the corresponding four-word Swedish phrase *Hon läser alltid tidningen*. The figure also illustrates the two components of an encoder-decoder model: The encoder and the decoder. The encoder is responsible of reading the input and computing an internal representation of it. This representation is then passed to the decoder, which produces the output sequence.

The encoder and decoder of the model illustrated in Figure 3.1 are based on recurrent neural networks (RNNs). This was the most common encoder-decoder architecture prior to the transformer. As the illustration shows, RNN-based models operate sequentially; The encoding of each token depends on the encoding of the previous token. This allows the model to keep track of the order of the sequence but has its drawbacks. First, the sequential operations cannot be parallelized. As a consequence, training of RNN-based models is slow and cannot benefit from the use of modern hardware which excels in parallel computation. Second, RNNs struggle to capture dependencies between

tokens that are far away from each other, and may forget information at the beginning of the sequence in long enough inputs.

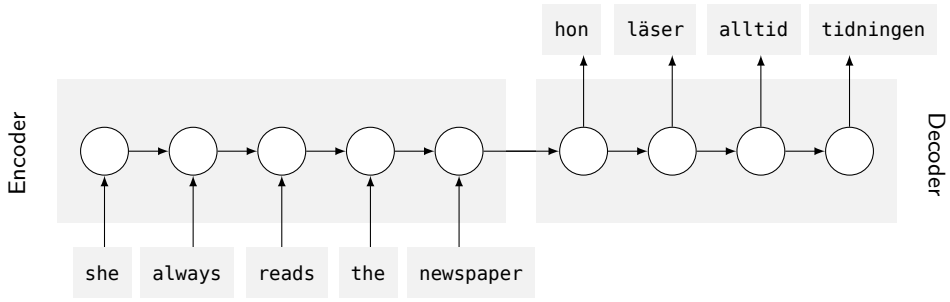


Figure 3.1: An illustration of translation from English to Swedish using an RNN-based encoder-decoder model. Each circle represents one RNN cell. The input sequence is encoded sequentially; The encoding of the last token, *newspaper*, is dependent on the encoding of all previous tokens.

3.2 *Attention*

To alleviate RNN-based models' forgetfulness, a mechanism called *attention* was introduced in 2015 [22]. With attention, all states of the encoding process are exposed to the decoder, rather than just a final vector. At each decoding step, the attention mechanism identifies the most relevant encoding states. Figure 3.2 illustrates how attention may be directed to different parts of the input sequence in each step of the decoding process. In the final decoding step, the attention mechanism gives the tokens *the* and *newspaper* high weight. Based on this information, the decoder outputs *tidningen*. How these weights should be distributed is learnt during training of the model.

Two years after the introduction of attention, the transformer was proposed by Vaswani et al. in a paper titled *Attention is all you need* [7]. As the title of the paper suggests, the transformer is fully based on attention. A simplified illustration of its architecture is shown in Figure 3.3. Like earlier models, it uses attention to connect the encoder and decoder ("cross attention" in transformer terminology), but it also uses attention within the encoder and decoder, instead of the traditional RNNs. This type of attention is known as self-attention, and is a way to model relations between tokens within the same sequence.

In contrast to RNN-based models, the transformer is able to process the entire input sequence all at once. It keeps track of sequence order by adding positional information to each token. As an example, the input tokens [*she, always, reads, the, newspaper*] could be encoded as [(*she,*

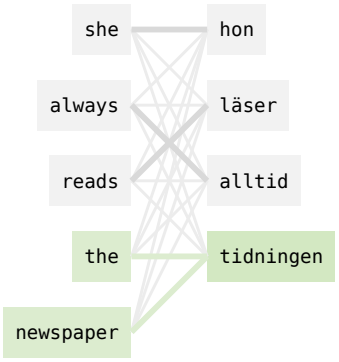


Figure 3.2: Illustration of encoder-decoder attention. In the last decoding step, the attention mechanism tells the decoder to focus on *the* and *newspaper*.

0), (*always*, *1*), (*reads*, *2*), (*the*, *3*), (*newspaper*, *4*)].[1] The input tokens, along with their positional information, are then passed to the self-attention mechanism, which for each token computes weights representing how related the token is to all other tokens in the sequence. For example, the token *newspaper* may have a stronger relationship with *the* and *reads* than with *always*. The self-attention weights for each token are combined in a standard feed forward network to produce the final encoding. This way, the transformer encodes (1) the input tokens, (2) their position in the sequence, and (3) the relationships between them, in a fully parallel manner.

Since the attention mechanism considers every pair of tokens in the sequence, it has quadratic complexity in terms of sequence length. At the same time, this allows the model to capture relations between tokens regardless of their distance from each other.

3.3 *Transfer learning*

Thanks to the parallelisable attention mechanisms, transformer models are much more efficient to train compared to earlier models. With enough resources, a transformer model can be trained on colossal amounts of data, much larger than typical labelled datasets. This has led to a popularisation of transfer learning, which is the practice of pre-training a model on large amounts of unlabelled data, and then fine-tuning it to a specific task with labelled data. The aim of pre-training is to let the model acquire general knowledge and language skills. Examples of pre-training objectives are predicting the following text [23] and filling

[1] In practice, more sophisticated positional encoding schemes are used.

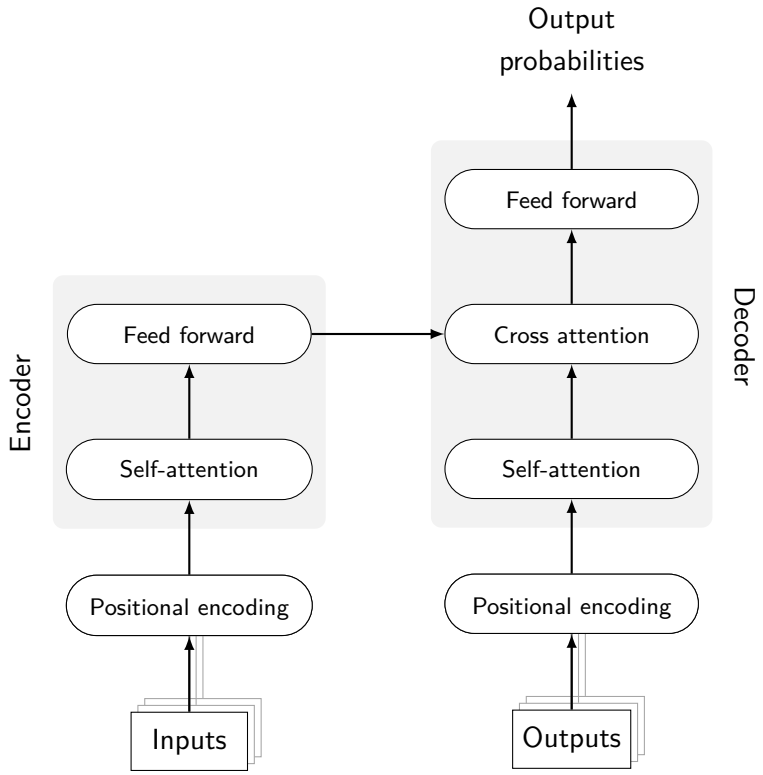


Figure 3.3: A simplified illustration of the transformer architecture proposed by [7], and used by T5 and ByT5. In practice, the encoder and decoder consist of several stacked layers. Note that the input tokens are fed simultaneously to the encoder rather than sequentially as in the RNN-based model in Figure 3.1.

in masked words [24]. In the fine-tuning step, the model is adapted to specific tasks such as translation or summarization.

Transfer learning generally leads to better performance compared to training a model from scratch using labelled data only [25]. At the same time, it cuts training time, since the same pre-trained model can be fine-tuned to virtually any task. This versatility was demonstrated by Raffel et al. [24], whose pre-trained transformer model T5 achieved state of the art results in not only seq2seq tasks such as summarisation and question answering, but also classification tasks mapped to the seq2seq format.

3.4 *ByT5*

ByT5 is a character-level version of T5, introduced by Xue et al. in 2021 [8]. Both T5 and ByT5 use the original transformer architecture as proposed by [7] and shown in Figure 3.3. As a character-level model,

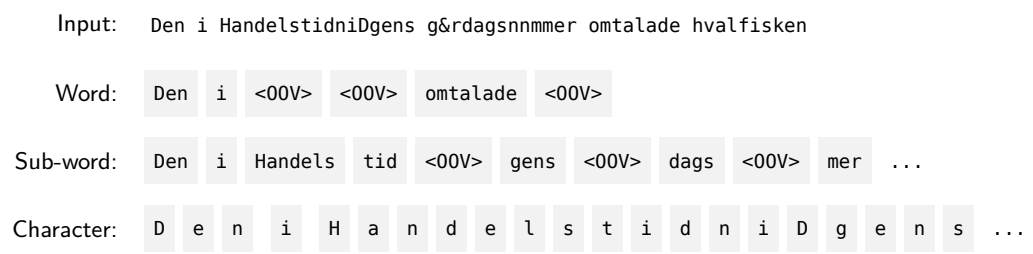


Figure 3.4: Word, sub-word and character-level tokenization of the sequence *Den i HandelstidniDgens g&rdagsnnmmer omtalade hvalfisken*. ByT5 uses character-level tokenization, which preserves words that are not covered by the model's vocabulary due to for example OCR errors (*HandelstidniDgens*) or age (*hvalfisken*), at the cost of increased sequence length.

ByT5 handles text as a sequence of characters, instead of as a sequence of word or sub-word tokens. The different tokenization approaches are illustrated in Figure 3.4. This low-level representation of text has shown to be suitable for tasks sensitive to character-level noise [8]. Since the sizes of language models' vocabularies are fixed, a word-level model would map all out-of-vocabulary words (e.g., misspelled or obsolete words) to the same out-of-vocabulary token `<OOV>`, as shown in Figure 3.4. As a result, all information about these words is lost. A character-level model like ByT5 preserves the information at the cost of increased sequence length.

ByT5 is trained on mC4, a dataset that was originally prepared as training data for the multilingual T5 (mT5) by Xue et al. [26]. The dataset consists of texts from Common Crawl, a non-profit initiative that provides monthly scrapes of the internet. With this data a starting point, Xue et al. create mC4 by filtering out source code[2], placeholder text[3], offensive language[4] and duplicate text. The resulting dataset contains 6.3 trillion tokens covering 107 languages, among them Swedish, which constitutes 1.61% of the dataset.

Xue et al. report that ByT5 achieves a comparable performance with its token-based counterpart, mT5, in many tasks. ByT5 does however outperform mT5 in tasks that concern spelling and pronunciation, which Xue et al. demonstrate on transliteration and grapheme-to-phoneme tasks. Their findings are supported by Stankevičius et al. who use the model to achieve results comparable to state-of-the-art in diacritics restoration in thirteen languages [27], and by Maheshwari et al. who find that ByT5 performs well in post-OCR correction in Sanskrit [28].

[2]Any page that contains curly brackets { }

[3]Any page that contains the phrase *Lorem impsum*

[4]Any page that contains words found on the *List of dirty, naughty, obscene and otherwise bad words*: `https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words`

4. *Data*

As discussed in Chapter 3, the model used in this project, ByT5, is pre-trained on web-scraped data. Like other transfer learning models, it is adaptable to new tasks by fine-tuning on labelled task-specific data. For the task of post-OCR correction, this fine-tuning data consists of OCR-processed texts and their corresponding ground truth.

This project's main source of such data is a manually transcribed subset of Kungliga biblioteket's digitized newspapers. In addition to this dataset, three other datasets are used: one dataset containing OCR-processed literature and two datasets containing OCR-processed black-letter texts. An overview of all datasets is given in Table 4.1. This chapter describes the datasets in detail and the specific processing steps applied to the respective datasets.

Table 4.1: An overview of the datasets

Dataset	Partition	Time period	Characters	CER (%)
Newspapers	*Tesseract*	1818–2018	6 956 748	4.86
	Abbyy Finereader		6 928 171	3.85
Literature		1836–2001	7 266 866	1.63
Swedish fraktur		1626–1816	282 432	17.61
Then swänska Argus		1732–1734	259 468	19.06

4.1 *Newspapers*

The newspapers dataset was originally prepared as an evaluation set for the two-OCR engine method proposed by Dannélls et al. [3]. The dataset consists of almost 44,000 text segments, identified by layout analysis of 400 newspaper pages, two pages per year between 1818 and 2018. The content of the segments varies from headlines of a few words to paragraphs spanning several lines. One of the 400 pages is shown in Figure 4.1.

Each text segment is manually transcribed by the German company Grepect using double-keying. The transcriptions are annotated with tags that indicate font styles. Ligatures are marked with an underscore,

for example, the ⟨tz⟩ ligature is notated as ⟨t_z⟩, with an exception of the ⟨ſs⟩ ligature, which has its own Unicode representation ⟨ß⟩.

Two OCR transcriptions exist of each segment: one produced by Tesseract and one produced by Abbyy Finereader. A third version, which is the combination of the two described in [3], is also included in the dataset. Over the entire set, Dannélls et al. report that Tesseract achieves a WER of 12.6% and Abbyy Finereader 16.6%. However, the older parts of the collection have higher error rates. The WER of material published in the period 1818–1906 varies from 17.6% (Tesseract) to 28.0% (Abbyy Finereader). They also estimate that 75% of the material published in this period is printed in Fraktur [3].

The dataset reflects two hundred years of development of the Swedish language. It covers the transition from Modern Swedish (*nysvenska*, 1525–1900) to Contemporary Swedish (*nusvenska*, 1900–), which involved changes in both grammar and spelling [29]. The contemporary Swedish spelling was largely settled with the 1889 and 1906 spelling reforms [30], which means that the dataset contains both modern and historical spelling, for example *vad* and *hvad*, *kvarn* and *qvarn*.

Parts of the dataset are available for download through Språkbanken Text under the names *Swedish newspapers 1818–1870* and *Swedish newspapers 1871–1906*. Due to copyright restrictions, material published later than 1906 is not publicly available.

Pre-processing

We used Lundberg and Torstensson's processed version of this dataset as a starting point [6], which is stripped of transcriber annotations. However, some disagreements, not caused by OCR errors, remain between the OCR outputs and their transcriptions, including:

- The Abbyy Finereader output does not distinguish between ⟨ſ⟩ and ⟨s⟩, and uses only the latter. This is unlikely an OCR error, but instead a result of the OCR software regarding ⟨ſ⟩ as a typographic variation of ⟨s⟩ and not a separate character.

- The blackletter hyphen ⟨⸗⟩ is transcribed as the regular hyphen ⟨-⟩ in the OCR outputs, most probably by the same reason as above.

- Layouts within a segment (e.g., small tables) are represented by tabs in the transcriptions, while the OCR output only use spaces.

- The characters ⟨a̋⟩ and ⟨ő⟩ appear occasionally in the transcriptions. While these characters may theoretically appear in foreign

words in Swedish texts, the majority of them can safely be assumed to represent ⟨ä⟩ and ⟨ö⟩.

Before use in this project, these disagreements were solved by adjusting the transcriptions accordingly. All ⟨ſ⟩ in transcriptions paired with Abbyy Finereader output were replaced with ⟨s⟩. Equal signs followed by a line break were replaced by a hyphen, tabs were replaced by regular spaces, and the double-accented ⟨a̋⟩ and ⟨ő⟩ were replaced by ⟨ä⟩ and ⟨ö⟩.

4.2 *Literature*

The literature dataset consists of 79 titles of Swedish literature printed between 1836 and 2001. In total, it contains about seven million characters, making it roughly the same size and from the same period as the newspapers (see Table 4.1). The OCR quality is generally much higher than the newspapers', most likely because of higher paper quality and simpler page layout, which can be seen in Figure 4.1.

The data was provided as XML files by the Swedish Literature Bank, *Litteraturbanken* [1]. The formats of the OCR outputs and manual transcriptions are slightly different. The manual transcriptions are encoded line by line, as they appear on the page, while the OCR transcription is a list of the page's words and their location on the page. For this project, both formats were converted to plain text. Apart from this, no other processing was done to this set.

4.3 *Swedish fraktur*

The dataset *Swedish fraktur* is the first of the two blackletter datasets provided by Språkbanken Text.[2] It was prepared within the project *A free cloud service for OCR* [31], a collaboration between Språkbanken Text and Gothenburg University Library. The texts are taken from 199 pages published between 1626 and 1816 from the library's collections and transcribed by Grepect. One of these pages is shown in Figure 4.2. The material is available in plain text format through Språkbanken Text. The transcriptions are annotated with typographic information such as font changes and ligatures; these were removed for this project. Aside from this, no further processing was done.

[1] `https://litteraturbanken.se/`

[2] `https://spraakbanken.gu.se/en/`

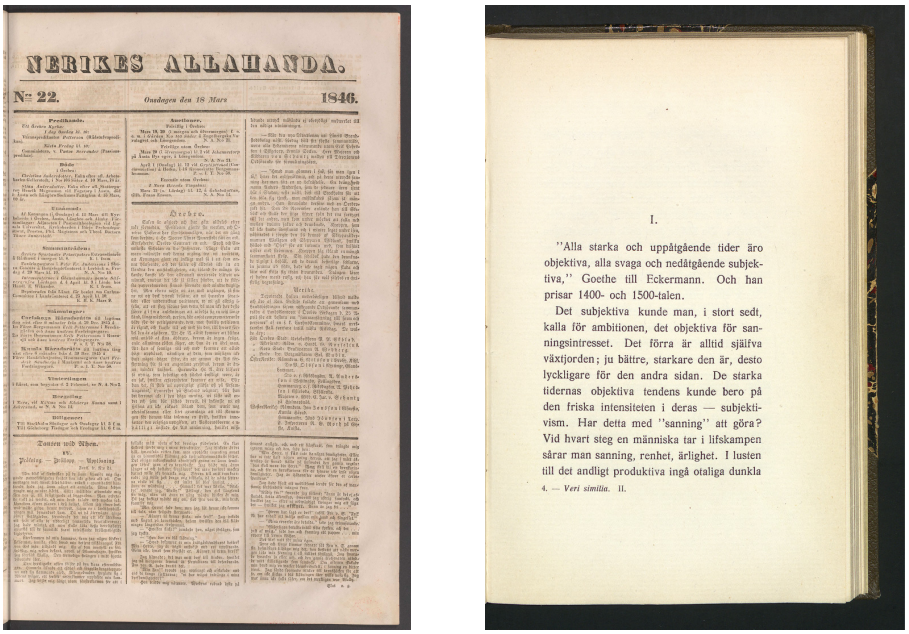


Figure 4.1: Sample pages from the newspaper dataset (left) and the literature dataset (right).

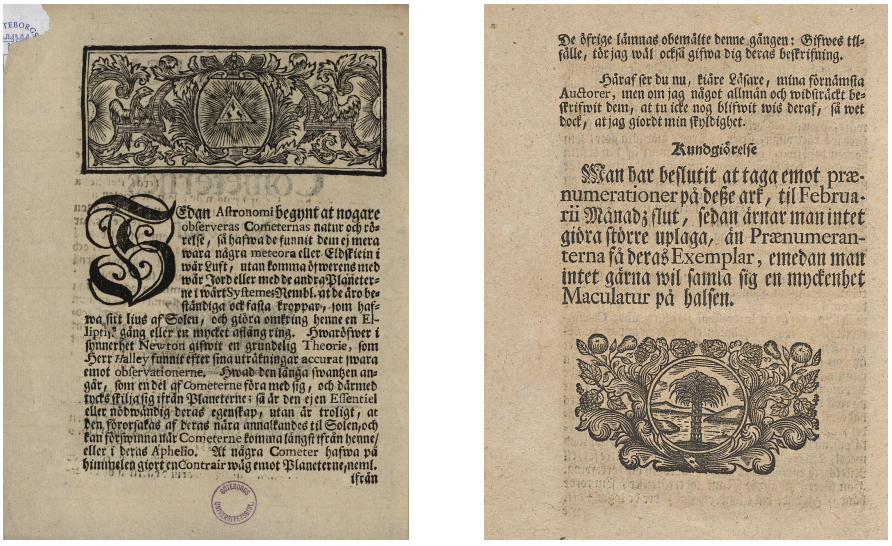


Figure 4.2: Sample pages from *And. Celsii Tanckar om cometernes igenkomst*, part of the *Swedish fraktur* dataset (left), and *Then swänska Argus* (right).

4.4 *Then swänska Argus*

The second source of blackletter data is *Then Swänska Argus*, which was prepared within the same project as *Swedish fraktur* [31]. This dataset consists of 25 issues of *Then swänska Argus*, a periodical by Olof von Dalin published between 1732 and 1734. It is significantly older than the newspaper texts, but the language is casual and simple for its time, and it is often cited as the first example of younger Modern Swedish (*yngre nysvenksa*).

The transcription is based on a transcription made at Uppsala University, with some modification by [31]. The original *Then swänska Argus* has been republished at least once, and the transcription does not appear to be directly tied to a specific printing. As a result, the transcription does not contain any line breaks. In order to align it with the OCR output, line breaks were inserted in the transcription. The line breaks were inserted such that the sum of the CER of each individual line was minimized.

5. *The model*

Chapter 4 described the four datasets used in this project. After some processing, each dataset consists of pairs of OCR output and their ground truth, each in plain text format. This data forms the foundation for fine-tuning and evaluating the post-OCR processing model.

This chapter aims to present the post-OCR processing model. First, we describe how the training, testing, and evaluation data was prepared from the four datasets. This is followed by descriptions of the training setup and three different versions of the model. We also present a strategy for how to use the model to process long texts.

5.1 *Preparing the data*

Alignment

The lengths of the texts in the four datasets vary from a few characters to several thousand. Most of them were too long to be processed by a single pass through the model (a discussion of the model's size limit follows in Section 5.2). The main data pre-processing step was thus to split the texts into shorter pieces. Careful consideration was taken to keep the OCR output and its ground truth aligned, since misaligned training samples may encourage the model to delete or insert text. Since an OCR output and its ground truth may differ in length, simply slicing the texts at the same character or word index does not work. We do however assume that the OCR software and human transcribers agree on line breaks.

Each data sample was aligned using a modified version of Myers' difference algorithm [32]. The algorithm, in its original implementation, compares two files A and B, and returns (1) which lines are present in both files, (2) which lines are only present in file A, and (3) which lines are only present in file B. In its original implementation, it considers two lines the same if they are equal. When comparing OCR files and their transcription, this is naturally not the case. We modify the algorithm such that it considers two lines "equal" if their character

error rate is low enough; the threshold was set to 30-50% depending on the source.

Filtering

With all data samples aligned line-by-line, they were split using line indexes. Again, we split the segments on line breaks, to avoid misalignment. Some samples do however contain multiple lines if the lines are short enough. From these samples, the following types of samples were removed:

- Short samples, where either the OCR output or the ground truth is shorter than four characters. As Lundberg and Torstensson [6] reason, shorter samples cannot contain the context necessary for correction.

- Samples with character error rate above 50%, which may be the result of very poor OCR or failed alignment. In the first case, the large amount of errors may obscure the context needed for correction. In the latter case, the model may learn to remove or insert words in an unwanted way.

- Samples where the ground truth contains ⟨@⟩, which is used in the transcriptions of the newspapers and *Swedish fraktur* to indicate illegible text, and not something the model should learn to generate.

Figure 5.1: Alignment of OCR output and its ground truth. In this sample, taken from the page of *And. Celcii Tanckar om cometernes igenkomst* shown in Figure 4.2, the OCR output (left) and its original ground truth (middle) differ significantly due to an ornamental page header. After alignment (right), the OCR output and the ground truth can be compared line-by-line.

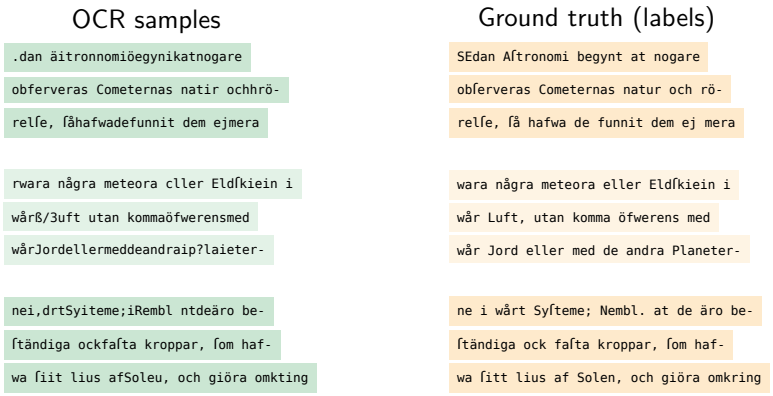


Figure 5.2: Examples of training data. After alignment (Figure 5.1), the texts were split in segments of at most 128 UTF-8 bytes each.

Data splits

The newspaper samples were randomly split into three subsets: train (70%), test (15%), and evaluation (15%). Here, test refers to the test set used during training, and evaluation a hold-out set that the model will not see during training. The two versions of each sample (one processed by Tesseract, one by Abbyy Finereader) were put in the same split. This way, there is no contamination between the sets. The remaining samples were split into two subsets each: train (85%) and test (15%).

5.2 *Fine-tuning setup*

The base model, ByT5, was accessed through Huggingface's Transformers library [33]. The 300 million parameter version, `byt5-small`, was used. This is the smallest of the five available versions of ByT5. Xue et al. report that in most downstream tasks, the model's performance increases with its size [8]. However, this tendency is not as strong in tasks that concern spelling or pronunciation, suggesting that a larger model's potential performance increase in post-OCR correction may not motivate its larger memory footprint.

The maximum input and output lengths were set to 128 UTF-8 bytes, which corresponds to slightly less than 128 characters[1]. This is significantly shorter than the original 1024 bytes [8] but assumed to be enough to cover the relevant context for correcting OCR errors. Lowering the input size improves the speed of the model since the attention mechanism has quadratic complexity in terms of sequence length.

[1]ByT5 uses UTF-8 encoding, in which most characters occupy one byte, but non-ASCII characters such as ⟨å⟩, ⟨ä⟩ and ⟨ö⟩ occupy two or more bytes.

The training was done using the Trainer API provided by Huggingface. The Adafactor optimizer [34] was used with a constant learning rate of 0.001, mimicking the original setup used by both Xue et al. in fine-tuning ByT5 [8] and by Raffel et al. in fine-tuning T5 [24]. The Adafactor optimizer has a lower memory footprint than Adam, its main alternative. The batch size was set to 32, giving a total batch size of $128 \cdot 32 = 2^{12}$ tokens per batch. The remaining hyperparameters were left to their default values as of Transformers version 4.34.0.

5.3 *Model instances*

Using the setup described in Section 5.2, three different versions of the model were fine-tuned: Model 1, Model 2, and Model 3. The only difference between the three is what data they were fine-tuned on. All three models were trained on the same newspaper data, but Model 2 and Model 3 were further trained on data from other sources. An overview of the three model instances is given in Table 5.1.

The first model, Model 1, was trained on newspaper data only. This model is comparable to instances 2–4 of Lundberg and Torstensson's model [6].

The second model, Model 2, was trained on a mix of newspaper and literature data. The ambition with this mix was to provide more examples of 19th and early 20th century Swedish, as it can be assumed that ByT5's pre-training data is mostly contemporary Swedish. The literature dataset has better OCR quality than the newspapers. At almost 190 000 training samples, Model 2 is the longest trained model.

The final model, Model 3, was trained on a mix of newspaper and blackletter data. The blackletter data consisted of the datasets Swedish fraktur and Then swänska Argus. It is estimated that 75% of the samples from 19th century newspapers are printed in blackletter [3]. These contain different kinds of errors compared to modern newspapers. By

Table 5.1: Training and test data for the three model instances (number of samples).

	Model 1		Model 2		Model 3	
Dataset	Train	Test	Train	Test	Train	Test
Newspapers	125 637	26 456	125 637	26 456	125 637	26 456
Literature	-	-	63 867	11 271	-	-
Swedish fraktur	-	-	-	-	2 487	453
Then swänska Argus	-	-	-	-	2 394	408
Total	125 637	26 456	189 504	37 727	130 518	27 317

adding more blackletter data to the training mix, the ambition was to improve performance on earlier texts.

5.4 *Processing long texts*

Since the model accepts inputs of at most 128 UTF-8 bytes, many input texts are too long to be processed in a single pass through the model. These texts may be handled by splitting them into segments of at most 128 bytes, processing each segment individually, and then piecing the outputs together to form the final output. This approach is illustrated in Figure 5.3.

When the input text is split, there is a risk that the context needed for correction of one text segment ends up in another. An extreme example of this would be if the OCR software has recognized one word as two, and the two word pieces end up in different segments. The risk of this happening is rather low, nonetheless, the context provided with each word depends on how the input text is split and may influence the model's prediction.

To minimize this risk and to obtain robust results, we segmented each input text in n different ways. Figure 5.4 illustrates alternative segmentations of an input text. The n versions were processed according to the procedure shown in Figure 5.3. This gave the model n opportunities to correct potential errors (and also n opportunities to refrain from editing correct characters), each with a slightly different context than the others. A final output was created by merging the n outputs. The merge was done by aligning the texts character-by-character. Any disagreements were solved by vote, and a majority vote was required to edit an original character. For example, if all n outputs give different suggestions, the original character was kept unedited. In section 6.2, the results with $n = 1$, 3, 5, and 7 are reported.

5.5 *Evaluation*

The models were evaluated on a 15% subset of the newspaper data. This evaluation set was randomly selected and not used in the training of any model. We prepared two versions of the set: EVAL-SHORT and EVAL-LONG. The first version, EVAL-SHORT, was processed in the same way as the training data. In the second set, EVAL-LONG, all samples were kept at their original lengths. Both sets were filtered on the same conditions as the training data, i.e., short samples and samples with too high error rate were removed (see Section 5.1 for details).

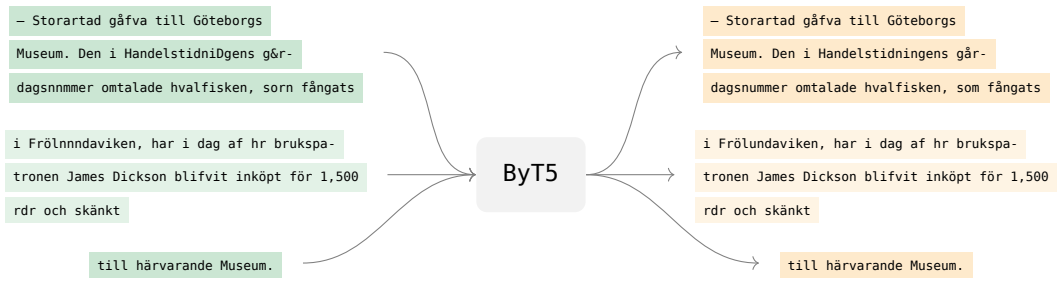


Figure 5.3: When processing texts that are longer than the model's limit of 128 UTF-8 bytes, the input text is split into chunks of at most 128 bytes each. Each chunk is individually processed by the model, and the outputted chunks are concatenated to form the final output.

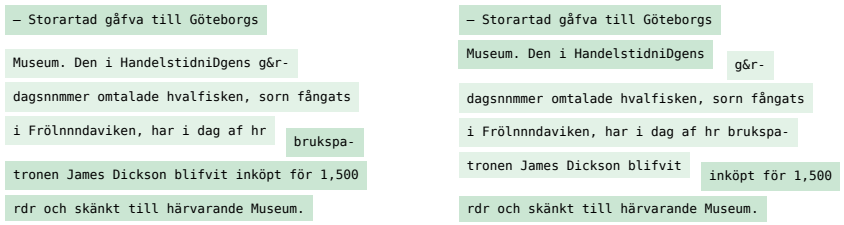


Figure 5.4: Examples of alternative segmentations of the input text in Figure 5.3.

First, the three models were evaluated on EVAL-SHORT. Since this set is prepared in the same way as the training and test sets, evaluation on this set indicates how well the model has learned the fine-tuning task. The predictions were computed using greedy decoding, i.e., at each decoding step, the highest-probability character was selected. The CER and WER were computed with the Python library `jiwer`[2], which also provides a utility for aligning texts. This alignment was used to compute the precision and recall. The results are reported in section 6.1. Next, the model that achieved the lowest CER on EVAL-SHORT was used for evaluating the strategy for processing long texts described in Section 5.4. For this task, we used EVAL-LONG. The error rates, precision, and recall were computed as for EVAL-SHORT, and are reported in Section 6.2.

[2]Version 3.0.3., available at `https://pypi.org/project/jiwer/`, accessed November 6, 2023)

6. *Results*

In this chapter, the results from the evaluation described in Section 5.5 are presented. We first compare the performance of the three model instances on the evaluation set EVAL-SHORT. This is followed by an evaluation of the strategy for processing long texts that was described in Section 5.4.

6.1 *Comparison of model instances*

Three versions of the model were trained: Model 1 on newspaper data, Model 2 on newspapers and literature, and Model 3 on newspapers and blackletter data. These were evaluated on EVAL-SHORT, an evaluation set with samples of at most 128 bytes each. A sample from this set with corrections suggested by each model instance is shown in Figure 6.1. Although the three suggestions differ slightly, the overall quality is similar. For example, the three model instances agreed that *ko»unqen«* was incorrect, but only Model 3 managed to correct it to *Konungen*. At the same time, Model 3 wrongly suggested *all* as a correction for *alk*, while the other two offered the correct *att*.

The three model instances tended to produce corrections of similar quality; see Figure A.1 and Figure A.2 for additional examples. This tendency is reflected in the error rates of the three model instances, which are listed in Table 6.1. The three models achieved similar error rates, and all successfully reduced the error rates at both character and word level. Although the differences between the three models were small, Model 1 and Model 3 consistently performed slightly better than Model 2. Over the entire set, Model 3 achieved the lowest CER of 2.04%, a reduction of 36% from the baseline CER of 3.20%. Model 1 achieved the lowest WER of 7.17%, a reduction of 45% from the baseline WER of 13.21%.

Models 1 and 3 outperformed Model 2 in terms of precision too. These results are listed in Table 6.2 together with the models' recall scores. Just like in Table 6.1, the differences between the three instances are small. All three showed a tendency for higher precision and recall in the earlier material, i.e., the more noisy texts. Over the entire

OCR — tz. M. ko»unqen« tillfriffFnanbe kor lock,
ligtwis nu sä fortgått alk H. M. den >6 för för,

GT — H. M. konungens tigfrisknande har lock-
ligtwis nu så fortgått att H. M. den 16 för för-

Model 1 — H. M. kommißens tillfrisknande kor lock,
ligtwis nu få fortgått att H. M. den 16 för för-

Model 2 — H. M. kommens tillfrisknande för lock-
ligtwis nu så fortgått att H. M. den 16 för för-

Model 3 — H. M. Konungens tillfrisknande kor lock-
ligtwis nu så fortgått all H. M. den 16 för för-

Figure 6.1: An 1852 sample from EVAL-SHORT with corrections suggested by the three model instances. Errors and corrections are highlighted in bold. Note that first line of the ground truth (GT) contains two mistakes, these are underlined.

set, Model 3 achieved the highest precision of 75.4%, while Model 1 achieved the highest recall of 54.8%.

Although the three model instances succeeded in lowering the error rates on EVAL-SHORT as a whole, they occasionally introduced new errors. For example, Model 3's precision of about 75% indicates that it introduced one new error for every three errors it corrected. It was found that Model 3 increased the WER in about 5.1% of the EVAL-SHORT samples. Some of these "corrections" showed to be repetitive nonsense, and was observed in output from all three models. Two examples of this phenomenon are shown in Figure A.3 and Figure A.4.

6.2 *Correction of long texts*

Section 6.1 reports the models' performance on EVAL-SHORT, an evaluation set with samples of at most 128 UTF-8 bytes. However, for practical use of the model, a strategy for processing longer texts is needed. Such a strategy is proposed in Section 5.4. In short, it includes segmenting the text in n different ways, processing each version individually, followed by merging the n outputs. This strategy was evaluated at different values of n on EVAL-LONG, using Model 3, which achieved the lowest CER on EVAL-SHORT.

Table 6.3 lists the post-OCR correction quality at different values of n on EVAL-LONG. At $n = 1$, the texts were passed through the model

Table 6.1: CER and WER of Model 1 (M1), Model 2 (M2) and Model 3 (M3) compared to baseline (BL) on EVAL-SHORT.

	CER (%)				WER (%)			
Period	BL	M1	M2	M3	BL	M1	M2	M3
1818–1859	8.39	4.39	4.63	**4.30**	32.46	**15.34**	15.80	15.54
1860–1899	4.04	**2.29**	2.61	2.38	16.51	**8.06**	8.63	8.22
1900–1939	2.60	2.01	1.97	**1.92**	11.24	7.03	7.08	**6.99**
1940–1979	1.46	1.39	1.49	**1.29**	6.45	4.43	4.54	**4.39**
1980–2018	0.83	**0.67**	0.75	0.73	3.74	**2.67**	**2.67**	**2.67**
1818–2018	3.20	2.06	2.20	**2.04**	13.21	**7.17**	7.42	7.23

Table 6.2: Precision and recall for Model 1 (M1), Model 2 (M2) and Model 3 (M3), on EVAL-SHORT.

	Precision (%)			Recall (%)		
Period	M1	M2	M3	M1	M2	M3
1818–1859	82.5	80.0	**85.0**	**60.6**	60.1	59.8
1860–1899	**79.7**	71.9	78.1	**58.2**	57.9	57.3
1900–1939	66.2	67.9	**70.3**	46.4	**46.5**	45.9
1940–1979	52.2	48.4	**56.6**	**47.7**	47.5	45.7
1980–2018	**64.2**	56.0	56.8	42.7	43.2	**46.2**
1818–2018	74.0	70.1	**75.4**	**54.8**	54.5	54.0

Table 6.3: Performance of different processing strategies on EVAL-LONG. The baseline CER is 3.26% and WER is 13.12%. Model 3 is used.

Merged versions (n)	CER (%)	WER (%)	Precision (%)	Recall (%)
1	2.62	8.17	62.9	49.9
3	2.05	7.28	79.0	50.2
5	2.03	7.23	80.0	**50.8**
7	**2.02**	**7.22**	**80.4**	**50.8**

only once. This resulted in a 19.6% reduction in CER (from 3.26% to 2.62%). Already at $n = 3$, a significant decrease in errors is seen on both character and word level. The error rates decrease further, albeit marginally, by increasing n to 5 and 7. While the improvement in error rates, especially the jump between $n = 1$ and $n = 3$, is mirrored in the precision metric, the recall stayed around 50%. The lowest error rates were achieved with $n = 7$: A 38% reduction in CER (from 3.26% to 2.02%) and a 45% reduction in WER (from 13.12% to 7.22%).

About a quarter of the texts in EVAL-LONG were affected by using $n > 1$ instead of $n = 1$. A manual inspection of these texts showed that the n different versions often agreed on whether a given source character should be edited or not, but the suggested replacement candidates could vary. An example of this is shown in Figure 6.2, which shows a sample from EVAL-LONG together with a correction obtained by merging three different outputs from Model 3. As can be seen in the figure, each version suggested a different correction for the OCR error ⟨rif rd⟩, only agreeing on changing ⟨f⟩ to ⟨k⟩. Since only edits that were suggested by the majority (here: two or three) are kept in the merged output, the original error was left unedited.

OCR	Sn underlig race att ſtudera, desfa upp- ſinnare! utropar en Londontidnings fro- nifôr. Wet ni hur ſtort antalet är af patenter, ſom ſiſtlidet är utfärdades i Bri- tish Patent Office? Jo, 14,000 ſty>en !! Det kan man ju fkalla en rif #rd! Fjor- ton tuſen uppfinninnar! Herre Gud, hwil- fet märkrwoärdigt tidehrvarf wi lefroa i!	nikör nikör nikör niför
Output	En underlig race att ſtudera, desſa upp- ſinnare! utropar en Londontidnings kro- nikör. Wet ni hur ſtort antalet är af patenter, ſom ſiſtlidet är utfärdades i Bri- tish Patent Office? Jo, 14,000 ſtycken! Det kan man ju kalla en rik #rd! Fjor- ton tuſen uppfinn#ar! Herre Gud, hwil- det märkwärdigt tidehwarf wi lefwa i#	rik #rd! rik rdr rik äro: rikets? uppfinn#ar uppfinningar uppfinn#ar uppfinn#ar
GT	En underlig race att ſtudera, desſa upp- [f]innare! utropar en Londontidnings kr[ö]- nikör. Wet ni hur ſtort antalet är af patenter, ſom ſiſtlidet år utfärdades i Bri- tish Patent Office? Jo, 14,000 ſtycken!! Det kan man ju kalla en rik ſkörd! Fjor- ton tuſen uppfinnin[g]ar! Herre Gud, hwil- ket märkwärdigt tidehwarf wi lefwa i!	

Figure 6.2: Sample from EVAL-LONG with $n = 3$ merged outputs from Model 3. The three versions disagreed at three locations, these are underlined in the OCR and output texts. The different candidates are shown in to the right. Errors and corrections are highlighted, missing characters are indicated with #. Three mistakes in the ground truth have been corrected and are indicated with square brackets.

7. *Discussion*

This thesis explores using ByT5 for post-OCR correction of 19th and 20th century Swedish newspapers. The work aims to answer three questions: How well does ByT5 perform on the task? How can long texts be processed efficiently? What is the impact of training on data from other domains? In this final chapter, we seek to discuss these questions based on the results reported in Chapter 6.

7.1 *Performance of ByT5*

The first research question of this thesis was whether fine-tuning ByT5 is a suitable approach to post-OCR correction of historical Swedish newspapers. We fine-tuned the 300M parameter version of ByT5, `byt5-small` for three epochs on three different datasets. Each of the these model instances successfully reduced the CER and WER in the evaluation dataset EVAL-SHORT.

The best character level result was achieved by Model 3, which was trained on newspapers and blackletter data. It accomplished a 36% reduction of CER (from 3.20% to 2.04%). This performance is comparable to the one reported by Janoarson et al. [18] whose fine-tuned ByT5 post-OCR correction model for historical Icelandic achieved a 35% CER reduction. It should be noted that they used `byt5-base`, which at 582M parameters is one size up from the version used in this project, `byt5-small`. This suggests that the choice of `byt5-small` was a good trade-off between size and performance.

Although it can be assumed that ByT5 has not been seen much old Swedish in its pre-training, the fine-tuned model did not seem to struggle with old Swedish spelling. In fact, both precision and recall were higher in the earlier material, as seen in Table 6.2, supporting the observation of Maheshwari et al. [28] that ByT5 is able to perform well in languages not seen during pre-training. It is possible that the error patterns are more predictable in the older material than the newer material. The older newspapers contain more systematic errors, for example confusion between ⟨ſ⟩ and ⟨f⟩, while errors in the newer ma-

terial are fewer and more random to their nature, and thus harder to find and correct.

All instances of the model were occasionally observed to produce gibberish repetitive output, as seen in Figure A.3 and Figure A.4. This is a common issue for generative language models, and was for example also observed by Lundberg and Torstensson in their LSTM-based model [6]. It has been shown that this behaviour can be reduced by using other decoding schemes instead of greedy decoding, which was used in this project [35].

This thesis did not delve into optimizing model hyperparameters, as it was found that the fine-tuning setup used by Xue et al. [8] yielded satisfactory results. However, it is likely that adjusting the fine-tuning setup and decoding strategy could further improve the model's performance.

7.2 *Processing long texts*

The second research question was how long texts should be processed. Primarily motivated by scarce computing power, the model's input and output lengths were shrunk from 1024 to 128 UTF-8 bytes. This was assumed to be long enough to cover the necessary context for the task. Any input text that is longer than the model's limit needed to be processed in chunks. This is the case regardless of the model's limit, however, the shorter limit increased the risk of splitting the input text in a way that affects the correction (e.g., if a sentence is split mid-way).

This issue was largely mitigated by processing the input text multiple times, each with a different segmentation of the input. This approach ensures that even if one segmentation splits the text at a bad location, the others will keep that text intact. The different outputs were merged and any disagreements were solved by vote.

This approach showed to be effective. By processing the texts of EVAL-LONG three times instead of once, the precision jumped from 63% to 79%, resulting in a further reduction of the baseline CER of 3.3% from 2.6% to 2.1% (see Table 6.3). The recall, however, remained around 50%, indicating that the higher accuracy was caused by fewer introduced errors, rather than more corrected errors. A slight further improvement in performance was achieved by merging more than three versions, but this marginal improvement may not be motivated out of a time perspective.

7.3 *Using training data from other domains*

The third and final research question was to evaluate how training data from other domains impacts the model's performance on newspapers. To answer this question, three models were trained. The first, Model 1, was trained only on the newspapers, while the other two were trained on the newspapers and further on data from other domains: Model 2 on literature, and Model 3 on blackletter data.

Model 1, which was only trained on the newspapers, achieved the best recall and WER. Training further on literature data did not yield any improvement in error rates, precision or recall. Adding the blackletter data to the training set, however, improved the CER and precision. It should be noted that the differences in performance between the three models were small.

7.4 *Future work*

This work showed that the existing training data for post-OCR correction of historical Swedish newspapers was enough to fine-tune ByT5 on the task. In the initial phase of the project, we considered augmenting the training data to create a larger training set. Data augmentation has shown to be a successful approach in situations where data is scarce, and may even reduce the need for pre-trained models [18]. Although not used in this project, it remains an interesting direction for further research.

7.5 *Conclusion*

This study set out to evaluate how effective a fine-tuned ByT5 is in the task of correcting OCR errors in historical Swedish newspapers. Using the 300M parameter version of ByT5, `byt5-small`, and the fine-tuning setup described in [8], three models were fine-tuned on different mixes of training data. All were trained on newspapers, but two models were further trained on data from other domains. All three successfully lowered the CER and WER on the evaluation set of OCR-processed Swedish newspapers published between 1818 and 2018. The most successful version in terms of precision and CER was trained on newspapers and blackletter data, and achieved a 37% reduction of CER. We further suggested a strategy for using the model on longer texts, which is applicable for any post-OCR correction model with limited processing capacity. Using the suggested approach, we achieved a 38% reduction of CER in longer texts.

Bibliography

[1] A. Bingham, "The digitization of newspaper archives: Opportunities and challenges for historians," *Twentieth Century British History*, vol. 21, no. 2, pp. 225–231, Apr. 2010. DOI: 10.1093/tcbh/hwq007.

[2] J. Burchardt, "Are searches in OCR-generated archives trustworthy?" *Jahrbuch für Wirtschaftsgeschichte / Economic History Yearbook*, vol. 64, pp. 31–54, Apr. 2023. DOI: 10.1515/jbwg-2023-0003.

[3] D. Dannélls, L. Björk, O. Dirdal, and T. Johansson, "A two-OCR engine method for digitized Swedish newspapers," in *CLARIN Annual Conference*, 2021.

[4] D. Van Strien, K. Beelen, M. C. Ardanuy, K. Hosseini, B. McGillivray, and G. Colavizza, "Assessing the impact of OCR quality on downstream NLP tasks," 2020.

[5] T. Nguyen, A. Jatowt, M. Coustaty, and A. Doucet, "Survey of post-OCR processing approaches," *ACM Computing Surveys*, vol. 54, pp. 1–37, Jul. 2021. DOI: 10.1145/3453476.

[6] A. Lundberg and M. Torstensson, "Deep learning for post-OCR error correction on Swedish texts," M.S. thesis, Chalmers University of Technology, Gothenburg, 2021.

[7] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010, ISBN: 9781510860964.

[8] L. Xue, A. Barua, N. Constant, *et al.*, *ByT5: Towards a token-free future with pre-trained byte-to-byte models*, 2021. DOI: 10.48550/ARXIV.2105.13626.

[9] M. Hammarstedt, A. Schumacher, L. Borin, and M. Forsberg, "Sparv 5 user manual," Göteborg, Tech. Rep., 2022.

[10] C. Tensmeyer and T. Martinez, "Historical document image binarization: A review," *SN Comput. Sci.*, vol. 1, no. 3, May 2020. DOI: 10.1007/s42979-020-00176-1.

[11] *Göteborgs handels- och sjöfartstidning*, Nov. 1865. [Online]. Available: https://tidningar.kb.se/3678898/1865-11-01/edition/144022/part/1/page/3/.

[12] H. Balk and L. Ploeger, "IMPACT: Working together to address the challenges involving mass digitization of historical printed text," *OCLC Systems & Services*, vol. 25, pp. 233–248, Oct. 2009. DOI: 10.1108/10650750911001824.

[13] *Fraktur*, in *Nordisk Familjebok*, T. Westrin, Ed., 2nd ed., Stockholm: Nordisk familjeboks förlags aktiebolag, 1908. [Online]. Available: http://runeberg.org/nfbh/0572.html.

[14] M. Piotrowski, *Natural Language Processing for Historical Texts*. Morgan & Claypool Publishers, 2012, ch. Aquiring historical texts, ISBN: 1608459462.

[15] S. Persson, "OCR post-processing of historical Swedish text using machine learning techniques," M.S. thesis, Chalmers University of Technology, Gothenburg, 2019.

[16] C. Rigaud, A. Doucet, M. Coustaty, and J.-P. Moreux, "ICDAR 2019 competition on post-OCR text correction," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 1588–1593. DOI: 10.1109/ICDAR.2019.00255.

[17] Q. Duong, M. Hämäläinen, and S. Hengchen, "An unsupervised method for OCR post-correction and spelling normalisation for Finnish," in *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, S. Dobnik and L. Øvrelid, Eds., Reykjavik, Iceland (Online): Linköping University Electronic Press, Sweden, May 2021, pp. 240–248.

[18] A. Jasonarson, S. Steingrímsson, E. F. Sigurðsson, Á. D. Magnússon, and F. Á. Ingimundarson, "Generating errors: OCR post-processing for Icelandic," in *The 24rd Nordic Conference on Computational Linguistics*, 2023.

[19] T. T. H. Nguyen, A. Jatowt, N.-V. Nguyen, M. Coustaty, and A. Doucet, "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*, ser. JCDL '20, Virtual Event, China: Association for Computing Machinery, 2020, pp. 333–336, ISBN: 9781450375856. DOI: 10.1145/3383583.3398605.

[20] E. Soper, S. Fujimoto, and Y.-Y. Yu, "BART for post-correction of OCR newspaper text," in *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, W. Xu, A. Ritter, T. Baldwin, and A. Rahimi, Eds., Online: Association for Computational

Linguistics, Nov. 2021, pp. 284–290. DOI: 10.18653/v1/2021.wnut-1.31.

[21] G. Chiron, A. Doucet, M. Coustaty, and J.-P. Moreux, "ICDAR 2017 competition on post-OCR text correction," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, 2017, pp. 1423–1428. DOI: 10.1109/ICDAR.2017.232.

[22] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR*, May 2015.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[24] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 1, Jan. 2020, ISSN: 1532-4435.

[25] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

[26] L. Xue, N. Constant, A. Roberts, *et al.*, *mT5: A massively multilingual pre-trained text-to-text transformer*, 2021. arXiv: 2010.11934 [cs.CL].

[27] L. Stankevičius, M. Lukoševičius, J. Kapočiūtė-Dzikienė, M. Briedienė, and T. Krilavičius, "Correcting diacritics and typos with a ByT5 transformer model," *Applied Sciences*, vol. 12, no. 5, 2022, ISSN: 2076-3417. DOI: 10.3390/app12052636.

[28] A. Maheshwari, N. Singh, A. Krishna, and G. Ramakrishnan, "A benchmark and dataset for post-OCR text correction in Sanskrit," in *Findings of the Association for Computational Linguistics: EMNLP 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds., Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 6258–6265. DOI: 10.18653/v1/2022.findings-emnlp.466. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.466.

[29] *Språkhistoria*, in *Nationalencyklopedin*. [Online]. Available: http://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/svenska/spr%C3%A5khistoria.

[30] G. Pettersson, *Svenska språket under sjuhundra år*. Lund: Studentlitteratur, 2005, ISBN: 9789144039114.

[31] L. Borin, G. Bouma, and D. Dannélls, "A free cloud service for OCR / En fri molntjänst för OCR," Göteborg, Tech. Rep., 2016.

[32] E. W. Myers, "An O(ND) difference algorithm and its variations," *Algorithmica*, vol. 1, pp. 251–266, 1986.

[33] T. Wolf, L. Debut, V. Sanh, *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[34] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Oct. 2018, pp. 4596–4604. [Online]. Available: https://proceedings.mlr.press/v80/shazeer18a.html.

[35] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *International Conference on Learning Representations*, 2020.

A. *Appendix: Examples*

OCR	En Gosse fur plats nu genast ! inetallyrkc, JU 83 Drottninggatan.
GT	En Gosse får plats nu genast i metallyrke, No 83 Drottninggatan.
Model 1	En Gosse får plats nu genast i metallyrke, No 33 Drottninggatan.
Model 2	En Gosse får plats nu genast i metallyrke, No 83 Drottninggatan.
Model 3	En Gosse för plats nu genast i Metallyrke, No 33 Drottninggatan.

Figure A.1: A sample from EVAL-SHORT with corrections suggested by the three model instances. Note that Models 1 and 3 change 8 to 3. Errors and corrections are highlighted in bold.

OCR	Afgäiigstiden bestämmes wid fartyget» hlt-komst.
GT	Afgångstiden bestämmes wid fartygets hit-komst.
Model 1	Afgångstiden bestämmes wid fartygets hit-komst.
Model 2	Afgångstiden bestämmes wid fartygets här-komst.
Model 3	Afgångstiden bestämmes wid fartygets hit-komst.

Figure A.2: A sample from EVAL-SHORT with corrections suggested by the three model instances. Errors and corrections are highlighted in bold.

OCR	SKrifportföljer, SKrliunderlägg, Necessgarier. ländsticks-etuior, Serviett-band.
GT	Skrifportföljer, Skrifunderlägg, Necessarier, Tändsticks-etuier, Serviett-band.
Model 1	Skrifportföljer, Skrifportföljer, Skrifportföljer, Skrifportföljer, Skrifportföljer, Serviett-band.
Model 2	Skrifportföljer, Skrifunderlägg, Necessgarier, ländsticks-etuior, Servlett-band.
Model 3	Skrifportföljer, Skrifportföljer, Skrifportföljer, Skrifportföljer, Skrifportföljer, Serviett-band.

Figure A.3: A sample from EVAL-SHORT with corrections suggested by the three model instances. In this sample, Models 1 and 3 display repetitive behaviour. Errors and corrections are highlighted in bold.

OCR/GT	Ett antal annonsmarknader publiceras på Internet, http://www.svd.se. Annonsören förutsätts medge
Model 1	Ett antal annonsmarknader publiceras på Internet, http://www.svd.se. Annonsören förutsätts medge
Model 2	Ett antal annonsmarknader publiceras på Internet, http://wwwwwwwwwwwwww-wwwwwwwwwwwwwwwwwwwwwwwwww-wwwwwwwwwwwwwwwwwwwwwwwww
Model 3	Ett antal annonsmarknader publiceras på Internet, http://www.svd.se. Annonsören förutsätts medge

Figure A.4: An error-free sample from EVAL-SHORT with corrections suggested by the three model instances. Models 1 and 3 correctly reproduce the input, while Model 2 got stuck in a loop. Errors are highlighted in bold.

OCR
Cn tifiig och roff Yngling, förfebb meb
fäfra od) goba betpg cm fitt uppforan:
be, fan erbåda fonbition i 3ernf„(‚mlim'l'ft!
fimarcht närmare unberrättelfe crfmdCb
uti ≥r Srobbfrnmhanblaren C. SI. UBcberé
95ob mib lida 9iögatan.

Output
En tillig och rost Yngling, försedd med
säkra och goda betyg om sitt uppföran-
de, kan erbåda kondition i Jernf. Malmölks
simarcht närmare underrättelse erfwärd
uti _ Stockholmhandlaren C. M. Weders
Bod wid lida Sjögatan.

GT
En liflig och rask Yngling, försedd med
säkra och goda betyg om sitt uppföran-
de, kan erhålla kondition i Jernkramhandel;
hwarom närmare underrättelse erhålles
uti Hr Kryddkramhandlaren E. A. Webers
Bod wid lilla Nygatan.

Figure A.5: A very noisy sample from EVAL-LONG with correction by Model 3.