Enhancing Historical Dutch OCR Accuracy with Post-Correction & Synthetic Data

Thomas Eckhardt

TU Delft

Enhancing Historical Dutch OCR Accuracy with Post-Correction & Synthetic Data

by

Thomas Eckhardt

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday June 26, 2023 at 10:00 AM.

Student number:	4473302	
Project duration:	November 14, 2022 – June 26, 2023	
Thesis committee:	Dr. C. C. S. Liem,	TU Delft, supervisor
	Dr. C Lofi,	TU Delft

TU Delft

Abstract

This paper presents a novel approach to synthetic data generation for OCR post-correction, utilizing specific background and font variations tailored to specific timeperiods. The goal is to use synthetic data to enhance text accuracy in digitized historical documents. The proposed three-step process involves generating synthetic images that emulate the characteristics of historical documents from different years, incorporating year-specific backgrounds and fonts. Using these images, a dataset can be created. Multiple T5 sequence-to-sequence transformers are then fine-tuned on the generated dataset. The trained models demonstrate the capabilities of improving the OCR, and aligning them with the ground-truth text. The effectiveness of the approach is evaluated through various performance metrics, highlighting the benefits of using year-specific synthetic data for training. This work contributes to the field of OCR post-correction by providing a powerful framework for improving the accuracy of OCR systems in historical OCR text tasks.

Preface

This thesis represents the culmination of a journey that began with my enrollment at TU Delft in 2015. It signifies the final milestone in my pursuit of a Master's Degree in Computer Science, specializing in the field of Data Science & Technology.

First and foremost, I would like to express my sincere gratitude to my thesis supervisor, Cynthia Liem, from the Multimedia Computing Group. Her unwavering support, expert guidance, and invaluable insights have been pivotal in shaping this research. I am deeply appreciative of her continuous assistance and mentorship throughout this journey.

I would also like to express my appreciation to the individuals at the Koninklijke Bibliotheek, particularly Mirjam Cuper, for generously providing me with access to their resources and datasets. Their contribution has played a vital role in the successful execution of this research, and I am truly grateful for their support. Furthermore, I would like to thank Christoph Lofi for graciously accepting the responsibility of serving as a member of my thesis committee and for devoting his time and expertise to review my work.

In addition, I would like to acknowledge the unwavering support and encouragement of my family and friends throughout the past eight years. I am particularly grateful to my father, who selflessly allowed me to utilize his computer for training the numerous machine learning models required for this research. All training time combined came to roughly 380 hours, resulting in an energy usage over that time of around 140 KwH.

Thomas Eckhardt
Delft, June, 2023

Contents

List of Figures

List of Tables

Nomenclature

CER	Character Error Rate
$DBNL$	Digitale Bibliotheek voor de Nederlandse Letteren (Digital Library for Dutch Literature)
$ICDAR$	International Conference on Document Analysis and Recognition
OCR	Optical Character Recognition
$T5$	Text-to-Text Transfer Transformer
WER	Word Error Rate

1 Introduction

Writing has been an integral part of human communication for thousands of years. The earliest evidence of writing dates back to the Upper Palaeolithic era in Europe, around 42,000 BP [1]. While the symbols used at that time were likely used for practical purposes such as counting animals or marking the passage of time, over time, writing became a way to record information and pass on ideas to future generations. Writing is essential for research as it enables us to build upon the work of others, advancing our understanding of the world and improving upon existing knowledge. In this way, writing and research are vital tools for the evolution of society and the betterment of humankind.

The invention of the printing press in 1436 by Johannes Gutenberg [27], was a significant milestone in the history of writing. The ability to quickly create copies of books and documents allowed for the more rapid spread of knowledge than was possible before. This invention led to the first book printed at scale called the Gutenberg bible. Or in other words the 42-line bible, related to the amount of lines per page [23]. The printing press also helped to standardize the way information was presented. Prior to the printing press, the spelling and grammar of written languages varied widely from region to region. With the advent of movable type and standardized printing techniques, however, it became possible to establish more consistent rules for spelling and grammar, which helped to make written communication more clear and accessible.

1.1. Main problem of OCR

When it comes to archiving large amounts of data, there can be significant challenges, particularly when the aim is to preserve history in its most pristine form. One of the main challenges is the transcription of text from printed documents into digital format. While scanning printed articles manually is one task, transcribing the text from an image is a different matter entirely. As a solution, software has been developed that enables the easy extraction of text from images. This technology is known as Optical Character Recognition (OCR), which allows the input image to be quickly converted into editable text.

Despite the advancements in OCR technology, errors can still occur in the transcription process, particularly when the scanned images are of low quality. To mitigate this issue, post-correction techniques are employed to clean the OCR output and rectify any inaccuracies. While human intervention has been utilized to aid in this process, it can be time-consuming and expensive, particularly when dealing with large amounts of data. In addition, human correctors require a deep understanding of the text domain and must be trained to use the correction tools effectively. Moreover, humans are prone to making mistakes when identifying similar characters that may cause confusion during the correction process [11].

In this thesis, the objective is to establish a framework that allows for the synthetic generation of OCR data utilizing ground-truth data. This generated data can then be utilized to train a sequence-to-sequence correction model, which can produce corrected texts based on the original OCR data. This framework is not limited to historical data and can be conveniently adapted to modern data as well. The proposed framework has the potential to streamline the transcription process significantly and provide a more accurate and efficient way of preserving historical documents.

1.2. Research questions

This section outlines the research questions that have guided the study and motivated the investigation into the effectiveness of various approaches for improving OCR accuracy and quality.

RQ1. How effective can synthetic data be generated, as closely resembling images from a given timeperiod?

RQ2. How effective are machine learning-based OCR post-correction methods trained on synthetic data in improving the accuracy and quality of OCR output?

RQ3. How much of an effect does a model pre-trained on Dutch data have on the performance of a fine-tuned model?

RQ4. How does the performance of a transformer model trained on one dataset compare when evaluated on a different dataset?

The first research question (RQ1) focuses on exploring the effectiveness of generating synthetic data that closely resembling images from a given timeperiod. By employing such synthetic images, datasets can be created that closely resemble the OCR output when compared to the ground-truth data. The aim is to assess the efficacy of this approach and its potential in training models specifically designed for post-processing OCR output. The models will be evaluated based on their capacity to enhance the accuracy and quality of OCR results. Furthermore, the impact of synthetic data on the performance of post-correction methods will be investigated to determine their effectiveness in refining OCR output.

Moving on to RQ2, it centers around evaluating machine learning-based OCR post-correction methods trained on synthetic data. Building upon the synthetic data generated in response to RQ1, the goal is to train models tailored for post-processing OCR output. These models will be evaluated to measure their effectiveness in improving the accuracy and quality of OCR results. The impact of synthetic data on the performance of these post-correction methods will be analyzed, thereby determining their value in refining OCR output.

RQ3 explores the influence of pre-training models on Dutch data in the context of fine-tuning for OCR post-correction. The extent to which incorporating pre-trained models specifically trained on Dutch data affects the performance of fine-tuned models will be investigated. By comparing the performance of models with and without Dutch pre-training, insights into the impact of transfer learning and the additional benefits of utilizing domain-specific pre-training in OCR post-correction tasks will be gained.

Finally, RQ4 focuses on evaluating the performance of a transformer model trained on one dataset when evaluated on a different dataset. This investigation aims to assess the generalization capabilities of the model and its ability to adapt to different OCR datasets. By comparing the model's performance across multiple datasets, valuable insights into its robustness and versatility in handling diverse OCR scenarios will be obtained.

1.3. Koninklijke Bibliotheek and Delpher

The Koninklijke Bibliotheek (KB) is the National Library of the Netherlands. The KB plays a crucial role in digitizing and preserving the country's cultural and historical publications. In order to provide the public access the vast quantities of historical data, they developed an application called Delpher, which is a digital platform for accessing and exploring mainly dutch historical newspapers, books, and other printed materials. It offers a vast collection of digitized resources, making it a valuable resource for researchers, historians, and the general public interested in exploring of the dutch cultural heritage. The full list of the amount of records is as follows:

- 2,094,960 Newspapers[1]
- 192,692 Books[2]

[1] http://jsru.kb.nl/sru?operation=searchRetrieve&version=1.1&recordSchema=dcx&x-collection=DDD_krantnr&query=*&maximumRecords=1&startRecord=1

[2] http://jsru.kb.nl/sru?operation=searchRetrieve&version=1.1&recordSchema=dcx&x-collection=BOEKEN_boek&query=*&maximumRecords=1&startRecord=1

- 458,576 Magazines [3]
- 1,474,359 Radio Bulletins[4]

The system incorporates a user-friendly web interface that grants users unrestricted access to an extensive pool of data. With a few simple steps, users can input keywords of interest and effortlessly retrieve relevant results. To enhance the search experience further, the web interface offers additional customization options. Users have the freedom to narrow down their searches such as selecting specific time periods and types of articles. This feature ensures, specific timeperiod can be investigated.

For instance, let's consider an illustrative example depicted in Figure 1.1. Imagine a user wants to explore the history of cycle racing in the Netherlands. They input the keyword "wielrennen", the Dutch term for cycle racing, into the search bar. By utilizing the interface's filtering capabilities, the user can refine their results to exclusively display articles from the 19th century. This filtering option allows them to delve specifically into historical perspectives and gain insights into the development and significance of cycle racing during that period. The search functionality in this application relies on the OCR text output obtained from the scanned images. By improving the accuracy of OCR by post-correction, the performance of keyword-based searches can be enhanced. This highlights the significance of enhancing OCR quality, as it directly translates into improved search results when seeking specific keywords or terms.

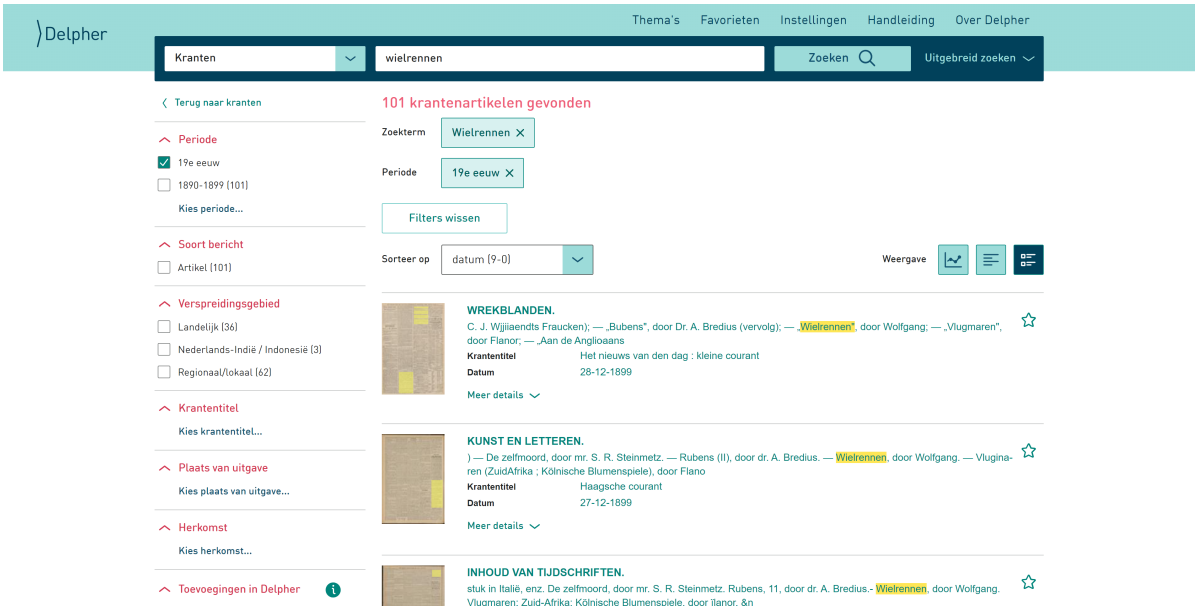


Figure 1.1: Search results for the query "wielrennen" in Delpher for the 19th century

1.4. Thesis outline

Chapter 2 serves as an introduction, delving into modern OCR (Optical Character Recognition) techniques and discussing the implications of low quality OCR. Additionally, it covers the functioning of transformers and their potential for fine-tuning. Moving forward, Chapter 3 presents a review of related work carried out in the field of post-correction, focusing on previous research related to synthetic data generation. Chapter 4 offers an overview of the pipeline that has been created for post-correction, and analyzes the datasets used throughout the thesis, providing valuable insights. Chapter 5 highlights the efforts made to synthetically generate datasets for post-correction using year dependant background and fonts. In Chapter 6, the focus shifts to the fine-tuning of a transformer model for the task of post-correction, showcasing the methods and techniques employed. Chapter 7 presents the experimental

[3] http://jsru.kb.nl/sru?operation=searchRetrieve&version=1.1&recordSchema=dcx&x-collection=DTS_document&query=*&maximumRecords=1&startRecord=1

[4] http://jsru.kb.nl/sru?operation=searchRetrieve&version=1.1&recordSchema=dcx&x-collection=ANP&query=*&maximumRecords=1&startRecord=1

results obtained during the course of this research, shedding light on the performance and effectiveness of various machine learning models used in the study, in combination with training on the synthetic data. Lastly, Chapter 8 concludes the thesis, summarizing the key findings and drawing conclusions. Furthermore, it offers valuable insights and suggestions for potential directions for future research in this domain.

Additionally, the report includes several sections accompanied by corresponding appendices. Appendix A provides additional information on some of the provided datasets. Appendix B presents the hyperparameters used during training. Appendix C contains additional data related to the Delpher Search API. Appendix D describes alternative approaches that have been explored to increase OCR accuracy. It should be noted that abbreviations will be written in full the first time they are mentioned in the report, and thereafter, the nomenclature can be referred to for further clarification. Lastly, Appendix E provides the link to my Github repository, containing all the code for this thesis.

2

Background

In this thesis, a comprehensive framework is introduced for post-correction of optical character recognition (OCR) using synthetic data. Before delving into the framework's details, it is crucial to establish a solid understanding of the concepts utilized throughout this thesis. This chapter serves the purpose of fulfilling that requirement by providing essential background knowledge. The key topics covered in this chapter are OCR and transformers, fundamental concepts that form the bedrock of this research.

2.1. OCR

Let's start by explaining the core concept of this thesis, and giving it some knowledge of it's origins. OCR is a technology that has become increasingly important in many industries. It refers to the process of converting an image of printed or handwritten text into a format that can be read by computers. OCR software analyzes the image, identifies the text patterns, and then converts them into a digital format that can be easily edited and searched.

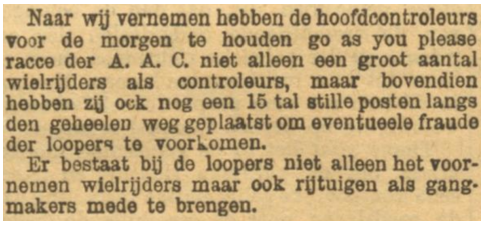
The earliest OCR systems were mechanical rather than computer guided. They were mechanical devices that used photoelectric cells to detect characters on a page. The first of this type was called GISMO and was developed by M. Sheppard [20]. It could read musical notations as well as words character by character. These mechanical type systems were slow and cumbersome, and their accuracy was limited. In the 1950s, computer-based OCR systems were developed that used digital image processing techniques to recognize characters. These systems were faster and more accurate than their mechanical counterparts, and they formed the basis of modern OCR technology.

The first computer-based OCR system was developed in the early 1960s by IBM. It was designed to read printed text and could recognize a limited set of characters. Over the next few decades, OCR technology continued to improve, and by the 1980s, OCR systems had become more accurate and could recognize a wider range of characters, including handwritten text. In the early 1990s, OCR technology underwent a revolution with the advent of neural network-based OCR systems. These systems were able to learn from examples and could recognize characters with a high degree of accuracy. This led to the development of OCR systems that could recognize a wide range of fonts and handwriting styles.

OCR has become a critical tool for archiving historical documents. By converting physical copies of books, newspapers, and other printed materials into digital formats, historians, scholars, and researchers can access information that might otherwise have been lost or inaccessible. This is particularly valuable for preserving and sharing cultural heritage and historical knowledge. But this technology is not only limited to historical documents, but can be used for modern purposes such as with medical data. The hard part for this type of data is that is does not conform to a general structure, but is a collection of patient documents [22].

Numerous modern OCR tools are available today, including ABBYY Finereader[1], Adobe Acrobat Pro DC[2], and Tesseract [3], to name a few. However, since the first two operate on a subscription basis,

[1] https://pdf.abbyy.com/
[2] https://www.adobe.com/acrobat/acrobat-pro.html
[3] https://github.com/tesseract-ocr/tesseract



Naar wij vernemen hebben de hoofdcontroleurs voor de morgen te houden go as you please racce der A. A. C. niet alleen een groot aantal wielrijders als controleurs, maar bovendien hebben zij ock nog een 15 tal stille posten langs den geheelen weg geplaatst om eventueele fraude der loopers te voorkomen.
Er bestaat bij de loopers niet alleen het voornemen wielrijders maar ook rijtuigen als gangmakers mede te brengen.

(a) De Amsterdammer - 25-10-1896

Naar **wy** vernemen hebben de hoofdcontroleurs voor de morgen te houden go as you please racce der A. A. C. niet alleen een groot aantal **wielrydors** als controleurs, maar bovendien hebben **zy ock** nog een 15 tal stille posten langs den **geheeleu** weg geplaatst om eventueele fraude der loopers te voorkomen. Er bestaat **by** de loopers niet alleen het voornemen **wlelryders** maar ook **rytuigen** als gangmakers mede te brengen.

(b) Text retrieved from the figure on the left using OCR

Figure 2.1: The scanned text along with its OCR equivalent

they are not ideal for use in this thesis. On the other hand, Tesseract is an open-source OCR engine initially developed by Hewlett-Packard but now maintained by Google. The program is predominantly written in C++, making it incredibly fast to use. Additionally, Tesseract is available as a python-package, making it simple to implement.

2.1.1. Examples of bad OCR

Despite significant advancements in OCR technology, it remains far from perfect. Extensive research has been conducted to address various factors that can compromise OCR accuracy, such as the thinness of paper, discoloration, imprecisions during printing, and more [31]. These issues can be especially problematic for older fonts or when scans of documents are of lower quality, reflecting the evolution of printing methods over time. It's important to note that the process of archiving historical data is an ongoing one that has been taking place for several decades. Earlier versions of scans used outdated OCR software, which has since been improved upon. Consequently, earlier scans may have higher error rates compared to modern OCR systems, which have benefited from significant advancements in technology over the years.

To illustrate the limitations of OCR technology, we can take a closer look at Figures 2.1a and 2.1b. Figure 2.1a displays a specific article from the *Amsterdammer* newspaper, dating back to 1986. In Figure 2.1b, we see the output provided by OCR software when this article was digitized. As we examine the text in Figure 2.1b, it becomes evident that it contains various imperfections, which can significantly impact its accuracy and reliability. For instance, the word *"wielrydirs"* is incorrectly transcribed instead of *"wielrijders"*, and *"rytuigen"* instead of *"rijtuigen"*. Such errors are not uncommon when using OCR technology, and they can significantly hinder the ability of researchers to extract accurate information from historical documents.

2.1.2. Implications of bad OCR

Errors in OCR data of historical archives can have several implications. First and foremost, it can make it difficult for researchers and historians to accurately interpret the content of the documents. Even a single error can change the meaning of a sentence or paragraph, which can have a significant impact on historical analysis and research. Previous research has shown, that in part-of-speech tagging, in which words in a text are categorized, the error rate of the tagger increases linearly with that of the OCR output [30] Furthermore, errors in OCR data can also impact the ability to perform searches and queries on digitized archives. If the OCR engine does not accurately recognize certain words or phrases, then they will not be included in search results. This can make it more difficult to locate relevant documents and information within large archives [15].

Finally, errors in OCR data can also affect the preservation of historical documents. If the OCR output is used as the primary means of accessing and interpreting the content of a document, then errors in the OCR data may result in inaccurate or incomplete preservation of the original document. Therefore, it's important to ensure that OCR data is as accurate as possible to preserve the integrity and authenticity of historical documents. In Section 6.2.1 it is explained how this quality can be quantified.

2.2. Transformers

A transformer is a neural network that processes sequential data, such as text or speech. It was introduced in a seminar paper by Vaswani et al. [40]. This paper proposed a new type of neural network architecture called the transformer, which uses self-attention mechanisms to process sequences of

input data. Before the input data can be processed it should first be split into smaller units called tokens, which can be individual words, subwords, or even characters. This process is called tokenization, and transformers relies on it to convert the input data into a format that can be processed by the neural network. Tokenization in transformers typically involves two steps: first, the input text is split into individual tokens, and second, each token is mapped to a unique integer ID. The tokenization process can vary depending on the specific transformer model being used and the task at hand. These tokens allow the transformer to fully capture complex patterns and relationships between tokens, which are an essential feature for many NLP tasks.

Using these tokens, the transformer can utilise a mechanism called self-attention, which is a mechanism that enables the network to weigh the importance of each input token based on its relevance to other tokens in the sequence. This allows the network to focus on the most relevant information and ignore irrelevant or noisy data. The transformer architecture is composed of an encoder and a decoder. The encoder processes the input sequence using multiple layers of self-attention and feed-forward neural networks to produce a sequence of hidden representations. The decoder then uses the same self-attention and feed-forward layers to generate an output sequence from the hidden representations.

The transformer architecture has several advantages over previous neural network architectures, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs). One major advantage is that the transformer can process sequential data in parallel, which makes it much faster than RNNs [21]. Additionally, the self-attention mechanism allows the network to capture long-range dependencies, which is difficult for RNNs and CNNs.

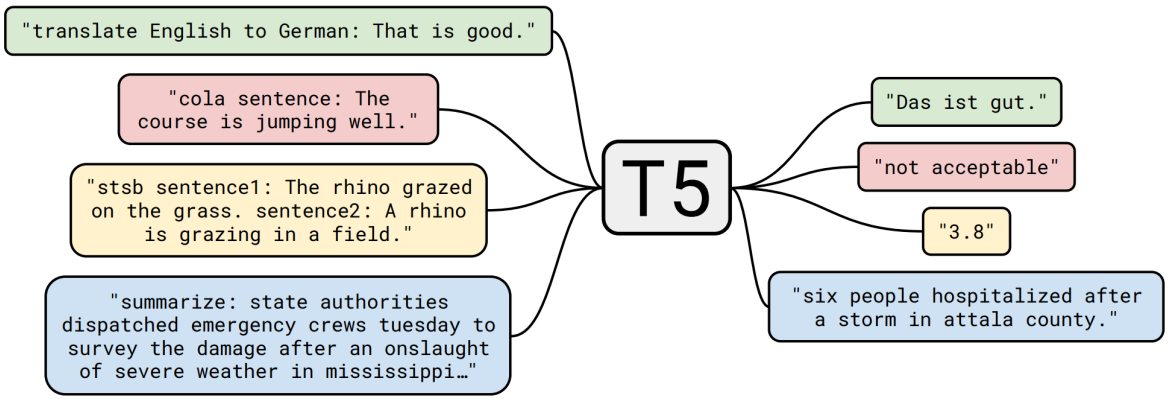


Figure 2.2: A diagram containing various tasks included in the original T5 model. Retrieved from [32]

2.2.1. Transfer Learning

Transfer learning is a powerful technique used in machine learning that involves leveraging knowledge acquired in one task to improve performance in a related but different task. In the context of transformers, transfer learning refers to the practice of taking a pre-trained transformer model and fine-tuning it for a specific downstream task. The pre-training phase of a transformer model involves training the model on a large corpus of text data using a self-supervised learning approach. During this phase, the model learns to understand the structure of language and to generate meaningful representations of words and sentences.

In the context of OCR post-correction, transfer learning can be used to fine-tune a pre-trained transformer model for the specific task of correcting errors in OCR-generated text. The pre-trained model can be trained on a large corpus of text data, such as the Common Crawl dataset, using a self-supervised learning approach. This pre-trained model can be further fine-tuned on a smaller dataset of OCR-generated text, with the objective of improving the accuracy of the text correction process.

3

Related Work

While the previous chapter focused on providing a preliminary grasp of the fundamental concepts, the purpose of this chapter is to delve deeper into specific concepts relevant to this thesis. Additionally, it explores previous research conducted in the respective fields, aiming to support the chosen direction for this research. Within this chapter, the concepts of Post-Correction will be thoroughly explained, encompassing both manual and (semi) automated approaches. Furthermore, specific examples of the latter will be discussed. Additionally, the contexts in which post-correction is typically employed will be examined, such as the historical context. The final sections of this chapter will focus on the utilization of synthetic data generation for the purpose of post-correction.

3.1. Post-Correction

OCR post-correction is a rapidly growing field of research that aims to improve the accuracy and efficiency of OCR systems. As mentioned earlier, OCR systems are not perfect and often produce errors due to various factors such as the quality of the input document, the complexity of the font, and the variability of the language. OCR post-correction aims to correct these errors and improve the accuracy of OCR output.

In the field of post-correction, there are different approaches available to achieve the desired objective. In this particular study, we classify these approaches into two main types: the manual approach and the (semi)automated approach, as defined by Nguyen et al. [30]. The next subsection will explain these concepts into more details, and give concrete examples.

3.1.1. Manual Approach

The manual approach involves human intervention for carrying out the post-correction process. However, this approach has a significant drawback as it requires considerable labor and often incurs high costs. One method to accomplish this is by utilizing crowdsourcing platforms like Amazon's Mechanical Turk [37]. While some projects attempt to involve volunteers in assisting with the post-correction of large datasets, it is important to note that this can still be a challenging task. An example illustrating this approach can be found in the work of Chrons, Sundell, and Bulevardi [5]. They proposed a strategy to crowdsource the error correction process through gamification. They created a game called *digitalkoot*, derived from the Finnish word *talkoot*, which refers to a group of people engaging in unpaid work. Through this game, they were able to enlist the efforts of volunteers, resulting in the completion of over 2.5 million tasks after investing a total of 2740 hours of work.

3.1.2. (Semi) Automated Approach

For post-correction the (semi) automated approach often involves a specific type of transformer, namely the sequence-to-sequence (often referred to as Seq2Seq) transformer. This type of transformer has been developed to transform one sequence of data into another sequence. The basic idea behind sequence-to-sequence transformers is to use two separate transformers: an encoder and a decoder. The encoder processes the input sequence, typically a sentence or a paragraph, into a fixed-length representation called a context vector, which captures the important information contained in the input

sequence. The decoder then takes the context vector as input and generates an output sequence, which can be another sentence or a translation of the input sequence, depending on the task at hand. An example of this is would be a transformer used for translation, in which you give as input a piece of text in one language, and then it's translated into another piece of text in another language. For post-correction the goal is to train a model that as input text that contains errors, and returns the text in which these errors are removed.

Over the last decade, several variations of sequence-to-sequence transformers have been developed, including BART [8], GPT-3 [3], and T5 [32]. BART, short for "Bidirectional Auto-Regressive Transformers" is a transformer-based language model that was designed for pre-training on large corpora of text to improve the performance of various natural language processing (NLP) tasks, such as text classification, question answering, and language generation. GPT-3, on the other hand, is a more recent and powerful language model that uses a massive neural network with billions of parameters to generate human-like text.

T5, which stands for "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer," is another type of transformer-based language model that was specifically designed for solving a wide range of NLP tasks using a single model architecture. The T5 model is unique in that it can perform various NLP tasks, including translation, summarization, question answering, and text completion, all using the same architecture. In Figure 2.2, you can see the original diagram of the T5 model as created by the developers at Google. The figure depicts four different types of actions, including translation and summarization, which the T5 model can perform. The original T5 model was trained using a massive dataset called Colossal Clean Crawled Corpus, which was created by the developers themselves.

Overall, while BART and T5 are both powerful transformer-based sequence-to-sequence models, T5's ability to perform multiple NLP tasks using a single architecture makes it a promising choice for a wide range of NLP applications, including post-correction.

3.1.3. Usage in post-correction

Recently, many researchers have chosen to utilize machine learning based techniques for post-correction tasks, where a model is trained on a specific ground-truth dataset. In 2017, Saluja and colleagues [36] employed an LSTM approach for post-correction in Indic languages, which are a group of languages primarily spoken in South Asian countries such as Hindi, Bengali, and Punjabi. They found that an LSTM was most suitable for correcting Out of Vocabulary (OOV) words.

Over the past few years, other machine learning methods have been explored, such as BERT [29] and BART [24], which is similar to BERT but incorporates an autoregressive-decoder. BART is a sequence-to-sequence model, which means it can be trained to generate output sequences of varying lengths from input sequences. This makes it ideal for post-correction, as it enables the correction of errors. Variations of BERT include RoBERTa [22], an extended version of BERT that uses a larger and more diverse training corpus and a modified training methodology to achieve better natural language processing performance. RoBERTa has a larger model size and was trained for more steps than BERT.

Another approach involves using a T5 transformer [25, 31]. T5 is similar to BART in that it is a transformer-based architecture that consists of a stack of transformer encoder-decoder layers capable of processing both input and output sequences. However, T5 includes a pre-processing step that converts input and output tasks into a unified text-to-text format, which allows for training and fine-tuning on a wide range of tasks using a single architecture and training process. This makes T5 an ideal approach for post-correction in OCR, as it can be easily fine-tuned.

3.2. Post-Correction in the historical domain

In Section 2, it was discussed how Optical Character Recognition (OCR) can be a valuable tool for archiving historical documents, and post-correction can further enhance the quality of the OCR results. However, one of the primary challenges encountered is that modern OCR technology is often ill-suited for handling historical documents. To address this, Martínek, Lenc, and Král [26] proposed a technique that utilizes a small amount of annotated data to effectively segment different parts of the text.

Several researchers have also explored post-correction methods specifically tailored to historical documents. For instance, Drobac et al. [12] conducted experiments on historical Finnish data, achieving an accuracy of 95.21% using the best OCR and post-processing models. They employed the

Ocropy OCR software[1] and utilized the DIGI and NATLIB datasets, which are part of a larger corpus of historical newspapers and magazines created by the Library of Finland. Each item in these datasets consists of image files containing individual lines of printed text, as well as the corresponding plain text contents. The alignment problem was one of the challenges they addressed during their research.

The archival process of historical documents has been ongoing for several decades, and various OCR applications have been employed during this time, each producing outputs of varying quality. Post-correction applications are often trained on specific OCR software but are expected to be applied to documents acquired using different software. Dannélls and Persson [7] proposed a post-correction method that utilizes three OCR systems: ABBYY Finereader, Tesseract, and Ocropus. They conducted their research on historical Swedish data from the Project Runberg OCR project, which covers the 17th and 19th centuries, as well as the Swedberg dataset, which covers the 18th century. It is worth noting that their post-correction was performed at the word level rather than the sentence level.

3.3. Post-Correction in other domains

Despite the focus of this thesis being in that of historical documents, OCR can play a role in many other fields. One in particular is the medical domain. Medical data often lacks a standardized structure and exists primarily in the form of unstructured, digitally generated, or scanned paper documents that are stored as part of a patient's medical records. To make this unstructured data accessible, Optical Character Recognition (OCR) technology is used to digitize it [22]. However, OCR accuracy can be a culprit when the quality is too low. This is especially true when dealing with scanned or handwritten documents, where text may be skewed, obscured, or illegible. Moreover, medical terminology used in the processed text is highly specialized and may not be part of general language lexicons, adding to the complexity of the OCR process.

The field of social media is another critical area where OCR technology has become increasingly important. With vast amounts of data being processed in this domain, human intervention is often not feasible due to the sheer volume of information being uploaded and shared on these platforms. Unfortunately, this large quantity of data also makes social media platforms vulnerable to malicious content, particularly when it comes to images. As noted in a recent study [17], many images uploaded to social media platforms may contain embedded text that contains malicious content, such as spam or phishing messages.

The problem is compounded by the fact that most social media systems rely heavily on automated processes to analyze and moderate content, with little human intervention. This means that malicious content can easily slip through the cracks and go undetected, posing a significant risk to users.

3.4. Post-Correction for Dutch data

Research has been done into the field of post-correction specifically for Dutch data. For instance, in the International Conference on Document Analysis and Recognition (ICDAR), which is a conference in which post-correction is addressed, besides other parts, a dutch dataset is included for training and evaluation [35].

One of the leading researchers into the field of post-correction for dutch literature, is Reynaert. In 2008 he proposed a system called "Text-Induced Corpus Clean-up" (TICCL) to reduce the level of OCR-induced typographical variation in large text collections [34]. The system focuses on high-frequency words and gathers all typographical variants that are within a predefined Levenshtein distance. Simple filtering techniques are used to retain true positives and discard false positives. The system was evaluated on a contemporary OCR-ed Dutch text corpus and a corpus of historical newspaper articles with lower OCR-quality and an older Dutch spelling. The results show that the system can effectively remove most OCR-induced typographical variation automatically up to a Levenshtein distance of 2. The paper also discusses the adaptation of the correction mechanism to OCR-error resolution.

3.5. Synthetic data generation

The generation of synthetic training data is not a novel concept and has been previously employed. Breuel et al. [2] utilized a tool called ocropus-linegen to artificially generate synthetic data images. Another instance can be found in the paper authored by Imam, Vassilakis, and Kolovos [18], where a

[1] https://github.com/ocropus/ocropy

synthetic data generator[2] was utilized to detect images containing malicious content. In this approach, individual sentences were placed on separate white images, and various OCR applications were employed to extract the text.

Expanding beyond the use of images to create data, Duong, Hämäläinen, and Hengchen [13] introduced a method to incorporate OCR-like errors into their dataset. They deliberately manipulated the dataset by deleting, adding, and replacing characters up to a predetermined ratio. The objective was to create a dataset that closely resembles real OCR errors. Similarly, Dong and Smith [9] uniformly introduced insertions, deletions, and substitutions to mimic authentic OCR errors. Furthermore, a sequence-to-sequence model was trained using the synthetic data, resulting in improved performance in terms of CER and WER scores on the validation dataset.

Determining the appropriate error ratios for generating OCR errors can be challenging, particularly when dealing with historical data. Historical articles, especially in the early 17th century, often employed gothic-style fonts, which can be more difficult to interpret, and older OCR software was used to read those texts. In contrast, newer articles are typically easier to read and interpreted by more modern OCR software, consequently increasing the quality of the output.

3.6. Evaluation dataset of ICDAR

ICDAR, the International Conference on Document Analysis and Recognition, is a renowned event that focuses on analyzing and recognizing documents. The contents of the conference is two-fold. Participants enter with models for either error detection or correction in OCR. One of the main areas of evaluation at the conference involves assessing models' capabilities in detecting and correcting errors in OCR (Optical Character Recognition) content [35]. In order to validate the results obtained by these models, different datasets were employed, including a Dutch subset of the IMPACT dataset. Their datasets are often used for evaluation purposes, aiding in comparing different machine learning models. However the evaluation datasets differ per individual language. This section aids to explain, why the choice was made to not use the ICDAR dataset for evaluation.

The validation files in the dataset comprised three lines: the *OCR_toInput* line, which represented the output generated by the OCR model, and the *OCR_aligned* and *GS_aligned* lines, which represented aligned versions of the OCR output and the corresponding gold standard text, respectively. Alignment was achieved by using @ symbols to match up the sentences in the two lines.

However, it is important to note that the dataset suffered from inconsistent data quality, as demonstrated by the extreme case of file *39.txt*. In this instance, the gold standard text consisted of only a few words, while the OCR output consisted of multiple sentences. Such inconsistencies pose additional challenges for post-correction models. Post-correction itself is already a demanding task, and it becomes even more difficult when dealing with inconsistent data. The presence of such inconsistencies highlights the importance of carefully considering the quality and characteristics of the datasets used for evaluation. In the case of post-correction models, having reliable and consistent data is crucial to ensure accurate assessments of the models' performance.

It is worth emphasizing that ICDAR places its primary focus on techniques pertaining to error detection and error correction, rather than the data generation process. Participants in the conference are provided with a designated training dataset and are expected to train their models using this dataset, instead of collecting or creating their own data. This standardized approach ensures a level playing field for evaluating and comparing different models. Attempting to compare a model trained on different data sources would introduce confounding variables and compromise the reliability of the results.

Furthermore, it is important to note that the IMPACT dataset, which serves as the primary dataset for the ICDAR conference, is limited in its temporal scope, encompassing data exclusively from the years 1777 to 1878, as indicated earlier in the section. Consequently, this dataset fails to encompass substantial portions of data that are included in the dataset utilized in our research work. Given this disparity in the data sources, taking into consideration the specific context and characteristics of our model, it was deemed inappropriate to employ the evaluation set of ICDAR for assessing the performance of our model.

By not using the evaluation set of ICDAR, we ensure that our model's performance is evaluated within the appropriate context and against datasets that encompass a more comprehensive representation of the target domain. This approach allows for a more accurate assessment of the model's

[2] https://github.com/Belval/TextRecognitionDataGenerator

[OCR_aligned] @XII VOORREDEN. badist in geen kwaden zin meende, dagt ik te mo gen hopen, dat alle verftandigen denzelven nu •,voorts wel eenvoudig zouden willen aanzien als de benaming van een zeker Leerftelfel, bij ons onder dien mam bekend•, en dat, zonder dien Naam, niet anders, dan telkens door eene zeer breede omfchrijving zou kunnen worden aangeduid. Meer bedoelde ik hier in niet; en nademaal het modig was, dat ik dien Naam in den tegenwoor- digen Brief wederom meenigemalen gebruikte, ver zoek ik elk, die goedvinden zal den zelven te le zen , dat hij dien in dezen, en in geenen ande ren zin gelieve te verf aan. Want de Naam La- badist is bij mij alzoo weinig een fcheldnaam, als die van Vitringist, Lampeaan of Honertiaan; wel ken men immers noemen kan, zonder iemand te beledigen? vfangezien zij niets anders aanduiden dan zekere denkwijzen, die allen, even als het Labadismus, tot het Cocceanismus behoor en, of daar van onderdeelen uitmaken. hierom heb ik ook niet dan met leetwezen gezien, dat de Heer Alethophilus van zig heeft kunnen verkrijgen, bl. 6. te fchrijven, dat de meefle leden onzer Kerke door mij voor Labadisten uicgekreten worden; en dan bl. 106 weer anders om; dat ik met labadie zand in de oogen geworpen heb. Hoe beide deze dingen uit dezelfde Pen hebben kunnen vloeijen, weet ik niet j maar
[GS_aligned] XII VOORREDEN. badist HIEROM maar

Figure 3.1: The contents of the file *39.txt* contained in the Dutch evaluation set of ICDAR

capabilities.

4

Overview and Data Preparation

The objective of this thesis is to leverage ground-truth data in order to train a model capable of automatically rectifying OCR errors. However, prior to utilizing the data to achieve this goal, it must undergo acquisition, analysis, and pre-processing. This chapter offers a detailed explanation of the steps involved in acquiring the data, conducting analysis, and performing necessary pre-processing procedures to adequately prepare the data for training the OCR error correction model. These steps play a critical role in ensuring the data is suitably prepared. Furthermore, in addition to the approaches discussed in this chapter, various other techniques were explored but deemed unsuitable or not effective enough. For a comprehensive overview of these techniques, please consult Appendix D.

This chapter will first provide an initial overview of the various parts of the framework, followed by how each of the datasets is pre-processed. Then an overview of the used datasets is presented, along with a more in depth look into what is contained in each one of them. Finally, all datasets are combined, and the total datasets is also examined. This section also contains a part about creating an evenly distributed dataset, which is a dataset containing equal amounts of information for each of the publication years.

In this chapter, an introductory overview of the different components of the framework is presented. Following that, we will delve into the pre-processing procedures applied to each of the datasets. Subsequently, an overview of the utilized datasets will be presented, accompanied by a detailed examination of the contents within each dataset. Finally, all the datasets are combined and the contents of the dataset as a whole is examined. This section will also address the creation of an evenly distributed dataset, which involves curating a dataset with equal amounts of information for each publication year. This last part is used in the next chapter to train the sequence-to-sequence models.

4.1. Overview of the system

In Figure 4.1, a visual representation of the pipeline of the system can be found. This explains the 3 steps that are performed in this thesis.

The initial component is the *data pre-processing stage*. Here, all the various datasets are loaded into the system along with accompanying information, such as the publication year, which plays an important role in the next steps. Furthermore, irrelevant data is removed, ensuring that the remaining dataset is in a suitable state for training purposes.

In the second step in the pipeline, we encounter the *synthetic data generation phase*. This stage leverages the data loaded in the previous step to create images that incorporate year-dependent backgrounds and fonts. With the help of OCR software, sentences displayed on the images can be accurately extracted and consolidated into a dataframe. This dataframe contains both the ground-truth and sentence after going through the OCR software, as well as the publication year.

Lastly, the final step revolves around training a *sequence-to-sequence model*. By utilizing the synthetically generated data from the second step, a sequence-to-sequence model can be trained for the goal of post-correction.

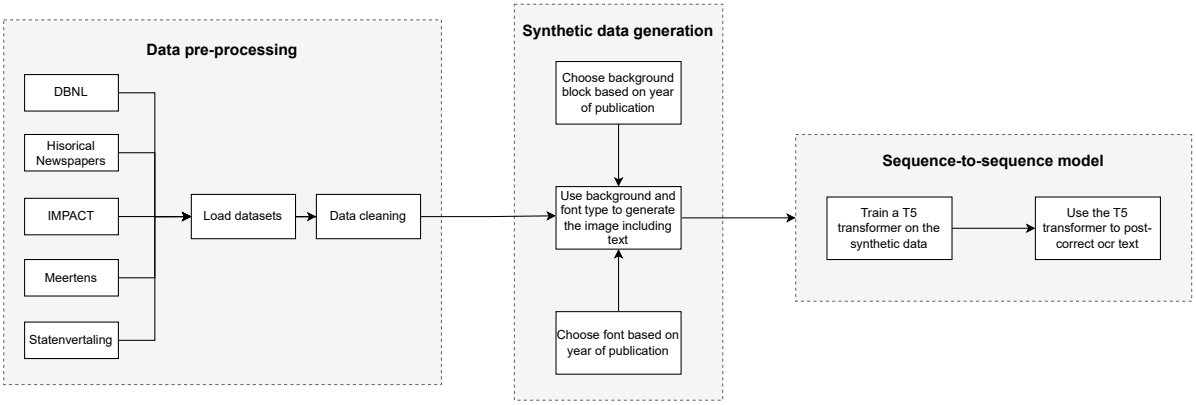


Figure 4.1: An overview of the full framework of the application, containing the 3 stages

4.2. Data Pre-Processing

Starting of with the first step, the *data pre-processing stage*. Before the actual datasets themselves are discussed, first the steps of data-cleaning will be discussed. The reason why this is explained first, is that in the next sections figures are displayed containing the amount of words after data-cleaning. Data cleaning involves identifying and rectifying any errors or inaccuracies, while pre-processing involves transforming the data into a suitable format for the model.

To begin with, the dataframes used in the system should consist of two columns: the ”target” column and the ”year” column. The ”target” column contains the ground-truth or target text that the model aims to generate. The ”year” column indicates the publication year of a specific article, which is helpful in the subsequent step of synthetic data creation. Data pre-processing specifically focuses on the ”target” column.

In the Python code, the initial step involved ensuring that both the ”year” and ”target” columns contain only string values. This step aims to maintain consistency across all rows in the dataframe. Next, regular expressions (regex) were employed to filter out any characters that do not meet the specified criteria. The criteria typically include allowing only alphabetic letters, numbers, and standard punctuation marks such as commas (,), periods (.), colons (:), and semicolons (;). Additionally, consecutive whitespaces were removed from the dataset. These pre-processing steps ensure that the data contains only relevant characters for the model while eliminating unwanted information.

In addition to cleaning and pre-processing, the function incorporates a filtering step based on sentence length. Initially, all sentences are split using semicolons (commonly used in 17th-century texts to separate sentences) and dots as delimiters. Subsequently, sentences that are deemed too short (less than 5 words) or too long (more than 50 words) are excluded from the dataset. This filtering ensures that the model is trained on high-quality data containing sufficient information for effective learning.

Furthermore, unwanted values are identified and removes rows where the target column contains NaN, None, or ’None’ values. These occurrences may arise due to errors or inconsistencies in the data. By eliminating such rows, the model is trained on more accurate and reliable data. Duplicate lines are also unwanted, and are therefore removed. These might occur when having overlapping datasets, especially with shorter sentences.

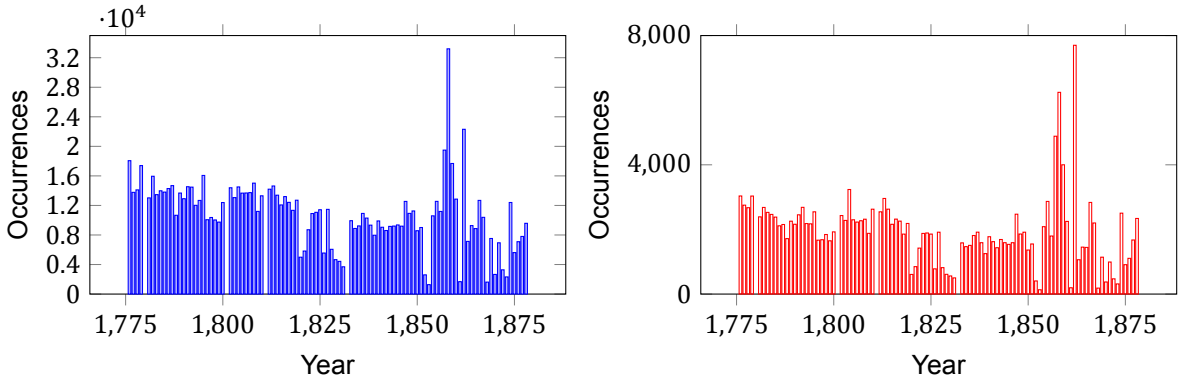
4.2.1. Overview of the datasets

The complete dataset comprises five distinct ground-truth datasets, which will be referred to as **DBNL**, **Historical Newspapers**, **IMPACT**, **Meertens** and **Statenvertaling**. Each of these datasets is described in detail in the following sections, along with a corresponding figure that shows their contents. It is worth noting that these figures are generated after the pre-processing step, as described in the previous section. Therefore, they represent the actual usable size of the dataset that is relevant for our analysis.

4.2.2. DBNL

DBNL is a dataset digitised by the 'Digitale Bibliotheek voor de Nederlandse Letteren' or 'Digital Library for Dutch Literature' in English. It contains a digitized version of 220 Dutch books, ranging from 1776 to 1878 written primarily in Dutch. It contains the original OCR version in a .txt format, and a manually corrected version in TEI-Lite[1] format. The full list of books can be found in Appendix A.1.

Figure 4.2 provides a visualization of this dataset and shows the number of words and sentences per year. The figures indicate that the dataset covers the period from 1776 to 1878. One interesting observation is that there is a significant peak in both the number of words and sentences in the year 1862. This could be due to a number of factors, such as an increase in the production of Dutch literature during that period or the publication of a particularly significant work. Further analysis would be needed to determine the cause of this peak. Another notable feature of the dataset is that there are several missing years, specifically 1780, 1801, and 1811. It is unclear why data for these years is missing, but it is possible that there was a lack of available literature during those periods or that the data has been completely removed during the data pre-processing procedure.



(a) Amount of words per year

(b) Amount of sentences per year

Figure 4.2: Two figures containing the amount of words and sentences per year for the *DBNL* dataset

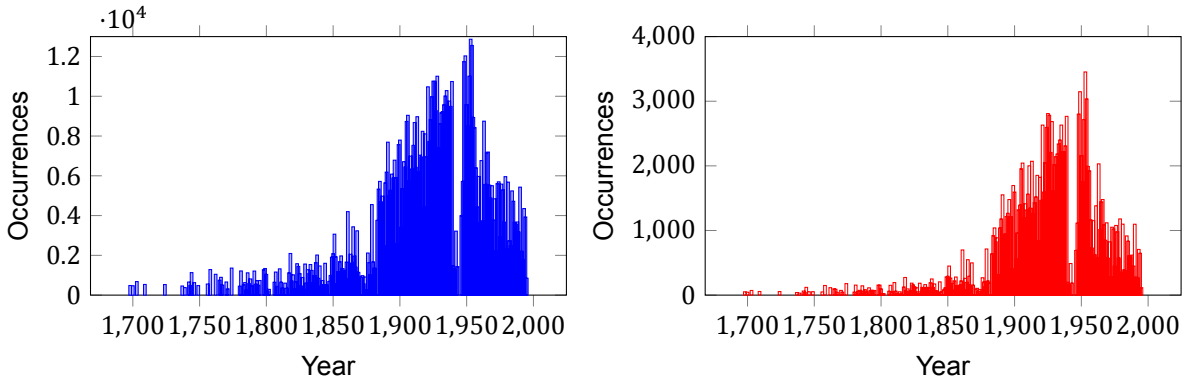
Historical Newspapers

Historical Newspapers is a dataset developed by Wilms, Nijssen, and Koster [41] for a research project. They manually correct 2000[2] pages of newspapers ranging from 1700 to 1995. The pages were correct to 99.95 percent accuracy for paper newspapers, and to 99.5 for newspapers digitised from microfilm. The data was rekeyed using the software Aletheia[3]. The set consists of 2000 images in JP2 format, accompanied by the original OCR files in ALTO format, and 2000 manually corrected OCR ground-truth files in ALTO format. The JP2 (JPEG 2000) image format is a highly compressed image format that uses wavelet technology to produce high-quality images with smaller file sizes than other image formats [16]. This style of image compression is used throughout the entire image database of the Koninklijke Bibliotheek. In addition to the image files and OCR files, there is an excel file included, which holds all the identifiers for each of the images, as well as other information such as the year of publication.

The provided figure, which can be seen in Figure 4.2, is a visual representation of this dataset, showing the number of words and sentences per year. The dataset spans a period of time from 1698 to 1995, covering a vast range of historical events, societal changes, and cultural shifts that occurred over nearly three centuries. The data indicates that there was a noticeable increase in the number of sentences and words in the dataset from around the year 1875 onward. This increase could be attributed to a number of factors, such as advances in printing technology, a greater demand for news and information, and an overall growth in the size and complexity of the media landscape during that

[1] https://tei-c.org/Vault/P4/
[2] https://lab.kb.nl/dataset/historical-newspapers-ocr-ground-truth
[3] https://www.primaresearch.org/tools/Aletheia

time period. However, there is also a noticeable drop in the number of sentences and words during the period of the second world war, from 1940 to 1945. This is to be expected, given the widespread disruptions and upheavals that occurred during that time, including censorship, shortages of paper and other materials, and the overall chaos of wartime conditions. One other observation that can be made from the figure is that there seem to be some gaps or missing years, particularly before the year 1750. This could be due to a variety of factors, such as the loss or destruction of historical records, the lack of available data in certain regions or areas, or the limitations of the data pre-processing procedures used to compile the dataset.



(a) Amount of words per year

(b) Amount of sentences per year

Figure 4.3: Two figures containing the amount of words and sentences per year for the *Historical Newspapers* dataset

IMPACT

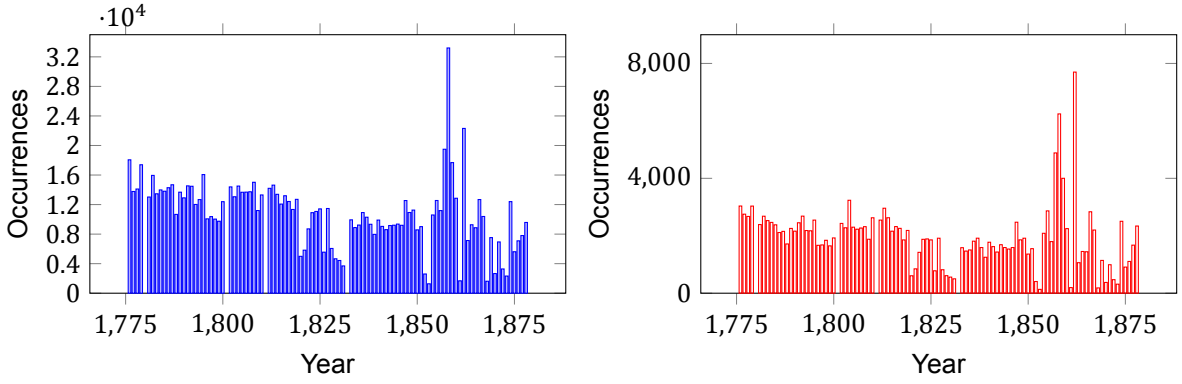
The dataset known as **IMPACT** [19] was acquired as part of the IMPACT project[4], a European initiative headed by the Koninklijke Bibliotheek (KB) aimed at promoting progress in OCR and language technologies. It comprises a selection of book pages, newspaper pages, parliamentary proceedings, and typewritten radio bulletins. Each category of text is accompanied by TIF image files and XML files containing the page content. The dataset also includes a spreadsheet that provides additional information, such as the publication year, enabling the linkage between the TIF and XML files. The text files are manually corrected to an accuracy of 99.95 percent. The full source for the dataset can be found in Appendix A.3

The dataset consists of historical data spanning from 1776 to 1878, and Figure 4.4 visually represents this data. In comparison to other datasets, the distribution of the data in *IMPACT* appears to be more uniformly spread out, with the only noticeable peak occurring in the year 1858. This observation could suggest that there were no major events or changes that occurred during this time period, or the dataset may have been intentionally configured to have a smoother distribution. Despite this, there are still a few gaps in the data, specifically in the years 1801 and 1811, which suggest that there might have been some missing data, which can either be related to the data itself or the data cleaning process.

Meertens

The Meertens dataset[5] contains a transcription of 6000 newspapers from the 17th century. The data has been transcribed via a crowdsourcing project, volunteers would manually type out the text, and then another correction step was performed. It was an effort that took 200 volunteers, 5 years to complete. The project was a cooperation between the Koninklijke Bibliotheek, Nationale Bibliotheek van Nederland and the Meertens Instituut. The dataset is focused on the data between 1618 and 1700, since that is the period in which Gothic and Roman fonts were used, which were difficult for OCR software to recognize.

[4] https://cordis.europa.eu/project/id/215064
[5] https://meertens.knaw.nl/2020/05/18/crowdsourcing-maakt-zeventiende-eeuwse-kranten-op-delpher-beter-door

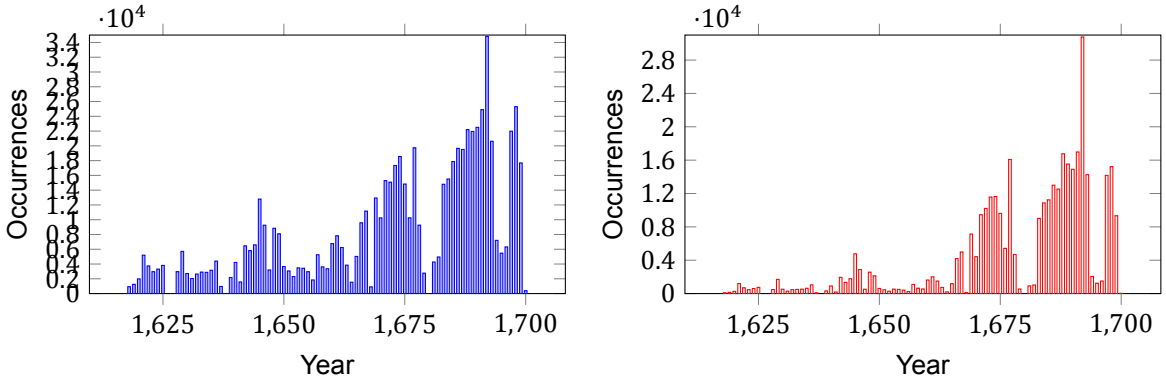


(a) Amount of words per year

(b) Amount of sentences per year

Figure 4.4: Two figures containing the amount of words and sentences per year for the *IMPACT* dataset

The Meertens dataset, as illustrated in Figure 4.5, encompasses data spanning from the 17th century, specifically from 1618 to 1700. The dataset shows an increasing trend in size as the years progress, with the year 1692 marking a peak. Unlike the previous datasets, there appear to be no missing years within the Meertens dataset.



(a) Amount of words per year

(b) Amount of sentences per year

Figure 4.5: Two figures containing the amount of words and sentences per year for the *Meertens* dataset

Statenvertaling

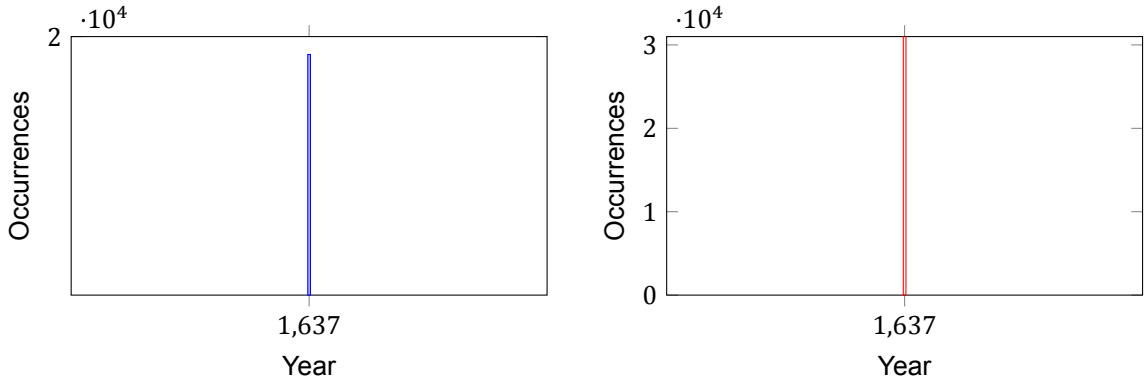
The *Synode van Dordrecht* was a gathering of the Dutch Reformed Church that took place in 1618-1619. During this Synod, it was decided that a new translation of the Bible from its original languages of Hebrew and Greek was needed [39]. The result of this decision was the *Statenvertaling*, a Dutch translation of the Bible that was completed in 1637. The Statenvertaling became an important cultural and literary work, helping to establish a common literary language for the Dutch Republic and shaping the Dutch language as we know it today [28].

Over time, numerous other translations of the Bible have been made, including modern and easy-to-read versions such as the *Basisbijbel*. Despite this, the Statenvertaling remains an important historical text, with its thorough transcription over the years providing valuable data for scholars interested in studying the language and culture of the 17th century.

Figure 4.6 displays the figures for the dataset[6] associated with the Statenvertaling. As expected, this dataset is not as extensive as others, as it only consists of one book published in the year 1637.

[6] `https://viaveritasvita.info/SV_downloadables/DutSV.ont`

Nonetheless, this dataset can still provide insights into the language and culture of the Dutch Republic during this time period, making it a valuable resource for this research.



(a) Amount of words per year

(b) Amount of sentences per year

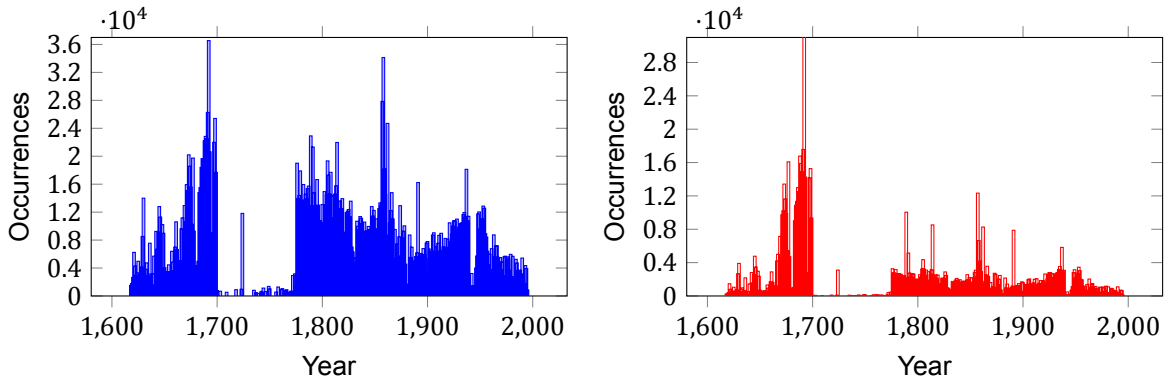
Figure 4.6: Two figures containing the amount of words and sentences per year for the *Statenvertaling* dataset

4.3. Full Dataset

The mentioned datasets have been concatenated into a single dataset, which will be utilized to train our sequence-to-sequence models. Details regarding the word and sentence counts can be found in Figures 4.7a and 4.7b. Upon examining these figures, it becomes apparent that there is a noticeable data gap between the years 1700 and 1775. Although the *Historical Newspapers* dataset includes some data from this period, the density of information is comparatively low when contrasted with other time periods.

Another significant observation is the significant increase in data for the year 1637, which can be attributed to the publication of the *Statenvertaling*. The combination of this dataset with the *Meertens* dataset has resulted in an extensive amount of data for that particular year. Additionally, an examination of the sentence count leading up to the year 1700 reveals a substantial number of sentences, corresponding to the word count for that period. Despite the presence of overlapping datasets, particularly between 1778 and 1878, this overlap is not clearly depicted in the figure.

Overall, the distribution of words appears to be relatively uniform across different time periods, with a slight decline as we approach the year 2000. In our workflow, all the datasets are merged into a single pandas dataframe, which is prepared for the subsequent step of synthetic data generation.



(a) Amount of words per year

(b) Amount of sentences per year

Figure 4.7: Two figures containing the amount of words and sentences per year for the entire dataset

4.3.1. Evenly spread dataset

Another crucial consideration is the creation of a single model capable of working effectively across various time periods. To achieve this objective, a specific dataset has been constructed, ensuring an even distribution of data points across different years of publication. Suppose we require a training dataset of 50.000 sentences. In such a scenario, sentences are selected based on their availability. If there is an insufficient number of sentences from a particular year, the remaining sentences are distributed among other years. This approach guarantees that it gives an equal representation of sentences for each year, thereby preventing over or under-representation of specific time periods. This hopes to ensure that the resulting models performance is evenly balanced over the different publication years.

5

Synthetic Data Generation

The main challenge in training post-correction models is the availability of sufficient training data. As discussed earlier, manual approaches to correcting texts can be labor-intensive and expensive. Additionally, matching each OCR text to its corresponding ground-truth text can be difficult, further increasing the problem of data availability. To overcome this challenge, it is possible to create a custom OCR dataset from ground-truth data. The general idea behind this approach is to create a background image and then overlay text onto the image. Finally, OCR software is applied to the image to generate the desired OCR output. With this approach the aim is to answer RQ1 of the research questions.

This approach offers several advantages, including its versatility across diverse fields. In Chapter 3.3, we examined its potential applicability to various domains, such as social media platforms. By employing synthetic data generation in such scenarios, where a continuous flow of textual data exists, the collection of ground-truth data becomes virtually cost-free. Utilizing synthetic data generation alongside the abundance of input data presents an opportunity to access an extensive pool of potential training data.

Moreover, this approach circumvents the need to address the alignment problem, not having to solve the requirement for matching the text before and after OCR. In the subsequent sections, we will delve into the details of the synthetic data generation framework. The initial stage entails creating a suitable background for text placement, which involves downloading images from Delpher and create building blocks out of those papers. These building blocks consist of square images with diverse backgrounds, serving as the foundation for constructing the background on which the text will be placed.

Considering the historical nature of the articles, which often exhibit variations in font styles, we will utilize multiple types of fonts. We will explain the process of selecting appropriate fonts for different time periods to ensure historical accuracy. In the final step, optical character recognition (OCR) is employed to convert the images containing the text into an actual text format, allowing for the creation of a full training dataframe.

5.1. Background creation

To generate artificial training data, the initial step involves crafting a canvas for the text. Given the evolution of paper over the last few decades, the type of paper used to display the text can significantly impact the OCR application's performance [31]. Consequently, the objective was to design a canvas that corresponds to the time period when a dataset was published. To achieve this, thousands of article images from 1618 to 1995 were downloaded using the Delpher search API[1]. With these images, building blocks for the canvas were created for each decade, covering the time periods from 1610-1620, 1620-1630, and so on. These building blocks each had a fixed width and height which was set to 80 pixels.

5.1.1. Downloading the images

Using the Delpher API, a function can be created to download a set of images from 1610 to 1995, for which we have data to create ground-truth datasets. The images are downloaded for every 10-year

[1] https://www.kb.nl/en/research-find/datasets/delpher-newspapers

time period, resulting in a set of downloaded images that are ready for the next step of the process. For every of the 10-year periods the maximum amount of images would be set to 1000. Examples of these images are shown in figures 5.1 and 5.3. The first image is a page from the newspaper *Algemeen Dagblad* from 02-01-1990, while the second figure is the oldest image on Delpher, the *Courante uyt Italien, Duytslandt* from 14-06-1618,

Figure 5.1: A page from the newspaper *Algemeen Dagblad* in 02-01-1990 in Rotterdam

5.1.2. Creating the building blocks

After obtaining the downloaded images of the newspaper pages as mentioned in the previous section, we can generate what are referred to as building blocks. These building blocks are used to compose the background of the synthetically generated images. The main challenge in creating these building blocks was finding clean areas that did not contain any text. To solve this problem, an algorithm was developed, as depicted in Algorithm 1. The algorithm takes an image that has been processed using the *InRange* function of OpenCV as input. By defining a range of colors in the HSV color space (16,30,30 to 150,255,255), the algorithm turns pixels outside of the desired range black, while turning those inside the range white. This effectively blocks out colors close to black and white, which are the dominant colors of the text and paper edges. The results of this algorithm can be observed in Figures 5.3 and 5.5. Text itself will become black, as well as the edges. The borders of the pages on the other hand will be almost completely white of colour. This property can now be used to identify where parts of the image are located, so that it can be used as a background.

The primary challenge encountered in creating these building blocks was identifying clean areas not containing any text. To overcome this hurdle, an algorithm was developed, as illustrated in Algorithm 1. This algorithm takes as input an image processed using OpenCV's *InRange* function. By defining a specific range of colors in the HSV color space (from 16,30,30 to 150,255,255), the algorithm sets pixels outside the desired range to black and those within the range to white. Consequently, colors close to black (representing text) and white (representing paper edges) are effectively suppressed. The outcomes of this algorithm can be observed in Figures 5.3 and 5.5. The text itself, as well as the edges, appear black, while the page borders exhibit nearly pure white color. This property enables the identification of the image's background areas, which can be subsequently utilized as backgrounds for further processing.

(a) (b)

Figure 5.3: A page from the *Courante uyt Italien, Duytslandt* on 14-06-1618, the earliest newspaper available on Delpher, accompanied by zoomed in sections of the newspapers.

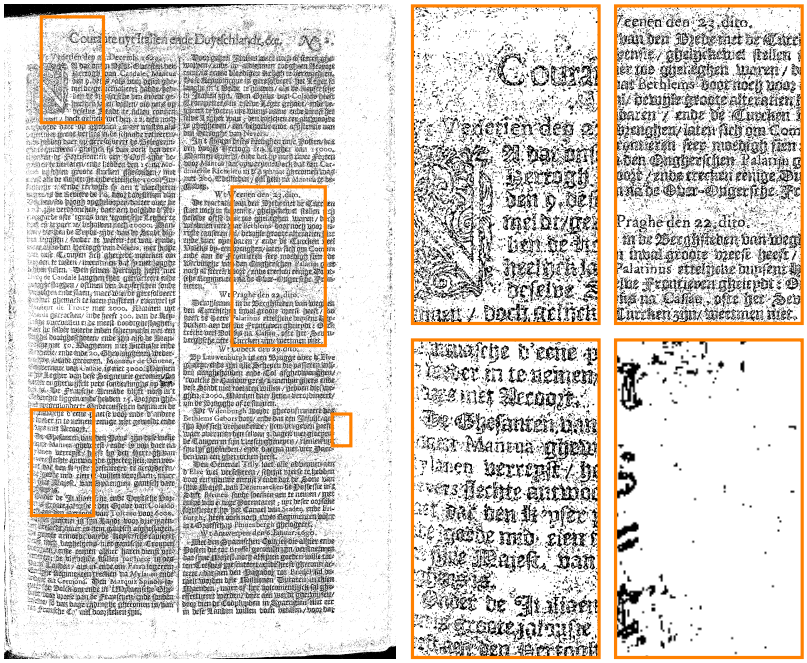
The algorithm for creating artificial training data employs a method for finding the cleanest 80 by 80 square in the image. This square has the highest sum of pixels and thus contains the least number of artifacts. Once this square is identified, it is saved for later processing and serves as a building block for constructing the background for the text. Since the quality of these building blocks can vary depending on the image, a subset of 20 images is selected for each decade, based on the highest sum using the *inRange* function. This means that for every 10-year period, there are 10 images that can serve as the background for the text. When selecting a building block, one is chosen at random from its' specific time period and used to create the entire background. The building blocks are not mixed, ensuring that the background remains consistent throughout the entire image. This process helps to create a more diverse set of training data, as each background is unique and corresponds to the time period in which the dataset was published.

Figure 5.6 provides an overview of the entire collection of building blocks. Notably, this image not only presents the assortment of building blocks but also provides insights into the evolution of paper over the past few decades. A transformation can be observed, particularly in the 17th century, where paper appeared much lighter in color, accompanied by a more pronounced bleeding of colors onto the page. The scans of early 1800s newspapers also exhibit a darker hue compared to more modern articles. Approaching the late 1800s and extending into the mid-1900s, the paper takes on a distinct, dark yellowish tone. Moreover, the margins of newspapers from the end of the 1900s progressively narrow, resulting in the appearance of sudden black bars within these eras.

By placing all these images into one large composition, we can visualise the evolution of paper types across the centuries. These variations serve as indicators of print and scan quality, which can significantly impact the final results of our analysis. Consequently, the importance of the decision to create images based on their respective publication dates becomes clear. These observations underscore the significance of considering the historical context, the characteristics of the paper used, and the potential implications on the OCR process.

5.2. What typeface to use

An important part of the visual representation of text on an image, is defined by the choice of typeface of the text. A typeface encompasses various elements, including the shape, weight, and style of the characters. These elements determine the overall look and feel of the typeface. Typeface families typically



(a) (b)

Figure 5.5: A page from the *Courante uyt Italien, Duytslandt* on 14-06-1618, the earliest newspaper available on Delpher, accompanied by zoomed in sections of the newspapers. This time the InRange function as described in Section 5.1.2 has been applied.

Algorithm 1 Finding White Square

```
procedure FindWhiteSquare(frame_ threshed, image_ size)
    highest_value ← 0
    x_min ← 0
    x_max ← image_size
    y_min ← 0
    y_max ← image_size
    for x in range(0, len(frame_threshed), int(image_size)) do
        for y in range(0, len(frame_threshed[x]), int(image_size)) do
            if x − len(frame_threshed) > image_size and y − len(frame_threshed[x]) > image_size then
                sum_value ← sum(frame_threshed[x : (x + image_size), y : (y + image_size)])
                if sum_value > highest_value then
                    highest_value ← mean_value
                    x_min ← x
                    x_max ← x + image_size
                    y_min ← y
                    y_max ← y + image_size
    return x_min, x_max, y_min, y_max
```

Figure 5.6: All the building blocks that were used for creating the background for the newspapers

consist of multiple variations, such as regular, bold, italic, and condensed, which offer different visual styles and weights within the same design. The words *font* and *typeface* are often used interchangeably although they are slightly different. A typeface is the broader design concept that encompasses the overall style, while a font refers to a specific variation or style within that typeface. Typefaces are like families, and fonts are the individual members within that family.

The choice of typeface can greatly impact the performance of an OCR application [31]. For the goal of synthetic image creation based on the year of publication, choosing the right typeface is crucial in giving a perspective as close to reality as possible. In order to discover which datasets were used in specific periods, we need to delve into the history of typefaces. As our dataset is only comprised of data from the year 1618 onward, we will only be discussing typeface that were used during that period.

During the early 17th century, Blackletter typefaces, especially the Fraktur font, dominated the typography landscape. These typefaces were characterized by their distinct Gothic appearance and ornate, calligraphic letterforms. Widely used in western Europe, Blackletter fonts remained popular until the late 17th century [10]. An early example of this typeface can be observed in Figure 5.3, which showcases the earliest available instance on Delpher. Amidst the prevalence of Blackletter fonts, another noteworthy typeface emerged, known as the Jenson roman typeface. Although it had been invented centuries earlier, its timeless design continued to be embraced during the 17th century.

Around the year 1720, a significant contribution to typography came from William Caslon, who introduced the Caslon typeface [14]. This typeface quickly gained recognition and prominence during that era, influencing the typography landscape for years to come.

As the 19th century unfolded, typography witnessed further evolution. One of the prevalent typefaces during this period was Didone, known for its elegant and high-contrast letterforms. Therefore, our application will utilize the Didone typeface to accurately represent the typography of the 19th century [38]. Additionally, the Baskerville font began to gain popularity during this time, reflecting a shift towards more refined and sophisticated typefaces. Towards the end of the 19th century, the typographic scene witnessed the emergence of crisp alternatives, such as the Bodoni typeface, which featured sharp serifs and geometric shapes.

The 20th century brought about a radical transformation in typography, driven by various artistic movements. The Art Nouveau movement, spanning the late 19th and early 20th centuries, introduced highly decorative and intricate typefaces inspired by organic forms and flowing lines. However, the most significant revolution occurred with the advent of the Bauhaus movement and the subsequent emergence of Swiss design in the mid-20th century. Renowned designers like Jan Tschichold and Max Miedinger championed a new approach to typography, emphasizing functionalism, simplicity, and geometric forms. It was during this period that iconic typefaces like Helvetica and Univers came into being, representing the pinnacle of minimalist yet impactful typography [38].

Throughout these historical periods, the evolution of typefaces mirrors not only changing aesthetics but also cultural and artistic influences. By understanding the significance of each typeface and its respective era, the application aims to faithfully represent the diverse typographic landscape across different time periods.

Although beyond the scope of our dataset, it is important to acknowledge the digital revolution of the late 20th century, which resulted in a diverse landscape of font types. Nowadays, there exists a vast array of typefaces, ranging from traditional and classic designs to more contemporary alternatives like Comic Sans. However, for the purpose of our application, we will primarily focus on the historical

and commonly used typefaces representative of specific periods.

Figure 5.7 visually presents examples of these selected fonts, showing the differences between each of the fonts. As mentioned earlier, our list of fonts includes two types of blackletter fonts. This decision was driven by the significant variations observed among blackletter fonts, making it essential to include multiple font types to capture the nuances of the era. The BreitkopfFraktur tends to have heavier strokes and more intricate letter forms, whereas the textur font is somewhat lighter, and little more refined in comparison. While the list of fonts is not exhaustive, it encompasses the most prevalent typefaces of the respective periods and is deemed suitable for our use case.

Figure 5.7: An image containing the fonts from old to new BreitkopfFraktur (Blackletter), Textur (Blackletter), Jenson, Bodoni, Caslon, Baskerville, Helvetica and Univers

Now that we have established the fonts to be used, the next step is to determine the appropriate periods for each font. However, achieving precise accuracy in this regard is impractical due to the extensive amount of data available. To address this challenge, a division table has been devised, as illustrated in Figure 5.8. This table allows us to navigate the overlaps between fonts during specific time periods. When overlapping occurs, a font is chosen randomly from the available options. Once a font is selected, the corresponding image is created exclusively using that particular font. Since images are generated for each individual year, all the relevant data from that year will be utilized with the corresponding font.

By employing this methodology, the aim is to create an application that accurately represents the typographic evolution across different periods, even though an exhaustive analysis of every single data point is not feasible.

5.2.1. Constructing the images

Now that enough building blocks have been created for the background, and the type of font for every timeperiod has been defined, it's time to put it all together and create images that can be used to gather data. The first step in the process of image creation is determining what the year is in which the data has been published. Now given this information, a building block corresponding to that period can be obtained from the results found in the previous section. Now the width of the image should be determined, as this allows for creating a horizontal image with the corresponding width. As the building blocks are of a fixed size of 80 by 80 pixels, the width of the horizontal image will also be rounded by 80. Now that a horizontal image is obtained, the goal is to obtain the vertical image size is needed to fit all the sentences. In this process, the font size and the distances between lines should also be taken into account, as this is an important point for separating these sentences later on in the process. In the case of a sentence that is longer than the page is wide, the sentence should be wrapped and proceed on the next line. Figure 5.9 shows a final example, but still containing the lines in which the sentences will be cropped when processing them individually.

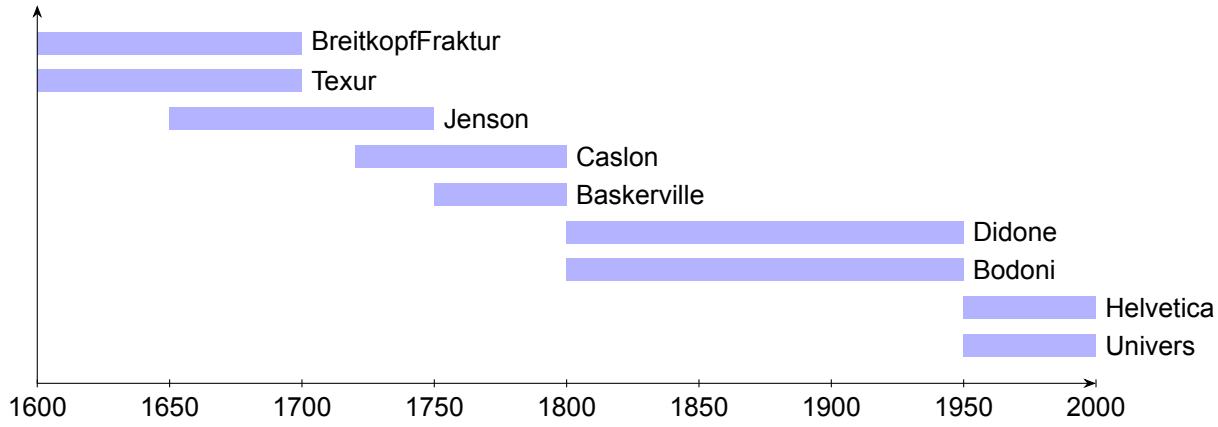


Figure 5.8: A figure containing the fonts that are used for creating the period dependent images, in the synthetic data generation step

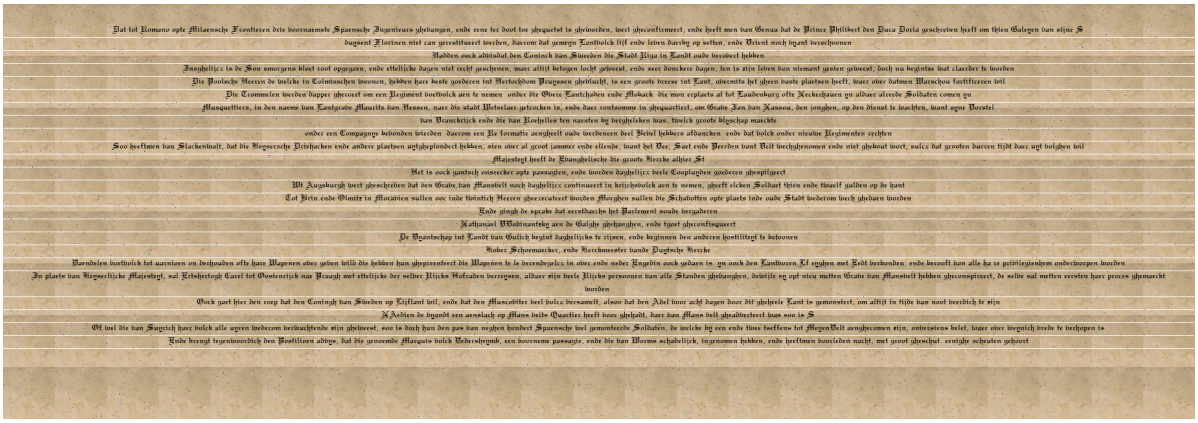


Figure 5.9: An example of a final image, but now also containing a visual representation of where the sentences will be cropped

Examples of the final images can be viewed in figures 5.10 and 5.11. These examples are taken of subsets of the dataset for the years 1629 and 1937 and are just meant to display what the final results would look like. What now also becomes visible is that the amount of data may vary according to year of the corresponding data. As discussed in Section 4.3, some years have a lot more data than the other ones, meaning that the images representing those years, will vary greatly in dimensions.

Another observation is that the sentences are positioned at the center of the page, as opposed to being aligned to the left side. This deliberate choice aims to replicate the appearance of authentic newspapers, where sentences frequently deviate from the left margin. By implementing this, coupled with the variation in sentence starting points across the background page, the goal is to create a more diverse dataset.

5.2.2. Generating a Dataset

With regards to this objective, the outcome produced by OCR is denoted as the *source* since it serves as the input for a post-correction model. Conversely, the ground-truth data is referred to as the *target* as it represents the expected output of a model. Building upon the approach described in the preceding sections, sentences can be extracted by cropping the text according to the predefined coordinates.

As explained in Section 2, our chosen program for obtaining OCR output from the images is tesseract. By applying this program to the list of images, we can generate the OCR output for each image. This procedure is repeated for every year since each year's dataset is contained in a separate image. Once we have the *source* output list, it can be merged with the sentences used to generate the images from the *target* column. Consequently, we obtain a unified dataframe that is ready to be used for training various models.

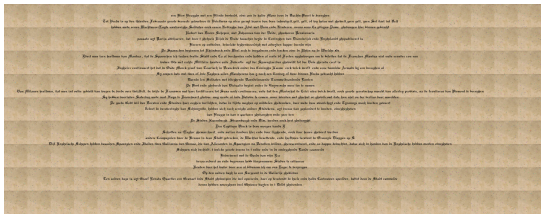


Figure 5.10: An image created with a subset of the data for the year 1629

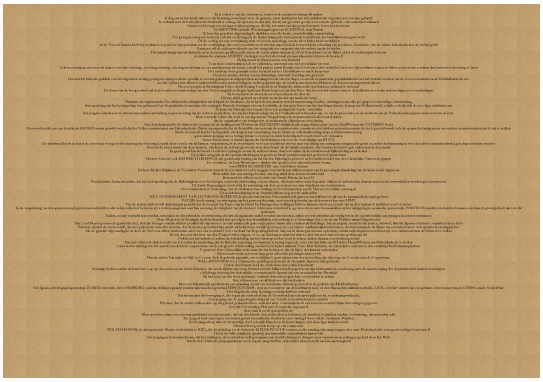


Figure 5.11: An image created with a subset of the data for the year 1937

5.2.3. Examples of generated data

To provide an illustration of the data generated using this method, refer to Figure 5.12. On the left, you can observe the original sentences as the ground-truth, while on the right, you can see the sentences after undergoing the synthetic data generation process. It is evident that after passing a sentence through the framework, it becomes somewhat jumbled compared to the original version. The sentence deliberately incorporates errors, mimicking text and it's OCR equivalent.

t welck de handelingh van Vrede vry te rugh stelt, t welck de handelingh van Vrede vry te rugh stelt, alsoo men hoope heeft gehad, die noch eenmael in haer Logement sou hervat werden	t **ir**elck de ha**m**delingh **b**an **rede G**ry te **cagh** stelt, **also** men hoo**g**e heeft ge**q**a**b**, die **mock** eenm**u**el im haer**e** Logement sou her**s**at **terder**

Figure 5.12: Example of generated data, with on the left the ground-truth and on the right data as it comes out of the framework

6

Sequence-to-Sequence Model

This chapter primarily focuses on the procedures involved in fine-tuning and evaluating models. It begins by providing an overview of the trained models, followed by a description of the evaluation process. The evaluation process includes a discussion of the various metrics employed. In the subsequent sections, three types of models are trained: models trained on an evenly distributed subsection of the datasets, as outlined in section 4.3.1. Furthermore the impact of further fine-tuning an existing model is explored, and the performance of models trained on one dataset is investigated when applied to another dataset. The results of these experiments can be found in Chapter 7.

6.1. Fine-tuning a model

Fine-tuning is a technique in machine learning that allows us to enhance the performance of pre-trained models. It involves taking a pre-existing model that has been trained on a large dataset and adapting it to a specific task or domain. Instead of starting the training process from scratch, which would require an extensive amount of data and computational resources, fine-tuning builds upon the knowledge already captured by the pre-trained model. This process not only saves time and resources but also enables us to leverage the learned features and patterns from the initial training, empowering the model to excel in new, specialized tasks.

During the fine-tuning process of a machine learning model, there are several hyperparameters that can be adjusted to optimize the performance of the final model. These hyperparameters play a crucial role in determining how the model learns and performs. Examples of such hyperparameters include the number of training epochs, the learning rate, and the choice of optimizer.

In machine learning, training epochs refer to the number of times the learning algorithm iteratively processes the entire training dataset during the training phase. Each epoch consists of one complete pass through the training data, where the model makes predictions, computes the loss, and updates its parameters based on the optimizer's rules. The goal of multiple epochs is to allow the model to gradually learn and refine its representations and predictions by exposing it to different instances and patterns in the data.

The learning rate, on the other hand, is a hyperparameter that determines the step size at which the model's parameters are updated during each of the epochs. It controls the magnitude of parameter adjustments made by the optimizer. A higher learning rate results in larger updates, allowing the model to converge faster, but it may risk overshooting the optimal solution. Conversely, a lower learning rate leads to smaller updates, which can result in slower convergence but potentially more precise parameter estimation. Finding an appropriate learning rate is essential for effective and stable training.

The choice of learning rate, along with the type of optimizer used, influences how quickly or slowly the model learns from the training data and in what manner. Different optimizers, such as gradient descent, stochastic gradient descent (SGD), or adaptive optimizers like Adam, incorporate various strategies to update the parameters and adjust the learning rate dynamically. Optimizers can include momentum, which helps smooth the updates and accelerate convergence, or adaptive techniques that adapt the learning rate based on the gradients or historical updates.

By controlling the learning rate and selecting an appropriate optimizer, machine learning practitioners can influence the training process. A high learning rate with momentum may lead to faster convergence, especially in the presence of noisy or sparse gradients. Adaptive optimizers can automatically adjust the learning rate to different parameters, improving convergence and handling different types of data. However, selecting the right optimizer and fine-tuning its hyperparameters is a critical step in achieving optimal model performance.

In summary, training epochs determine how many times the model is exposed to the entire training dataset, allowing it to learn and improve its predictions. The learning rate, in combination with the chosen optimizer, determines the step size and update strategy for the model's parameters during each epoch. Finding the right balance between the number of epochs, learning rate, and optimizer type is essential for effective and efficient training of machine learning models.

6.1.1. Full Dataset Models

For training a model on a subset of the full dataset, an evenly spread dataset was used as discussed in Section 4.3.1. To study the effects of varying sizes of training data, models were trained with this approach for dataset sizes of 50.000, 100.000 and 200.000 sentences.

To study the effects of what effect the trained data has on the final model in case of transfer learning, models should be compared against each other, with 1 trained on dutch data, and the other on non-dutch data. For this the models T5-Base[1] [33] and T5-Base Dutch have been chosen. The T5-Base model has only been trained on a dataset containing the languages English, French, Romanian and German. The T5-Base-Dutch model, has the original T5 architecture, but is fine-tuned on a cleaned version of the dutch part of the C4 multilingual dataset (mC4)[2]

In recent times, an apparently notable enhancement to the T5-Base model has emerged, known as T5-Flan. This improved version of the model aims to address certain limitations and further refine its capabilities [6]. The T5-Flan model, which can be found in the Hugging Face[3] model repository, incorporates advancements to augment its performance in various natural language processing tasks.

The purpose of comparing Dutch and non-Dutch containing base models is to investigate the effects of transfer learning, specifically to address the research question RQ3. This comparison helps in understanding the impact of transferring knowledge from one task to another. On the other hand, the comparison between all these models serves to answer research question RQ2, which aims to explore the performance disparities among the different models.

6.1.2. Further Finetuned Models

Furthermore, we will investigate the impact of additional finetuning. The most successful model among the previously selected models will undergo further training using additional data segments. This analysis aims to assess how further finetuning affects the model's performance and its ability to adapt to new information. By exposing the model to a broader range of data, we can evaluate improvements in accuracy, robustness, and predictive capabilities.

To summarize the steps outlined above, three base models are utilized for further finetuning: T5-Base, T5-Base-Dutch, and T5-Flan. In this evaluation phase, these base models are finetuned on datasets containing 50,000, 100,000, and 200,000 samples. The model that achieves the best performance will undergo additional finetuning to investigate its effects on overall performance. This process is also relevant to the concept of transfer learning and contributes to answering research question RQ3.

6.1.3. Individual Dataset Models

On top of that other models were trained on the individual datasets. To assess the performance of how a model trained on 1 dataset performs on another, models have to be trained on specific datasets. In our case this was done for each of the 5 datasets: *DBNL*, *Historical Newspapers*, *IMPACT*, *Meertens* and the *Statenvertaling*. For each of them a set of 50,000 sentences were randomly sampled out of the entire dataset. These sentences were then used to train each of the 5 models.

[1] `https://huggingface.co/t5-base`
[2] `https://huggingface.co/datasets/yhavinga/mc4_nl_cleaned`
[3] `https://huggingface.co/google/flan-t5-base`

6.2. Evaluations

This section focuses on evaluating all the models described in the preceding sections. Initially, it is important to establish the metrics employed and their interpretation. Moreover, the evaluation process itself is outlined, including the specific evaluation dataset utilized.

6.2.1. Metrics for evaluation

To evaluate the performance of a trained sequence-to-sequence model, it is necessary to conduct an assessment. Understanding the model's performance enables effective comparisons with other models. Evaluating the quality of OCR output involves utilizing various metrics, with Word Error Rate (WER) and Character Error Rate (CER) being the most commonly employed metrics, as highlighted by Carrasco [4]. WER and CER are error rate measures that gauge the disparity between the OCR output and the ground-truth. Additionally, the evaluation process incorporates the Jaccard similarity metric. Together, these metrics provide valuable insights into the post-correction application's performance.

WER is a measure of the percentage of errors in the transcription of spoken language compared to a reference transcription. The reference transcription in the instance of OCR, is the ground-truth text, which has already been manually revised. WER is calculated by dividing the number of errors (substitutions, deletions, and insertions) by the total number of words in the reference transcription. In the equation below can be seen how WER can be calculated. Here n_w is the number of words in the reference text, s_w is the number of words substituted, d_w the number of words deleted and i_w the number of words inserted required to transform the output text into the target.

$$WER = \frac{i_w + s_w + d_w}{n_w} \tag{6.1}$$

In Figure 6.1, an example of the Word Error Rate (WER) is presented. The reference sentence is displayed on the left, while the sentence containing errors is shown on the right. In this example, there are 4 substitutions (slowy, ober, hotizon, gouden), 1 insertion (a), and 1 deletion (tranquil). These errors account for a total of 15 words, resulting in an error rate of $\frac{4+1+1}{15} = 0.4$ or 40 percent. It is important to note that during a post-correction process, this score is expected to decrease, indicating a reduction in errors.

The sun slowly set over the horizon, casting a golden glow across the tranquil beach.	The sun **slowy** set **ober** the **hotizon**, casting a **gouden** glow across the ~~tranquil~~ beach.

Figure 6.1: Example text, aiding in explaining how to calculate the Word Error Rate (WER) and Jaccard similarity

In a similar vein, the Character Error Rate (CER) serves as a metric to quantify the percentage of errors in the written text transcription when compared to the ground-truth, focusing specifically on character-level errors. CER is computed by dividing the total number of errors (including substitutions, deletions, and insertions) by the number of characters in the reference transcription. In this context, let n represent the character count in the reference text, s denote the number of substituted characters, d indicate the count of deleted characters, and i reflect the number of inserted characters required to transform the output text into the target.

$$CER = \frac{i + s + d}{n} \tag{6.2}$$

The example for calculating the character error rate can be found in Figure 6.2. In the example 4 substitutions,1 insertion and 1 deletion can be found, all on a total of 85 characters, this comes to a score of $\frac{4+1+1}{85} = 0.071$ or 7.1%. The score of the CER can actually rise over a score, when there are a lot more characters in the new sentence than the original one. As the same for the WER, this score is also meant to go down in a post-correction process.

The sun slowly set over the horizon, casting a golden glow across the tranquil beach.	The sun slowły set o**b**er the hot**i**zon, casting a**a** go**u**den glow across tranqu**y**l the beach.

Figure 6.2: Example text, aiding in explaining how to calculate the Character Error Rate (CER)

The final metric that will be used for evaluation is the Jaccard similarity. This metric measures the amount of words 2 sentences have in common. It does this by dividing the amount of words in common by the total amount of words in both sentences. For this we take the same example as for the WER in Figure 6.1. In this example the total intersection is of size 10, and the total union is of size 19. This results in a Jaccard similarity of $\frac{10}{19} = 0.526$. This will output a score between 0 and 1, with a score of 1 meaning that the sentences are the same. In our post-correction process, the goal is thus to increase this score.

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{6.3}$$

Now given these metrics the evaluation of a model can easily be performed. First the baseline score of the metrics have to be calculated given the ground-truth sentence and the sentence after OCR. Then after applying the post-correction process, this score can be calculated again. For the WER and CER a lower score, indicates a better performance. the goal is the decrease this score, whereas the score for Jaccard similarity should go up.

6.2.2. Evaluation of full dataset models

The evaluation for this section will be performed on the 9 models described in Section 6.1.1. These models have undergone training and have learned to perform specific tasks based on a diverse dataset. To assess their performance, a set of 250 sentences has been randomly sampled from the full dataset. These sentences represent a representative subset of the overall data and will be used to evaluate how well the models handle various sentences. To quantify the models' performance in comparison to the original, a set of metrics, as outlined in Section 6.2.1, will be employed.

When performing post-correction on a dataset, the aim is to lower the amount of distinct words. To illustrate this, let's consider the Dutch word ”wielrennen” as an example. After undergoing OCR, this word can take various forms such as ”Wylrenne,” ”Wylrennen,” ”weilrennen,” or ”wielrenen.” All these variations should be corrected to the same word, but currently, they are counted as different words. Therefore, during post-correction, multiple words will be corrected into the same word, resulting in a reduction in the number of distinct words in the dataset. Therefore a graph will be created on a subset of the full dataset, such that the amount of words in the ground-truth of the subset, contain roughly 2000 words for each timeperiod. This method was performed on a subset, as it was unfeasible to perform it on the full dataset. AS explained the amount of distinct words should be lower after performing post-correction, eliminating words containing errors. Now given that the amount of words the ground-truth, OCR and post-correction texts will be displayed in a single figure, it allows for easy comparison of the three.

In addition to the evaluation mentioned earlier, a visual comparison will be conducted using the top-performing model. This comparison will focus on examining sentences both before and after the post-correction process. The intention behind this visual analysis is to showcase concrete examples that demonstrate the capabilities of the model.

6.2.3. Evaluation of individual dataset models

Given that models that were previously trained for the individual datasets as mentioned in Section 6.1.3, an evaluation can be performed on each of these models. This evaluation aims to provide answers to RQ4 of the research questions. For this task a set of 250 randomly sampled sentences were extracted from the original datasets. Important to note is that these sentences were not included in the set used to train each of the models. The models will thus each evaluated on each of these datasets and will be accompanied by a baseline score, which indicates what the comparison metrics were, *before* any post-correction was performed on them.

7

Results and Discussion

This chapter presents the evaluation results along with a discussion of their implications and reasons behind the observed outcomes. It provides an overview of the evaluation results for the models described in sections 6.1.1, 6.1.2, and 6.1.3. The evaluation process itself is described in sections 6.2.2 and 6.2.3. It also provides a visual example of one of the results.

7.1. Evaluation of the full dataset

The results presented in this section are summarized in Table 7.1, which includes a table showcasing the performance metrics. Upon analyzing these results, several observations can be made. Firstly, when considering training set sizes of 50,000 and 100,000, both T5-Base and T5-Base-Dutch exhibit relatively similar performance. However, for a training set size of 200,000, the T5-Base model noticeably underperforms compared to the T5-Base-Dutch model. This difference suggests that the Dutch-specific training data used in T5-Base-Dutch plays a noteworthy role in enhancing its performance, highlighting the benefits of transfer learning in the context of post-correction.

Another interesting finding is that the T5-Flan model consistently outperforms the other models across all tested scenarios. This implies that the improvements implemented in this particular model effectively contribute to its improved performance, even within the field post-correction. These results emphasize the success of the enhancements made in T5-Flan and reinforce its position as the top-performing model in this study.

Surprisingly, the model trained on a smaller corpus of 100,000 sentences outperformed its counterpart trained on a larger corpus of 200,000 sentences. This suggests that simply increasing the number of sentences does not guarantee better performance. This observation raises questions about the factors that contribute to model effectiveness. It implies that factors other than the sheer volume of training data, such as data quality, diversity, or the specific characteristics of the sentences, might play a role in determining the model's performance, but further exploration would be required to fully comprehend.

The findings of the experiment, detailed in Section 6.2.2, are presented in Figure 7.1. Upon analyzing the figure, several observations can be made. Firstly, it is evident that the number of words identified by the OCR system is generally higher compared to the ground-truth text. This outcome was anticipated due to the inherent variability in OCR-generated words. Additionally, the figure illustrates that the post-corrected version exhibits a significantly closer resemblance to the ground-truth text in terms of word count. This outcome aligns with expectations since a significant portion of word variations are intended to be eliminated through the post-correction process. These insights shed light on the behavior and performance of the OCR system and the effectiveness of the post-correction mechanism in aligning the OCR-generated text with the ground-truth text.

The number of errors in the OCR exhibits a declining trend, which can be attributed to the approach employed in our application, as this is related to the fact of year dependent font and background creation. As a result, the quantity of errors is initially higher during the early 17th century and gradually decreases as time progresses. This correlation between the decreasing error rate and the temporal context of the data demonstrates the effectiveness of our method in capturing and OCR errors over different historical periods.

The dataset size used for training	The base models used			
	Baseline	**T5-Base**	**T5-Base-Dutch**	**T5-Flan**
Baseline	WER: 0.281 CER: 0.077 Jaccard: 0.612			
50.000		WER: 0.175 CER: 0.065 Jaccard: 0.793	WER: 0.177 CER: 0.102 Jaccard: 0.775	WER: 0.139 CER: 0.05 Jaccard: 0.81
100.000		WER: 0.155 CER: 0.064 Jaccard: 0.799	WER: 0.159 CER: 0.067 Jaccard: 0.815	WER: 0.125 CER: 0.046 Jaccard: 0.829
200.000		WER: 0.165 CER: 0.074 Jaccard: 0.797	WER: 0.142 CER: 0.077 Jaccard: 0.827	WER: 0.158 CER: 0.077 Jaccard: 0.846

Table 7.1: A table depicting the results retrieved from the experiments, on the full dataset, for a variation of training sizes and base models

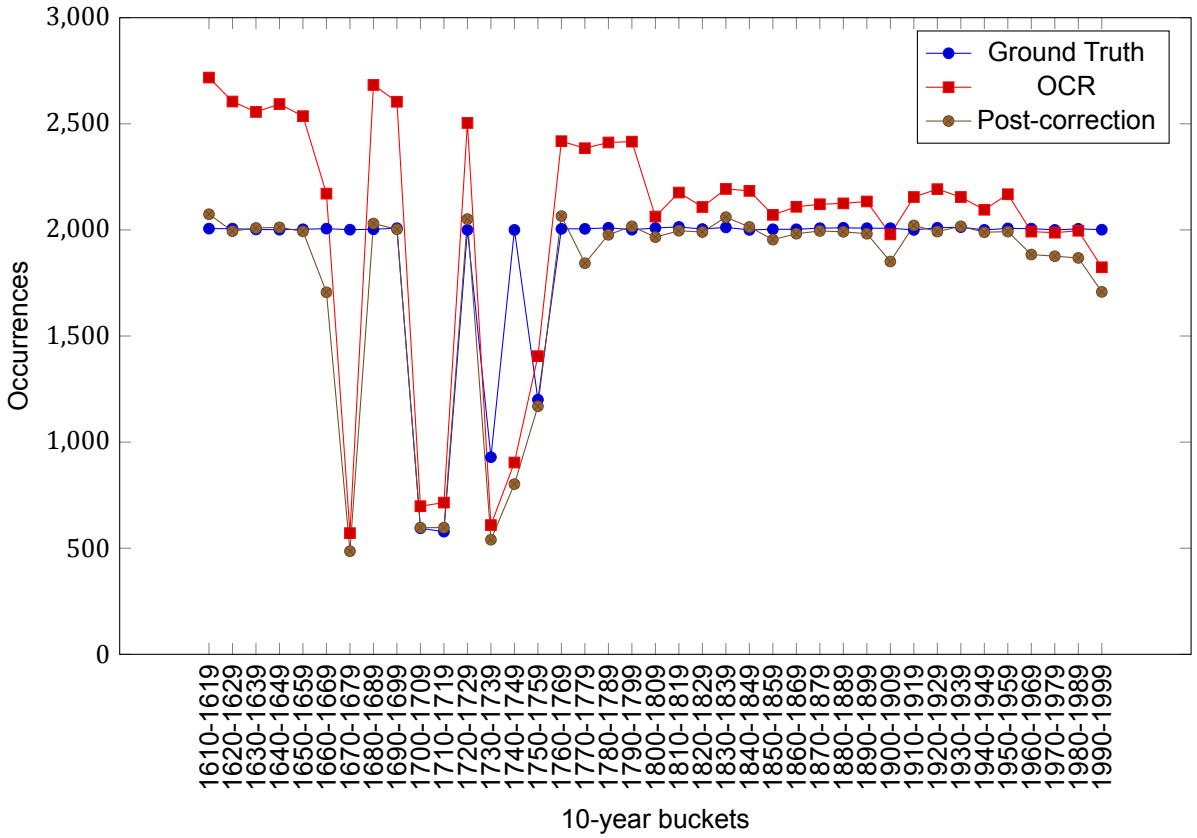


Figure 7.1: The amount of words for the ground truth, OCR and post-correction

Baseline	T5-Flan	T5-Flan 1st further fine-tuning	T5-Flan 2nd further fine-tuning
WER: 0.266 CER: 0.075 Jaccard: 0.631	WER: 0.125 CER: 0.046 Jaccard: 0.829	WER: 0.098 CER: 0.043 Jaccard: 0.863	WER: 0.108 CER: 0.054 Jaccard: 0.853

Table 7.2: A table depicting the results of the accuracies of further fine-tuning the T5-Flan model trained on 100,000 sentences.

7.2. Evaluation of the further finetuned models

As discussed in Section 6.1.1, the impact of further finetuning on performance is explored. To achieve this, the highest-performing model from Table 7.1 has been selected, which is the T5-Flan model trained on 100,000 sentences. Subsequently, we further finetune this model twice using separate portions of the dataset, each containing 100,000 sentences. The results of these further finetuned models are presented in Table 7.2.

Analyzing the aforementioned table, we can conclude that further finetuning contributes to performance improvement. For both the first as the second session of further finetuning, all the metrics indicate that However, it is worth noting that the second session does not outperform the initial session.

As discussed in Section 6.1.1 it will also be investigated what effects retraining has on the performance. For this goal, the best performing model was chosen from Table 7.1, namely the T5-Flan model trained on 100.000 sentences. This model was now retrained using other parts of the dataset, also containing 100.000 sentences each, twice. The results of these models are displayed in Table 7.2. From this table, the conclusion can be drawn retraining helped improving the performance, although after the second retraining is not performing better than the first one.

7.3. Evaluation per individual dataset

Based on the previous findings indicating the superior performance of models pre-trained on Dutch data, it has become clear that these models outperform their counterparts that lack Dutch pre-training in most instances. Therefore, for the purpose of fair comparison, the focus of this study will be solely on models that have been trained using pre-trained models specifically tailored to Dutch data.

The evaluation results for the individual dataset can be found in figures 7.3 and 7.4, with the first having T5-Flan as base model and the second one T5-Base-Dutch. In this figure the row indicates on which the model has been trained on, and the column on which data the model is evaluated on. The top row indicates the baseline, which are the values *before* any post-correction was performed on the data. Each column contains the word error rate (WER), character error rate (CER) and the Jaccard similarity as explained in Section 6.2.1.

There are several observations that can be derived from the information presented in this table. Let us begin with the most apparent finding, which is that the T5-Flan clearly outperforms the T5-Base-Dutch model, which is comparable behaviour as in the evaluation of the full models from the previous section. With this, it comes to the same conclusion that the T5-Flan model is better suited for post-correction than the T5-Base-Dutch model.

Upon closer examination of the results, it becomes evident that a model trained on a specific dataset consistently outperforms other models when evaluated on the same dataset, even when it incorporates additional data that was not utilized during the training process. However, it is important to note that the presence of overlapping sections within the timespans of the varying datasets does not always guarantee success.

Another intriguing observation is related to the Meertens model, which exclusively relies on data from the 17th century. Surprisingly, it consistently ranks as the second worst performing model in the list, often falling behind the baseline and often worsening the quality of the OCR instead of improving it. The only exception to this trend is when evaluating the model on the *Statenvertaling*, where some improvement is observed, likely attributable to the utilization of 17th-century data in both cases.

On the other hand, the poorest performing model overall is the one trained on the *Statenvertaling* dataset. Apart from cases where the evaluation is conducted on the *Statenvertaling* dataset itself, this

The datasets used for training	The datasets used for Evaluation				
	DBNL	**Historical Newspapers**	**IMPACT**	**Meertens**	**Statenvertaling**
Baseline	WER: 0.278 CER: 0.071 Jaccard: 0.618	WER: 0.19 CER: 0.058 Jaccard: 0.734	WER: 0.302 CER: 0.077 Jaccard: 0.586	WER: 0.461 CER: 0.113 Jaccard: 0.379	WER: 0.503 CER: 0.143 Jaccard: 0.346
DBNL	WER: 0.153 CER: 0.043 Jaccard: 0.785	WER: 0.084 CER: 0.04 Jaccard: 0.894	WER: 0.172 CER: 0.051 Jaccard: 0.752	WER: 0.31 CER: 0.085 Jaccard: 0.532	WER: 0.3 CER: 0.111 Jaccard: 0.585
Historical Newspapers	WER: 0.208 CER: 0.065 Jaccard: 0.719	WER: 0.066 CER: 0.036 Jaccard: 0.915	WER: 0.21 CER: 0.065 Jaccard: 0.714	WER: 0.395 CER: 0.111 Jaccard: 0.449	WER: 0.417 CER: 0.132 Jaccard: 0.427
IMPACT	WER: 0.195 CER: 0.063 Jaccard: 0.722	WER: 0.102 CER: 0.057 Jaccard: 0.883	WER: 0.199 CER: 0.109 Jaccard: 0.794	WER: 0.21 CER: 0.059 Jaccard: 0.665	WER: 0.254 CER: 0.087 Jaccard: 0.61
Meertens	WER: 0.347 CER: 0.133 Jaccard: 0.538	WER: 0.22 CER: 0.093 Jaccard: 0.69	WER: 0.252 CER: 0.091 Jaccard: 0.644	WER: 0.099 CER: 0.032 Jaccard: 0.835	WER: 0.331 CER: 0.114 Jaccard: 0.522
Statenvertaling	WER: 0.364 CER: 0.145 Jaccard: 0.525	WER: 0.247 CER: 0.12 Jaccard: 0.665	WER: 0.346 CER: 0.153 Jaccard: 0.562	WER: 0.446 CER: 0.179 Jaccard: 0.406	WER: 0.076 CER: 0.03 Jaccard: 0.875

Table 7.3: A table depicting the results retrieved from the experiments, conducted on the T5-Flan as base model

model consistently ranks lower than any of the other models. This could be attributed to the limited language usage and vocabulary present in a religious text such as the Bible.

Now let us shift our focus to the better performing models. When using T5-Flan as the base model, the model trained on the *DBNL* dataset demonstrates superior performance compared to the *IMPACT* dataset when evaluated on both the Historical Newspapers and the IMPACT dataset. The only exception to this pattern occurs with the Meertens and Statenvertaling datasets, where the rankings are reversed. However, when T5-Base-Dutch serves as the base model, the *IMPACT* dataset outperforms *DBNL* in all instances. This difference highlights the effect various base models have on the final results, even when the same training dataset is employed.

The most balanced performance can be observed fo the model trained on the *Historical Newspapers* dataset in comparison to the other models, particularly when T5-Base-Dutch is employed as the base model.

7.4. Visual Examples

To provide an illustration of the application's final output, we can refer to Figure 7.2, which presents an example. The figure showcases three distinct forms of the same sentence: the OCR version, the post-corrected version, and the ground-truth sentence. Upon observing the results after the post-correction process, several observations can be made. Firstly, the model accurately identified the letter *f* in the word ”verfcheide,” which was historically pronounced as *s* and can be found in articles dating back to the 17th century. However, the model chose to correct the word ”wy” to ”wij,” a spelling commonly used in more modern versions of Dutch. The same can be observed for the words ”proeve” and ”proef,” with the latter being a more recent variation.

The datasets used for training	The datasets used for Evaluation: DBNL	Historical Newspapers	IMPACT	Meertens	Statenvertaling
Baseline	WER: 0.278 CER: 0.071 Jaccard: 0.618	WER: 0.19 CER: 0.058 Jaccard: 0.734	WER: 0.302 CER: 0.077 Jaccard: 0.586	WER: 0.461 CER: 0.113 Jaccard: 0.379	WER: 0.503 CER: 0.143 Jaccard: 0.346
DBNL	WER: 0.199 CER: 0.074 Jaccard: 0.741	WER: 0.231 CER: 0.11 Jaccard: 0.829	WER: 0.251 CER: 0.101 Jaccard: 0.704	WER: 0.355 CER: 0.12 Jaccard: 0.501	WER: 0.318 CER: 0.125 Jaccard: 0.568
Historical	WER: 0.328 CER: 0.192 Jaccard: 0.647	WER: 0.08 CER: 0.056 Jaccard: 0.905	WER: 0.286 CER: 0.15 Jaccard: 0.666	WER: 0.477 CER: 0.221 Jaccard: 0.407	WER: 0.521 CER: 0.275 Jaccard: 0.415
IMPACT	WER: 0.268 CER: 0.124 Jaccard: 0.669	WER: 0.114 CER: 0.075 Jaccard: 0.868	WER: 0.153 CER: 0.074 Jaccard: 0.805	WER: 0.238 CER: 0.098 Jaccard: 0.646	WER: 0.313 CER: 0.166 Jaccard: 0.59
Meertens	WER: 0.416 CER: 0.246 Jaccard: 0.514	WER: 0.345 CER: 0.267 Jaccard: 0.622	WER: 0.484 CER: 0.344 Jaccard: 0.533	WER: 0.304 CER: 0.246 Jaccard: 0.645	WER: 0.461 CER: 0.294 Jaccard: 0.435
Statenvertaling	WER: 0.401 CER: 0.215 Jaccard: 0.504	WER: 0.251 CER: 0.146 Jaccard: 0.706	WER: 0.384 CER: 0.215 Jaccard: 0.553	WER: 0.504 CER: 0.255 Jaccard: 0.361	WER: 0.075 CER: 0.04 Jaccard: 0.885

Table 7.4: A table depicting the results retrieved from the experiments, conducted on the T5-Base-Dutch as base model

‘uit de veri**c**heide tre**t**fende **Laterecien** Kiezen wy het volgende ter pro**c**ve.	- uit de verfcheide treffende Lateren kiezen wij het volgende ter proef	uit de verfcheide treffende Tafereelen kiezen wy het volgende ter proeve

Figure 7.2: Examples for the OCR, post-correction and ground-truth text

8

Conclusion

This work introduces a method for generating synthetic data specifically designed for post-correction on OCR. The results demonstrate that the model produces usable data, which can be effectively employed to train different post-correction methods. Multiple T5 models have been trained using this data, showcasing a substantial improvement in data quality. It is important to note that the approach proposed in this thesis should not be considered as a replacement for existing synthetic data generation procedures, but rather as a complementary approach that offers a distinct advantage. The generated data can be utilized to train other state-of-the-art models, contributing to the advancement of the field.

This chapter aims to answer the research questions for this thesis, along with summarizing the main findings. It also provides a section on potential work for future research directions.

RQ1: How effective can synthetic data be generated, as closely resembling images from a given timeperiod?

With the process as described in Chapter 5, a procedure was outlined to facilitate the creation of images with attention to historical details. These images encompassed not only background elements but also specific characteristics of the typeface used. By incorporating both these factors, it became possible to generate images that closely resemble the time period during which the original texts were published. This approach ensures that the created images capture the essence and visual attributes associated with the respective historical era.

RQ2: How effective are machine learning-based OCR post-correction methods trained on synthetic data in improving the accuracy and quality of OCR output?

The results presented in Section 7 show that the models trained on synthetic data, can show great performance in when correcting OCR. Notably, when these models undergo additional fine-tuning, as shown in Table 7.2, further improvements are observed. Given that these datasets can be created, whenever ground-truth data is present, it can become of great importance, also outside of the field of historical data. One of the surprising conclusion that can be drawn from the data, is that having more sentences does not always mean it will output a better performing model. In our tests the one trained on 100.000 sentences outperformed the one trained in 200.000 sentences. Potentially, this was done due to overfitting on the training set, but further research would need to be done to figure out why exactly this was the case.

RQ3: How much of an effect does a model pre-trained on Dutch data have on the performance of a fine-tuned model?

The answer to this question is presented in Table 7.1, which demonstrates that the performance of T5-Base and T5-Base-Dutch is comparable. However, when considering T5-Flan, it exhibits noticeable improvements over the other two models. It is worth mentioning that T5-Flan is a newer and more advanced model.

Furthermore, Table 7.2 presents the outcomes of models that underwent further fine-tuning using supplementary data. These models were initially trained on a base model and subsequently subjected

to the fine-tuning process once again. During this process, the first revision demonstrated a notable improvement in performance. However, it appears that the second revision led to a decline in performance, possibly indicating overfitting.

RQ4: How does the performance of a transformer model trained on one dataset compare when evaluated on a different dataset?

This question is related to the generalizability of the models, and the corresponding results can be observed in Table 7.3 and Table 7.4. These tables reveal that the models do not exhibit equal performance across different datasets. They demonstrate that datasets containing data from more diverse time periods generally yield better overall performance. However, it is evident that most models encounter difficulties when they have not been trained on data from a specific period. This is likely attributed to the evolution of language over time.

8.1. Future work

Other Machine Learning Models

This study utilized a T5 sequence-to-sequence model trained on synthetic data. It is worth noting that the methodology discussed in section 3 can be extended to other machine learning models, potentially enhancing their performance. Additionally, the base models examined in this thesis, also are available in versions trained on larger data sizes, presenting an opportunity to explore the described approach with larger variations of these models and compare their performance. Unfortunately, this was not feasible due to hardware limitations.

Models by timeperiod

In future work, it is recommended to train models specifically on distinct time periods to investigate the performance tradeoffs associated with using them. This approach would involve training models exclusively on datasets from the 16th, 20th, and 21st centuries, for example. By comparing the performance of these models on evaluation data from different time periods, it would provide valuable insights into the impact of language progression over time. It is hypothesized that a model trained on 16th century data would likely perform worse on 21st century evaluation data compared to a model trained on 20th century data. Further exploration in this direction would contribute to a deeper understanding of language evolution and its implications for language models.

Modern Data Acquisition

In the case of this application, gold standard data is required. For this thesis data was used, that had to be manually corrected. Fortunately, there are now numerous sources from which this data can be obtained, including online news media and other websites that contain text. These texts are of sufficient quality to be used for creating a dataset.

Automatic typeface recognition

Choosing the right typeface for the image creation step can have a significant impact on performance. Throughout the ages, many different types of typefaces have been used. In this work, a study was performed to research when certain typefaces were most dominant in time. This included delving into the field of typography research. It was also observed that, especially in more modern works, often more than one typeface was used on a single page. An additional enhancement that could be made is automatic typeface recognition, where a set of scanned images can be inputted, and a list of typefaces with corresponding publication dates would be generated as output. This would further diversify the dataset in comparison, making it a valuable addition to the current setup.

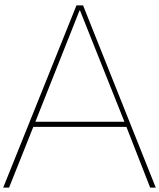
Differentiation in typeface models

In our approach, we assigned the typeface based on the specific year of publication, recognizing the potential influence of typographical changes over time. However, an alternative avenue for exploration is to investigate the effects of machine learning models trained exclusively on a dataset using a single typeface and subsequently evaluated on the same dataset but created with an alternative typeface. This approach would enable us to examine how much of an impact the choice of typeface has on the performance of the models. Such research will allow us to gain insights into the robustness and generalizability of machine learning models across various typographic styles.

Stable Diffusion
In future research, an alternative approach that can be explored is the manner in which images are generated. In this thesis, Stable diffusion models like DallE and Midjourney were briefly examined, but they were not considered suitable for the intended application because they were not dependable enough to display text on images. As a result, other diffusion models that allow for simple image creation based on input could be developed. For instance, an input could be ”Can you add the text: *”Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.”* on a paper from 1678?” While this aspect was not thoroughly investigated during this thesis due to time constraints, it has the potential to enhance the image generation process.

Other OCR Software
The goal of this thesis was to explore methods for artificially creating data, with a particular focus on the Tesseract OCR software. Tesseract was chosen due to its ease of use, in contrast to subscription-based options like ABBYY or Adobe Acrobat Pro DC. However, it would be worthwhile to investigate the performance of other OCR software packages, including those that excel in recognizing handwritten text like Loghi[1], and compare their results to Tesseract.

[1] https://www.nationaalarchief.nl/beleven/nieuws/unieke-transcriptie-software-nu-open-source-beschikbaar



Datasets

A.1. DBNL

The DBNL datasets contains digitized versions of the books:

- Al de volksdichten
- Algemeen Nederduitsch en Friesch Dialecticon
- Algemeen wijsgeerig, geschiedkundig en biographisch woordenboek voor vrijmetselaren
- Alle de werken
- Avondschemering
- Beatrijs en Carel ende Elegast
- Beginsels der woordvorsching
- Bijdragen en Mededeelingen van het Historisch Genootschap
- Biographisch woordenboek der Nederlanden
- De dichtwerken van Bilderdijk
- De dichtwerken van vrouwe Katharina Wilhelmina Bilderdijk
- De gierigaard
- De Hollandsche natie
- De nachtegaal en het lijstertje
- De Nederlandsche kerkgeschiedschrijver Geeraardt Brandt
- De taal- en letterbode
- De Taalgids
- De Tijdspiegel
- Der leken spieghel
- Die Dietsce Catoen
- Dietsche Warande
- Familie en kennissen
- Gezamenlijke dichtwerken
- Gideon Florensz. Deel 1
- Handleiding tot de kennis van onze vaderlandsche spreekwoorden en spreekwoordelijke zegswijzen, bijzonder aan de scheepvaart en het scheepsleven, het dierenrijk en het landleven ontleend
- Het land, in brieven
- Het leven en de uitgelezen verzen van Elizabeth Wolff-Bekker

- Histoire de la littérature flamande
- Huibert en Klaartje
- Kunstwoordenboek
- Leçons élémentaires et pratiques de langue flamande. Lecture, grammaire, lexicologie
- Lenteloveren
- Letterkundige schetsen
- Leyden ontzet, in 1574
- Los en vast. Jaargang 1870
- Madelieven
- Nieuwe winde-kelken
- Ongeloof en revolutie
- Oude Vlaemsche liederen
- Over kinderpoëzy
- Parthonopeus van Bloys
- Proeve van Bredaasch taal-eigen
- Roman van Cassamus
- Roman van Karel den Grooten en zijne XII pairs
- Roman van Moriaen
- Romantische werken
- Snippers van de schrijftafel
- Spieghel historiael
- Tooneelspelers
- Torec
- Uit het leven voor het leven
- Vaderlandsche letteroefeningen
- Van enen manne die gherne cnollen vercoopt ene goede boerde
- Van vrouwen ende van minne
- Vanden vos Reinaerde
- Verspreide en nagelaten gedichten
- Zeemans-woordenboek

A.2. Historical Newspapers

This dataset consists of a set of newspapers that has been

Years of publication/software	ABBYY 8.1	ABBYY 9.0	ABBYY 10.0	Total
1700-1882	237	0	37	275
1883-1947 (minus 1940-1945)	652	21	494	1166
1948-1995	436	11	112	559
Total	1325	32	643	2000

Table A.1: A table containing the years in which the newspapers were retrieved for the *Historical Newspapers* dataset, as retrieved from this website

A.3. IMPACT

The dataset consists of:

- 2055 book pages, ranging from 1630 until 1796 from Early Dutch Books Online[1] and Digitale Topstukken[2]
- 1024 newspaper pages, ranging from 1618 until 1885 from Delpher
- 1179 parliamentary proceedings, ranging from 1814 until 1945 from Staten Generaal Digitaal
- 205 typewritten radio bulletins from 1937, from Delpher

[1] https://www.delpher.nl/nl/boeken/results?query=digitizationProject+any+%22dpo%22&coll=boeken
[2] https://www.kb.nl/zoeken/content?f%5B0%5D=categorie%3A99

B

Hyperparameters

Training batch size	4
Validation batch size	4
Training epochs	5
Learning rate	0.00005
Random seed	42
Source max length	256
Target max length	256

Table B.1: The hyperparameters used during fine-tuning and validation

C

Delpher API

C.1. Response from Delpher Search API

Listing C.1: The output for the Delpher

```
<srw:searchRetrieveResponse xmlns:srw="http://www.loc.gov/zing/srw/" xmlns:tel="http://krait.kb.nl/coop/tel/handbook/telterms.html" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ddd="http://www.kb.nl/ddd" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dcx="http://krait.kb.nl/coop/tel/handbook/telterms.html" xmlns:facets="info:srw/extension/4/facets">
<srw:version>1.2</srw:version>
<srw:totalMilliSeconds>8</srw:totalMilliSeconds>
<srw:searchEngineMilliSeconds>6</srw:searchEngineMilliSeconds>
<srw:kbmdoMilliSeconds>-1</srw:kbmdoMilliSeconds>
<srw:numberOfRecords>122</srw:numberOfRecords>
<srw:records>
<srw:record>
<srw:recordPacking>xml</srw:recordPacking>
<srw:recordSchema>http://www.kb.nl/ddd</srw:recordSchema>
<srw:recordData>
<ddd:paperurl>http://resolver.kb.nl/resolve?urn=ddd:010633666:mpeg21</ddd:paperurl>
<ddd:accessible>1</ddd:accessible>
<ddd:yearsdigitized>1629-166X</ddd:yearsdigitized>
<ddd:metadataKey>ddd:010633666:mpeg21:a0001</ddd:metadataKey>
<ddd:papertitle>ijTdinghe uyt verscheyde quartieren</ddd:papertitle>
<ddd:edition>Dag</ddd:edition>
<dc:source>Kungl. Biblioteket Stockholm</dc:source>
<dc:title>Wt Romen den 12, April.</dc:title>
<dc:type>artikel</dc:type>
<ddd:spatialCreation>Amsterdam</ddd:spatialCreation>
<ddd:forerunner>Uut Romen</ddd:forerunner>
<ddd:pageurl>ddd:010633666:mpeg21:p001</ddd:pageurl>
<ddd:spatial>Landelijk</ddd:spatial>
<ddd:issued>1629-</ddd:issued>
<dc:identifier>http://resolver.kb.nl/resolve?urn=ddd:010633666:mpeg21:a0001:ocr</dc:identifier>
<dc:date>1620/05/02 00:00:00</dc:date>
<dcx:DelpherPublicationDate>Wed Nov 20 01:00:00 CET 2013</dcx:DelpherPublicationDate>
<zones>[{"x":252,"y":51,"w":580,"h":60, "image": "DDD_010633666_001_access.jp2"},{"x":235,"y":125,"w":759,"h":202, "image": "DDD_010633666_001_access.jp2"},{"x":53,"y":335,"w":931,"h":166, "image": "DDD_010633666_001_access.jp2"},{"x":60,"y":136,"w":167,"h":191, "image": "DDD_010633666_001_access.jp2"},{}]</zones>
<ddd:ppn>832688312</ddd:ppn>
<ddd:publisher>s.n.</ddd:publisher>
<ddd:page>1</ddd:page>
</srw:recordData>
<srw:recordPosition>1</srw:recordPosition>
</srw:record>
</srw:records>
<srw:extraResponseData>
<facets:facetedSearchParameter>
```

```
<facets:indexes/>
<facets:excludedFilters/>
<facets:sort>index</facets:sort>
</facets:facetedSearchParameter>
<facets:facetedSearchResult>
<facets:facet>
<facets:name>periode</facets:name>
<facets:values>
<facets:value frequency="122">0/17e_eeuw/</facets:value>
</facets:values>
</facets:facet>
<facets:facet>
<facets:name>type</facets:name>
<facets:values>
<facets:value frequency="1">advertentie</facets:value>
<facets:value frequency="121">artikel</facets:value>
</facets:values>
</facets:facet>
<facets:facet>
<facets:name>spatial</facets:name>
<facets:values>
<facets:value frequency="122">Landelijk</facets:value>
</facets:values>
</facets:facet>
</facets:facetedSearchResult>
</srw:extraResponseData>
<srw:echoedSearchRetrieveRequest>
<srw:version>1.2</srw:version>
<srw:query>(date within "01-01-1610 31-12-1620")</srw:query>
<srw:startRecord>1</srw:startRecord>
<srw:maximumRecords>1</srw:maximumRecords>
<srw:recordSchema>ddd</srw:recordSchema>
<srw:resultSetTTL>300</srw:resultSetTTL>
<srw:extraRequestData>
<fields>zones</fields>
<facets:facetedSearchParameter>
<facets:indexes/>
<facets:excludedFilters/>
<facets:sort>index</facets:sort>
</facets:facetedSearchParameter>
<collection>DDD_artikel</collection>
</srw:extraRequestData>
</srw:echoedSearchRetrieveRequest>
</srw:searchRetrieveResponse>
```

D

Other techniques

In addition to the techniques explored in this thesis, several alternative approaches were investigated but ultimately not incorporated into the final work. This section aims to explore these alternative approaches, highlighting the range of research conducted and providing insights into why they were not successful as originally intended.

D.1. Translation approach

One of the researched approaches involved translating a given text into a more contemporary version and then translating it back. The idea behind this approach was to filter out OCR errors by comparing the translated version with the original text. However, this approach faced a significant drawback due to the unavailability of the required dataset. The closest available dataset was the *Statenvertaling* bible, which had a more modern version called the *basisbijbel*. The *basisbijbel* was a user-friendly and modernized translation of an earlier bible version. Unfortunately, the limited vocabulary of the bible made it challenging to apply this approach to other types of texts. As a result, this approach was not utilized in the final work.

D.2. Single sentence approach

An alternative approach for the synthetic data generation step involved placing each sentence on a separate image instead of creating a single image containing all sentences. Figure D.1 illustrates an example of this approach. The construction of these images followed a similar process as described in section 5.2.1. The images were resized proportionally to accommodate the length of the sentences, meaning longer sentences resulted in longer images.

However, this approach was not pursued for a couple of reasons. Firstly, it posed scalability challenges when dealing with larger datasets containing a substantial number of sentences. Secondly, all the generated images exhibited a similar appearance, leading to a less diverse dataset. As a result, this approach was not chosen for implementation in the final work.

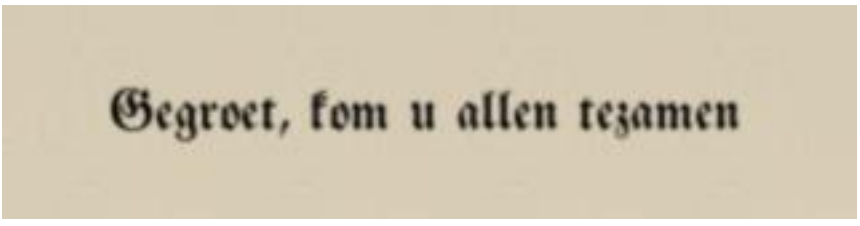


Figure D.1: A single sentence on a background, as used in of the earlier versions of the application

E

Code

All code belonging to this project can be found in my Github repository.

Bibliography

[1] Bennett Bacon et al. “An Upper Palaeolithic Proto-writing System and Phenological Calendar”. In: *Cambridge Archaeological Journal* (2023), pp. 1–19. doi: 10.1017/S0959774322000415.

[2] Thomas M. Breuel et al. “High-Performance OCR for Printed English and Fraktur Using LSTM Networks”. In: *2013 12th International Conference on Document Analysis and Recognition*. 2013, pp. 683–687. doi: 10.1109/ICDAR.2013.140.

[3] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. url: https://arxiv.org/abs/2005.14165.

[4] Rafael C. Carrasco. “An Open-Source OCR Evaluation Tool”. In: *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*. DATeCH ’14. Madrid, Spain: Association for Computing Machinery, 2014, pp. 179–184. isbn: 9781450325882. doi: 10.1145/2595188.2595221. url: https://doi-org.tudelft.idm.oclc.org/10.1145/2595188.2595221.

[5] Otto Chrons, Sami Sundell, and Microtask Bulevardi. *Digitalkoot: Making Old Archives Accessible Using Crowdsourcing. Digitalkoot: Making Old Archives Accessible Using Crowdsourcing*. Tech. rep. 2011. url: https://www.researchgate.net/publication/221604727.

[6] Hyung Won Chung et al. *Scaling Instruction-Finetuned Language Models*. 2022. doi: 10.48550/ARXIV.2210.11416. url: https://arxiv.org/abs/2210.11416.

[7] Dana Dannélls and Simon Persson. *Supervised OCR Post-Correction of Historical Swedish Texts: What Role Does the OCR System Play?* Tech. rep. 2020. url: https://github.com/tesseract-ocr/.

[8] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Tech. rep. url: https://github.com/tensorflow/tensor2tensor.

[9] Rui Dong and David Smith. “Multi-Input Attention for Unsupervised OCR Correction”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 2363–2372. doi: 10.18653/v1/P18-1220. url: https://aclanthology.org/P18-1220.

[10] G Dowding. *An introduction to the history of printing types: An illustrated summary of main stages in the development of type design from 1440 up to the present day: an aid to type face identification*. Clerkenwell: Wace, 1962.

[11] Senka Drobac and Krister Lindén. “Optical character recognition with neural networks and post-correction with finite state methods”. In: *International Journal on Document Analysis and Recognition* 23.4 (Dec. 2020), pp. 279–295. issn: 14332825. doi: 10.1007/s10032-020-00359-9.

[12] Senka Drobac et al. *OCR and post-correction of historical Finnish texts*. Tech. rep. 131. 2017, pp. 23–24.

[13] Quan Duong, Mika Hämäläinen, and Simon Hengchen. “An Unsupervised method for OCR Post-Correction and Spelling Normalisation for Finnish”. In: *CoRR* abs/2011.03502 (2020). arXiv: 2011.03502. url: https://arxiv.org/abs/2011.03502.

[14] Craig Eliason. ““Transitional” Typefaces: The History of a Typefounding Classification”. In: *Design Issues* 31.4 (Oct. 2015), pp. 30–43. issn: 0747-9360. doi: 10.1162/DESI_a_00349. eprint: https://direct.mit.edu/desi/article-pdf/31/4/30/1715512/desi_a_00349.pdf. url: https://doi.org/10.1162/DESI%5C_a%5C_00349.

[15] Guilherme T. B. et al. *Advances in Information Retrieval*. Tech. rep. 2020, pp. 102–109. url: http://www.springer.com/series/7409.

[16] Henrik Helin et al. “Optimized JPEG 2000 Compression for Efficient Storage of Histopathological Whole-Slide Images”. In: *Journal of Pathology Informatics* 9.1 (2018), p. 20. issn: 2153-3539. doi: https://doi.org/10.4103/jpi.jpi_69_17. url: https://www.sciencedirect.com/science/article/pii/S2153353922003339.

[17] Niddal H. Imam, Vassilios G. Vassilakis, and Dimitris Kolovos. “OCR post-correction for detecting adversarial text images”. In: *Journal of Information Security and Applications* 66 (May 2022). issn: 22142126. doi: 10.1016/j.jisa.2022.103170.

[18] Niddal H. Imam, Vassilios G. Vassilakis, and Dimitris Kolovos. “OCR post-correction for detecting adversarial text images”. In: *Journal of Information Security and Applications* 66 (2022), p. 103170. issn: 2214-2126. doi: https://doi.org/10.1016/j.jisa.2022.103170. url: https://www.sciencedirect.com/science/article/pii/S2214212622000552.

[19] IMPACT Project. *MPACT KB Ground-truth*. url: http://lab.kb.nl/dataset/ground-truth-impact-project.

[20] Noman Islam, Zeeshan Islam, and Nazia Noor. *A Survey on Optical Character Recognition System*. Tech. rep. 2. 2016.

[21] Shigeki Karita et al. “A Comparative Study on Transformer vs RNN in Speech Applications”. In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., Dec. 2019, pp. 449–456. isbn: 9781728103068. doi: 10.1109/ASRU46091.2019.9003750.

[22] Srinidhi Karthikeyan et al. “An OCR Post-Correction Approach Using Deep Learning for Processing Medical Reports”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 32.5 (May 2022), pp. 2574–2581. issn: 15582205. doi: 10.1109/TCSVT.2021.3087641.

[23] Bruce H. Kusko et al. “Proton milliprobe analyses of the Gutenberg Bible”. In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 3.1 (1984), pp. 689–694. issn: 0168-583X. doi: https://doi.org/10.1016/0168-583X(84)90464-6. url: https://www.sciencedirect.com/science/article/pii/0168583X84904646.

[24] Mike Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880. doi: 10.18653/v1/2020.acl-main.703. url: https://aclanthology.org/2020.acl-main.703.

[25] Ayush Maheshwari et al. “A Benchmark and Dataset for Post-OCR text correction in Sanskrit”. In: (Nov. 2022). url: http://arxiv.org/abs/2211.07980.

[26] Jiří Martínek, Ladislav Lenc, and Pavel Král. “Building an efficient OCR system for historical documents with little training data”. In: *Neural Computing and Applications* 32 (Dec. 2020). doi: 10.1007/s00521-020-04910-x.

[27] Philip B Meggs and Alston W Purvis. *”Meggs” history of graphic design*. 6th ed. May 2016.

[28] Nauta D. et al. “De Statenvertaling 350 jaar”. In: (2005).

[29] Thi Tuyet Hai Nguyen et al. “Neural machine translation with bert for post-ocr error detection and correction”. In: *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*. Institute of Electrical and Electronics Engineers Inc., Aug. 2020, pp. 333–336. isbn: 9781450375856. doi: 10.1145/3383583.3398605.

[30] Thi Tuyet Hai Nguyen et al. *Survey of Post-OCR Processing Approaches*. July 2021. doi: 10.1145/3453476.

[31] Maciej Ogrodniczuk. “Fine-Tuning OCR Error Detection and Correction in a Polish Corpus of Scientific Abstracts”. In: *Communications in Computer and Information Science*. Vol. 1716 CCIS. Springer Science and Business Media Deutschland GmbH, 2022, pp. 450–461. isbn: 9789811982330. doi: 10.1007/978-981-19-8234-7{_}35.

[32] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: (Oct. 2019). url: http://arxiv.org/abs/1910.10683.

[33] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. url: http://jmlr.org/papers/v21/20-074.html.

[34] Martin Reynaert. “Non-interactive OCR post-correction for giga-scale digitization projects”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 4919 LNCS. 2008, pp. 617–630. isbn: 354078134X. doi: 10.1007/978-3-540-78135-6{_}53.

[35] Christophe Rigaud et al. “ICDAR 2019 competition on post-OCR text correction”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. IEEE Computer Society, Sept. 2019, pp. 1588–1593. isbn: 9781728128610. doi: 10.1109/ICDAR.2019.00255.

[36] Rohit Saluja et al. “Error Detection and Corrections in Indic OCR Using LSTMs”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. Vol. 1. IEEE Computer Society, July 2017, pp. 17–22. isbn: 9781538635865. doi: 10.1109/ICDAR.2017.13.

[37] Omri Suissa, Avshalom Elmalech, and Maayan Zhitomirsky-Geffet. “Toward the optimized crowdsourcing strategy for OCR post-correction”. In: *Aslib Journal of Information Management* 72.2 (Apr. 2020), pp. 179–197. issn: 20503814. doi: 10.1108/AJIM-07-2019-0189.

[38] J. Tselentis et al. *Typography, Referenced: A Comprehensive Visual Guide to the Language, History, and Practice of Typography*. Rockport Publishers, 2012. isbn: 9781610582056. url: https://books.google.nl/books?id=2-W4vxVM-8gC.

[39] Robartus Johannes van der Spek. “De Statenvertaling van de Bijbel en het Oude Nabije Oosten”. Dutch. In: *Phoenix* 65.2 (Dec. 2019), pp. 3–5. issn: 0031-8329.

[40] Ashish Vaswani et al. “Attention Is All You Need”. In: (June 2017). url: http://arxiv.org/abs/1706.03762.

[41] L. Wilms, R. Nijssen, and T. Koster. *Historical newspaper OCR ground-truth data set. KB Lab: The Hague*. 2020.