UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Data Science Curriculum

Mattias Väli

Optical Character Recognition of Estonian Fraktur Script

Master's Thesis (15 ECTS)

Supervisor(s): Aleksei Dorkin, MSc
Kairit Sirts, PhD

Tartu 2025

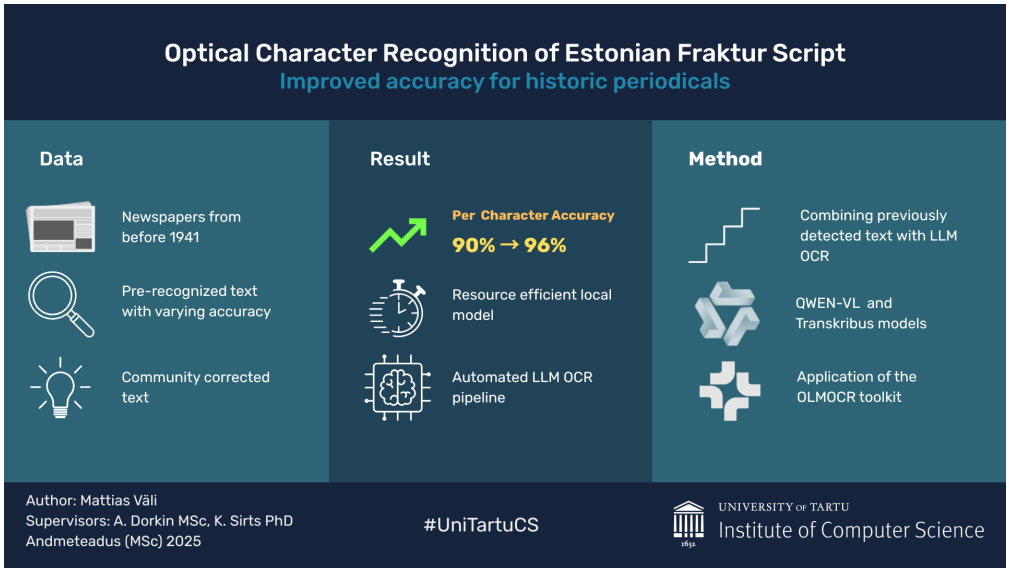
Optical Character Recognition of Estonian Fraktur Script

Abstract:

The DIGAR portal of the National Library of Estonia hosts a diverse collection of historical Estonian newspapers. This publicly accessible dataset provides valuable resources for historians and other researchers, supporting a wide range of scholarly inquiries. For example, it can be used to investigate contemporary public opinion, trace the activities of individuals, and document historical locations. The National Library of Estonia's Digilab also supplies machine-recognized text; however, recognition accuracy is often limited, particularly for older newspapers and publications printed in Fraktur script.

This study focuses on newspapers published prior to 1944, many of which are regional titles characterized by lower print quality and more limited circulation. The primary objective is to enhance the accuracy of existing machine-recognized text corpora by leveraging state-of-the-art text recognition technologies. Specifically, the project employs advanced models from the Qwen2.5-VL family alongside the Transkribus platform. The proposed framework enables efficient and traceable local processing of data retrieved from the digital archive with predefined storage architecture. The resulting cleaned datasets are prepared for downstream processing on other platforms, and accompanying code is provided to facilitate model training. The data, models, and the associated code base are freely available in Huggingface, Transkribus and Github.

Visual Abstract:



Keywords: OCR, LLM, VLM, Transkribus, OLMOCR, QWEN, DIGAR, Digital Archive, Fraktur

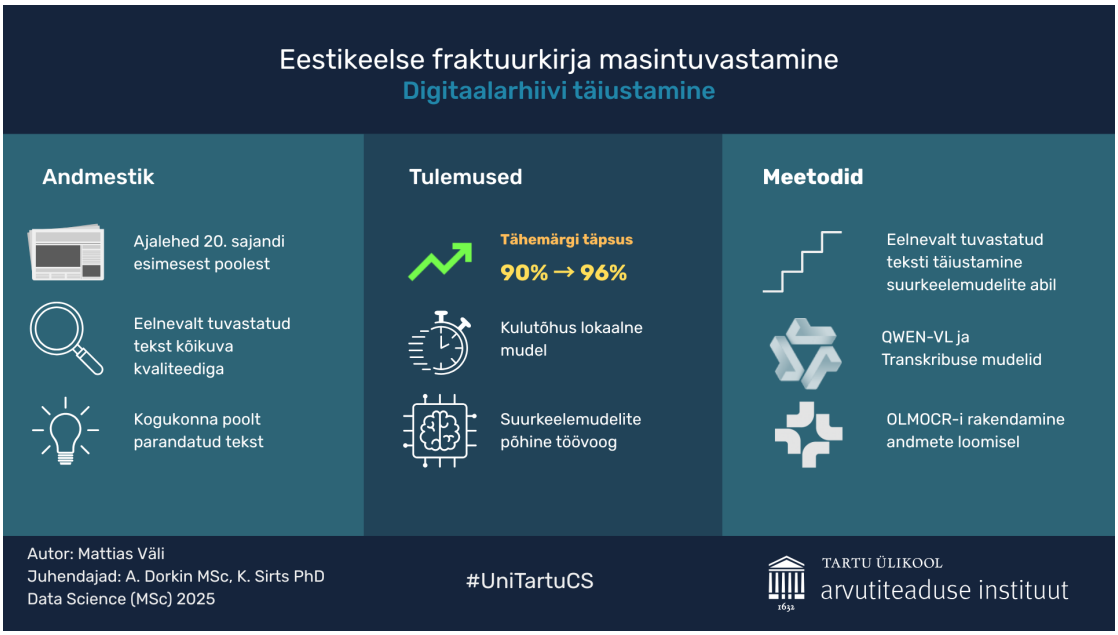
CERCS: P175 Informatics, systems theory, P176 Artificial Intelligence

Eestikeelse fraktuurkirja masintuvastamine

Lühikokkuvõte:

Eesti Rahvusraamatukogu DIGARi portaal on Eesti trükiste digitaalne andmebaas, mis sisaldab pea kogu Eesti trükiperioodikat. See andmestik on avalikult kättesaadav ning võimaldab ajaloolastel ja teistel huvilistel rakendada seda erinevateks uurimustöödeks. Näiteks saab seda kasutada omaagse meelsuse, üksikisikute kui ka asukohtade uurimiseks. Rahvusraamatukogu Digilabor pakub ka masintuvastatud teksti, kuid tuvastatud tekst pole alati täpne, seda eriti vanemate ja fraktuuri kirjapildis perioodika puhul. Antud uurimustöö keskendub ajalehtedele, mis on välja antud enne 1944. aastat. Mitmed vaadeldavad ajalehed on maakonna ajalehed, mille trükikvaliteet on madalam ja levik väiksem. Antud töö eesmärk on otsida võimalusi, kuidas teha täpsemaks seni masintuvastatud tekstikorpusi kasutades modernseid tekstituvastuslahendusi. Selleks kasutatakse Qwen2.5-VL-Instrukt ja Transkribuse platvormi täiustatud mudeleid. Pakutud raamistik võimaldab andmeid digiarhiivist sisse lugeda, neid tõhusalt ja jälgitavalt töödelda ning talletada. Puhastatud andmed on valmis edasiseks kasutamiseks teistel platvormidel ning on loodud ka kood mudelite treenimiseks. Andmed, mudelid ja kood on vabalt kättesaadavad Hugging Face'i, Transkribuse ja GitHubi platvormidel.

Visuaalne kokkuvõte:



Võtmesõnad: Tekstituvastus, Fraktuurkiri, Visuaalsed keelemudelid , Transkribus, OL-

MOCR, QWEN, DIGAR, Digiarhiiv

CERCS: P175 Informaatika, süsteemiteooria, P176 Tehisintellekt

Contents

1 Introduction **7**

2 State of the Art **10**

3 Data **13**

4 Methodology **16**
4.1 Data Modeling and Flow 16
4.2 Large Language Model Use for OCR 19
4.3 Transkribus 20
4.3.1 Platform overview 20
4.3.2 Full page solution 21
4.3.3 Block-based solution 21
4.4 OLMOCR framework 23
4.5 QWEN-2.5-VL 24
4.5.1 Base model 24
4.5.2 Fine-tuning QWEN-2.5-VL 24

5 Results **26**
5.1 Metrics 26
5.2 Transkribus 27
5.2.1 Full-page results 27
5.2.2 Block-based results 28
5.3 QWEN fine-tuned model 28
5.4 LLM 31

6 Future Research Directions **33**

7 Conclusion **34**

References **37**

Appendix **38**

A Repositories **38**

B List of used newspaper with dates **38**

C Document-Anchoring Prompt **40**
II. Licence 41

1 Introduction

Optical Character Recognition (OCR) is used to detect text and other elements from images. These could be, for example, business documents, handwritten letters, or, in the case of this thesis, scanned historical periodicals. The detected text becomes machine-readable and, with enough accuracy, allows the end user to find, manipulate, and categorize this data for their target purpose. This can significantly speed up the research process. Removing constraints such as the user's limitations in working with uncommon scripts, difficult-to-translate words, or hard-to-understand handwritten notes makes the data more accessible. The end user in this case could be academic researchers as well as individuals interested in local or family history.

While there is continuous improvement in the OCR field, these new techniques are not universally applied, and performance metrics may be too focused on specific areas and types of text. OCR efficiency can also be heavily impacted by how texts vary by the writing script used, image quality and type. Much focus is given to performance metrics in English, Chinese or other very widely used languages, while there is less focus on smaller language groups.

The focus of this paper is on OCR of text in Estonian, especially written in the Fraktur script, an archaic script used in Estonia until the Second World War. This data creates challenges due to the small corpora of Estonian on the Internet, especially for texts using archaic spelling. An even smaller amount of text can be found that is properly validated and marked as accurate by end users. It can be challenging for even large language models to properly deal with these issues, especially if some of the wording can be almost totally unseen by their training data.

The dataset for this study was sourced from the publicly accessible digital archive of the National Library of Estonia DIGAR. This corpus is composed of digitized documents that contain an initial layer of text generated via OCR, which exhibits varying quality. Crucially, the dataset also includes human-annotated corrections provided by a community of registered users who can edit or validate text segments through the archive's interface. These community-verified texts, noted for their high quality, serve as the ground truth.

To explore the error-correction capabilities of modern solutions, the initial OCR output and corresponding document images were processed using the APIs of leading Large Language Models (LLMs), namely Google's Gemini-2.5-pro and OpenAI's GPT-4o. The objective is to leverage these LLM-generated corrections as a high-quality baseline to train and evaluate a new, specialized model designed to rectify errors in the original OCR output.

To improve the quality of the available text, this thesis uses Transkribus and parts of the OLMOCR framework [LLH+24]. The former is a European Union-backed project aimed at historical documents with their own platform. OLMOCR is a framework created by the Allen Institute for Artificial Intelligence and is an open-source tool designed to

create a plain text representation of whatever documents are given to it. Neither project is focused on providing an Estonian-based solution. Transkribus has larger models that work in multiple European languages and OLMOCR is focused on English with their model training. The underlying model of OLMOCR and one of the model used for baseline training is Qwen2.5-VL-3B-Instruct [BCL+25].

The fine-tuned models were trained on a single page of a newspaper or a block-based dataset. These blocks come from the initial data available through DIGAR and generally denote a headline, article, or other elements. For larger pieces of text, like articles, they could be divided into multiple blocks. The same block definition is given to small fragments, such as page numbers and other basic metadata. An example of how the blocks are divided can be seen in Figure 1.

Figure 1. Blocks
Blocks can be headings, subheadings , articles and parts of articles.

The goal of this thesis was to try and create a reusable framework which can be applied to the digitized periodicals. To achieve that, different solutions were observed that can handle our dataset of the Estonian Fraktur script, a more obscure script of a smaller language. It was necessary to check if it is viable to use only the already available

community-corrected text when comparing it with an application like Transkribus. This paper also tries to answer the question if it is viable to create a baseline dataset using large language models and is the final dataset robust enough that models can be fine-tuned with it. The dataset, scripts, and models were, as much as possible, made available for further use.

This thesis is divided into four main chapters. First, the current state of the art demonstrates what should be achievable. Next, the data is described discussing how the images and already detected text plays a role in training new models and using LLMs. The methodology and results of test data creation is described in the following chapter. Finally, the results of different models are compared with each other after the performance metrics have been calculated. A final comparison is made between the different solutions that were used to improve the quality of the text.

Appendix A contains the repositories where the code and data produced for this thesis are stored. Appendix B is a list of the papers used and their date of publication. Appendix C contains the prompts used for document-anchoring.

LLMs such as GPT-4o and Gemini-2.5-Pro were utilized to process and improve the phrasing of the text in this thesis. These models, along with Microsoft Copilot, also assisted with writing and debugging the code.

2 State of the Art

Automated Text Recognition (ATR) is of interest to a diverse group of researchers from many different fields. It seemed logical to start the research from papers and projects which specifically deal with historic artifacts, such as the European Union-funded NewsEye project which similarly focused on newspapers [Com20]. The goal of this project was to continue digitizing on the part of archives and national libraries in different European countries. Digitization can be an ongoing process that requires additional work to improve the quality of the detected text. The main contributors to this project came from Germany, Austria, France and Finland. Papers are published based on parts of the projects done on different languages.

Their approach, as detailed in the paper on Automatic Text Recognition for the German language [ML20], is based on Connectionist Temporal Classification (CTC) and Sequence-to-Sequence (Seq2Seq) models. Both models combine convolutional neural networks (CNNs) with recurrent layers (BLSTM) and are trained end-to-end. The CTC model uses a convolutional-BLSTM stack, while the Seq2Seq model follows an encoder-decoder structure with attention mechanisms. The two approaches yield broadly similar results, with the CTC being slightly faster to train and use. The Austrian National Libraries newspaper dataset is used, consisting of 232 pages and 88786 lines, to reach a Character Error Rate (CER) of below 1%. The final result is also able to generalize on different inputs, but for this, the languages must not differ significantly. As the model is trained on the German language, it cannot properly work on text from Arabic newspaper, as an example.

A successor and a participant of projects like this is Transkribus, a platform which provides transformer-based super models for wider use and Pylaia based models that can be fine tuned by users [Traa]. This site contains many ready-to-be-used models and has free demos for them, but text recognition is only free up to a certain amount of documents. The platform allows for publishing and sharing of the trained models; as of July 2025 there is a single model available that is trained on Estonian text.

To address the multitude of different fonts present in historic prints, a 2023 paper explores the use of specific models for different scripts [Chr]. It presents two approaches in which one tries to detect only a single font to focus on and another which is able to deal with multiple fonts at the same time. The research recommends using the specific font model unless there are frequent changes in the script of the text. This solution is based on text lines; it is a CTC-based Convolutional Recurrent Neural Networks model. Their character error rate was around 1% for texts where there were no multiple different fonts.

Large vision-language models (LVLM) have allowed the combination of language processing with visual perception. They typically consist of a visual encoder, a cross-modal projector, and a LLM [BCL+25]. Visual language models such as those from Alibaba Group's Qwen team allow for efficient scanning of documents. It is able to

detect bounding boxes and parse text with good accuracy, besides being able to also parse other forms of input. It handles inputs of different sizes and allows for scanning of tables and images. The conversational format of the model allows for it to describe objects that couldn't be detected by text detection alone. The wide range of model sizes available make it ideal for this project, as they allow for exploration of what works without the need for more costly hardware. These QWEN models are the same that the OLMOCR framework uses and which they fine-tune to their data. The data used to train these models is mostly in English and Chinese, but other languages are also present. The combination of such models with training data generated by larger models such as provided by Google and OpenAI allows a pretty seamless multiplication of training data. Larger QWEN models should also be able to compete with the larger models from the above mentioned companies.

Another way to look at the chosen approach is to view it as OCR text correction. The use of LLMs on Finnish, a language that is related to Estonian although with a larger amount of users, for OCR correction has been studied in a 2025 paper[KLKG25]. The result shows improvements as of using GPT-4o in English papers, but the performance was poor for Finnish. This was especially true without additional normalization, such as changing certain characters to others due to their historical spelling differences. For example, the normalization of the old spelling of the character "v" for "w", which is also applicable to Estonian papers.

Research in historical document analysis has placed a significant focus on German-language manuscripts and printed works. A recent study by Greif et al. (2025) evaluated the efficacy of LLM models for processing German documents from the 18th and 19th centuries. Their findings demonstrate the high efficiency of Gemini-2.0-Flash, which achieved a Character Error Rate (CER) of 1.27% in preprocessed text, in contrast to GPT-4o which yielded a CER of 6.31%. The study achieved its optimal result, a CER below 1%, using a hybrid methodology. This approach used the Transkribus Print M1 model for initial OCR transcription (which has a standalone error rate of under 4%) and subsequently employed Gemini-2.0-Flash for post-correction. The lowest reported Word Error Rate (WER) from this process was approximately 5.5% [GGG25].

As it is an important topic for the usability of character recognition, it is worth to point out the same paper's results on named entity recognition. With strict matching of strings, they are able to produce a 74.92% accuracy on all of their different documents with Gemini-2.0-Flash using noisy OCR data. But on the same data they are able to produce an accuracy of 88.61% when accounting for changes that do not interfere with the user interfacing with the data. This means that extra periods are not read in as mistakes and neither are spelling differences which won't confuse researchers dealing with the data.

The viability of many computational methods in historical research is directly contingent upon the quality of Optical Character Recognition (OCR). Quantitative analyses,

such as tracking vocabulary shifts over time [HRMT21] or linking entities across vast newspaper collections [BLPCD$^+$20], are only reliable if the underlying textual data is accurate. Therefore, while the pursuit of more precise OCR technologies is essential, future progress must also prioritize inclusiveness. This includes extending state-of-the-art methods to smaller and less-resourced languages and adapting them to handle diachronic linguistic complexities, such as non-standardized archaic spellings.

3 Data

For this project, digitalized regional newspapers from the National Library, from the period between 1919–1944 have been used. These periodicals come from different parts of Estonia and vary in physical size and release frequency. Some smaller newspapers were printed weekly while larger ones were daily. Most papers are standard news outlets, while Eesti Kirik is more focused on what is going on in the Estonian Evangelical Lutheran Church.

The quality of the digitalized pictures of the accessible periodicals varies, but overall the quality is adequate. Some newspapers have defects that might influence the overall results, such as smudges. The printing quality seems otherwise pretty uniform. A large issue is also the differing layout of the pages that can vary in size and presentation. For example, article headlines, advertisements, and dating ads can appear in different boxes and with irregular placements. Images, tables, and crossword puzzles are also present. The newspapers used for the current training data have a white background, but other papers can have a slightly tinted background.

The final amount of data collected can be seen in Table 1. A list of newspapers used can be seen in Appendix B The data from Table 1 is the data that makes it to the output layer of the final storage and will be given to the model to train on. The final amount of newspaper pages is 245. The total text length is 1,495,435 characters, and the final number of blocks is 3666. While examining the data, a much larger amount of rows is visible for the community corrected blocks, but most of these either don't have actual text data or contain a single number. As they are void or almost empty of any data, they are not used.

During data preparation, some smaller blocks—such as those containing only page numbers—were removed from the text flow, as they were not relevant for training and therefore did not need to be sent to GPT-4o. Instead, the focus was on larger blocks of article text. Articles, for which an LLM was used for ground truth, could also be cut because they did not respond or were against the model's guidelines. For example, GPT-4o in multiple cases found that the text given was trying to jailbreak it. Some blocks would also be cut by the data loader if the number of tokens was considered too high for the GPU used.

Type	Amount
Pages	245
Characters	1495435
Blocks	3666

Table 1. Amount of gold layer data

For all of the observed papers, there are also accompanying recognized texts. The quality varies a lot paper by paper and some parts of the texts are misclassified; for example, letters are taken to be punctuation. But this text is the baseline against which we can compare and from which test data can be produced. Several articles, such as government announcements, were printed almost identically in multiple papers simultaneously. There was an assumption that extra data could be gathered in such a way. But as these announcements could vary by date of publication, text length or layout, this route was not pursued.

As the language used in the newspapers is more archaic, there are mismatches in how certain words are spelled and differences in form. For example, the modern suffix of "lik" was spelled "line". The use of "w" as opposed to "v" is also prevalent. From a usability point of view, it doesn't matter as much if the models produce either variation of the spelling as long as it follows an uniform path that can later be smoothed with post-processing. This means that for calculating error rates, these "w" and "v" spelling mistakes can be set aside. However, hallucinations of completely different spellings of words are to be counted as mistakes. This is a factor to be aware of when using language models that are trained on a modern corpus.

The data, as presented on the National Library website, has bounding boxes to identify where the text is located. This is a useful tool as the quality of this placement is accurate for the observed cases. This means that these coordinates can be given alongside the text and that would give models extra information as to what is actually on the image. This data is structured in an Alto XML format and so needs to be modified for further processing. The frameworks and methods used have their own requirements for what their input should look like. The bounding boxes and their ordering made sense from observation and could be read in a logical order. These boxes are usually grouped by articles, but there are many cases where the header is in a separate box or an article is divided between multiple group ids. This creates an issue when trying to link different detected text parts together with full page processing. If the order of the next article is not in sync or there is a difference in what is considered an article, word error rates and character error rates are more difficult to define. A solution to this problem was to treat each article as a separate entity and not focus on the whole newspaper page at once.

To increase the amount of available data, additional data was gathered using crawling. Variations of these scripts are available in the accompanying Github repository. The received XML files had flags for community corrected data. These flags indicated blocks that could be used as the ground truth. On further inspection, some articles were falsely flagged as they still considered typos, but overall this is a valuable input for the creation of test data and for validation.

PDFs can contain text that can be retrieved with tools such as Pypdf [FSpz+24]. Since the Digital Archives website already provides detected text when retrieving data, this functionality was not applied directly to all newspapers. For community-corrected

text, the text available in Pypdf was not updated and appeared to be of lower quality; it appears to be the original text when the image was scanned. This was used as a base to create the ground truth data. It gave us a larger corpus of text where one could see the original automatically generated text and the text that someone had corrected or confirmed.

Not all of the papers available have PDFs related to them. Usually, the web interface allows for the download of images and presents a button available in the browser. When contacting the Digital Archive team, they were also not able to provide images for certain newspapers like "Elu : Võru-Valga-Petserimaa töörahva häälekandja". To obtain these images scanning of snippets from the pages and reconstructing the images from fragments was possible.

4 Methodology

4.1 Data Modeling and Flow

As one of the goals of this project is to create a framework which can be reused and improved upon, an initial flow and data storage is proposed. The code is available in the Github repository [Vä25]. The dataset collected is available on Huggingface. The framework is able to first gather the data from the National Library's site and store the data as PDF files for images and XMLs. Data is also stored in a DuckDB database. Data is processed through local scripts and APIs such as the ones for Gemini and GPT. The high-level view of the flow can be seen in Figure 2.

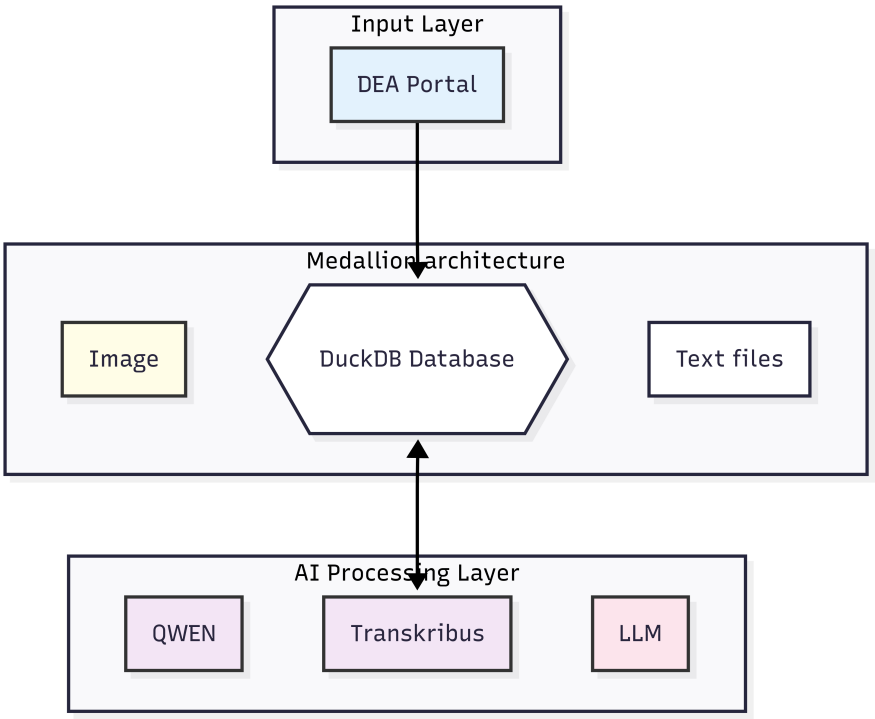


Figure 2. Database diagram

Data for this project was provided by the National Library's Digital Archive team and was given in an Alto XML format with accompanying PDFs. In addition, extra data is available and attainable from the website. PDFs of the newspaper and full text are easy to obtain for most papers. Block-specific data requires a user to be sourced. The Git repository also contains scripts to stitch files together as some newspapers do not have easily available PDF files [Vä25]. The images can be scanned piece by piece and combined to obtain the final PDF.

A variation of the Medallion architecture was chosen for file storage, which is a common pattern for data lakehouses. This architecture organizes data into three distinct layers: a bronze layer for raw, ingested data; a silver layer for augmented data; and a gold layer for aggregated, consumer-level data. The term "Medallion architecture" was coined by Soham Bhatt and Deepak Sekar in an article published by Databricks, which is a key reference for this approach [Dat22].

This architecture was adopted to simplify data lineage and debugging, even though it wasn't a strict requirement for this project. Given the project's small scale, some text data was also stored in a separate database. Data lineage was managed by storing timestamps of changes and including columns that link each row to its origin, a practice reflected in the final storage on Huggingface.

The initial files are stored in the initial bronze subdirectory with an additional subdirectory for each edition of a newspaper. These are then split into a silver layer subdirectory per page. As initial attempts at using LLMs on full page text gave null values for some newspaper editions, with no explanation about the type of errors, then the pages were split and stored in such a manner. Each of these page subdirectories has, based on the layer they are in, additional subdirectories for the image files. This might include the full-page PDF, full-page JPG files and images created per article block. Additional subdirectories include text, JSON and Transkribus files. All of the non-image files are read into the database as well when needed. The gold images contain all of the information needed to move the data for processing, such as the Tartu University High Performance Computing (HPC) cluster.

The data is then parsed to suit the different inputs that frameworks require. The full text is extracted with metadata on how many blocks per page have a community-marked correctness tag. The XML is converted to another XML format for Transkribus, which uses PageXML. A JSON file is created from the full text with coordinates per row to be used for OLMOCR framework. This data can be used to create prompts for LLMs. Almost all of the text processing also ends up in the final database.

The solution in place was first tried with images from a full page of text and texts to accompany them. This is reflected in the storage. As that route seemed to be hampered by the complexity of the pages in question, it was reflected in early result, which showed a higher error rate than expected, a block-based approach was added. This means that all storages had to account for different type of inputs and database tables got an extra column to differentiate blocks and data, such as the text, from a full newspaper page.

A separate workflow was created for newspapers and blocks that had community-corrected values present in their original source. To use these for training and validation the corrected text gathered was used as the target and the text in the PDFs, which seemed to be the original detected text, was queried out using a script based on a similar script from the OLMOCR framework. This text alone constitutes the bulk of the available data for training.

The main tool used for databases has been DuckDB as it is free, easy to deploy, and use. The main storage for tables is in parquet files, and the storage remains separate from the Medallion architecture type for the files themselves. First, the state of table is queried to understand what data is available and how much of it is already validated. The information per block in a page is then extracted and stored as a separate table. The Block_Id in this case comes from the initial data. The extracted full text is stored in a separate table with each of the new sources and types also used as the primary index. In case of the text being detected for example by the gemini-2.5-flash model, the Source will be the models name and the type of prompt used will be stored under the Type column. Different combinations of the same basic element are used to differentiate between block-based data, which has block id stored in the block column. The texts that have the full text from a page in the newspaper are denoted by the string 'All'.

Parquet tables are created to store the final information that will be used to create the final datasets for training. Previously gathered text is aggregated per block and page as needed and stored in a way that they can be converted into a Hugging Face Datasets element. Data is split to train, test, and validation.

The images had to be converted from PDF to JPG for processing. Those JPG files additionally had to be split to account for block-based training. All of the coordinates were scaled accordingly, and the coordinate system was reduced to only the x and y axis coordinates for block-based data. This was done due to the way that the OLMOCR anchoring logic works, as it produces a final value without width and height.

A separate validation table is created to store the different types of validation. The initial version consists of a different column for each metric. With columns for Character error rate and Word error Rate (WER). A more comprehensive description will be given in the results chapter. The database schema can be seen in Figure 3.

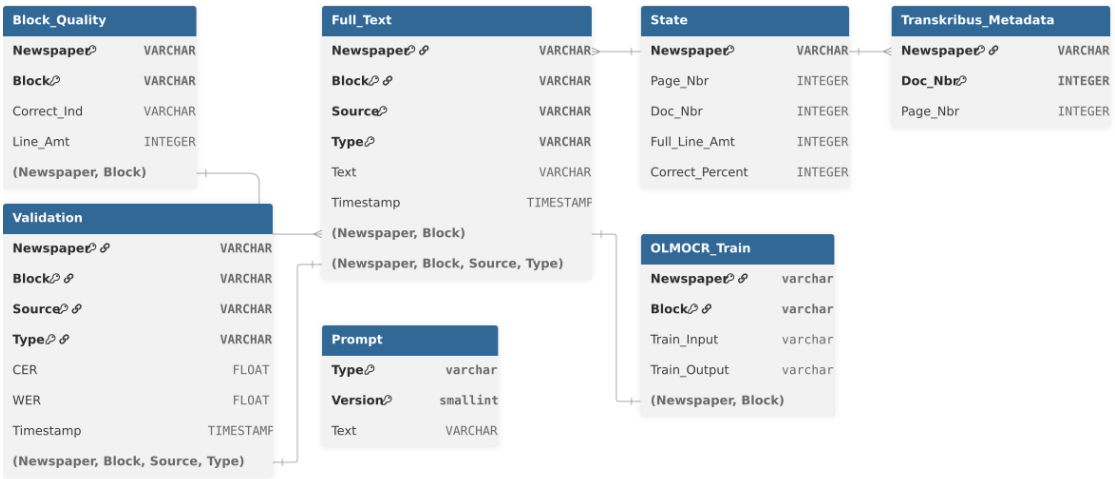


Figure 3. Database diagram

To interact with the Transkribus REST API a separate table is created to store their document and page numbers assigned to our metadata. Separate scripts are used to check the available documents and to see how the documents are assigned. The loading the images to their servers required the usage of the website. To make this easy, as soon as the data needed was collected, for example, for block-based data all correct blocks are identified and stored in block_quality table, the images are gathered to a single folder and then uploaded all at once. This solution creates a single document with n number of pages. The pages keep their original names, and after pinging the server this gives us an idea of which image now needs to have which page of XML uploaded for it. This part of the flow could be more automated, but as the intended use case is to preprocess data all at once and then use it for training, the current system shouldn't cause larger stoppages. The API also changes during 2025 [Trab].

For training the QWEN2.5-VL-Instruct model [Qwe24], the HPC of the University of Tartu was used. Data stored in the servers correspond to the LLM detected text from Full_Text table. Image files are also transferred from local storage to the cloud for model training. Scripts are available in the Git repository in the model training directory [Vä25].

4.2 Large Language Model Use for OCR

At the time of writing this project Gemini has a significant free to use tier for their API. To enhance the data that was given and to test out the capabilities of LLMs in identifying text from an image, this was extensively used. Some additional tokens were used through a paid tier. This was done alongside ChatGPT using an API key provided by Tartu NLP research group.

The main usage was similar to how the OLMOCR framework proposes to use language models. The data given to it is in a format of a JSON file with text and the location of that text on the image. The coordinates are given as only x and y while ommiting height and width, as that is how they are returned by the OLMOCR repositories script for the original pypdf solution. As the images provided are all in Estonian and are not flipped or at an angle, these extra fields were not given as opposed to the original format.

The initial prompt used was an almost unchanged version of what could be seen in the OLMOCR paper, the only exception being some modification of the [0,0] coordinates and a specification of data type to match our data. This query can be seen in the appendix C and is referred to in the tables as type original_olmocr.

When using the Gemini API and targeting blocks instead of pages, the main restriction became the number of articles that could be gone through as the free tier restricts the use of Gemini-2.5-pro to only a hundred requests. Gemini-2.5-flash seemed to just return the input text instead of meaningfully modifying it, so it was mostly left aside. Using full-page text for blocks seemed challenging, as assigning the rows to specific areas

seemed easy to make error-prone. As this limitation was not really a factor with paid tiers and GPT-4o, articles were queried out separately.

GPT-4o did have an issue that it did not return responses for some content. For Gemini no such errors were observed, although no real check was done for the cases where the response was just missing. Gemini did return null values with its responses, but this was allowed according to the prompt.

While providing the text and the images in the same format as in the OLMOCR paper, for 1459 queries the final token count use reported by the Google AI Studio was 675 600. The total Gemini-2.5-Pro cost for this projects paid tier is reported in the console to be a total of 38.36 euros. The average amount of blocks per page for our dataset, that have text in them that is longer than 1 character, is 35. The cost does not take into account the use of GPT-4o use or the use of Gemini-2.5-Pro through the free tier and so does not cover the full cost of using LLMs for this thesis.

4.3 Transkribus

4.3.1 Platform overview

Transkribus is a platform that started in 2013 with the support of European Union funded projects [Trab]. It has an application programming interface (API) and a web browser user interface(UI) which can be validated with a registrable user. It allows for free model training and data storage, at least for the extent of data used on this project. From a usability perspective there weren't any longer queues and data could also be modified by the web UI.

The models that are available for training in Transkribus are based on Pylaia [CKC25], an open source tool for line-based text recognition. It relies on the CTC loss function and makes decisions based on the shapes of the letters. It also relies on the data that it gets from lines and doesn't have the larger context that an LLM might have [TSGL+24].

For Transkribus a few different base models were tried out first to try and gauge their effectiveness. These models are freely accessible and are divided into two categories: Super models and the rest. Super models cannot be fine tuned on your data as they are rather large according to the documentation of Transkribus. Models have demos available, and one can upload a single image to understand their accuracy. Most of the models tried had large amount of errors for the full text, both super models and other models. For the block-based approach, the already present models were much more accurate, but still returned some errors.

Using the already present Transkribus models outside of the demos requires credits; a small amount of these credits is given each month, but extra credits require a subscription or are purchasable separately.

4.3.2 Full page solution

Attempts were made with both, the Transkribus Print M1 model, the continuously updated base model, and with the Danish Newspapers 1750-1850 [HD] model. The final fine-tuned model with the highest achieved accuracy was based on the Danish Newspapers 1750-1850 model. The data provided to it came from the community-corrected newspapers where the confirmed number of correct lines per page was greater than 50%. The validation data was selected to have a higher percentage as initial test placed great importance on the need to have high quality. Most of the parameters can't be modified, except for a few ones like the number of epochs and the proportion of the validation data. The final amount of data available to the final model with parameters:

- Training pages: 50
- Validation pages: 5
- Number of words : 102 003
- Number of lines 19 037
- Max number of epochs : 100
- Batch size : 24
- Learning rate : 3.0E-4
- RNN Type : LSTM
- RNN Layers: 3
- RNN Units: 256

4.3.3 Block-based solution

A block, as defined in the data gathered from the National Library's website, means separate block of text, such as those of titles of articles, images or parts of articles. A single example of a block can be seen in Figure 4. For this example nothing else remains on the image except for a single column of text.

For this approach, the model was fine-tuned on these blocks alone and not the full picture of the newspaper page. This theoretically reduces the errors that arise from understanding where the text is. It eliminates complex paragraph localization and images. The data gathered for this originates from the community-corrected text, but also from where the Gemini-2.5-Pro and GPT-4o models returned less than 5% difference from the original text. This means that if a model has a difference of only 1% from the original text gathered from the National Library then the original archive text was used.


22. juulil 1634 a. oli kiriku wiſitatſioon, kus ſõja läbi äralõhutud kirikuſt ainult neli ſeina olid järele jäänud, milliſed prügiga kaetud. Kirikus olid kaks kõlbliku pinki leida, mis lajeb oletada, et ſeal ſeinte wahel ka jumalateenistuſt peeti. Seinad olnud kõwadeſt teliskiwideſt tehtud ja muidu heas korras. Õpetaja maja puudunud tol korral täiesti. Rannu mõiſa wolinik kammerhärra andis kiriku ametnikule Grasſile käſu, et ſee pidi kiriku parandamiſe tarwis materjali oſtma. Selleks otstarbeks osteti Tarwaſtuſt mituſada palki ja weeti nad talwel üle järwe kohale, nii et ſuwel wõiſid kiriku parandamiſe tööd algada. Rannu kogudus awaldas weel ſiinjuures ſoowi, et Puhja kogudus temal awitaks kirikut üles ehitada, ſeſt et nad warem Puhja koguduſel kiriku ehitamiſe juures abiks olid olnud. Rannu koguduſe ſoow täideti, nii et Puhja kogudus aitas neil kiriku uuesti üles ehitada. Ülejäänud materjal weeti Rõngu kiriku juure. Eſialgne kirik on olnud wõlwitud laega, mis aga ſõja läbi rikuti ja ſiſſe langes. Pärastine kirik olnud laudadeſt lae ja katuſega kaetud. Niiſama ſaid ka ukſed ja aknad endiſelt korda ſeatud. 24. weebr. 1669 a. kiriku wiſitatſioonil tuli ilmſiks, et kirikul weel mõned puuduſed olid, mis

Figure 4. A single block

To try and gauge the differences between the two models, their accuracy rating for the original text was compared against the community-corrected one. Of the 350 examples, a difference between one model returning a text with under a 5% difference and the other over the 5% happened 40 times in total. The differences seemed to come from one of the LLMs adding lines, reading the half cut-off text from the top of the snippet, and not returning the full text. Based on the examples observed and the overall accuracy of these models, this additional way of adding extra data to the original validated was used.

A total of 1644 blocks were given for training. This totaled 114,159 words. Other parameters remained the same as for the full-page attempt. The input files had bounding boxes for each line which were also given for training, they did seem slightly tall for some instances, however, as this wasn't uniform and minimal changes were already done, this was not further modified. For this training, the Transkribus Print M1 model was used as a baseline. The few demo examples that were tried showed its advantage over other models such as Danish Newspapers 1750-1850 model that was used for the full-page solution.

4.4 OLMOCR framework

OLMOCR is a framework with a 7 billion parameter model on Huggingface[All24]. It provides a free to use demo version which indicated that the framework has a working baseline for dealing with our data. The logic behind the framework using the metadata gathered from PDFs in a way that is called document-anchoring. The text and location of the text is gathered from PDSs using packages such as pypdf, something that is quite effective when it comes from files that were created and not scanned in, and then providing it to LLMs. Their output is further used to train their model with a base of a visual language model(VLM), at the time of writing that means using the Qwen2-VL-7B-Instruct checkpoint. As the 7b parameter model needs larger cards then were freely available at the time of writing , the Qwen2.5-VL-3B-Instruct model was used to test the framework and train the final model.

According to the paper, the use of document-anchoring allows for a very efficient and low-cost way of detecting text and also enables to understand tables, graphs and mathematical notation. The training data is based on English only, but as the framework uses LLMs such as Gemini then it has all the necessary requirements to produce extra trainable data from other sources.

For the current use case, the aim was to fine-tune the model to work on the Estonian text. Because of the way in which the National Library's data is structured, the use of pypdf was not really necessary to create a baseline. The data already had bounding boxes and an initial text accompanying it. An attempt was made to see what data came out of pypdf – the text appeared to be the original OCR text. That means that the community improved upon text was not there. Due to this fact the initial parts of the framework did not feel as important, but as the experimentation began with fine-tuning, it proved useful as a tool to get the original OCR-d data from the PDF files. This allowed for an instant large increase in the overall amount of training data available. Similar prompts were used to those that were in the OLMOCR paper, only slight modifications were made because of the coordinate system being slightly different. The JSON schemas, initially containing data on language and page rotation, were also simplified because of the uniform nature of the scanned newspapers.

The data used for training needed some modification due to the way the VLMs are structured. This means a conversational data format. The messages given are split into three parts, with the initial being the system prompt for the input, and the text being largely unchanged from the OLMOCR paper. The second part is the input text and image, which in this case means the initial version of text gathered from dea.digar and the image that was then converted to a jpg format. The image data is a reference to the actual directory where the image is. The final part was the assistant, which is told to output the LLM corrected text. The training data was 81%/ 9%/ 10% split for training, validation and testing. The data was loaded to the University of Tartu HPC servers with the dataset with all of the text and a reference to the images, the images were uploaded under a

separate dataset folder.

4.5 QWEN-2.5-VL

4.5.1 Base model

QWEN-2.5-VL is a visual language model that was released in 2025. It is able to understand multimodal data such as video and images. In our cases it is given both text and images. The images are encoded with native dynamic resolution which allows it to understand scales inherently. It encodes height and width data for image tokens while keeping such information under a single identical position IDs for text. The image features extracted are fed to a two-layer perceptron and projected in a way that they match with the text embeddings in the LLM. The combined multimodal representations from both the text and the image are decoded by the LLM which is the QWEN2.5 LLM model [BCL+25].

Training of the model QWEN-2.5-VL was done in a way that the model saw many differing elements such as tables, charts and images, the same types of images that also appear in newspapers. Training data consisted of documents from many languages such as French, German, Italian, Spanish, Portuguese, Arabic, Russian, Japanese, Korean, and Vietnamese [BCL+25].

4.5.2 Fine-tuning QWEN-2.5-VL

The training for the fine-tuned model was performed using the Qwen2.5-VL-3B-Instruct checkpoint due to issues with running out of GPU memory on the HPC. How much this potentially changes the final outcome is not easy to gauge. The benchmark that is discussed in the Qwen2.5-VL paper for multilingual OCR use is CC-OCR OmniDocBench. There, the 72B model gains a score of 79.8, the 7B model drops to 77.8 and the 3B model gets a final score of 74.5 [BCL+25]. This benchmark is multilingual and has many different types of documents, but Estonian is not included in this dataset [OQZ+24].

The larger elements in the dataset could not be processed on the 40GB A100 cards and had to be filtered out of the initial training. This means that tokenized input ids were restricted to no more than 2100 tokens. Batch sizes were kept at 1 due to this same issue. The image files were reduced to at most 1024 pixels width or height. An overall restriction was placed for the maximum token-level to be at 8192 as in the OLMOCR paper.

The training scripts were made from a modified example script from the TRL repository [Fac23]. The learning rate was set to 5e-5 and the loss used was the default for the loader used: ForCausalLMLoss. Gradient accumulation steps were set to 2. The number of epochs was set to 10 after the initial few tries and the training would be cut short if the evaluation loss did not improve in 3 consecutive epochs.

Upon completing the training, an inference script was used to process the test set and generate responses, which were subsequently loaded into the database. These responses were then evaluated against the ground truth to compute the CER and WER metrics.

5 Results

5.1 Metrics

The baseline data for our comparison was the community-corrected data. This is separate data with an extra indicator from the National Library source, and some manually corrected data. This was compared against the LLM produced data and from the new model trained with the base of Qwen2.5-VL-3B-Instruct and the model trained by Transkribus.

Comparing full-page texts is more complex than a simple character-by-character or word-by-word comparison. This issue comes from the fact that different sources have a different reading order from one another. For texts that have clear sentence structures, this might be less of an issue, but in this case the text is in multiple columns and has a lot of headlines and sub-headlines. Due to this fact, it is more difficult to combine different pieces of text and it does not seem from a usability point of view to be an error if the ordering is different. As long as the articles are separated, readability does not necessarily suffer. For example, for LLMs the prompts used from the OLMOCR framework did not specify to return value locations even though these were used as inputs. As the final version of the QWEN model finetuning was based on blocks, then this issue mostly disappeared.

The main metric used to determine the accuracy of the given text is the character error rate and the word error rate. Character error rate is summation of how much of the characters needed to be deleted, inserted, or updated. The word error rate checks for these differences just on a word-per-word basis. The package used to calculate CER and WER was jiwer. A python package, initially designed for use in automatic speech recognition systems [Vae25].

To construct a baseline accuracy metric, the text extracted from pypdf was used. The logic being that we can see what is the difference between the originally detected text and the new ground truth community-corrected text. The final result was an error rate of 10%. An additional check was done to see if this differs by the papers observed. This data can be seen in table 2 where it shows the three largest papers with error rates, the mean, and the three smallest. The name string consists of the publication name, the date when it was published and the page at the end.

Newspaper	CER
Nommesona19341124.1.1	0.33
Eestikirikeelk19260401.1.5	0.26
Eestikirikeelk19260325.1.3	0.24
Average	0.10
Maahaal19331023.1.5	0.02
Nommesona19341124.1.3	0.01
Eestikirikeelk19260429.1.3	0

Table 2. Error rates for the original source OCR

5.2 Transkribus

5.2.1 Full-page results

The character error rate reported by the Transkribus application was 15.2%. For the validation pages, the result was moderately good at detecting line boundaries. Mistakes did not come so much from block misalignment rather the incorrect recognition of individual characters. In multiple parts, half of the word or even row was missing.

Additional 4 unseen pages of the periodical Järva Teataja were used to see how accurate the output would look like. The results were not very promising, the high error rate was mostly coupled with issues of identifying where the text was. This contrasted with the performance observed on the validation data. Some parts of the text were detected without any fault, but large parts of the lines seemed to only partially be covered. Other parts of the lines spanned two columns of text and so created issues from an usability point of view.

An example article can be seen in Figure 5, the blue lines shown are baselines. Due to the fact that the QWEN and the block-based Transkribus model showed much more promising results, the model was left as is for now. More work is most likely needed to create more input for training to produce better results. One of the benefits of using such a tool is the amount of manual corrections that can be done in the Transkribus application. Other base models could also be trained, and maybe if the larger Super models are opened for training then there would be a benefit in trying again. According to the documentation, this application will be continued to be updated during 2025 with a new API [Tra25]. The current model could not be published because it has a higher character error rate than the limit of 12%.

Figure 5. Transkribus full page baseline detection

5.2.2 Block-based results

The block-based version of the Transkribus model had a higher accuracy rate compared to the full page attempt. The training improved continuously as the error rate dropped even at later epochs of the data. The final test date CER achieved was 5.71% with a WER of 19.61%. For validation data the Transkribus application reported a CER of 7.1%.

The block-based on full-page models differed by what type of data was given to them and the model. The result for this block-based model was two times better than the page-level model version created by this thesis. For Estonian, this block type of model could be used to generate data that still needs manual corrections until there is enough full-page data to attempt to train that type of model again. The use of the Transkribus Print M1 model [Tra22] could also have an influence on the final result. This in comparison to using the Danish Newspapers 1750-1850 model [HD].

5.3 QWEN fine-tuned model

Model training was done on the basis of the Qwen2.5-VL-3B-Instruct model. Different attempts with more data yielded similar results when it came to training. The model started overfitting in only a few epochs. With a limit of the 3 epochs with no improvement in evaluation loss, the training was cut short. In the last run this can be seen happening in the second epoch already in Figure 6. The training took 3 hours on a single A100 40 GB GPU. The final evaluation loss was 0.243.

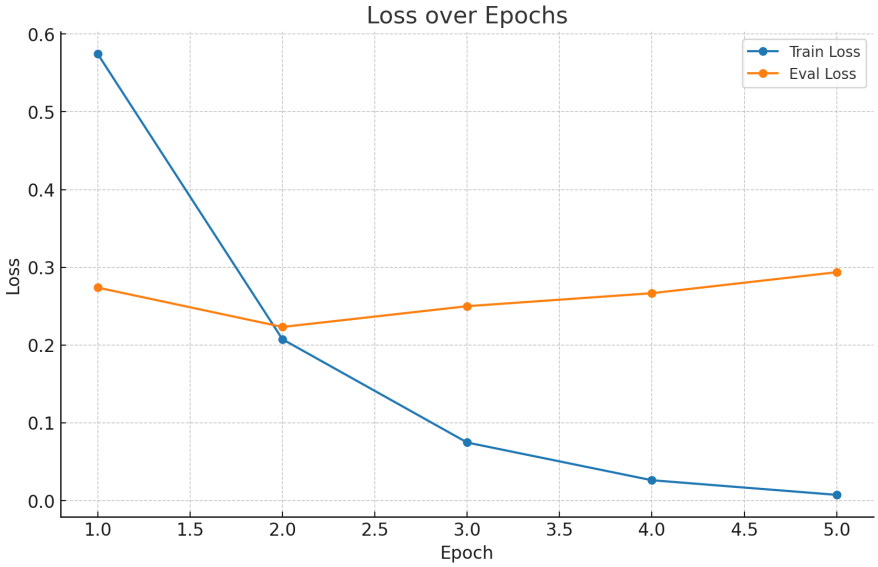


Figure 6. Training loss

When comparing the baseline model with the new fine-tuned model, the differences are easily observable in Table 3. The character error rate drops to 4% and the word error rate to 7%. The baseline model has high error rates in comparison and is not usable without fine-tuning.

Model	**CER**	**WER**
`Fine tuned model`	0.04	0.07
`Qwen2.5-VL-3B-Instruct`	1.43	1.68

Table 3. Average metrics for base and fine tuned models

The overall quality of the text can be observed in Figure 4. There is an immediately observable difference between the base Qwen2.5-VL-3B-Instruct and the fine-tuned version. The largest amount of recognized images dropped from the 5% to 50% error range to less than 1%. The ranges for these errors came from trial and error, but the underlying assumption about the low error rate of model performance comes from another attempt at understanding Fraktur script from printed books [Chr].

Primary sources of inaccuracy included model either omitting a response entirely, returning no values, or providing only a fragment of the complete text. This can be seen in Table 4. Manual checks of the images confirmed that the original text is present and readable. There are still mistakes being made in the higher error rate examples by misattributed characters, and multiple issues can be found in the same block. While the

best results appear to be for shorter text on average, the under 5% error rate rows seem to be the larger texts where the text was not lost, at least not to such a large extent. The text omitted does seem to be the largest indicator that the text is of low quality when comparing all groups. Compared to the base model, there seem to be less hallucinations or completely nonsensical responses. For example, the base model had a response in Spanish with a description of the text appearing to be from a historical document.

Different from the experience using a large LLM model, this fine-tuned model can also return non-words. For example *riigiwanema* gets turned into *eiigimana*. The second part of the phrase isn't a word, at least not in Estonian. Another example that, for example, Gemini can understand is *Hulljulge rööwimine Riias.* This gets turned into *Kull julge rööwimine Riias*. The input text for this, created by the original OCR used by the National Library, also misses the first word *JCttlljulge rööwimine Riias*. This is an example where the model does not seem to fall back on more plausible predictions for Estonian. Without any additional lines or the image Gemini-2.5-Pro would predict *Küll julge rööwimine Riias* which is still wrong, but is an actual plausible output.

Category	**Count**	**Len_Diff**	**Len_Avg**
Base <1\%	5	0	146.2
Base <5\%	10	10	241.5
Base >5\%	101	395	585.2
Base >50\%	19	1283	411.8
Base >90\%	106	6	169.2
Tuned <1\%	198	6	280.3
Tuned <5\%	21	31	782.3
Tuned >5\%	12	920	804.0
Tuned >50\%	7	1374	803.1
Tuned >90\%	3	663	270.0

Table 4. CER error rate visualized

5.4 LLM

Similar to how OLMOCR framework operates, this project created a lot of its training data by giving LLMs the input text with coordinates and the picture itself. The final values for what the differences were with the community-corrected texts can be seen in Figure 5. This is a check to see if the language models hallucinate or otherwise change the text in ways that would make this approach susceptible to errors. From a usability standpoint, it is not clear whether confusing a "w" for a "v" is actually an issue. Therefore, for the comparison, all such characters were unified as the letter "v". The main difference here was for GPT-4o, it got more accurate results by 1% over all cases.

Another attempted change was to unify the way the text is structured. The initial error is still shown as it seems to be standard practice for language models to change the format. Words are not split by multiple lines but written into a single word. Responses also get rid of multiple rows and return the text rather in a single row unless the text context does not allow for that. In Figure 5 the average CER is then without any modifications and Unified_Avg_CER is with the previously mentioned changes.

For the remaining errors observed, the issue for Gemini appears to be missing text. For example the first line gets omitted or only a single line is returned out of multiple. GPT-4o sometimes fixes errors that are still present in the original correct text, but other times it hallucinates the words. In one example article *kuldkäeuur* gets changed to *kuldkäekell*. This is an example of an archaic word being replaced with a more modern one, the meaning of the words *uur* and *kell* is watch. As that same word appears twice in the same article, and is corrected by GPT-4o both times, then this is not a solitary issue, but might be indicative of certain phrases disappearing if only these models are used for detection without additional oversight.

Using LLMs for validation data was also observed to not give any values for some of the articles. This was seen when using GPT-4o and an extra run of blocks was used as a trial to gauge how much this affects the final data. For 550 blocks it was observed that 12 were deemed to be against the content management policy. The initial assumption was that this is due to violence being described, but it turned out to be that the images and text were triggering jailbreak errors. This means that the system has identified that the prompts were trying to bypass safety protocols which keep the model from producing harmful content. These errors could be triggered by keyword-based rules or sensitive content detection models [HJB+25]. For our purposes, as there is no reason to assume anything malicious is being forwarded, this just means an issue with losing, by this example run, 2,18% of the total, usable data to overzealous filters.

Model	Total_Obs	Avg_CER	Unified_Avg_CER
GPT-4o	408	4.5%	1.5%
Gemini-2.5-Pro	408	4.7%	2%

Table 5. LLM metrics after modernization

Due to how the responses from LLMs are structured, at least with the scripts used that aren't tied to any coordinates, they are not easily placed into the current National Library solution. That storage is split into two, full text storage for each article and a line based solution in another view. If LLMs are to be used with the same type of no-coordinates response, then it makes sense to think about overhauling the current solution. If there is a reliable way of accounting for split words and validate it together with errors, then this is of course not needed.

6 Future Research Directions

Further work needs to be dedicated to investigate if the original detected text can be improved upon. For example what happens when the scope is widened and older newspapers with lesser quality, such as discoloration, are scanned by more modern methods. Additionally the viability, from a usability point of view, of providing the full LLM based detection on the more popular papers, which don't already contain community corrections. As a higher quality OCR may bring in more users to correct the text, different approaches should be tried. As the large scale detection of different documents by proprietary language models can be expensive, cost saving measures be explored. For example user interest in certain types of periodicals or articles can be used to first focus on them. The use of VLMs to try and get a description of each image could be also worthwhile if that would create more community interest in the materials.

Different Transkribus models were used for the full page and block based approaches. As the M1 model gets continually updated, and of course other new models appear, other models could be retried on the already gathered data to see which one produces the best results. As Transkribus is easy to use and it aligns well with the row based approach of the National Library, then it would be a tool that could continue to be useful.

7 Conclusion

This thesis demonstrates how modern approaches to optical character recognition work on historical Estonian data with archaic script and writing. The test data consisted of digitalized newspapers from 1919-1944 which already had earlier detected text alongside it in the DIGAR database. Community-corrected text was used as ground truth data with LLMs used to add to this data for training of models.

Tools provided in the GitHub repository offer a multi-step solution to ingest data, store it effectively and resiliently, and format it for external training. Coordinated data can be provided by full pages or blocks to the target models or platforms. Responses can be read in and validated based on the database design.

With LLMs good results were achieved if these models are given both the previously detected text and the image. With the text as is the error rate is below 5% for both GPT-4o and Gemini-2.5-Pro. With some modification to the text, to remove line breaks and modernize the vocabulary a little, the accuracy is 2% for Gemini-2.5-Pro and 1.5% for the observed cases for GPT-4o. LLMs suffer from missing text in their responses, some text just isn't generated and only half of the text is in the response. Also, these LLMs change words to a more modern Estonian which is an issue for retaining archaic phrases.

QWEN-2.5-VL-Instruct is also given anchored text. The original QWEN-2.5-VL-Instruct and the new fine-tuned created model are compared. QWEN-2.5-VL-Instruct is not adept at understanding Estonian texts out of the box with a character error rate over 0.1, but with fine-tuning the character error rate drops to 0.04. The best word error rate achieved is 0.07. The OLMOCR framework was used to obtain training data for this model through the use of LLMs. Coordinated data was fed as the input and output was expected to be a plain text representation of the characters that were seen. For this project the 3 billion parameter model was used, the results might be better with a larger model. This is not conclusive due to the focus on larger languages in the benchmark used to determine the effectiveness of the model.

Two different models were trained on the Transkribus platform. The first used community-corrected data to train a model on the full pages where at least some community-corrected data was present. This model wasn't effective as it could not properly detect where the text was and as such had a final character error of 15%. Another attempt with blocks of community-corrected data was made with the Transkribus M1 model and it showed improved results by reaching a CER of slightly higher than 7%.

Based on the error rates, the LLM based approaches are the most effective way to improve the quality of the current detected text. From the two paid models tested, GPT-4o outperformed Gemini-2.5-VL. Some additional post-processing is needed and this can be accomplished with the framework provided by this project for ingesting, enhancing and preparing the training data.

References

[All24] AllenAI. olmOCR-7B-0225-preview. https://huggingface.co/allenai/olmOCR-7B-0225-preview, 2024.

[BCL+25] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025.

[BLPCD+20] E. Boros, E. Linhares Pontes, L. A. Cabrera-Diego, A. Hamdi, J. G. Moreno, N. Sidère, and A. Doucet. Robust named entity recognition and linking on historical multilingual documents. Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum (CLEF-HIPE 2020), Thessaloniki, Greece, September 2020.

[Chr] Vincent Christlein. Combining OCR Models for Reading Early Modern Printed Books. Document Analysis and Recognition - ICDAR 2023. ICDAR 2023. Lecture Notes in Computer Science, vol 14191. Springer, Cham. https://arxiv.org/abs/2305.07131/.

[CKC25] Giorgia Crosilla, Lukas Klic, and Giovanni Colavizza. Benchmarking large language models for handwritten text recognition. https://arxiv.org/abs/2503.15195, 2025.

[Com20] European Comission. Newseye: A digital investigator for historical newspapers, 2020. https://cordis.europa.eu/project/id/770299 Accessed 07 August 2025.

[Dat22] Databricks. Data Warehousing Modeling Techniques and their Implementation on the Databricks Lakehouse Platform. *The Databricks Blog*, June 2022.

[Fac23] Hugging Face. Transformers reinforcement learning (trl). https://github.com/huggingface/trl, 2023. Accessed: 2025-08-07.

[FSpz+24] Mathieu Fenniak, Matthew Stamy, pubpub zz, Martin Thoma, Matthew Peveler, exiledkingcc, and pypdf Contributors. The pypdf library, 2024. See https://pypdf.readthedocs.io/en/latest/meta/CONTRIBUTORS.html for full list of contributors.

[GGG25] Gavin Greif, Niclas Griesshaber, and Robin Greif. Multimodal llms for ocr, ocr post-correction, and named entity recognition in historical documents. https://arxiv.org/abs/2504.00414, 2025.

[HD] Aalborg Universitet Historisk Datalaboratorium. https://app.transkribus.org/models/text/danish-newspapers-1750-1850 Accessed 10 August 2025.

[HJB+25] Linghan Huang, Haolin Jin, Zhaoge Bi, Pengyue Yang, Peizhou Zhao, Taozhao Chen, Xiongfei Wu, Lei Ma, and Huaming Chen. The tower of babel revisited: Multilingual jailbreak prompts on closed-source large language models. https://arxiv.org/abs/2505.12287, 2025.

[HRMT21] Simon Hengchen, Ruben Ros, Jani Marjanen, and Mikko Tolonen. A data-driven approach to studying changing vocabularies in historical newspaper collections. https://doi.org/10.1093/llc/fqab032, 11 2021.

[KLKG25] Jenna Kanerva, Cassandra Ledins, Siiri Käpyaho, and Filip Ginter. Ocr error post-correction with llms in historical documents: No free lunches. https://arxiv.org/abs/2502.01205, 2025.

[LLH+24] Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: on the hidden mystery of OCR in large multimodal models. Science China Information Sciences. 67(12), 2024. https://link-springer-com.ezproxy.utlib.ut.ee/article/10.1007/s11432-024-4235-6.

[ML20] Johannes Michael and Roger Labahn. Newseye d2.5: Automatic text recognition (final). EU Horizon 2020 Project Deliverable D2.5, University of Rostock, Rostock, Germany, 2020. EU Horizon 2020 Project No. 770299, NewsEye: A Digital Investigator for Historical Newspapers.

[OQZ+24] Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. Omnidocbench: Benchmarking diverse pdf document parsing with comprehensive annotations. https://arxiv.org/abs/2412.07626, 2024.

[Qwe24] Qwen Team. Qwen2.5-VL-3B-Instruct. https://huggingface.co/Qwen/Qwen2.5-VL-3B-Instruct, 2024.

[Traa] Transkribus. https://help.transkribus.org/super-models Accessed 10 August 2025.

[Trab] Transkribus. https://readcoop.org/what-we-do/ Accessed 07 August 2025.

[Tra22] Transkribus (READ-COOP). Transkribus print m1. https://app.transkribus.org/models/text/transkribus-print-m1, 2022. Extended multi-language Transkribus print model, including antiqua and blackletter prints, typewriter, computer print outs and decorative fonts Accessed 11 August 2025.

[Tra25] Transkribus, 2025. https://www.transkribus.org/ Accessed 07 August 2025.

[TSGL+24] Solène Tarride, Yoann Schneider, Marie Generali-Lince, Mélodie Boillet, Bastien Abadie, and Christopher Kermorvant. Improving automatic text recognition with language models in the pylaia open-source library. https://arxiv.org/abs/2404.18722, 2024.

[Vae25] Nik Vaessen. JiWER: Similarity measures for automatic speech recognition evaluation. https://github.com/jitsi/jiwer, 2025. Python package, version 4.0.0; Apache-2.0 license.

[Vä25] Mattias Väli. Ee-gothic-script-ocr. https://github.com/MattValSE/EE-Gothic-Script-OCR, 2025. Accessed: 2025-08-07.

Appendix

A Repositories

The code, the models and Parquet files created for this thesis are publicly available in the following repositories:

- **Hugging Face:** The dataset can be found at `https://huggingface.co/datasets/MatValSE/EE_Pre1944_Newspapers`.
- **Hugging Face:** The trained model can be found at `https://huggingface.co/MatValSE/EE_Fraktur_OCR`.
- **GitHub:** The source code is available at `https://github.com/MattValSE/EE_Fraktur_OCR.git`.
- **Transkribus:** The model is available at: `https://app.transkribus.org/models/text/383577`

B List of used newspaper with dates

- **Postimees**
  - 1927-12-14
  - 1924-08-06
  - 1936-12-06
  - 1933-04-01
  - 1937-12-12
  - 1937-04-26
  - 1937-04-24
  - 1924-04-01
  - 1939-07-16
  - 1928-03-13
- **Kaja**
  - 1935-09-17
  - 1929-08-13
  - 1927-07-10
  - 1935-07-01
  - 1922-09-17
  - 1921-04-17
  - 1929-08-25
- **Maa Hääl: maarahva ajaleht**
  - 1933-10-23
  - 1934-08-03
  - 1934-08-05
  - 1934-08-07
  - 1934-08-10
  - 1934-08-12
  - 1934-08-14
  - 1934-08-17
  - 1934-08-19

  - 1934-08-21
  - 1934-08-24
  - 1934-08-26
  - 1934-08-28
  - 1934-08-31
- **Sakala**
  - 1936-11-18
- **Nõmme Sõna**
  - 1934-11-24
- **Järva Teataja**
  - 1927-06-10
  - 1938-08-01
  - 1934-01-04
  - 1927-12-16
- **Lõuna-Eesti**
  - 1929-08-29
  - 1938-03-23
- **Tallinna Teataja**
  - 1920-03-15
  - 1917-05-05
  - 1917-05-06
- **Eesti Kirik**
  - 1926-03-25
  - 1935-10-24
  - 1926-04-29
  - 1926-02-04
  - 1926-02-25
  - 1926-03-18
  - 1926-04-01
  - 1926-05-27

C Document-Anchoring Prompt

```
prompt = [
    image_path,
    {
"text": f"""Below is the JPG image of one page of a PDF document ,
as well as some raw textual content that
was previously extracted for it that includes position
information for each image and
block of text ( The origin [0 x0 ] of the coordinates
is in the upper left corner of the
image ).
Just return the plain text representation of this document
as if you were reading it
naturally .
Turn equations into a LaTeX representation , and tables
into markdown format . Remove the
headers and footers , but keep references and footnotes .
Read any natural handwriting .
This is likely one page out of several in the document ,
so be sure to preserve any sentences
that come from the previous page , or continue onto the next page ,
exactly as they are .
If there is no text at all that you think you should read ,
you can output null .
Do not hallucinate .
RAW_TEXT_START
{ layout_text }
RAW_TEXT_END
"""
    }
]
```

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Mattias Väli**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Optical Character Recognition of Estonian Fraktur Script,
   (title of thesis)

   supervised by Aleksei Dorkin and Kairit Sirts.
   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Mattias Väli
12/08/2025