electronics

MDPI

Article

Treating OCR Output as a Language (TOOL): A Pipeline for OCR Post-Correction †

Thomas Asselborn [1,*], Magnus Bender [2], Ralf Möller [3] and Sylvia Melzer [1,3]

[1] Institute of Information Systems (IFIS), University of Luebeck, Ratzeburger Allee 160, 23562 Luebeck, Germany; sylvia.melzer@uni-hamburg.de
[2] Department of Management, Aarhus University, Universitetsbyen 61, 8000 Aarhus C, Denmark; magnus@mgmt.au.dk
[3] Institute for Humanities-Centered Artificial Intelligence (CHAI), University of Hamburg, Warburgstraße 28, 20354 Hamburg, Germany; ralf.moeller@uni-hamburg.de
* Correspondence: thomas.asselborn@uni-luebeck.de
† This paper is an extended version of our paper published in 20th Conference on Computer Science and Intelligence Systems (FedCSIS 2025).

Abstract

Optical Character Recognition (OCR) enables the digitisation and analysis of large document collections, but recognition errors remain a challenge, particularly for historical documents. This paper presents TOOL (Treating OCR Output as a Language), a modular approach that formulates OCR post-correction as a sequence-to-sequence translation task. A fine-tuned MarianMT encoder–decoder model transforms noisy OCR output into corrected text without modifying the underlying recognition engine. We evaluate TOOL on a collection of historical German vocational education and training documents spanning the period from 1871 to the present day. On 139 held-out test documents, TOOL reduces mean word error rate from 0.792 for Tesseract alone to 0.600, corresponding to a relative reduction of approximately 24.3%. Improvements are also observed in Jaccard bigram similarity, BLEU-4, METEOR, and ROUGE-2, whereas mean normalised Levenshtein distance increases. Comparisons with Hunspell, Phi-4, and Qwen2.5-VL show that the larger language and vision-language models achieve higher accuracy on several metrics. However, TOOL offers a competitive quality–runtime trade-off, with an average end-to-end processing time of 30.12 s per document, compared with 96.57 s for Qwen Transcribed. The results support the feasibility of translation-based OCR post-correction as an additional processing stage in existing OCR workflows, while generalisation to other document collections, OCR engines, and languages remains to be investigated.

Keywords: OCR; Post-OCR Processing; Optical Character Recognition; Natural Language Processing; TOOL; Seq2Seq; LLMs; Marian

Check for updates

Academic Editors: Arkaitz Zubiaga, Andrzej Paszkiewicz and Chintan Bhatt

Received: 21 August 2026
Revised: 18 October 2026
Accepted: 6 November 2026
Published: 18 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The rapid growth of document digitisation has led to a large volume of scanned textual data that requires reliable automatic processing [1]. OCR (Optical Character Recognition) has therefore become a fundamental technology within computer vision, document analysis, digital libraries, and IDP (Intelligent Document Processing) [2–4]. Despite remarkable progress in deep learning-based OCR systems, recognition errors remain a challenge when processing historical documents, low-quality scans, multilingual collections, or complex layouts [5]. These errors can propagate into subsequent processing stages and affect information extraction, semantic search, NER (Named Entity Recognition), knowledge graph

construction, and RAG (Retrieval Augmented Generation) applications [6]. Consequently, reliable OCR post-correction remains an important research topic at the intersection of computer vision, NLP (Natural Language Processing), and machine learning [7].

An important practical consideration is that domain researchers often rely on established OCR workflows and software environments. Replacing an existing recognition system with a vision-language model may require changes to infrastructure, computational resources, and established processing procedures. Therefore, an automated post-correction component that operates on the textual output of an existing OCR engine can provide an alternative to replacing the recognition system. This approach is particularly relevant for digital humanities projects in which existing tools and workflows need to be maintained. Furthermore, OCR errors often exhibit systematic patterns that can be learned from aligned examples [8].

To address this application scenario, TOOL (Treating OCR Output as a Language) treats noisy OCR output as a textual representation that can be transformed into corrected text using Seq2Seq (Sequence-to-Sequence) learning. Rather than modifying the underlying recognition engine or relying exclusively on handcrafted correction rules, TOOL introduces an additional processing stage that learns a mapping between OCR output and corresponding reference transcriptions. In the implementation evaluated in this paper, this mapping is learned using a fine-tuned MarianMT model, a Transformer-based encoder–decoder architecture [9]. The resulting correction component operates on textual input and can be integrated into an existing OCR workflow without modifying the recognition engine itself.

The proposed pipeline uses aligned pairs of noisy OCR output and corresponding GT (Ground Truth) text to train a collection-specific correction model. Its modular structure separates text recognition from post-correction and therefore permits the correction stage to be adapted independently of the underlying OCR engine. However, the performance of a trained correction model depends on the characteristics of the training data, including the document collection, language, and recognition errors. Consequently, the present study evaluates the approach on a specific collection of historical German documents rather than claiming demonstrated generalisation across languages, collections, or recognition engines.

The main contributions of this paper are threefold. First, we present TOOL as a modular approach for integrating OCR post-correction into established document-processing workflows without replacing the underlying recognition engine. Second, we describe the implementation of the pipeline, including the preparation of aligned OCR and reference text pairs, fine-tuning of a Transformer-based Seq2Seq model, and automated correction of OCR-generated documents. Third, we provide a comparative evaluation of conventional and transformer-based correction and transcription approaches on German documents spanning the period from 1871 to the present day, examining both transcription quality and computational cost. The evaluation includes runtime measurements and an analysis of the trade-offs between correction accuracy and processing time.

This article extends our previous conference contribution [8]. Whereas the earlier work introduced the TOOL concept and investigated its feasibility, the present journal version expands the experimental comparison to include additional correction and transcription approaches, including Hunspell, Phi-4, and Qwen. It further examines processing times at different stages of the workflow and evaluates accuracy–runtime trade-offs using Pareto-front analysis.

The remainder of this paper is structured as follows: Section 2 reviews related work on OCR post-correction and document processing. Section 3 introduces the historical vocational education and training document collection used in the study. Section 4 presents the TOOL approach and its pipeline architecture. Section 5 describes the experimental

setup and presents the evaluation results. Section 6 discusses the limitations of the present study. Finally, Section 7 concludes the paper and outlines directions for future research.

2. Related Work

Seq2Seq modelling forms the foundation of numerous NLP tasks, including machine translation, text summarisation, text normalisation, and more recently, OCR post-correction. The original Seq2Seq pipeline was introduced by Sutskever et al. (2014) [10], who proposed using RNs (Recurrent Networks) with LSTM (Long Short-Term Memory) units to encode an input sequence into a fixed-length vector and decode it into an output sequence. While this approach achieved promising results in machine translation, its reliance on a fixed-size context vector limited its ability to model long input sequences. To overcome this limitation, Bahdanau et al. (2015) [11] introduced the attention mechanism, enabling the decoder to dynamically focus on different parts of the input sequence during generation. This innovation significantly improved translation quality and became a core component of subsequent Seq2Seq models.

A breakthrough was achieved with the Transformer architecture proposed by Vaswani et al. (2017) [12], which replaced recurrent computation entirely with self-attention mechanisms. This design significantly improved parallelisation, training efficiency, and the modelling of long-range dependencies, establishing Transformers as the dominant architecture for modern Seq2Seq tasks. Building upon the Transformer, Lewis et al. (2020) [13] introduced BART (Bidirectional and Auto-Regressive Transformers), a denoising autoencoder for Seq2Seq generation. BART combines a bidirectional encoder (similar to BERT (Bidirectional Encoder Representations from Transformers) [14]) with an autoregressive decoder (similar to GPT (Generative Pre-trained Transformer) [15]), making it well-suited for text generation, correction, and translation tasks, particularly when the input text is noisy, such as in OCR outputs.

Around the same time, Marian [9] was developed as a highly efficient, production-grade translation system optimised for multilingual environments. As shown by Junczys-Dowmunt et al. (2018) [9], Marian is particularly effective in low-resource settings, making it a strong candidate for OCR correction across diverse languages.

Recent research has increasingly focused on Transformer-based approaches specifically for OCR post-correction. Nguyen et al. (2021) [5] provide a comprehensive survey of OCR post-correction techniques, documenting the transition from rule-based and statistical methods to Seq2Seq architectures. Their survey identifies Transformer-based models as a particularly promising direction because they can learn systematic OCR error patterns directly from aligned OCR and ground-truth text.

More recently, Boros et al. (2024) [16] demonstrated that modern Transformer architectures substantially outperform traditional statistical and recurrent approaches across multiple historical document collections. Their results further indicate that pretrained language models improve robustness, particularly for historical corpora and low-resource languages where annotated training data are scarce.

The increasing availability of LLMs (Large Language Models) has further advanced research in the field of OCR post-correction. Recent studies have examined instruction-optimised and generative Transformer models for correcting OCR errors, which demonstrate high generalisation ability while reducing the need for manually created correction rules. However, these approaches generally require significantly more computational resources than specialised Seq2Seq models, making them less suitable for large-scale digitisation workflows [7,17].

TOOL builds upon these developments while adopting a different perspective. Rather than treating OCR post-correction solely as a text normalisation problem, TOOL treats

OCR output as a language in its own right and formulates post-correction as a Seq2Seq translation task. This formulation enables modern Transformer architectures such as Marian and BART to learn systematic OCR error transformations directly from aligned OCR and ground-truth text without requiring modifications to the underlying OCR engine. As a result, the proposed pipeline provides a model-agnostic and adaptable solution that can be integrated into existing document analysis and IDP workflows.

3. Use Case

This section introduces the application domain used to evaluate the proposed TOOL pipeline. It first provides a brief overview of VET (Vocational Education and Training) and CVET (Continuing Vocational Education and Training), then describes the historical document collection and the OCR-related challenges that motivate the proposed post-correction approach.

3.1. VET and CVET

VET prepares learners for specific occupations by combining theoretical instruction with practical, work-based learning. In many European countries, and particularly in Germany, VET has long been a cornerstone of workforce qualification, supporting sectors such as manufacturing, healthcare, commerce, and technology [18,19]. Historical VET documents therefore provide valuable evidence for studying the evolution of occupational standards, qualification requirements, and vocational education policies. Beyond initial vocational education, CVET supports lifelong learning through professional development, reskilling, and upskilling. The German system differentiates between initial vocational training (German: *Ausbildung*), retraining (German: *Umschulung*), advanced continuing training (German: *Weiterbildung*) and upgrading training (German: *Fortbildung*). Upgrading training is often governed by federal regulations such as the Vocational Training Act (German: *Berufsbildungsgesetz*, BBiG) [20] and the Crafts Code (German: *Handwerksordnung*, HwO) [21]. Together, historical VET and CVET documents form an important source for analysing the development of vocational qualifications and educational policies over time.

3.2. OCR Processing of Documents for VET

The use case considered in this work consists of a collection of historical German VET and CVET documents dating from the late nineteenth century to the present day. These documents include training regulations, examination requirements, curricula, and administrative publications describing qualification standards and occupational competencies across different professions.

Figure 1 illustrates one representative document from the collection. It contains examination requirements for lithography teachers and exemplifies the structure and typography commonly found throughout the corpus. Many of these documents were published or supervised by governmental institutions responsible for vocational education, such as the Imperial Institute for Vocational Training in Trade and Industry (German: *Reichsinstitut für Berufsausbildung in Handel und Gewerbe*), which played an important role in standardising vocational education within the German Reich.

Digitising these historical sources using OCR enables full-text search, information retrieval, and large-scale analysis of the historical development of vocational education. However, the documents also represent a particularly challenging benchmark for OCR systems.

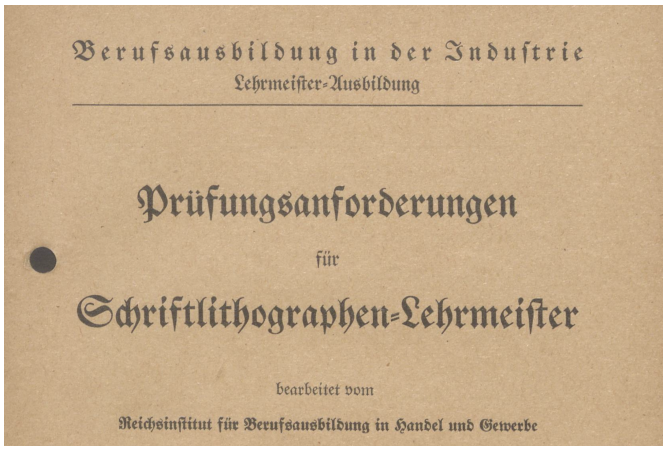


Figure 1. Exam requirements for a type lithographer teacher (orig. in German: *Prüfungsanforderungen für Schriftlithographen-Lehrmeister*).

An important problem is the typographical style of the original prints: Many documents from the early 20th century use Fraktur or broken Antiqua fonts, which differ greatly from modern Latin script. These fonts have unusual letter forms (e.g., long s "ſ" versus short s "s") and dense ligatures that standard OCR engines poorly recognise. The result is a high rate of recognition errors, such as character confusion, word fragmentation, or misreadings that distort key terms (e.g., the German words: *Lehrmeister*, *Prüfungsanforderung*, etc.). In addition, the print quality of many originals contributes to noisy OCR output with smudges, uneven contrast and marginal notes that are interpreted as content. These characteristics make historical VET and CVET documents a demanding benchmark for OCR post-correction. Rather than modifying the OCR engine itself, the proposed TOOL pipeline addresses these systematic recognition errors.

4. Method

To introduce TOOL, we begin by analysing the broader context. Specifically, we consider a comprehensive dataset comprising scans of VET documents, which is a collection of image files representing a set of text-based records. The primary objective is to generate high-quality transcriptions of these documents as automatically as possible. Additionally, a limited set of manually transcribed documents is available for training.

A typical IDP workflow consists of several processing stages. Documents are first acquired through scanning or image capture and converted into machine-readable text by an OCR engine. Downstream components then process the recognised text for document classification, information extraction, semantic search, knowledge graph construction, or RAG. Since the quality of these downstream tasks depends directly on the OCR output, recognition errors may propagate throughout the entire workflow. TOOL addresses this problem by introducing a dedicated post-processing stage between OCR and downstream processing, transforming the intermediate OCR representation into corrected text before further analysis.

Rather than adapting or retraining the underlying OCR engine, TOOL introduces a separate text-based post-correction component. The pipeline architecture does not depend on a particular recognition engine, provided that its output is available as text. This architectural independence does not imply that a correction model trained on one document collection or OCR engine will achieve the same performance on another. Such applications may require additional fine-tuning with representative OCR and reference text pairs.

Generally, this situation is a common use case for OCR, but as stated before, the OCR results do not fulfil our needs. Furthermore, the available set of transcribed documents is too small to be used for training a full custom OCR model. Training a new OCR model

would additionally require substantially more manually annotated image-text pairs than are available for many historical document collections. Therefore, improving the OCR output by means of post-correction represents a considerably more practical solution. However, we recognise that the unsatisfactory OCR output follows its own characteristics. Typical and recurring errors include, e.g., (i) the Umlaut "ü" in scanned documents often resulting in the sequence "ii" in the recognised text, or (ii) the long s "ſ" being recognized as "f".

These errors are not purely random but occur systematically as a consequence of typography, print quality, and the characteristics of the OCR engine. As a result, the generated OCR text exhibits recurring lexical and orthographic patterns that can be learned automatically.

We observe that the OCR output is not correct German, but some kind of inherent *language*. Instead of interpreting OCR output merely as corrupted text, TOOL models it as an intermediate textual representation, referred to as "OCR German." This representation follows its own statistical regularities and can therefore be translated into standard German using Seq2Seq learning. Hence, the output can be treated as a language, which leads us to the approach of TOOL: Translating from the OCR output *language* to a correct natural language, in our case German.

TOOL consists of two steps: the OCR step and the *translate* step.

The OCR step relies on an existing recognition engine and does not require additional model training within TOOL. The pipeline can process textual output from different OCR engines, although its correction performance across engines and languages has not been evaluated in this study. In contrast, the *translate* step requires a task-specific model, although training can be performed with only a comparatively small amount of aligned OCR and ground-truth data.

While OCR post-processing is commonly formulated as a correction task, TOOL formulates the second stage as a translation problem. Conventional OCR post-correction approaches aim to transform erroneous text into its correct form. In contrast, TOOL learns a mapping between two textual domains: the intermediate "OCR German" representation and standard German. Rather than predicting isolated corrections, the model learns a transformation between two textual representations with distinct statistical characteristics. This enables the exploitation of contextual information across complete sequences while learning systematic OCR transformations directly from aligned OCR and ground-truth text. Furthermore, because the translation component operates exclusively on textual input, it remains independent of the underlying OCR engine and can therefore be integrated as a modular post-processing component within IDP workflows.

State-of-the-art translation systems commonly use Transformer architectures, including encoder–decoder models suitable for the translation step of TOOL. The selection of a specific model depends on the desired trade-off between correction quality, computational efficiency, deployment complexity, and training effort. The following sections therefore discuss three representative approaches: (a) Encoder, (b) Instruction LLM, and (c) Seq2Seq.

4.1. Encoder

One potential approach involves the use of an encoder-only model such as BERT. Encoder-only architectures are highly effective at learning contextual representations but are not designed to generate corrected output sequences directly. Therefore, additional decoding mechanisms would be required for OCR post-correction. BERT processes an input sequence of words and outputs a corresponding sequence of contextualised embedding vectors. The idea is to utilise the vector space of the embedding vectors of a default BERT and train a custom BERT in a way that it generates for the OCR output embeddings in

the same vector space as a default BERT. The big advantage over the instruction LLMs, such as Llama and GPT, is that, in contrast, only a small BERT needs to be fine-tuned for TOOL and a shared embedding vector space is used, i.e., techniques like RAG for retrieving documents from a collection can be applied directly. However, there is also a need for the correct German transcriptions of the scans and text generation from BERT embeddings is not straightforward. Additionally, there is no GT for the vector embeddings as our training data contains transcribed documents and not embeddings.

4.2. Instruction LLM

Another possibility is to apply an instruction LLM, like Llama or GPT, similar to [22]. Recent instruction-tuned LLMs have demonstrated impressive capabilities for text correction and therefore represent an attractive alternative for OCR post-correction. Instead of training a model for TOOL's translation step, the instruction LLM receives the OCR output and is prompted to *correct* it. This approach works well, but there are disadvantages. While requiring no training is an advantage, the downside is that the process does not speed up once training is complete. An instruction LLM is a relatively large model, and running it requires special hardware and a lot of resources.

4.3. Seq2Seq

Overall, encoder–decoder architectures naturally match the formulation proposed by TOOL because they directly learn a mapping from noisy OCR sequences to corrected text. TOOL's translation step can use a comparatively compact Transformer-based encoder–decoder model that maps an input text sequence to a corrected output sequence. A popular choice for such Seq2Seq models is Marian [9]. Using Marian for TOOL's translation step solves both issues stated before: First, there is no need for sharing an embedding vector space as required when using BERT. Second, Marian is a comparatively small model and can be fine-tuned in a reasonable amount of time. Additionally, after fine-tuning, the model can be used with standard hardware.

For the implementation evaluated in this study, we selected MarianMT, a Transformer-based encoder–decoder Seq2Seq model [9]. Its architecture directly supports the transformation of noisy input sequences into corrected output sequences and permits task-specific fine-tuning using aligned text pairs. Compared with the instruction-based alternatives considered in this study, the selected model also offers a practical option for repeated inference on larger document collections. The computational requirements of fine-tuning and inference are considered separately in the experimental evaluation.

The overall TOOL architecture is illustrated in Figure 2. The input consists of scanned VET documents, shown on the left-hand side of the figure. The first stage applies an existing OCR engine to obtain a textual representation of each document. In the second stage, this text is divided into chunks that satisfy the input-length constraints of the correction model. Each chunk is then processed by a fine-tuned MarianMT model, which has been trained separately using aligned OCR and GT text pairs, as described in Section 5.1. Finally, the generated chunks are concatenated in their original order to produce the post-corrected document, shown on the right-hand side of the figure.



Figure 2. Process flow for the TOOL-based OCR translation pipeline. *Input files* are first passed through the *Do OCR* process. Then, the following processes are executed: *Split into Chunks*, *Translate with Fine-tuned Seq2Seq Model*, and *Recombine Chunks*. Together, these steps produce the *Final Output*.

Algorithm 1 summarises the document-processing stage of the pipeline. Let X denote the text output of the OCR process O applied to the input data I. The `SegmentText` process

takes X as input and divides it into a sequence of sentences S. The `PackChunks` process packs the sequence S into an ordered sequence of chunks, each with a maximum length of L. Each chunk c is then passed to the `translate` process, which uses the fine-tuned correction model to generate a corrected chunk y. Finally, the corrected chunks are concatenated in their original order to produce the continuous corrected text T.

Algorithm 1 TOOL document processing

Require: Document image I, OCR engine O, fine-tuned correction model M, maximum input length L
Ensure: Post-corrected document T

$X \leftarrow O(I)$ ▷ Perform OCR
$S \leftarrow \textsc{SegmentText}(X)$
$C \leftarrow \textsc{PackChunks}(S, L)$
$Y \leftarrow [\,]$
for all $c \in C$ **do**
  $y \leftarrow \textsc{Translate}(M, c)$
  $\textsc{Append}(Y, y)$
end for
$T \leftarrow \textsc{ConcatenateInOrder}(Y)$
return T

This algorithm describes inference rather than training-data preparation. Ground-truth alignment is performed exclusively during preprocessing for fine-tuning, as described in Section 5.1. No ground-truth transcriptions are required during inference.

The fine-tuned MarianMT model is the only trainable correction component of the pipeline. Text segmentation, chunk packing, and recombination are processing operations that enable document-level inference under the model's input-length constraints.

Overall, the big advantages of the TOOL approach are as follows:

- Any OCR tool can be used in the first step.
- For the second step, a pre-trained sequence-to-sequence translation model, such as MarianMT, can be fine-tuned.
- It does not require much training in terms of fine-tuning.
- The resulting pipeline, OCR and translation, is quite fast and resource efficient, especially compared to applying an instruction LLM [22].

Next, we describe the technical details of TOOL, perform our evaluation and present the results.

5. Experiments and Results

In order to verify the method described in this paper, experiments were conducted using the texts described in Section 3. First, the process of fine-tuning with its subprocesses (a) OCR, (b) Preprocessing for Fine-Tuning, and (c) Fine-Tuning is described.

5.1. Preprocessing and Fine-Tuning

A process flow diagram for fine-tuning the model can be seen in Figure 3. The experimental setup does not aim to identify the best possible models and parameters, but rather to show that the method works in practical scenarios.

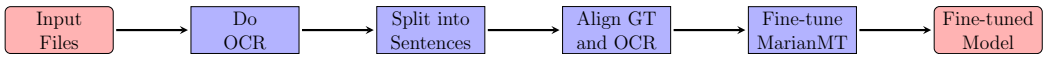


Figure 3. Process flow for OCR to MarianMT fine-tuning pipeline. *Input files* are first passed through the *Do OCR* process. Then, the following processes are executed: *Split into Sentences*, *Align GT and OCR*, and *Fine-tune MarianMT*. Together, these steps produce the *Fine-tuned Model*.

5.1.1. OCR

As a first step, OCR is performed on all files from our dataset. We have decided to use Tesseract [2] as our OCR engine because it is an open-source and widely adopted solution that supports a broad range of languages and scripts. Its flexibility and ease of integration into automated workflows make it particularly suitable for processing large and heterogeneous datasets.

5.1.2. Preprocessing for Fine-Tuning

The document collection was initially divided into two disjoint subsets of approximately equal size at the document level. One subset was used for model development, while the other was reserved for the final evaluation. OCR output and corresponding ground-truth transcriptions were subsequently segmented and aligned within their respective subsets. The aligned sentence pairs from the development subset were further divided into 90% training and 10% validation data for fine-tuning. The held-out evaluation documents were not used for model training or validation.

As a next step, the texts in each file are split into separate sentences using the NLTK (Natural Language Toolkit) [23] punkt tokeniser. This is done for the OCR and GT texts to obtain text that can be used for fine-tuning the MarianMT model.

For the final preprocessing step, the corresponding sentences from the GT (see Figure 2) and after the OCR process need to be aligned in order to obtain corresponding sentence pairs. To do so, we used a LaBSE (Language-agnostic BERT Sentence Embedding) model [24] based on the BERT architecture. This language-agnostic model allows us to encode texts from different languages into the same vector space. Using a similarity measure, such as cosine similarity, we can pair sentences that are close in this vector space. The cosine similarity, as used, is defined as

$$\mathrm{sim}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|} \tag{1}$$

In this application, $\mathbf{A}$ denotes the LaBSE embedding of a sentence obtained from the OCR output, while $\mathbf{B}$ denotes the LaBSE embedding of a candidate sentence from the corresponding ground-truth transcription. Cosine similarity is used to estimate whether the two sentences represent the same textual content despite recognition errors. The resulting value is in the range of $[-1, 1]$, with -1 denoting oppositely directed vectors, 0 denoting orthogonal vectors, and 1 denoting vectors pointing in the same direction.

Sentence alignment was performed separately within each document pair. Sentence pairs were selected greedily according to their similarity scores, subject to a minimum threshold of 0.7, with each ground-truth sentence assigned at most once. Sentences for which no suitable correspondence was identified were excluded from the aligned dataset. The threshold was used as a heuristic to filter potentially unreliable matches; it was not selected through a systematic sensitivity analysis. Consequently, alignment errors and the possible exclusion of particularly difficult OCR passages remain limitations of the training data.

Figure 4 shows the distribution of LaBSE cosine similarity scores obtained from the retrospective alignment analysis of all 284 documents. The scores cover a broad range, indicating considerable variation in the similarity between OCR sentences and their corresponding ground-truth candidates. While many sentence pairs exhibit relatively high similarity scores, a substantial proportion falls below the threshold of 0.7 used in the original experiments.

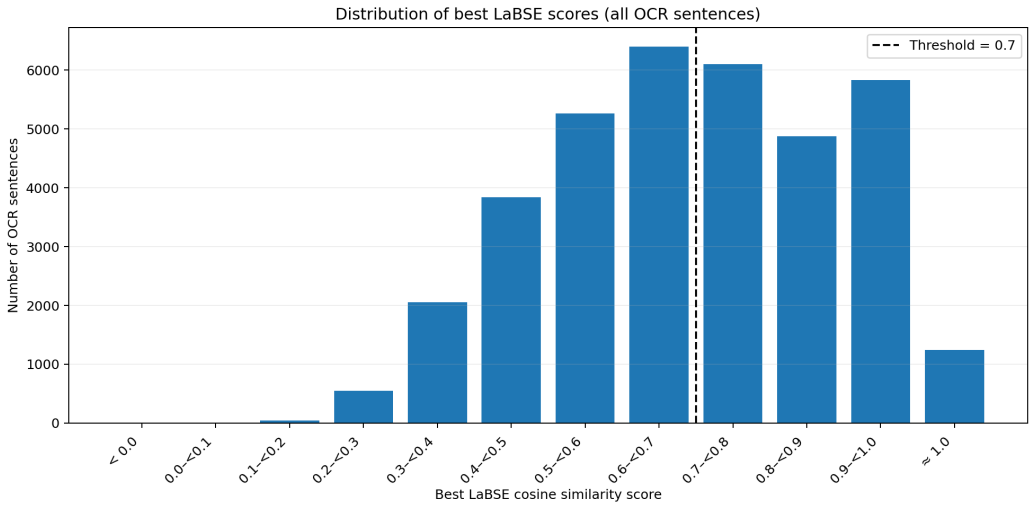


Figure 4. Distribution of LaBSE cosine similarity scores for candidate OCR–ground-truth sentence alignments across all 284 documents. The dashed line indicates the similarity threshold of 0.7 used for selecting training pairs in the original experiments. The distribution was obtained from a retrospective analysis of the complete document collection, rather than exclusively from the original training subset.

This distribution illustrates the trade-off involved in threshold-based sentence alignment. A threshold close to 1.0 would retain only nearly identical sentence representations, potentially excluding valid alignments affected by substantial OCR errors. Conversely, a threshold approaching 0.0 would admit increasingly dissimilar sentence pairs, raising the risk of incorrect alignments. The selected threshold of 0.7 therefore represents a heuristic compromise between retaining sufficient training data and limiting alignment noise. The quantitative effect of varying this threshold is examined in Table 1.

Table 1. Sensitivity analysis of the LaBSE similarity threshold. The table reports the number of accepted OCR–ground-truth sentence pairs at different cosine similarity thresholds, based on a retrospective analysis of all 284 documents.

Similarity Threshold	Accepted Sentence Pairs
0.50	25,698
0.60	21,978
0.65	19,355
0.70	**16,603**
0.75	13,804
0.80	11,018
0.85	8555
0.90	6430

Bold values indicate the threshold used in the original experiments and the corresponding number of accepted sentence pairs in the retrospective analysis.

To assess the sensitivity of the sentence alignment procedure to the similarity threshold, we retrospectively repeated the alignment analysis across all 284 documents using thresholds ranging from 0.50 to 0.90. As shown in Table 1, the number of accepted sentence pairs decreases from 25,698 at a threshold of 0.50 to 6430 at 0.90. At the threshold of 0.70 used in the original experiments, 16,603 sentence pairs were accepted across the complete corpus. This analysis quantifies the effect of the threshold on the number of retained alignments; it does not assess downstream post-correction performance at alternative thresholds.

The original document-level split yielded 10,492 aligned sentence pairs in the development subset, which were used for training and validation of the MarianMT model.

While the original conference study evaluated the model on aligned sentence pairs, the present extended study evaluates all compared methods on complete held-out documents. The document-level evaluation does not apply LaBSE-based sentence filtering to the test texts.

5.1.3. Fine-Tuning

The aligned sentence pairs are used to fine-tune a MarianMT encoder–decoder model [9]. We selected this architecture because it supports sequence-to-sequence text transformation and is comparatively compact, making it suitable for repeated post-correction of OCR output. While fine-tuning was performed on GPU-equipped server hardware in our experiments, inference can be performed separately and does not require the same training setup.

Based on the specific language(s) needed, the user must choose the appropriate base model. Since we are dealing with texts written in German, we have used the "Helsinki-NLP/opus-mt-de-de" (https://huggingface.co/Helsinki-NLP/opus-mt-de-de, accessed on 5 November 2026) model as our basis for fine-tuning. This model was pre-trained to translate from German into German, with our goal after fine-tuning being to translate from "OCR German" into correct German. For the experiments here, the following hyperparameters have been used:

- Number of epochs: 30;
- Learning rate: 5.0000×10^{-5};
- Weight decay: 0.05.

The hyperparameters were selected based on prior experience with fine-tuning sequence-to-sequence models rather than through a systematic hyperparameter search. They therefore represent a practical experimental configuration and are not claimed to be optimal. In particular, the learning rate, weight decay, and number of epochs may need to be adjusted for other document collections or OCR error distributions.

The model was fine-tuned in a supervised sequence-to-sequence setting, using OCR sentences as source sequences and their aligned ground-truth transcriptions as target sequences. Training was performed with the Hugging Face `Seq2SeqTrainer`, using the standard token-level cross-entropy objective. Source and target texts were tokenised with the MarianMT tokenizer and truncated to a maximum sequence length of 512 tokens, with dynamic padding applied within each batch. The per-device batch size was 64 for both training and validation. The training and validation split used a random seed of 42. Validation and checkpoint saving were performed after each epoch. The model obtained after completing all 30 epochs was saved for subsequent inference.

Fine-tuning was performed using two NVIDIA A100 GPUs, each equipped with 80 GB of GPU memory, in an NVIDIA DGX A100 system (The datasheet can be found here: https://images.nvidia.com/aem-dam/Solutions/Data-Center/nvidia-dgx-a100-datasheet.pdf, accessed on 5 November 2026). The fine-tuning process took approximately 36 min. The resulting model was then saved for use in the subsequent OCR post-correction experiments.

5.2. Translation

The inference pipeline follows the architecture shown in Figure 2. First, an existing OCR engine produces a textual representation of each document. For the experiments, previously generated OCR output was used directly.

The OCR text was divided into paragraphs and subsequently split into smaller segments at sentence boundaries and line breaks. Consecutive segments were greedily combined into chunks according to their token lengths, using the MarianMT tokenizer. The

maximum input length was set to 512 tokens. No individual sentence exceeded this limit in the evaluated dataset. Chunks were processed in batches of up to eight using the fine-tuned MarianMT model.

Output sequences were generated using the model's stored generation configuration, without explicitly overriding the decoding parameters. The generated sequences were decoded into text and concatenated in their original order. Paragraph boundaries were preserved when reconstructing the post-corrected document.

The generated chunks were recombined without a separate boundary-correction step. The pipeline does not perform automatic confidence estimation or verify generated text against the source document. Consequently, incorrect or spurious model output may be retained in the reconstructed document.

Because OCR output and text segmentation were already available from the preprocessing stage, these operations were reused rather than repeated during the experimental inference runs.

For comparison, the same inference procedure was also applied using the original, non-fine-tuned `Helsinki-NLP/opus-mt-de-de` model. This baseline allows us to examine the effect of task-specific fine-tuning on the historical OCR material.

Inference was performed on the NVIDIA DGX A100 system. Processing the held-out test dataset with the fine-tuned MarianMT model took approximately 40 min.

5.3. Performance Metrics

Before discussing the results of our experiments, a brief introduction to the performance metrics used is given.

5.3.1. (Normalised) Levenshtein Distance

The Levenshtein distance is a measure used to calculate the distance between two sequences. In our case, we calculate the distance between two sequences of characters. To calculate the Levenshtein distance, the minimum number of single-character edits to transform a provided sequence A, e.g., the texts after the TOOL pipeline, into the other sequence B (here, the GT texts) is counted. These single-character edits can be insertions, deletions, or substitutions. A longer text will more likely need more edits to be transformed into the other text, and thus, we have normalised the Levenshtein distance by dividing the results by the number of characters in the GT text:

$$\mathrm{NLD}(A, B) = \frac{d_{\mathrm{Lev}}(A, B)}{|B|}, \tag{2}$$

where $d_{\mathrm{Lev}}(A, B)$ denotes the character-level Levenshtein distance between the generated text A and the ground-truth text B, and $|B|$ is the number of characters in the ground-truth text.

5.3.2. Word Error Rate (WER)

The WER (Word Error Rate) is based on the Levenshtein edit distance at the word level rather than the character level. It measures the number of word substitutions, deletions, and insertions required to transform the generated text into the ground-truth transcription. As an example, "I like a sandwich" and "I like this sandwich" have four character edits but only one word substitution, which is relevant for the WER. For computing the WER, the number of word substitutions, deletions, and insertions is counted. The resulting value is then divided by the total number of words in the GT text:

$$\mathrm{WER} = \frac{S + D + I}{N}. \tag{3}$$

where the following definitions are used:

- S being the number of substitutions;
- D being the number of deletions;
- I being the number of insertions;
- N being the total number of words in the GT text.

5.3.3. Jaccard Similarity

The Jaccard similarity is a set-based similarity measure between two sets A and B. Because sets do not have an order, the similarity measure does not take the order of the words in the text into account. However, this measure can still provide interesting insights into how well a user searching for a specific term may find this term in the results. Formally, it is defined as

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{4}$$

The resulting value will be between 0 and 1, with 1 denoting identical sets and 0 meaning no overlaps at all. In order to still preserve some form of ordering, we have measured the Jaccard similarity with regard to word bigrams, i.e., the sets consist of combinations of two words as entries. This takes immediate context into account and makes the measure stricter.

5.3.4. BLEU(-4)

The BLEU (Bilingual Evaluation Understudy) score is a value between 0 and 1, and it evaluates the quality of one text compared to another. If the resulting value is getting closer to one, both texts are getting more similar. Initially, it was designed for machine translation tasks, but it can also be applied to other similar tasks. In essence, BLEU calculates how many words or n-grams in the candidate text, i.e., the text after TOOL, match those of the reference text. Detailed information can be found in its original paper by Papineni et al. [25].

We have decided to use the BLEU-4 score version that considers n-grams up to 4-g, i.e., sequences of one up to four words in length. The precision for each n-gram is calculated, and then the geometric mean over all scores is taken:

$$\text{BLEU-4} = \text{BP} \cdot \exp\left(\sum_{n=1}^{4} w_n \log p_n\right) \tag{5}$$

where the following definitions are used:

- BP is the brevity penalty;
- p_n is the precision for n-gram of size $n \in \mathbb{N}$;
- w_n is the weight for each n-gram precision (in our case, we have set it uniformly to $\frac{1}{4}$).

The brevity penalty is used to penalise translations that are too short when compared to the reference:

$$\text{BP} = \begin{cases} 1, & \text{if } t \geq g, \\ \exp\left(1 - \frac{g}{t}\right), & \text{if } 0 < t < g. \end{cases} \tag{6}$$

where the following definitions are used:

- t is the total word length of the translation;
- g is the total word length of the GT text.

Thus, if the translation is longer than the GT text, nothing happens, but if the translation is shorter, the BLEU score gets penalised exponentially.

5.3.5. METEOR

Unlike BLEU, METEOR (Metric for Evaluation of Translation with Explicit Ordering) considers both exact word matches and matches based on word stems. In our evaluation, we used the NLTK implementation of METEOR with a German stemmer and disabled WordNet-based synonym matching. This configuration accounts for morphological variation without relying on an English-language synonym resource. Further details on METEOR can be found in Banerjee and Lavie [26]. It is calculated as follows:

$$\text{METEOR} = F_{\text{mean}} \cdot (1 - \text{Penalty}). \tag{7}$$

The individual components of this equation are given as

$$\text{Precision} = \frac{m}{w_t}, \quad \text{Recall} = \frac{m}{w_{gt}}, \text{where} \tag{8}$$

- m is the number of matched unigrams;
- w_t is the number of words in the translation;
- w_{gt} the number of words in the GT texts.

Using precision and recall, the F_{mean} score can be calculated:

$$F_{\text{mean}} = \frac{\text{Precision} \cdot \text{Recall}}{\alpha \cdot \text{Precision} + (1 - \alpha) \cdot \text{Recall}}. \tag{9}$$

where $\alpha \in (0, 1)$ is a parameter controlling the relative weight of recall and precision.

With $\alpha = 0.9$, recall receives a higher weight than precision.

Finally, the penalty term is computed as

$$\text{Penalty} = \gamma \cdot \left(\frac{ch}{m}\right)^{\beta} \tag{10}$$

where the following definitions are used:

- ch is the number of chunks;
- m is the number of matches;
- β and γ being tuning parameters.

Also for the penalty term, the parameters β and γ have been kept their default values ($\beta = 3$, $\gamma = 0.5$).

5.3.6. ROUGE(-2)

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score was first described by Lin [27] in 2004. In contrast to BLEU, ROUGE is generally a measure of recall. However, it can also be F1-score based, which is the version we have used in our evaluation. Additionally, we calculated the score using word bigrams to capture more context. This F1-score version using bigrams is calculated as follows:

$$\text{ROUGE-2}_{F1} = \frac{2 \cdot P_2 \cdot R_2}{P_2 + R_2}, \tag{11}$$

where P_2 and R_2 denote bigram precision and bigram recall, respectively.

Precision and recall are as defined in Equation (8) but with word bigrams instead of unigrams.

The metrics provide complementary perspectives on post-correction quality. WER and normalised Levenshtein distance quantify edit errors at the word and character levels, respectively, whereas Jaccard similarity, BLEU, METEOR, and ROUGE-2 capture different

forms of lexical and n-gram overlap with the ground-truth transcription. All reported metrics are automatic reference-based measures; no separate human assessment of readability or semantic fidelity was conducted.

5.4. Results

The evaluation of TOOL was conducted on the held-out test dataset, which was not used during fine-tuning. The evaluation was conducted on 139 complete held-out documents without LaBSE-based sentence filtering.

For the comparison reported here, the generated outputs were evaluated against the corresponding ground-truth transcriptions using the six metrics described in Section 5.3. Scores were aggregated at the document level, and the mean and median across documents were calculated for each method.

Table 2 compares Tesseract without post-correction, the original MarianMT model, and the fine-tuned TOOL model. The base MarianMT configuration serves as a reference for assessing the effect of task-specific fine-tuning. The best mean and median values for each metric are highlighted in bold.

Table 2. Evaluation metrics for Tesseract and MarianMT-based post-correction methods. Mean and median are reported across documents. Lower values are better for normalised Levenshtein distance and WER (arrow down); higher values are better for Jaccard bigram similarity, BLEU-4, METEOR, and ROUGE-2 (arrow up).

Metric	Norm. Lev. Dist. ↓		WER ↓		Jaccard Bigrams ↑		BLEU-4 ↑		METEOR ↑		ROUGE-2 ↑	
Model	**Mean**	**Median**	**Mean**	**Median**	**Mean**	**Median**	**Mean**	**Median**	**Mean**	**Median**	**Mean**	**Median**
Tesseract only	**0.4141**	**0.1829**	0.7920	0.6529	0.1452	0.0876	0.2203	0.1548	0.4124	0.3522	0.2348	0.1701
Base MarianMT	0.4493	0.2648	0.8422	0.7155	0.1424	0.0844	0.2155	0.1470	0.4071	0.3378	0.2314	0.1593
TOOL	0.4564	0.2232	**0.5996**	**0.3867**	**0.3705**	**0.3932**	**0.4640**	**0.4731**	**0.6236**	**0.6302**	**0.5459**	**0.5914**

Bold values indicate the best result in each column.

Figure 5 shows the same data also in bar chart form. Orange bars are for the experiments with Tesseract alone, blue bars are the experiments after using the pre-trained MarianMT, and magenta bars are for TOOL. Solid bars are used for the mean, while hatched bars are for the median.

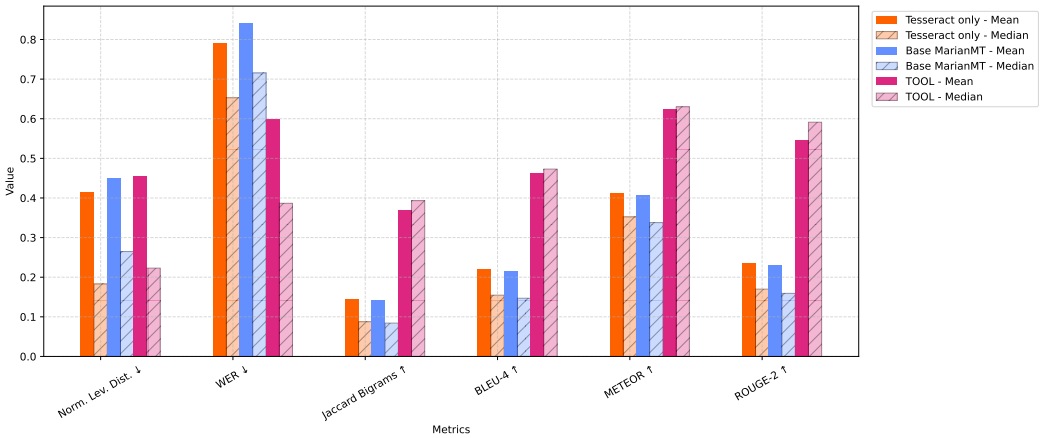


Figure 5. Evaluation metrics shown as bar charts. Tesseract is compared to base MarianMT post-correction and TOOL. Solid bars indicate the mean, while hatched bars are the median. Downward arrows indicate metrics for which lower values are better; upward arrows indicate metrics for which higher values are better.

The performance metrics are calculated as described in Section 5.3. For the columns "Normalised Levenshtein Distance" (abbreviated to "Norm. Lev. Dist.") and the "Word Error Rate" ("WER" in the table), a smaller value denotes a better performance. In contrast, for the "Jaccard Bigrams", "BLEU-4", "METEOR" and "ROUGE-2", a higher value denotes a better performance. For brevity, in this section we will refer to the BLEU-4 score as just BLEU and the ROUGE-2 score as just ROUGE.

Considering the mean document-level scores, TOOL improves five of the six evaluated metrics compared with Tesseract alone. Mean WER decreases from 0.792 to 0.600, corresponding to a relative reduction of approximately 24.3%. Jaccard bigram similarity, BLEU-4, METEOR, and ROUGE-2 also increase substantially.

However, mean normalised Levenshtein distance increases from 0.414 to 0.456, indicating worse performance at the character level. The two edit-based metrics therefore provide different assessments of the corrections. While TOOL reduces word-level errors on average, its generated output can contain character-level deviations, including repeated or unnecessary text, that increase the number of character edits required to recover the ground truth.

A document-level comparison further illustrates this discrepancy. For 62 of the 139 evaluated documents, TOOL reduced WER while increasing normalised Levenshtein distance. This indicates that improvements in word-level agreement can coexist with additional character-level deviations. Moreover, the increase in mean normalised Levenshtein distance was disproportionately influenced by a small number of documents with particularly large deteriorations. Consequently, the opposing trends in the two metrics should not be interpreted as contradictory results, but as evidence that the correction model exhibits different error characteristics at the word and character levels.

These results demonstrate an improvement in several automatic measures of textual agreement, but they do not establish that every generated correction is accurate. In particular, the deterioration in normalised Levenshtein distance highlights the importance of considering multiple metrics when evaluating generative OCR post-correction.

The comparison between Tesseract alone, the original non-fine-tuned MarianMT model, and TOOL provides a limited assessment of the contribution of task-specific fine-tuning. While applying the non-fine-tuned model increases mean WER from 0.792 to 0.842, the fine-tuned model reduces it to 0.600.

The non-fine-tuned model also performs worse than Tesseract alone across all six evaluated metrics, whereas TOOL improves five of the six metrics compared with Tesseract. These results support the importance of adapting the encoder–decoder model to the OCR error distribution of the investigated document collection.

However, this comparison does not isolate the individual effects of text segmentation, chunk packing, or document reconstruction and therefore does not constitute a comprehensive module-level ablation study.

In addition to the mean, the median was calculated across the evaluated documents to provide a complementary view of the score distributions. For all three methods, median WER and normalised Levenshtein distance are lower than their respective means, indicating that documents with comparatively high error rates increase the mean scores.

For TOOL, the median WER is 0.387, compared with a mean of 0.600, while the median normalised Levenshtein distance is 0.223, compared with a mean of 0.456. The median Jaccard, BLEU-4, METEOR, and ROUGE-2 scores are also higher than their respective means. Overall, the median results confirm the improvement of TOOL over Tesseract alone in five of the six metrics, while normalised Levenshtein distance remains worse.

5.4.1. Statistical Analysis

To assess whether the observed improvement in WER was consistent across the held-out documents, we performed a paired comparison between Tesseract alone and TOOL. The analysis used the document-level WER scores for the same 139 test documents.

Mean WER decreased from 0.792 for Tesseract alone to 0.600 for TOOL, corresponding to an absolute reduction of 0.192 and a relative reduction of approximately 24.3%. TOOL achieved a lower WER on 129 of the 139 documents.

A paired bootstrap analysis with 10,000 document-level resamples [28] yielded a 95% confidence interval of [0.163, 0.221] for the mean document-level WER reduction. A paired Wilcoxon signed-rank test [29] also indicated a statistically significant difference ($p \approx 1.5 \times 10^{-20}$).

However, this improvement did not extend to all metrics. Mean normalised Levenshtein distance increased from 0.414 to 0.456. Thus, the statistically supported improvement in word-level accuracy should not be interpreted as a corresponding improvement in character-level accuracy.

5.4.2. Qualitative Analysis

To illustrate the behaviour of TOOL beyond aggregate evaluation metrics, Tables 3 and 4 present selected excerpts from the held-out test documents. The examples include successful corrections, partial corrections, and undesirable changes introduced during generation.

Table 3. Selected OCR, TOOL, and ground-truth excerpts from held-out test documents. The examples illustrate both successful corrections and remaining deviations from the reference transcription.

Source	Text Excerpt
Example 1: Corrected historical OCR errors (`berufearchiv_1847`)	
OCR	Lehrzeit: 3 Sabre
TOOL	Lehrzeit: 3 Jahre
GT	Lehrzeit: 3 Jahre
Example 2: Corrected words and numerical information (`berufearchiv_1845`)	
OCR	Wegbinden und Einzwingen von runden und Rachen Borstzwingenpinseln bis zur Größe Nr. 10.
TOOL	Wegbinden und Einzwingen von runden und flachen Borswingenpinseln bis zur Größe Nr. 16.
GT	Wegbinden und Einzwingen von runden und flachen Borstzwingenpinseln bis zur Größe Nr. 16.
Example 3: Corrected spelling with remaining errors (`berufearchiv_1841`)	
OCR	Berufsausbildung in der Snduftrie
TOOL	Berufsausbildung In der Industrie
GT	Berufsausbildung in der Industrie

Table 4. Selected limitations of the fine-tuned TOOL model observed in held-out test documents. The excerpts illustrate residual recognition errors and repeated or spurious generated text.

Document	Observed TOOL Output and Error Behaviour
`berufearchiv_1845`	**Incomplete word correction:** “Wegbinden und Einzwingen von runden und flachen Borswingenpinseln bis zur Größe Nr. 16.” The reference contains “Borstzwingenpinseln”, which remains incorrectly generated.
`berufearchiv_478`	**Repeated and spurious output:** “für Berufeausbilacns BP. A 34” “für Berufeausbilacns BP. A 34” The generated document also contains an extended sequence of periods that is not supported by a corresponding passage in the OCR output.

The examples demonstrate that TOOL can recover words and characters affected by historical typography and OCR recognition errors. However, the corrections are not uniformly reliable. Some generated words remain different from the ground truth even when surrounding text is improved.

More severe errors can occur when the model generates repeated or spurious text. Such behaviour is particularly relevant for character-level evaluation because additional output can substantially increase edit distance. The current pipeline does not include an automatic verification or confidence-based rejection mechanism for generated corrections. Consequently, applications requiring high transcription fidelity may still require manual review of the corrected output.

5.4.3. Comparison with Other Methods of OCR Post-Correction

TOOL was also compared to other methods of OCR post-correction. The comparisons include the following:

- **Hunspell**—Hunspell is an open-source spell checker and morphological analyser that is used in a few popular open-source tools like LibreOffice or the Firefox web browser, among others [30]. In order for Hunspell to do its work, a dictionary needs to be used. For our task, we have used a German dictionary called "German (de-De frami)" (https://extensions.openoffice.org/en/project/german-de-de-frami-dictionaries, accessed on 5 November 2026). The idea is to iterate over the output from Tesseract and check whether this word is valid based on the Hunspell dictionary used. If it is not valid, Hunspell is used to search for the word that is closest to the wrong word and replace it. In contrast to all the other methods used in this article, this method is a purely dictionary-based method and does not use any form of machine learning. For the timing comparisons later on, two different approaches were tested. One method is a method where each wrong word invokes the complete Hunspell correction pipeline, while the other method uses a global cache. This cache stores the previously identified words and the used correction. Since the resulting output texts will be identical, the method without a cache is omitted for the performance metrics except for the timing.
- **Phi-4**—Phi-4 is a LLM developed by Microsoft [31]. It is available as a version with 14 billion parameters. The goal is to have a model that works as well as models with more parameters, but due to its smaller number of parameters, it can be more cost- and resource-efficient. Because of this, these models are also called SLMs (small language models). Phi-4 is only available in instruction-fine-tuned variation. The prompt for Phi-4 was as follows: *Correct the following text, which may contain errors resulting from optical character recognition (OCR). Improve spelling, grammar, and character recognition while preserving the original structure, formatting, and meaning of the text. Output only the corrected text.*
- **Qwen Vision-Language Model**—In contrast to the other methods described here, Qwen is not an OCR post-corrector. Qwen, or more specifically Qwen 2.5 VL, was used for this comparison. Qwen 2.5 VL is a vision-language transformer model [32]. The vision-language transformer does not do classical OCR. Rather, it splits up the input into patches that are encoded by the vision transformer part of Qwen VL. Afterwards, the encoded patches are further processed, together with the original user prompt, by the classical LLM part of Qwen VL. Thus, Qwen VL is not dependent on OCR but rather a full end-to-end pipeline. The concrete version of Qwen was the version using 7 billion parameters. The reason behind this is that this is the largest model that worked on the hardware used without taking a prohibitively long time. Two different prompts for Qwen have been used and compared:

- **Qwen Transcribed**—For this run, Qwen was explicitly asked only to transcribe the text. If it is unsure, the model should rather leave blanks than try to correct the output. The prompt was as follows: *You are given an image containing German text. Transcribe the complete text as accurately as possible. Preserve capitalisation, punctuation, spelling, and line breaks. If unsure, leave blank spaces in the output. Output only the transcribed text.*
- **Qwen Reconstructed**—For this run, Qwen was also asked to reconstruct if some words were damaged. The prompt was as follows: *You are given an image containing damaged German historical text. Reconstruct the intended original text. Use visible fragments together with linguistic and semantic context to restore damaged or missing characters and words. Preserve capitalisation, punctuation, and line breaks. Output only the reconstructed text.*

The evaluated methods differ in their input modalities and processing strategies. Hunspell, base MarianMT, TOOL, and Phi-4 operate on text previously recognised by Tesseract and therefore represent OCR post-correction approaches. In contrast, Qwen 2.5 VL processes the document images directly and generates text without requiring a separate OCR stage. Consequently, the Qwen configurations serve as end-to-end transcription and reconstruction baselines rather than direct text-based post-correction baselines.

All methods were evaluated on the same set of 139 held-out documents. These documents were excluded from the training and validation data used for fine-tuning TOOL, ensuring that the comparison was conducted on previously unseen evaluation material.

Table 5 presents the mean document-level scores for all evaluated methods. Figure 6 visualises a selection of these results.

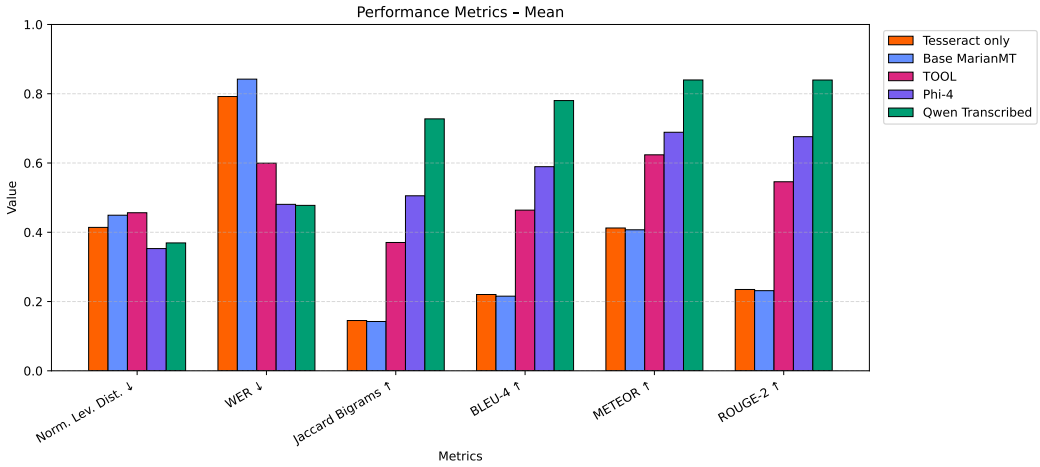


Figure 6. OCR post-corrector metrics shown in mean. Tesseract only and Qwen using the transcription prompt are shown as base-line comparison and best-case. Downward arrows indicate metrics for which lower values are better; upward arrows indicate metrics for which higher values are better.

For normalised Levenshtein distance, Phi-4 achieves the lowest mean value (0.353), followed by Qwen Transcribed (0.369). Tesseract alone achieves 0.414, while TOOL reaches 0.456. This indicates that TOOL's improvements in word-level accuracy do not extend to the mean character-level edit distance.

Qwen Transcribed achieves the lowest mean WER (0.478), closely followed by Phi-4 (0.481) and Qwen Reconstructed (0.492). Among the text-based post-correction methods, TOOL achieves a mean WER of 0.600, outperforming Hunspell (0.742) and base MarianMT (0.842).

Table 5. Mean document-level evaluation metrics for all evaluated methods. Lower values are better for normalised Levenshtein distance and WER; higher values are better for Jaccard bigram similarity, BLEU-4, METEOR, and ROUGE-2. Best results are shown in bold.

Model	Norm. Lev. Dist. ↓	WER ↓	Jaccard Bigrams ↑	BLEU-4 ↑	METEOR ↑	ROUGE-2 ↑
Tesseract only	0.4141	0.7920	0.1452	0.2203	0.4124	0.2348
Hunspell Cache	0.5083	0.7419	0.1772	0.2619	0.4726	0.3002
Base MarianMT	0.4493	0.8422	0.1424	0.2155	0.4071	0.2314
TOOL	0.4564	0.5996	0.3705	0.4640	0.6236	0.5459
Phi-4	**0.3529**	0.4806	0.5053	0.5894	0.6888	0.6760
Qwen Transcribed	0.3693	**0.4777**	**0.7274**	**0.7803**	**0.8398**	**0.8396**
Qwen Reconstructed	0.3992	0.4922	0.7143	0.7650	0.8169	0.8300

Bold values indicate the best result for each metric.

For Jaccard bigram similarity, BLEU-4, METEOR, and ROUGE-2, Qwen Transcribed achieves the highest mean scores, followed by Qwen Reconstructed. Phi-4 ranks next, followed by TOOL. Hunspell, Tesseract alone, and base MarianMT obtain lower scores across these four metrics.

Overall, the results show that the vision-language approaches achieve the strongest lexical and n-gram-based agreement with the ground truth. Among the evaluated text-based post-correction methods, Phi-4 generally achieves higher quality scores than TOOL, while TOOL substantially improves most metrics compared with the conventional and non-fine-tuned baselines.

Table 6 reports the median document-level scores, while Figure 7 provides a graphical comparison.

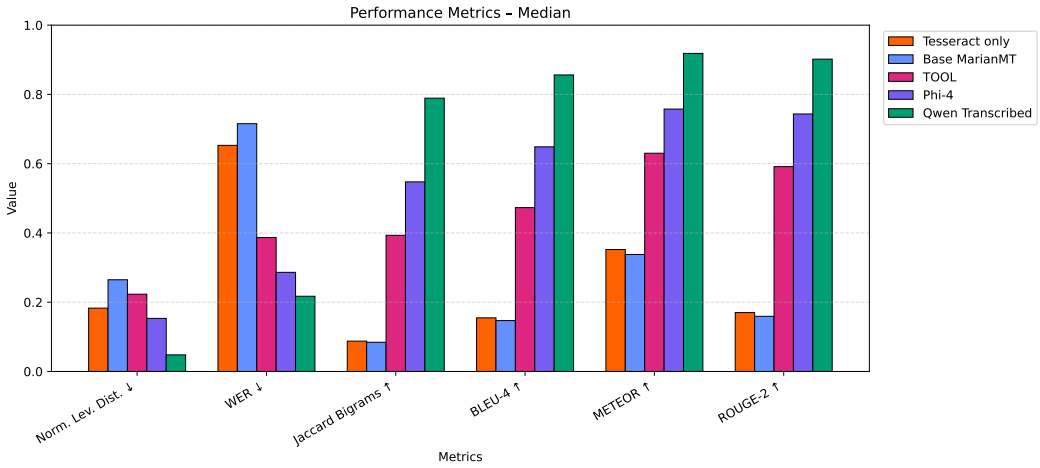


Figure 7. OCR post-corrector metrics shown in median. Tesseract only and Qwen using the transcription prompt are shown as base-line comparison and best-case. Downward arrows indicate metrics for which lower values are better; upward arrows indicate metrics for which higher values are better.

The median results broadly confirm the ranking observed for the mean scores. Qwen Transcribed achieves the best median values across all six metrics, followed by Qwen Reconstructed for most metrics. Phi-4 also performs strongly, while TOOL consistently improves WER and the n-gram-based metrics compared with Tesseract, Hunspell, and base MarianMT.

For every evaluated method, median normalised Levenshtein distance is lower than its mean. This indicates that the distributions include documents with comparatively high character-level error rates that increase the mean. However, the difference between mean

and median alone does not establish how many such documents are responsible for the observed variation.

Table 6. Median document-level evaluation metrics for all evaluated methods. Lower values are better for normalised Levenshtein distance and WER; higher values are better for Jaccard bigram similarity, BLEU-4, METEOR, and ROUGE-2. Best results are shown in bold.

Model	Norm. Lev. Dist. ↓	WER ↓	Jaccard Bigrams ↑	BLEU-4 ↑	METEOR ↑	ROUGE-2 ↑
Tesseract only	0.1829	0.6529	0.0876	0.1548	0.3522	0.1701
Hunspell Cache	0.2956	0.5723	0.1398	0.2178	0.4460	0.2644
Base MarianMT	0.2648	0.7155	0.0844	0.1470	0.3378	0.1593
TOOL	0.2232	0.3867	0.3932	0.4731	0.6302	0.5914
Phi-4	0.1533	0.2862	0.5474	0.6486	0.7576	0.7435
Qwen Transcribed	**0.0480**	**0.2172**	**0.7892**	**0.8562**	**0.9184**	**0.9018**
Qwen Reconstructed	0.0695	0.2412	0.7740	0.8359	0.8905	0.8905

Bold values indicate the best result for each metric.

Overall, TOOL occupies an intermediate position in terms of transcription quality. Qwen Transcribed achieves the best mean results for five of the six metrics, while Phi-4 achieves the lowest mean normalised Levenshtein distance. The Qwen models, however, operate directly on document images, whereas TOOL is a text-based post-correction component.

Quality metrics alone therefore do not determine which approach is most suitable for a particular application. The following section examines processing time as an additional consideration.

5.4.4. Time Consumption

Computational cost is an important consideration when selecting a transcription or post-correction method, particularly for research projects with limited hardware resources. We therefore measured the processing time of all evaluated approaches under a common hardware setup.

The experiments were conducted in a Docker container on a GPU-equipped server with two AMD EPYC 9124 CPUs (https://www.amd.com/en/products/processors/server/epyc/4th-generation-9004-and-8004-series/amd-epyc-9124.html, accessed on 14 August 2026). The container was allocated up to 32 CPU cores and 400 GB of system memory, with access to two NVIDIA A100 GPUs, each equipped with 80 GB of GPU memory.

All methods were evaluated in this environment to provide a consistent basis for comparing their observed runtimes. However, execution on the same hardware does not imply identical resource utilisation or minimum hardware requirements. In particular, the measurements should not be interpreted as benchmarks of performance on consumer laptops.

We first consider the runtime of the post-correction stage, excluding the preceding Tesseract OCR step. Table 7 reports the total processing time and the mean and median runtime per document for the five evaluated post-correction configurations.

Table 7. Runtime of the post-correction stage on the common 139-document test subset. Tesseract OCR runtime is excluded. Total runtime and mean and median runtime per document are reported.

Method	Total [min]	Mean [s/doc]	Median [s/doc]
Hunspell Cache	20.45	8.83	3.85
Hunspell No Cache	40.28	17.39	8.31
Base MarianMT	35.33	15.25	7.43
TOOL	40.48	17.47	8.32
Phi-4	251.17	108.42	50.61

Hunspell with caching is the fastest approach, requiring an average of 8.83 s per document. Without caching, its average runtime increases to 17.39 s. The cache avoids repeated dictionary suggestion calls for previously encountered words and reduces the total post-correction runtime by approximately 49%, without changing the generated output.

Base MarianMT requires 15.25 s per document, compared with 17.47 s for TOOL. The fine-tuned model is therefore somewhat slower in this experiment, although the difference is small compared with the runtime of Phi-4.

Phi-4 requires an average of 108.42 s per document for post-correction, approximately 6.2 times the runtime of TOOL. This illustrates the substantially higher inference cost of the evaluated instruction-based model under the same experimental conditions.

Table 8 reports the end-to-end runtime for the same 139 documents, with the corresponding comparison visualised in Figure 8. For the text-based post-correction methods, these measurements include both Tesseract OCR and the subsequent correction stage. The Qwen configurations instead process the document images directly.

Table 8. End-to-end runtime on the common 139-document test subset. For the post-correction approaches, the reported runtime includes Tesseract OCR and the subsequent correction stage. The Qwen-based approaches process the source documents directly. All runtime experiments were conducted on the same hardware.

Method	Total [min]	Mean [s/doc]	Median [s/doc]
Tesseract only	29.30	12.65	6.19
Hunspell Cache	49.76	21.48	10.01
Hunspell No Cache	69.58	30.03	14.10
Base MarianMT	64.63	27.90	13.21
TOOL	69.78	30.12	14.28
Phi-4	280.47	121.07	57.00
Qwen Transcribed	223.72	96.57	48.31
Qwen Reconstructed	228.03	98.43	50.63

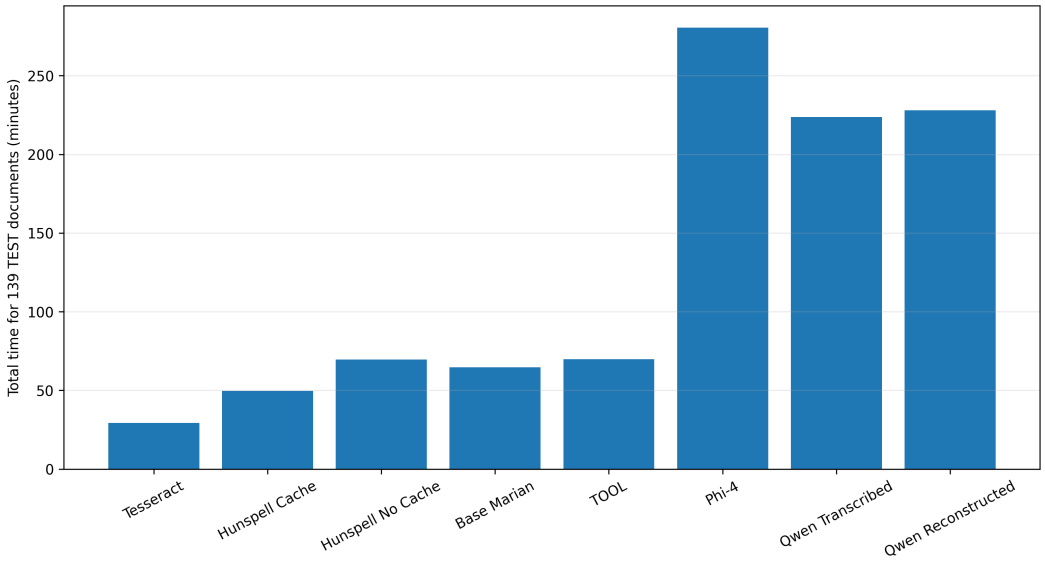


Figure 8. End-to-end runtime of all the tested methods visualised as bar charts. The values can be seen in Table 8.

Tesseract alone is the fastest configuration, requiring an average of 12.65 s per document. Hunspell with caching increases this to 21.48 s, while TOOL requires 30.12 s per document. Phi-4 has the highest mean end-to-end runtime among the evaluated methods, with 121.07 s per document.

Qwen Transcribed and Qwen Reconstructed require 96.57 and 98.43 s per document, respectively. Both are faster than the evaluated Tesseract-plus-Phi-4 pipeline, but substantially slower than TOOL. Thus, the higher transcription quality of the Qwen configurations comes with a considerable increase in processing time relative to the conventional and MarianMT-based approaches.

Figure 9 separates the Tesseract OCR runtime from the subsequent post-correction time. The OCR component is shared across all text-based post-correction methods.

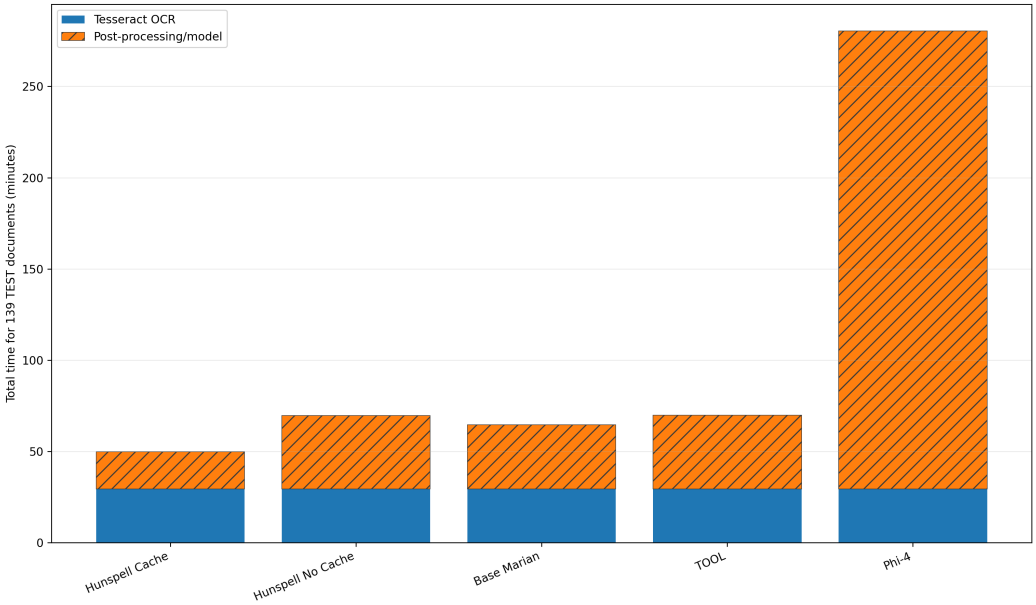


Figure 9. End-to-end processing times for all OCR post-correction methods across 139 test documents. The stacked bars distinguish the Tesseract OCR stage (solid blue) from the subsequent post-correction stage (hatched orange). Times are reported in minutes.

For Hunspell with caching, OCR accounts for the larger share of the end-to-end runtime. For base MarianMT and TOOL, the OCR and post-correction stages contribute comparable amounts of processing time. In contrast, Phi-4 inference dominates the runtime of its end-to-end pipeline.

Unlike the other evaluated configurations, TOOL requires task-specific fine-tuning before inference. For the dataset and hyperparameters used in this study, fine-tuning took 35.71 min. This is a one-time preparation cost, which is not included in the per-document inference runtimes reported above.

When processing larger document collections, the fixed fine-tuning cost is distributed across more documents. Its practical significance therefore depends on the number of documents to be processed and whether an already fine-tuned model can be reused.

5.4.5. Quality–Runtime Trade-Off

The previous evaluations considered transcription quality and runtime separately. As already mentioned previously, practical applicability is dependent on both dimensions simultaneously. If a method that provides higher transcription quality also requires significantly higher computational resources, it might become less attractive compared to methods with lower transcription quality but also significantly lower resource requirements. To take that into consideration, we additionally analysed the trade-off between those dimensions. The mean runtime per document is considered jointly with its respective mean quality metric.

Pareto-optimal methods were identified to characterise this trade-off [33]. A method is considered Pareto-optimal if no other evaluated method achieves an equal or better quality score at an equal or lower runtime, with a strict improvement in at least one of these dimensions. Methods outside the Pareto frontier are dominated by at least one alternative under the respective quality metric.

The connecting lines between Pareto-optimal points in the following figures serve only to visualise the observed Pareto frontiers. They do not represent fitted or extrapolated performance curves.

Figure 10 illustrates this trade-off for METEOR, while Figure 11 illustrates it for the WER. Both metrics provide complementary perspectives because a higher value indicates better performance for METEOR while a lower value indicates better performance for the WER. In both cases, a lower end-to-end runtime is preferable.

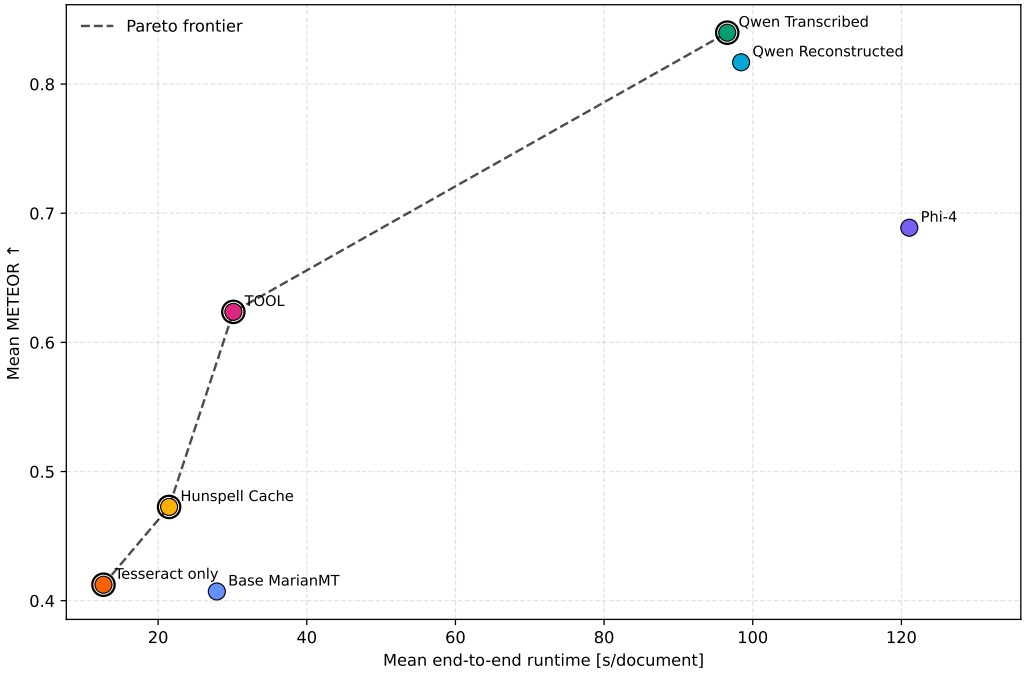


Figure 10. Mean METEOR score versus mean end-to-end runtime per document. The upward arrow indicates that higher METEOR scores are preferable. The dashed line connects the Pareto-optimal methods. No other evaluated method has an equal or lower runtime and an equal or higher METEOR score, with a strict improvement in at least one dimension.

For both WER and METEOR, four methods lie on the observed Pareto frontier: Tesseract only, Hunspell with caching, TOOL, and Qwen Transcribed. These methods represent different trade-offs between transcription quality and processing time.

Tesseract alone provides the lowest runtime, while Hunspell with caching offers a modest quality improvement at a relatively small additional computational cost. TOOL achieves a more substantial improvement in both quality metrics at a mean end-to-end runtime of 30.12 s per document. Qwen Transcribed achieves the best quality scores for both metrics, but requires 96.57 s per document on average.

The remaining evaluated configurations are dominated by at least one of these methods when considering the respective quality metric and mean end-to-end runtime.

The quality–runtime analysis shows that no single method is preferable under all practical constraints. Qwen Transcribed achieves the best mean WER and METEOR scores, whereas Tesseract alone requires the least processing time. Hunspell with caching provides a lightweight intermediate option.

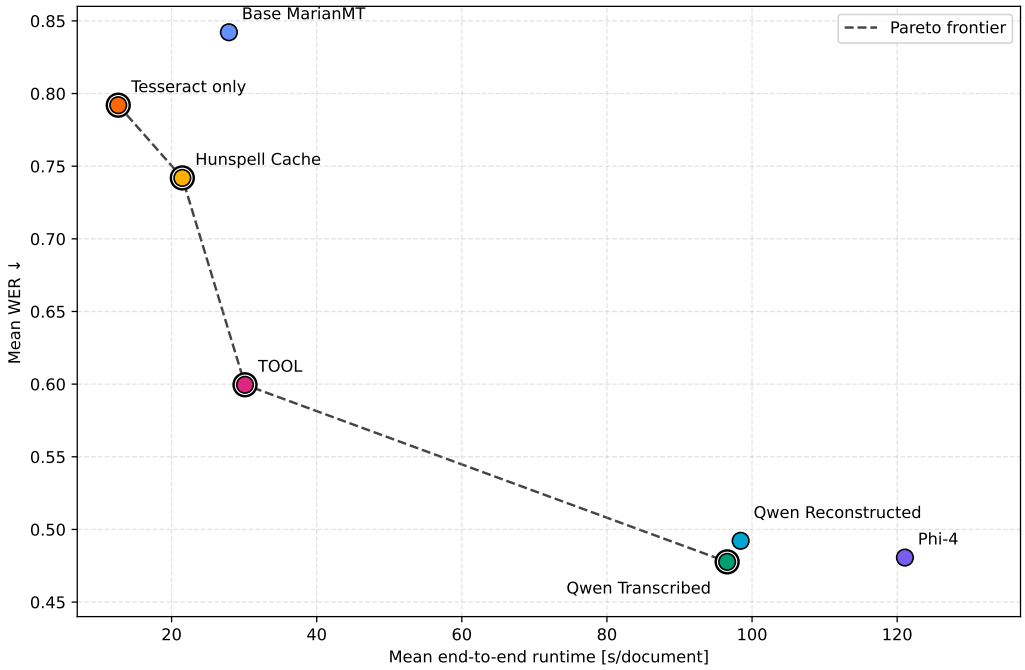


Figure 11. Mean WER versus mean end-to-end runtime per document. The downward arrow indicates that lower WER values are preferable. The dashed line connects the Pareto-optimal methods. No other evaluated method has an equal or lower runtime and an equal or lower WER, with a strict improvement in at least one dimension.

The fine-tuned TOOL pipeline occupies a favourable position between these approaches: it substantially improves word-level and n-gram-based agreement with the ground truth compared with Tesseract and Hunspell, while requiring considerably less processing time than the evaluated instruction-based and vision-language models. The most suitable approach therefore depends on the available computational resources and the required transcription quality.

5.4.6. Practical Implications

The results suggest different method choices depending on the requirements of the application. Tesseract alone provides the shortest processing time and may be sufficient when throughput is the primary concern. Hunspell with caching offers a lightweight post-correction option with limited additional processing time. TOOL provides a substantial reduction in word error rate compared with uncorrected Tesseract output while maintaining considerably shorter processing times than the evaluated larger models.

When correction quality takes precedence over runtime, Qwen Transcribed is the strongest option among the evaluated methods for most accuracy metrics. However, unlike the text-based post-correction methods, it transcribes document images directly and therefore represents an alternative to the OCR pipeline rather than a drop-in post-correction component. Phi-4 achieves the lowest mean normalised Levenshtein distance, but its comparatively long processing time makes it less attractive when word error rate or METEOR is considered together with runtime.

Consequently, the preferred method depends not only on the evaluation metric and available computational resources but also on whether an existing OCR workflow must be retained. In particular, TOOL offers a practical compromise for applications seeking improved text accuracy without replacing their OCR engine.

6. Limitations

Although the results demonstrate the feasibility of translation-based OCR post-correction, several limitations must be considered when interpreting the findings.

First, the evaluation is based on a single collection of historical German vocational education and training documents and on OCR output produced by one recognition engine, Tesseract. The held-out evaluation therefore assesses performance on unseen documents from the same collection rather than generalisation to independent corpora, other OCR engines, or additional languages and writing systems. Furthermore, performance may vary across historical periods, document layouts, and typographic characteristics, which were not evaluated as separate subgroups.

The available training data are also relatively limited, comprising 10,492 aligned sentence pairs for training and validation. Although this was sufficient to demonstrate the feasibility of the proposed approach on the investigated collection, it does not establish how performance scales with larger or more diverse training datasets. The representativeness of the aligned pairs may additionally be affected by the exclusion of heavily corrupted OCR passages during preprocessing.

Second, the training data depend on an automatic sentence-alignment procedure using LaBSE embeddings and a cosine similarity threshold of 0.7. The threshold of 0.7 was originally selected heuristically. The retrospective sensitivity analysis reported in Table 1 quantifies how the number of accepted sentence pairs changes with the threshold, but does not evaluate the effect of alternative thresholds on downstream correction performance. Because sentence pairs below the threshold are excluded, the resulting training data may underrepresent particularly noisy OCR passages. Consequently, the influence of alignment quality and threshold selection on correction performance remains insufficiently characterised.

Third, the experiments use a single training configuration and one document-level data split. Although the paired statistical analysis quantifies the observed improvement on the held-out documents, it does not establish the stability of the results across different random seeds, alternative data splits, or repeated model training runs. The reported findings should therefore be interpreted as results for the evaluated configuration rather than estimates of variability across training runs.

Fourth, improvements in word-level accuracy do not necessarily imply improvements at the character level. While TOOL reduced mean WER, mean normalised Levenshtein distance increased. The qualitative examples also reveal residual correction errors and occasional repeated or spurious output. The current pipeline generates corrections without an additional confidence estimation, verification, or rejection mechanism. It therefore cannot guarantee that every generated change preserves the content of the source document.

Finally, the runtime measurements describe the evaluated implementations and hardware environment. They provide a basis for comparing the observed quality–runtime trade-offs but do not directly establish performance on consumer hardware or under alternative deployment configurations.

7. Conclusions and Outlook

In this article, we presented TOOL, an approach that treats OCR post-correction as a sequence-to-sequence translation task from noisy recognised text to a corrected transcription. To demonstrate the feasibility of this approach, we applied TOOL to historical vocational education and training documents spanning the period from 1871 to the present day. The implementation uses a fine-tuned MarianMT encoder–decoder model to transform an "OCR German" representation into text that more closely matches the corresponding ground-truth transcriptions.

On the held-out evaluation documents, TOOL reduced mean word error rate from 0.792 for Tesseract alone to 0.600, corresponding to a relative reduction of approximately 24.3%. The approach also improved Jaccard bigram similarity, BLEU-4, METEOR, and ROUGE-2. However, mean normalised Levenshtein distance increased, and qualitative examples revealed occasional repetition and spurious text generation. In the broader comparison, Phi-4 and the Qwen vision-language configurations achieved higher accuracy on several metrics. Nevertheless, TOOL offered a favourable quality–runtime trade-off, requiring an average end-to-end processing time of 30.12 s per document compared with 96.57 s for Qwen Transcribed.

The results support the feasibility of translation-based OCR post-correction, while the present evaluation remains limited to a single historical document collection and one underlying OCR engine. Future work will investigate larger and more diverse datasets, additional languages and writing systems, and different OCR engines to assess the generalisability of the approach. Further directions include alternative Transformer architectures, automatic confidence estimation and verification of generated corrections, and integration into broader IDP workflows.

Author Contributions: Conceptualization: T.A., S.M. and M.B.; methodology: T.A. and M.B.; software: T.A. and M.B.; validation: T.A. and M.B.; formal analysis: T.A.; investigation: T.A. and M.B.; resources: T.A. and M.B.; data curation: T.A. and see Acknowledgments; writing—original draft preparation: T.A., S.M. and M.B.; writing—review and editing: T.A., S.M. and M.B.; visualization: T.A.; supervision: S.M., R.M. and M.B.; project administration: S.M. and M.B.; funding acquisition: S.M., R.M. and M.B. All authors have read and agreed to the published version of the manuscript.

Funding: The research was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2176 'Understanding Written Artefacts: Material, Interaction and Transmission in Manuscript Cultures', project no. 390893796.

Data Availability Statement: The source code for TOOL is available at GitHub at https://github.com/CHAI-UHH/Paper-TOOL (accessed on 5 November 2026). The dataset is available at Zenodo [34].

Acknowledgments: We would like to thank the German Federal Institute for Vocational Education and Training (Bundesinstitut für Berufsbildung, BIBB), and specifically Jens Dörpinghaus, for allowing us to use their data for this article. During the preparation of this manuscript, the author(s) used GitHub Copilot, DeepL and Grammarly to format TeX tables and diagrams, to translate words and to check grammar and spelling. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Doermann, D. The Indexing and Retrieval of Document Images: A Survey. *Comput. Vis. Image Underst.* **1998**, *70*, 287–298. [CrossRef]
2. Smith, R. An Overview of the Tesseract OCR Engine. In Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Paraná, Brazil, 23–26 September 2007; pp. 629–633. [CrossRef]
3. Patel, K.; Sheehan, B.; Bhada, S. Exploring Intelligent Document Processing Technology for Discovering Requirements Debt in Systems Engineering Management Plans. *Procedia Comput. Sci.* **2025**, *268*, 437–446. [CrossRef]
4. Cui, L.; Xu, Y.; Lv, T.; Wei, F. Document AI: Benchmarks, Models and Applications. *arXiv* **2021**, arXiv:2111.08609. [CrossRef]
5. Nguyen, T.T.H.; Jatowt, A.; Coustaty, M.; Doucet, A. Survey of Post-OCR Processing Approaches. *ACM Comput. Surv.* **2021**, *54*, 124. [CrossRef]
6. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.t.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NIPS '20, Proceedings of the 34th International Conference on Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2020.
7. Beshirov, A.; Dobreva, M.; Dimitrov, D.; Hardalov, M.; Koychev, I.; Nakov, P. Post-OCR Text Correction for Bulgarian Historical Documents. *Int. J. Digit. Libr.* **2025**, *26*, 4. [CrossRef]

8. Asselborn, T.; Bender, M.; Möller, R.; Melzer, S. Treating OCR Output as a Language (TOOL) – Improving OCR Output with Seq2Seq Translation. In *Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS 2025)*; Bolanowski, M., Ganzha, M., Maciaszek, L.A., Paprzycki, M., Ślęzak, D., Eds.; Annals of Computer Science and Information Systems; Polish Information Processing Society: Warsaw, Poland, 2025; Volume 43, pp. 471–478. [CrossRef]
9. Junczys-Dowmunt, M.; Grundkiewicz, R.; Dwojak, T.; Hoang, H.; Heafield, K.; Neckermann, T.; Seide, F.; Germann, U.; Aji, A.F.; Bogoychev, N.; et al. Marian: Fast Neural Machine Translation in C++. In *Proceedings of the ACL 2018, System Demonstrations*; Liu, F., Solorio, T., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2018; pp. 116–121. [CrossRef]
10. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. *arXiv* **2014**, arXiv:1409.3215.
11. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2015**. [CrossRef]
12. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**. [CrossRef]
13. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. 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*; Jurafsky, D., Chai, J., Schluter, N., Tetreault, J., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 7871–7880. [CrossRef]
14. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805,
15. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. *Improving Language Understanding by Generative Pre-Training*; Technical Report; OpenAI: Francisco, CA, USA, 2018.
16. Boros, E.; Ehrmann, M.; Romanello, M.; Najem-Meyer, S.; Kaplan, F. Post-Correction of Historical Text Transcripts with Large Language Models: An Exploratory Study. In *Proceedings of the 8th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature (LaTeCH-CLfL 2024)*; Bizzoni, Y., Degaetano-Ortlieb, S., Kazantseva, A., Szpakowicz, S., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 133–159. [CrossRef]
17. Guan, S.; Lin, M.; Xu, C.; Liu, X.; Zhao, J.; Fan, J.; Xu, Q.; Greene, D. PreP-OCR: A Complete Pipeline for Document Image Restoration and Enhanced OCR Accuracy. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 15413–15425. [CrossRef]
18. Busemeyer, M.R.; Trampusch, C. *The Political Economy of Collective Skill Formation*; Oxford University Press: Oxford, UK, 2011. [CrossRef]
19. Cedefop. *Vocational Education and Training in Europe, 1995–2035: Scenarios for European Vocational Education and Training in the 21st Century*; Number 114 in Cedefop Reference Series; Publications Office of the European Union: Luxembourg, 2020. [CrossRef] [PubMed]
20. Bundesregierung der Bundesrepublik Deutschland. *Berufsbildungsgesetz (BBiG) in der Neufassung vom 23. März 2005*; Bundesgesetzblatt Teil I Nr. 20: Bonn, Germany, 2005; pp. 931–968.
21. Bundesregierung der Bundesrepublik Deutschland. *Handwerksordnung (HwO) in der Neufassung vom 24. September 1998*; Bundesgesetzblatt Teil I Nr. 67: Bonn, Germany, 1998; pp. 3074–3111.
22. Asselborn, T.; Dörpinghaus, J.; Kausar, F.; Möller, R.; Melzer, S. Enhancing Text Recognition of Damaged Documents through Synergistic OCR and Large Language Models. In *Annals of Computer Science and Intelligence Systems, Volume 41*; Polish Information Processing Society: Warsaw, Poland, 2024; pp. 29–36. [CrossRef]
23. Loper, E.; Bird, S. NLTK: The Natural Language Toolkit. In *ETMTNLP '02, Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics—Volume 1*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2002; pp. 63–70. [CrossRef]
24. Feng, F.; Yang, Y.; Cer, D.; Arivazhagan, N.; Wang, W. Language-agnostic BERT Sentence Embedding. *arXiv* **2022**, arXiv:2007.01852.
25. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. Bleu: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*; Isabelle, P., Charniak, E., Lin, D., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2002; pp. 311–318. [CrossRef]
26. Banerjee, S.; Lavie, A. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*; Goldstein, J., Lavie, A., Lin, C.Y., Voss, C., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2005; pp. 65–72.
27. Lin, C.Y. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the Text Summarization Branches Out*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2004; pp. 74–81.
28. Efron, B.; Tibshirani, R.J. *An Introduction to the Bootstrap*; Monographs on Statistics and Applied Probability; Chapman & Hall: London, UK, 1993; Volume 57.
29. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biom. Bull.* **1945**, *1*, 80–83. [CrossRef]
30. Németh, L.; Foundation, F. Hunspell. Available online: https://hunspell.github.io/ (accessed on 5 November 2026).

31. Kamar, E. Introducing Phi-4: Microsoft's Newest Small Language Model Specializing in Complex Reasoning. 2024. Available online: https://techcommunity.microsoft.com/blog/aiplatformblog/introducing-phi-4-microsoft%E2%80%99s-newest-small-language-model-specializing-in-comple/4357090 (accessed on 15 February 2025).
32. Bai, S.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; Song, S.; Dang, K.; Wang, P.; Wang, S.; Tang, J.; et al. Qwen2.5-VL Technical Report. *arXiv* **2025**, arXiv:2502.13923.
33. Ehrgott, M. *Multicriteria Optimization*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2005. [CrossRef]
34. Steiner, P.; Waechter, M.; Dörpinghaus, J. *BIBB Berufearchiv*; Zenodo: Geneva, Switzerland, 2024. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.