Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 471–478
DOI: 10.15439/2025F1103
ISSN 2300-5963 ACSIS, Vol. 43

Treating OCR Output as a Language (TOOL) – Improving OCR Output with Seq2Seq Translation

Thomas Asselborn*†, Magnus Bender‡, Ralf Möller*, Sylvia Melzer*†
* University of Hamburg, Institute for Humanities-Centered AI, Warburgstraße 28, 20354 Hamburg, Germany
Email: {thomas.asselborn, ralf.moeller, sylvia.melzer}@uni-hamburg.de
https://orcid.org/0009-0005-3011-7626, https://orcid.org/0000-0002-1174-3323, https://orcid.org/0000-0002-0144-5429
† University of Hamburg, Centre for the Study of Manuscript Cultures, Warburgstraße 26, 20354 Hamburg, Germany
‡ Aarhus University, Department of Management, Fuglesangs Allé 4, 8210 Aarhus V, Denmark
Email: magnus@mgmt.au.dk, https://orcid.org/0000-0002-1854-225X

***Abstract*—Optical Character Recognition (OCR) systems are frequently used to digitise text, but often produce noisy results, especially with historical, poor-quality or multilingual data. Despite advances in OCR technology, post-processing remains a significant bottleneck. We propose TOOL (Treating OCR Output as a Language), a new approach that understands OCR correction as a machine translation task. By treating noisy OCR text as a language in its own right, TOOL employs sequence-to-sequence models like Marian to translate it into clean, standardised text. This method is scalable, model-independent and language-flexible. We demonstrate this approach by translating "OCR German" to Standard German from around 1871 to the present day, improving accuracy at the token level by using matched training pairs of OCR output and base text.**

I. Introduction

OCR (**O**ptical **C**haracter **R**ecognition) systems are widely used to digitise printed and handwritten documents. However, their output is often noisy, particularly when dealing with low-quality scans, historical materials or multilingual content. While recent advancements in computer vision and language modelling have enhanced OCR (**O**ptical **C**haracter **R**ecognition) accuracy, post-processing remains a critical bottleneck, especially in error-prone scenarios. OCR text recognition errors tend to follow systematic, learnable patterns, similar to those found in low-resource language translation or noisy text correction.

TOOL (**T**reating **O**CR **O**utput as a **L**anguage) introduces a novel paradigm: treating OCR output as a textual language that can be translated into clean text (free from errors) using Seq2Seq (**Seq**uence-**to**-**Seq**uence) models. By leveraging translation models such as BART (**B**idirectional and **A**uto-**R**egressive **T**ransformers) [14], Marian [13] or other LLMs (**L**arge **L**anguage **M**odels), TOOL frames OCR correction as a machine translation task. This enables robust, model-agnostic and domain-adaptable post-processing. The approach offers a scalable and language-independent framework for improving OCR results without requiring any modifications to the OCR engine itself.

The research was 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.

There is a need for such a flexible, general-purpose solution that can learn to correct OCR errors in a data-driven manner. This approach addresses the problem by training Seq2Seq models on aligned pairs of noisy OCR output and the corresponding GT (**G**round **T**ruth) text. In this paper, we specifically address the task of translating from "OCR German" to standard German from 1871 to the present day to improve token accuracy.

II. Related Work

Seq2Seq modelling plays a foundational role in tasks such as machine translation, text summarisation, and now, OCR post-correction. The original Seq2Seq framework was introduced by Sutskever et al. (2014) [24], who proposed using RNs (**R**ecurrent **N**etwork**s**) with LSTM (**L**ong **S**hort-**T**erm **M**emory) units to encode an input sequence into a fixed-length vector and decode it into an output sequence. This approach demonstrated promising results for translations but had difficulties with long sequences due to information bottlenecks in the fixed-size context vector. To address this limitation, Bahdanau et al. (2015) [3] introduced the *attention mechanism*, allowing the decoder to access different parts of the input sequence dynamically. This innovation significantly improved translation quality and became a core component of subsequent Seq2Seq models.

The Transformer architecture by Vaswani et al. (2017) [26] replaced recurrence entirely with self-attention mechanisms, enabling more efficient training and better performance on long sequences. Transformers rapidly became the standard architecture for Seq2Seq tasks. Building on the Transformer, Lewis et al. (2020) [14] introduced BART, a denoising autoencoder for Seq2Seq generation. BART combines a bidirectional encoder (similar to BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) [9]) with an autoregressive decoder (similar to GPT (**G**enerative **P**re-trained **T**ransformer) [21]), making it well-suited for text generation, correction and translation tasks; particularly when input text is noisy, such as in OCR outputs.

Around the same time, Marian [13] was developed as a highly efficient, production-grade translation system optimised for multilingual environments. As shown by Junczys-

Dowmunt et al. (2018) [12], Marian is particularly effective in low-resource settings, making it a strong candidate for OCR correction across diverse languages.

III. Use Case

This section begins by outlining the background of education, followed by an examination of the data availability concerning VET (**V**ocational **E**ducation and **T**raining), which serves as the primary data source and use case for the proposed TOOL approach.

A. Education Stages

Education comprises a sequence of structured stages, each contributing uniquely to an individual's personal and intellectual development [19]. The stages usually begin with ECEC (**E**arly **C**hildhood **E**ducation and **C**are), which includes programmes and services for children from birth to five years of age. This period is vital for the development of foundational cognitive, social and emotional skills. Evidence suggests that high-quality ECEC programs can have long-term positive effects on academic performance and social integration [17], [25].

Following ECEC, school education is generally categorised into primary and secondary levels. Primary education lays the groundwork for essential academic skills such as reading, writing, mathematics and science while fostering critical thinking and communication. Secondary education expands on this base, introducing more complex subjects and sometimes offering specialised or vocational tracks to prepare students for higher education or employment [10].

Higher education encompasses academic institutions like universities and colleges that confer undergraduate and postgraduate qualifications. These institutions provide in-depth disciplinary expertise and facilitate the development of research, innovation and professional competencies. Higher education is thus a significant driver of socio-economic growth and individual career advancement [1].

VET is a parallel pathway focusing on practical and occupational skills, tailored to specific sectors such as healthcare, manufacturing, hospitality and technology. VET programs often involve work-based learning, such as apprenticeships, to ensure graduates meet industry needs. These programs also support lifelong learning by offering reskilling and upskilling opportunities, thereby enhancing workforce adaptability [7], [8].

CVET (**C**ontinuing **V**ocational **E**ducation and **T**raining) plays a crucial role in supporting ongoing employability and national competitiveness. Many European countries, including Germany, have implemented policies to support lifelong learning and professional development through national strategies [22]. 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) [6] and the Crafts Code (German: *Handwerksordnung*, HwO) [5]. In contrast, other forms of CVET are typically more decentralised and less regulated, though equally important for continuous skill development.

B. OCR Processing of Documents for VET

Fig. 1 shows a document which is part of a collection of historical VET. It is just one example of many similar materials that outline training regulations, exam requirements and pedagogical standards for skilled trades in early 20th-century Germany. These documents were typically issued or supervised by state institutions such as the Imperial Institute for Vocational Training in Trade and Industry (German: *Reichsinstitut für Berufsausbildung in Handel und Gewerbe*), which played a central role in standardising vocational training across the German Reich. The aim is to process such

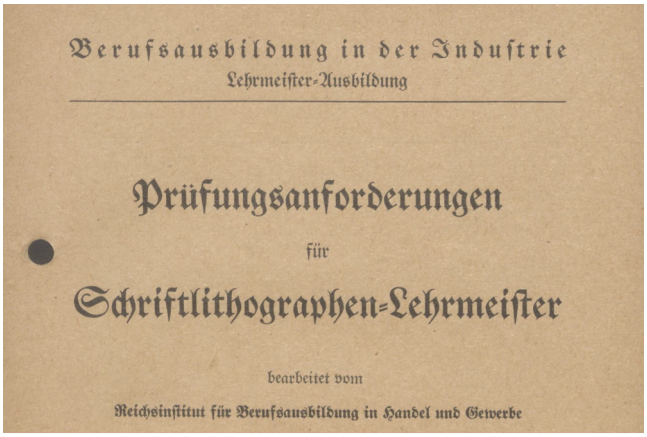

Fig. 1. Exam requirements for a type lithographer teacher (orig. in German: *Prüfungsanforderungen für Schriftlithographen-Lehrmeister*).

documents using OCR to make them digitally searchable and analysable. By converting these historical printed documents into machine-readable text, we aim to simplify research into the development of vocational training systems, qualification frameworks and occupation-specific pedagogical methods. However, processing these materials presents significant technical challenges.

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 are very different from modern Latin script. These fonts have unusual letter forms (e.g., long s "ſ" versus short s "s") and dense ligatures that are poorly recognised by standard OCR engines. 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. Therefore, in the following, we show a new approach to dealing with these problems without having to change the OCR process itself.

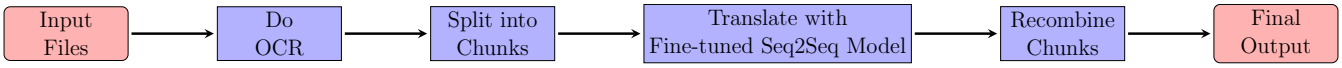

Fig. 2. Process flow for the TOOL-based OCR translation pipeline. *Input files* are first passed through the *Do OCR* process. Then the following processes are executed: *Split into Chunks*, *Translate with Fine-tuned Seq2Seq Model*, and *Recombine Chunks*. Together, these steps produce the *Final Output*.

IV. 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 in a manner that is as automated as possible. Additionally, a limited set of manually transcribed documents is available, which can be used for training purposes. 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 of a full custom OCR model. However, we recognise that the not-satisfying OCR output follows its own characteristics. Typical and recurring errors are, e.g., (i) the Umlaut "ü" in a scanned document often results in the sequence "ii" in the recognized text or (ii) the long s "ſ" is recognized as "f".

We observe that the OCR output is not correct German, but some type of inherent *language*. 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, German in our case. Thus, TOOL consists of two steps: the OCR step and the *translate* step.

The OCR step does not require any further training of models and generalises well for different OCR models and languages. Even though the *translate* step still requires an individually trained model, this training requires only a small amount of training data and resources.

State-of-the-art translation models are based on LLMs, just like the translation step of TOOL. There are many different methods for solving the OCR problem described above, several of which we briefly outline here. These include a) Encoder, b) Instruction LLM, and c) Seq2Seq.

a) Encoder: One potential approach involves the use of an encoder-only model such as BERT. 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 (**R**etrieval **A**ugmented **G**eneration) 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.

b) Instruction LLM: Another possibility is to apply an instruction LLM, like Llama or GPT, similar to [2]. Instead of training a model for TOOL's translation step, the instruction LLM gets 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 pretty large model, and running it requires special hardware and a lot of resources.

c) Seq2Seq: After all, TOOL's translation step should use a small transformer based LLM, i.e., a model taking a sequence of words as input and returning a transformed, e.g., translated, version of the words. A popular choice for such Seq2Seq models is Marian [13]. 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. Following a thorough evaluation, we selected the Seq2Seq model Marian, or more specifically, its Python implementation called MarianMT[1], to perform the translation step within TOOL.

The full TOOL pipeline is shown in Fig. 2. On the left-hand side, the red box depicts the input files, i.e., the scanned VET documents. The first two purple boxes depict TOOL's first step, i.e., the OCR. The remaining three purple boxes represent TOOL's second step, the translation using a custom fine-tuned MarianMT. As the input length of MarianMT is limited, the OCR output is split into chunks before it is translated by MarianMT. Finally, the translated chunks are concatenated, and the results are ready to be stored (the red box on the right-hand side).

Overall, the big advantages of the TOOL approach are:

- Any OCR tool can be used in the first step.
- For the second step, a pre-trained Seq2Seq LLM for translation, like variants of MarianMT, can be used.
- 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 [2].

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

[1]https://huggingface.co/docs/transformers/model_doc/marian

V. Experiments and Results

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

A. Preprocessing and Fine-Tuning

A process flow diagram for fine-tuning the model can be seen in Fig. 3. The experimentation setup does not serve as a mean to find out the best possible models and parameters to use but rather it aims to show that the method works in practical scenarios.

a) OCR: As a first step, OCR is performed on all files from our dataset. We have decided to use Tesseract [23] as our OCR engine of choice as it is open source and widely adopted, with libraries available for many programming languages like Python[2].

b) Preprocessing for Fine-Tuning: The files are randomly split into a dataset for training and validation, and another dataset for testing after training. The second testing dataset is not used during the training phase, but to compute the metrics later on to verify how well the model generalises. For this experiment, we have split the dataset to have around 50% of the text files in the training and validation dataset and the other 50% in the testing dataset. Thus, each dataset has around 145 text files in total.

As a next step, the texts in each file are split into separate sentences using the NLTK (**N**atural **L**anguage **T**ool**k**it) [16] punkt tokeniser. This is done for the texts from OCR and the GT texts to have texts that can be used for fine-tuning the MarianMT model.

For the final preprocessing step, the corresponding sentences from the GT (see Fig. 2) and after the OCR process need to be aligned in order to have the corresponding sentence pairs. To do so, we have used a LaBSE (**L**anguage-**a**gnostic **B**ERT **S**entence **E**mbedding) model [11] which is based on the BERT architecture. This language-agnostic model allows us to encode texts from different languages into the same vector space. Using a distance measure, like the cosine similarity, we can pair sentences that are close in this vector space. The cosine similarity, as used, is defined as

$$\text{cosine similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}, \text{ where} \tag{1}$$

- $\mathbf{A}$ and $\mathbf{B}$ are two n-dimensional vectors, $n \in \mathbb{N}$,
- $\mathbf{A} \cdot \mathbf{B}$ is the dot product between two vectors $\mathbf{A}$ and $\mathbf{B}$ and
- $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ being the Euclidean norms of $\mathbf{A}$ and $\mathbf{B}$.

The vectors $\mathbf{A}$ and $\mathbf{B}$ are the embeddings of the GT texts and after the OCR process. The resulting value is in the range of $[-1, 1]$, with -1 denoting exactly opposite vectors, 0 denoting orthogonal vectors and 1 denoting exactly the same vectors.

[2]https://pypi.org/project/pytesseract/

Since our goal is to have sentence pairs that are highly likely to correspond, yet they may differ in a few aspects, we have set a minimum cosine similarity between the pairs to 0.7. In the end, we obtained approximately 10000 sentence pairs for training and verification during training and 8000 sentence pairs for testing afterwards.

c) Fine-Tuning: After generating the sentence pairs, fine-tuning is possible. We have chosen to use a MarianMT model [13] because this class of models is relatively small and lightweight. This not only ensures that fine-tuning is possible on moderate hardware but also makes inference possible on modern consumer computers.

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"[3] 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: 5×10^{-5} and
- weight decay: 0.05.

The hyperparameters were chosen based on past experiences and are thus not necessarily the ones that provide the best results. Instead, the goal is to show that the method does work in general. Based on the specific use case, the hyperparameters need to be found, e.g., using Optuna[4]. For the fine-tuning, 90% of the training dataset has been used for training while the remaining 10% has been used for validation.

Fine-tuning was performed on a single Nvidia DGX A100 with 80GB of video RAM[5]. After around 1.5 hours, fine-tuning was finished, and the resulting model was stored to be used in the translation pipeline.

B. Translation

The translation pipeline follows the same process flow described in Fig. 2. In the first step, standard OCR needs to be performed on the text corpus. This output then needs to be split up into chunks of appropriate length. The reason behind splitting up is that most Seq2Seq models are only able to process a certain number of tokens. This value needs to be researched by the user and is set to 512 tokens for MarianMT. One possible way to generate these chunks is to split the input into sentences. Since the experiments performed for this article already did both steps for the pipeline presented prior, OCR and splitting up were not done in the translation pipeline, where it would have been typically done in a real-world scenario. The next step is then the translation, which is done with the fine-tuned MarianMT model. Additionally, for comparison in this work, translation was also done using the base MarianMT that is pre-trained for translating German into

[3]https://huggingface.co/Helsinki-NLP/opus-mt-de-de

[4]https://optuna.org/

[5]The datasheet can be found here: https://images.nvidia.com/aem-dam/Solutions/Data-Center/nvidia-dgx-a100-datasheet.pdf

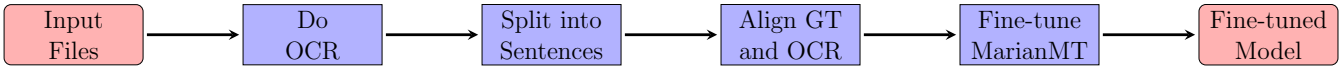

Fig. 3. Process flow for OCR to MarianMT fine-tuning pipeline. *Input files* are first passed through the *Do OCR* process. Then the following processes are executed: *Split into Sentences*, *Align GT and OCR*, and *Fine-tune MarianMT*. Together, these steps produce the *Fine-tuned Model*.

German, but not with the special cases we have in our corpus, like the ſ. After translating the text sentence by sentence, the results need to be recombined to get the full text back again.

This translation step was performed on a standard MacBook Pro 14 inch with M3 processor and 16GB of RAM[6]. For all the examples from the testing dataset, translation took around 1.5 hours.

C. Performance Metrics

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

a) Hunspell Errors: One of the goals of TOOL is to have valid German afterwards. Thus, we have used Hunspell to verify that. 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 [18]. 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)"[7]. The number of errors given by Hunspell based on this dictionary are counted. To have a value that then also accounts for the length of the text, as longer texts are more likely to have a higher number of errors produced, the resulting value is then divided by the total number of words in the GT text.

$$\text{Hunspell value (text)} = \frac{E}{N}, \quad \text{where} \tag{2}$$

- E is the number of Hunspell-detected spelling errors and
- N is the number of words in the GT text.

b) (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. Similar to the case with the Hunspell errors, 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.

c) Word Error Rate (WER): The WER (**W**ord **E**rror and **R**ate) is calculated in a similar way to the Levenshtein distance. In contrast to the Levenshtein distance, though, the WER counts the minimum number of words needed to change a text A into another text B. As an example, "I like a sandwich" and "I like this sandwich" have three character errors but only one word error, which is viewed at 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.

$$WER = \frac{S + D + I}{N}, \text{where} \tag{3}$$

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

d) 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:

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

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 regards 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.

e) BLEU(-4): The BLEU (**Bil**ingual **E**valuation **U**nderstudy) 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. [20].

We have decided to use the BLEU-4 score version that considers n-grams up to 4-grams, 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.

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

- BP is the brevity penalty,
- p_n the precision for n-gram of size $n \in \mathbb{N}$ and

[6]For technical specifications, visit https://web.archive.org/web/20250518115318/https://support.apple.com/en-gb/117735

[7]https://extensions.openoffice.org/en/project/german-de-de-frami-dictionaries

- w_n 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.

$$\text{BP} = \begin{cases} 1 & \text{if } t > g \\ \exp\left(1 - \frac{g}{t}\right) & \text{if } t \leq g \end{cases}, \text{where} \tag{6}$$

- t is the total word length of the translation and
- 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.

f) METEOR: Unlike BLEU, the METEOR (**M**etric for **E**valuation of **T**ranslation with **E**xplicit **Or**dering) score is not only looking at exact matches, but it is extended to also include stem matching (e.g., "read" and "reading") and synonym matching (e.g., "bike" and "bicycle"). More details can be found in its original paper by Banerjee and Lavie [4]. It is calculated as follows:

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

The individual components of this equation are given as:

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

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

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

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

- $\alpha \in (0, 1)$ being a parameter giving weight to recall over precision.

By default, α is set to 0.9, meaning that recall is weighted slightly above precision.

Finally, the penalty term is computed as:

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

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

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

g) ROUGE(-2): The ROUGE (**R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation) score was first described by Lin [15] 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 have calculated the score using text bigrams to capture more of the context of the words. This F1-score version using bigrams is calculated as follows:

$$\text{ROUGE-F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{11}$$

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

D. Results

The evaluation for TOOL was performed on the previously held-back testing dataset. This dataset contains around 8000 sentence pairs consisting of the texts after OCR, base MarianMT or TOOL respectively, and the corresponding GT sentences. All the text used for this dataset was chosen at random from our text corpus. Using that dataset, eight performance metrics were calculated. First, they were calculated for each sentence individually and then combined using both the mean and the median. Table I shows all the metrics for all the test datasets, all eight performance metrics and both the mean and median. Row "Tesseract only" shows the performances for only using the Tesseract OCR engine. In row "Base MarianMT" are the results shown for using the base MarianMT German into German translation model after the Tesseract OCR. This is done to compare the performance of the pre-trained MarianMT with both Tesseract alone and also TOOL. Finally, the row "TOOL" contains the performances for the approach presented in this article. The best values for both mean and median for each performance metric are highlighted in boldface.

Figure 4 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.

The performance metrics are calculated as described in Subsection V-C. For the columns "Hunspell Errors", "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.

At first, we look at the mean of each performance metric. Overall, it can be seen that the TOOL pipeline improves performance for all performance metrics. The number of errors identified using Hunspell was, on average, almost halved. Also, the Jaccard similarity for word bigrams as well as the BLEU and ROUGE scores doubled compared to the baseline Tesseract only. The normalised Levenshtein distance also got reduced. All of this indicates that the resulting text after TOOL gets closer to the GT, showing that the pipeline works.

Comparing the results after Tesseract alone and TOOL to the base pre-trained MarianMT was done in order to judge how much of the ability to correct the OCR errors is already inherent to the base model and how much was added via fine-tuning. Additionally, this serves as a baseline to judge whether the still relatively small dataset for fine-tuning has an impact on performance or not.

The MarianMT base model is consistently outperformed by our TOOL pipeline, demonstrating that fine-tuning sig-

TABLE I
EVALUATION METRICS FOR DIFFERENT MODELS (MEAN, MEDIAN).

Metric / Model	Hunspell Errors		Norm. Lev. Dist.		WER		Jaccard Bigrams		BLEU-4		METEOR		ROUGE-2	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median
Tesseract only	0.3670	0.4568	0.2275	0.1834	0.5357	0.5347	0.2079	0.1435	0.3047	0.2559	0.5977	0.5710	0.2913	0.2270
Base MarianMT	0.3455	0.4311	0.3387	0.3090	0.6252	0.5975	0.1975	0.1383	0.2338	0.1749	0.5083	0.4737	0.2755	0.2150
TOOL	**0.1963**	**0.1925**	**0.2175**	**0.1587**	**0.3626**	**0.2716**	**0.4723**	**0.4954**	**0.5695**	**0.6211**	**0.7761**	**0.8052**	**0.5781**	**0.6053**

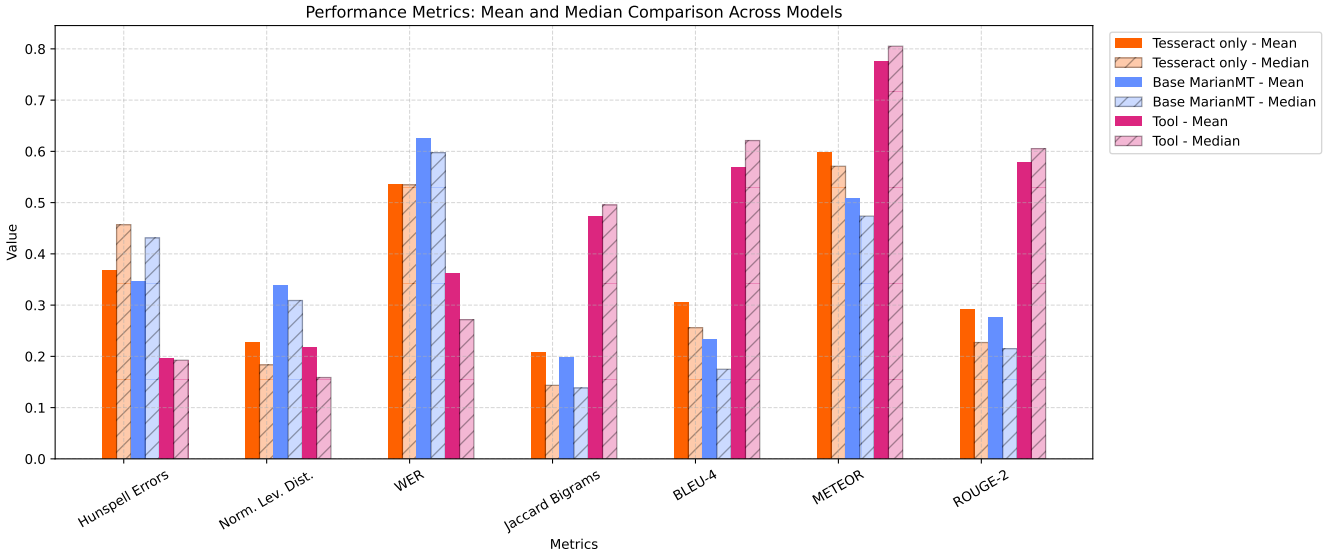

Fig. 4. Evaluation metrics shown as bar charts. Solid bars indicate the mean, while hatched bars are the median.

nificantly improves the results, even with a relatively small dataset. Moreover, the base MarianMT is also almost always worse than just Tesseract alone. It only outperforms Tesseract alone in the number of Hunspell errors, which is an indicator that the language produced after the translation is closer to German than the result after OCR alone. However, in correcting these mistakes, the results diverge from the GT leading to overall worse performance. Thus, using the base MarianMT without first fine-tuning to this specific OCR task is not helpful and even degrades results.

In addition to the mean, we also calculated the median value of each performance metric over all results. This was done in order to judge whether a few examples are skewing the results in one or the other direction. Looking at all performance metrics, the median is mostly worse than the mean for Tesseract only, but also for the base MarianMT model. The two exceptions are the normalised Levenshtein distance and the word error rate, where the results are getting slightly better. This is an indicator that there are a few outliers skewing the results slightly in a mostly worse direction. For our TOOL pipeline, the value is, however, almost always very close in both mean and median. The trend here is the same, with base MarianMT producing the worst results, while Tesseract alone performs slightly better, apart from the Hunspell errors, as already described in the mean value, and TOOL is the best.

VI. CONCLUSION AND OUTLOOK

In this article, we introduced TOOL, a novel method that reconceptualises OCR correction as a translation task from noisy to clean text. We demonstrated TOOL, which uses the Seq2Seq model MarianMT to translate the "OCR German" output into Standard German from around 1871 to the present day. We implemented TOOL with this lightweight model that runs efficiently on consumer-grade hardware. While the results are promising and show performance improvements, the current study relies on a relatively small dataset and serves as a proof of concept.

Future work will focus on scaling to larger datasets, which is expected to enhance performance and robustness. Extending this approach to other languages also represents a promising direction for future research.

REFERENCES

[1] Altbach, P.G., Reisberg, L., Rumbley, L.E.: Trends in Global Higher Education: Tracking an Academic Revolution. UNESCO Publishing, Paris (2009), https://unesdoc.unesco.org/ark:/48223/pf0000183219, a report prepared for the UNESCO 2009 World Conference on Higher Education

[2] Asselborn, T., Dörpinghaus, J., Kausar, F., Möller, R., Melzer, S.: Enhancing Text Recognition of Damaged Documents through Synergistic OCR and Large Language Models, pp. 29–36. Polish Information Processing Society (Sep 2024). https://doi.org/10.15439/2024F7400

[3] Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate (2016), https://arxiv.org/abs/1409.0473

[4] Banerjee, S., Lavie, A.: METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In: Goldstein, J., Lavie, A., Lin, C.Y., Voss, C. (eds.) Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. pp. 65–72. Association for Computational Linguistics, Ann Arbor, Michigan (Jun 2005), https://aclanthology.org/W05-0909/

[5] Bundesregierung der Bundesrepublik Deutschland: Handwerksordnung (HwO) in der Neufassung vom 24. September 1998. Bundesgesetzblatt Teil I Nr. 67, Bonn (1998), http://www.bgbl.de/xaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl&jumpTo=bgbl198s3074.pdf

[6] Bundesregierung der Bundesrepublik Deutschland: Berufsbildungsgesetz (BBiG) in der Neufassung vom 23. März 2005. Bundesgesetzblatt Teil I Nr. 20, Bonn (2005), https://www.bgbl.de/xaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl&jumpTo=bgbl105s0931.pdf

[7] Busemeyer, M.R., Trampusch, C.: The Political Economy of Collective Skill Formation. Oxford University Press (11 2011). https://doi.org/10.1093/acprof:oso/9780199599431.001.0001

[8] Cedefop: Vocational education and training in Europe, 1995–2035: Scenarios for European vocational education and training in the 21st century. No. 114 in Cedefop reference series, Publications Office of the European Union, Luxembourg (2020). https://doi.org/10.2801/794471

[9] Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805

[10] European Commission: Education Levels (nd), https://education.ec.europa.eu/education-levels, accessed: 2025-05-09

[11] Feng, F., Yang, Y., Cer, D., Arivazhagan, N., Wang, W.: Language-agnostic BERT Sentence Embedding (2022), https://arxiv.org/abs/2007.01852

[12] Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Aji, A.F., Bogoychev, N., Martins, A.F.T., Birch, A.: Marian: Fast Neural Machine Translation in C++. In: Liu, F., Solorio, T. (eds.) Proceedings of ACL 2018, System Demonstrations. pp. 116–121. Association for Computational Linguistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/P18-4020

[13] Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Aji, A.F., Bogoychev, N., Martins, A.F.T., Birch, A.: Marian: Fast Neural Machine Translation in C++ (2018), https://arxiv.org/abs/1804.00344

[14] 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: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 7871–7880. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.703

[15] Lin, C.Y.: ROUGE: A Package for Automatic Evaluation of Summaries. In: Text Summarization Branches Out. pp. 74–81. Association for Computational Linguistics, Barcelona, Spain (Jul 2004), https://aclanthology.org/W04-1013/

[16] Loper, E., Bird, S.: NLTK: the Natural Language Toolkit. In: Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1. p. 63–70. ETMTNLP '02, Association for Computational Linguistics, USA (2002). https://doi.org/10.3115/1118108.1118117

[17] Melhuish, E.C., Ereky-Stevens, K., Petrogiannis, K., Aricescu, A.M., Penderi, E., Rentzou, K., Tawell, A., Slot, P., Broekhuizen, M., Leseman, P.: A review of research on the effects of Early Childhood Education and Care (ECEC) upon child development. Technical report, European Commission (2015), https://eprints.bbk.ac.uk/id/eprint/16443/, cARE project; Curriculum Quality Analysis and Impact Review of European Early Childhood Education and Care (ECEC)

[18] Németh, L., Foundation, F.: Hunspell, https://hunspell.github.io/, accessed: 2025-02-15

[19] OECD: Education at a Glance 2021: OECD Indicators. OECD Publishing, Paris (2021). https://doi.org/10.1787/b35a14e5-en

[20] Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a Method for Automatic Evaluation of Machine Translation. In: Isabelle, P., Charniak, E., Lin, D. (eds.) Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. pp. 311–318. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA (Jul 2002). https://doi.org/10.3115/1073083.1073135

[21] Radford, A., Narasimhan, K.: Improving Language Understanding by Generative Pre-Training. In: Computer Science, Linguistics (2018), https://api.semanticscholar.org/CorpusID:49313245

[22] Schiersmann, C.: Weiterbildungsberatung im Kontext der Nationalen Weiterbildungsstrategie: Finanzielle und strukturelle Aspekte. Hessische Blätter für Volksbildung **72**(1), 43–53 (2022)

[23] Smith, R.: An Overview of the Tesseract OCR Engine. In: ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition. pp. 629–633. IEEE Computer Society, Washington, DC, USA (2007), https://storage.googleapis.com/pub-tools-public-publication-data/pdf/33418.pdf

[24] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to Sequence Learning with Neural Networks (2014), https://arxiv.org/abs/1409.3215

[25] UNESCO, UNICEF: Global Report on Early Childhood Care and Education: The Right to a Strong Foundation. UNESCO and UNICEF, Paris (2024). https://doi.org/10.54675/FWQA2113

[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need (2023), https://arxiv.org/abs/1706.03762