Advancing Post-OCR Correction: A Comparative Study of Synthetic Data

Shuhao Guan, Derek Greene
Insight Centre for Data Analytics, Dublin
School of Computer Science, University College Dublin, Ireland
shuhao.guan@ucdconnect.ie, derek.greene@ucd.ie

Abstract

This paper explores the application of synthetic data in the post-OCR domain on multiple fronts by conducting experiments to assess the impact of data volume, augmentation, and synthetic data generation methods on model performance. Furthermore, we introduce a novel algorithm that leverages computer vision feature detection algorithms to calculate glyph similarity for constructing post-OCR synthetic data. Through experiments conducted across a variety of languages, including several low-resource ones, we demonstrate that models like ByT5 can significantly reduce Character Error Rates (CER) without the need for manually annotated data, and our proposed synthetic data generation method shows advantages over traditional methods, particularly in low-resource languages[1].

1 Introduction

Digital libraries, like the Internet Archive, offer a vast collection of historical and culturally important books in image formats, including works written in low-resource and endangered languages. However, their image-only format limits content accessibility, hindering the use of these essential resources. Therefore, Optical Character Recognition (OCR) technologies are evidently useful in this context. However, OCR outputs frequently contain errors, particularly when working with texts featuring complex styles, archaic fonts, or unconventional layouts. These errors may include character recognition mistakes, formatting issues, and hyphenation problems, which are particularly prominent when dealing with low-resource languages (Ignat et al., 2022). Poor quality OCR can reduce the usefulness of these digital texts, adversely affecting downstream tasks (Linhares Pontes et al., 2019; Koudoro-Parfait et al., 2021).

Post-OCR correction is crucial for multiple reasons, including cultural heritage preservation (Jarlbrink and Snickars, 2017), expanding the accessibility of knowledge and information (Bazzo et al., 2020), and supporting further downstream tasks in cultural analytics (Stubbs, 1996). Additionally, accurate historical text data is extremely important for training large language models (LLMs) (Bubeck et al., 2023), which require high-quality data to improve their ability to handle complex queries, especially those involving history and culture.

Crowdsourcing has been the primary approach to acquire post-OCR training data in this domain (Clematide et al., 2016; Richter et al., 2018; Maheshwari et al., 2022). While this can provide highly accurate training data, the process is often time consuming and expensive. With the advent of Transformer architecture and attention mechanisms (Vaswani et al., 2017), deep learning models have emerged as the standard approach for post-OCR tasks. These models require large amounts of data for training. Thus, synthetic data has been increasingly adopted in this context. (D'hondt et al., 2017; Davydkin et al., 2023; Jasonarson et al., 2023). However, most existing literature on generating synthetic data relies on additional existing data for generation, and no comprehensive comparison has been made between different methods to understand how the synthetic data generated in various ways affects post-OCR performance.

To address these issues, this paper explores the impact of data volume and data augmentation methods on the performance of post-OCR models. We examine several common methods for creating synthetic data in the post-OCR domain and propose a novel method based on feature detection algorithms from computer vision to calculate glyph similarity for synthetic data construction. We conduct experiments on eight languages, including several low-resource languages, achieving significant CER reductions ranging from 12.41% to 48.18%.

[1]Code and data are available at https://github.com/NikoGuan/P_OCR

Findings of the Association for Computational Linguistics ACL 2024, pages 6036–6047

2 Related Work

Popular OCR systems include the Google Vision API OCR system (Fujii et al., 2017) and the Tesseract OCR engine (Smith, 2007). Jatowt et al. (2019) performed statistical analysis on the types of errors commonly produced by OCR systems.

Post-OCR correction, while often overlooked, is an important NLP task. Lexical approaches to post-correction concentrate on character and word level inaccuracies, primarily employing dictionaries and heuristic rules. Bassil and Alwani (2012) exploited Google's search suggestions for context-based corrections, circumventing the need for exhaustive dictionaries. Strategies specific to certain domains, such as those proposed by Furrer and Volk (2011), Estrella and Paliza (2014), and Kettunen (2016), highlight the necessity of tailored dictionaries for texts that possess unique features, like historical typefaces. Wemhoener et al. (2013) focused on aligning and merging outputs from various scans, including those from different editions, to rectify errors. Recent studies have framed post-OCR tasks as Seq2Seq tasks, with researchers applying both Statistical Machine Translation (SMT) and Neural Machine Translation (NMT) models (Amrhein and Clematide, 2018). Nguyen et al. (2020) and Soper et al. (2021) employed BERT (Devlin et al., 2018) and BART (Lewis et al., 2019) models, respectively. Maheshwari et al. (2022) conducted a comparison between pre-trained models and traditional Seq2Seq models, with their findings indicating that pre-trained models like ByT5 (Xue et al., 2022) outperform the conventional models. Ramirez-Orta et al. (2022) segmented documents into character n-grams, before aggregating their corrections into the final output via majority voting (Lam and Suen, 1997), essentially acting as an ensemble of sequence models.

Data is fundamental to the success of deep learning, as an increasing amount of research is directed towards leveraging data-driven strategies. These strategies aim to significantly improve model performance by optimizing data usage, rather than modifying the underlying model structure (Tarafdar et al., 2019; Mazumder et al., 2022). These efforts often involve generating large quantities of synthetic data (Choi and Park, 2023), involving data manipulation measures like filtering (Koehn et al., 2020), data augmentation (Shorten and Khoshgoftaar, 2019; Li et al., 2022), and noise injection (Izumi et al., 2003). Such synthetic data has been widely used in various NLP tasks, such as grammatical error correction (Ingólfsdóttir et al., 2023), language identification (Ahmadi et al., 2023), question answering (Puri et al., 2020), and named entity recognition (Liu et al., 2021).

For the task of text denoising, the primary method for constructing synthetic data is noise injection (Izumi et al., 2003). This technique involves inserting errors into clean text to generate training data pairs. D'hondt et al. (2017) added artificial OCR errors into sentences using a random process, while Jasonarson et al. (2023) focused on the low-resource Icelandic language for post-OCR tasks. The authors extracted OCR errors from real digitised documents and inserting them into clean text in similar proportions. Grundkiewicz et al. (2019) analyzed real data to create replacement sets for each word. Davydkin et al. (2023) adopted a different approach, developing a system to generate handwritten image data which were then processed with OCR. The resulting OCR outputs and the original texts were subsequently used to train a T5 model (Raffel et al., 2020) for correction purposes. Ignat et al. (2022) synthesized text image datasets by manipulating parameters, like font spacing and image saturation, then compare the original text with the OCR output text, to evaluate OCR systems' performance on various low-resource languages, and they enhanced Machine Translation (MT) through backtranslation (Sennrich et al., 2015).

Some studies have explored improving post-OCR text correction by analyzing the visual forms of characters, known as glyphs. Chen and Zhou (2023) attempting to use the CharBERT model (Ma et al., 2020) for post-OCR. Their method consists of two parallel CNN encoders and a transformer decoder, taking CharBERT and glyph embedding as inputs. Amrhein and Clematide (2018)'s experiment included NMT models and glyph embedding, but it did not enhance model performance. Other research has focused on the detection of homoglyphs, with Ginsberg and Yu (2018) employing a grid method to assess the similarity of glyphs by counting the number of overlapping grids between characters. The similarity of glyphs is actually based on visual features. In the field of computer vision, feature detection and matching algorithms, such as SIFT (Ng and Henikoff, 2003), ORB (Rublee et al., 2011), and AKAZE (Alcantarilla and Solutions, 2011) can extract feature points from an image and match them with those appearing in other images.

3 Methods

We now describe three common methods for generating synthetic data in the post-OCR domain, before introducing a new method based on *glyph similarity* in Section 3.4. Each method makes use of a clean corpus A. As a common preprocessing step, we segment A into multiple chunks by initially dividing sentences based on punctuation marks and then concatenating these sentences, ensuring that the total length of each chunk does not exceed a fixed limit of 230 characters, with the exception of Russian and Telugu, where the limits are 140 and 90 characters, respectively.

3.1 Method ① Random Injection

The random injection method (D'hondt et al., 2017) generates a synthetic OCR corpus $\hat{A}$ by randomly inserting errors into an existing clean corpus A.

First, we filter out infrequently occurring characters in A, as the corpus may inevitably contain some noise. The remaining characters are used for replacements and insertions. Next, for each chunk from A, we randomly select a target error rate $p \in [0, 15]$, which controls the level of noise to be introduced. According to the analysis by Jatowt et al. (2019), the average rate of OCR errors involving substitution, insertion, and deletion is approximately 5:1:1. Therefore, in our implementation, each character has a probability of $\frac{5}{7} \times p$ of being replaced by a random character and a probability of $\frac{1}{7} \times p$ of being deleted. Additionally, any two characters have a probability of $\frac{1}{7} \times p$ of having a random character inserted between them.

3.2 Method ② Image Creation

Following some OCR-related studies that add noise to text images (Jaderberg et al., 2014; Krishna et al., 2018; Boros et al., 2022), we also investigate the simulation of real-world correction scenarios by creating synthetic text images. Specifically, each chunk from corpus A is used to create a separate image which is manipulated to add OCR-like noise. These images are then processed through a standard OCR system to obtain $\hat{A}$.

The complete procedure is as follows. Initially, for every text chunk, a random font from the set F, which matches the language of corpus A, is chosen. Then, the text from each chunk is converted into an image. These images are subjected to random rotations of ± 5 degrees, followed by the insertion of random noise pixels. Following this, the images undergo random dilation and erosion to simulate text variability, and their resolution are randomly reduced within a range of 0 to 50%. At the end of this process, the chosen OCR system is applied for text recognition, resulting in the output $\hat{A}$.

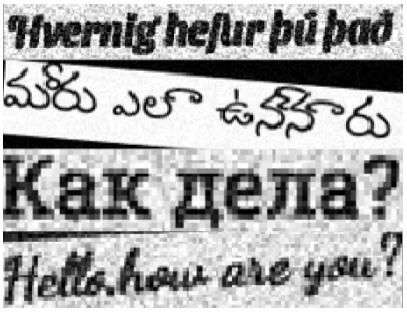

Figure 1: Examples of synthetic OCR images in various languages, generated using the process in Section 3.2.

Figure 1 shows sample synthesized images with noise generated using this process, for several different languages. Although synthetic images are used, this method simulates the OCR process under real-world conditions, the output text is derived from the OCR system, hence we consider it to be representative of authentic OCR text.The corpora A and $\hat{A}$ can be used to construct test sets, training sets, or to extract distributions of OCR errors.

3.3 Method ③ Real-World Injection

Several studies have considered the generation of synthetic data through the analysis of error distributions in existing datasets. The primary approach involves embedding OCR errors into the data at rates mirroring their occurrence in real-world settings (D'hondt et al., 2017; Grundkiewicz et al., 2019; Jasonarson et al., 2023). This strategy addresses the discrepancies between existing datasets and actual application domains, offering more control over the quality of the generated text.

To obtain a realistic OCR error distribution, in this work we use additional clean corpus C coming from the same domain and apply the method described in Section 3.2 to obtain OCR-processed corpus $\hat{C}$. Subsequently, the Recursive Text Alignment Scheme (RETAS) (Yalniz and Manmatha, 2011) is employed to perform the automatic alignment of the OCR text and the original clean text, allowing us to extract the probabilities for character substitution, deletion, and insertion. By adjusting the probability of these errors, we can control the CER of the synthetic data. For instance, if the CER between text C and its corresponding OCR version

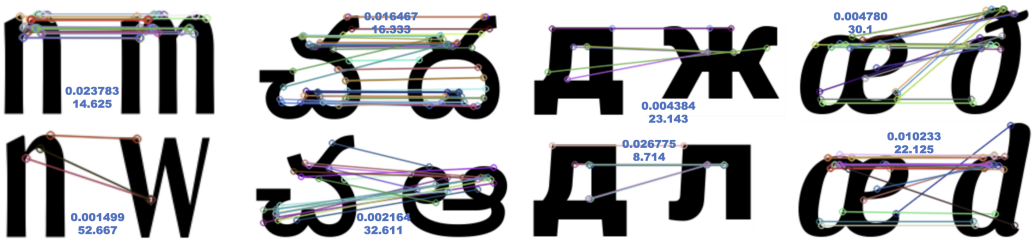

Figure 2: Visualizations of feature matching using ORB for fonts from different character sets, where matched feature points connected by colored lines. For each pair of characters, two numbers are displayed: the upper number represents the Jaccard Index J of overlapping feature points, and the lower number indicates the average distance D.

$\hat{C}$ is p, then doubling all of the error probabilities would result in an expected CER of $2p$ for the newly generated synthetic text. In practice, we set the expected CER for each chunk in A to a random value $\in [0, 15]$ to generate the synthetic data in $\hat{A}$.

3.4 Method ④ Glyph Similarity

Given that OCR replacement errors frequently happen between characters with visually similar glyphs (Jatowt et al., 2019), we can also use the information from glyph similarities to construct synthetic data. Specifically, we apply the following procedure. Firstly, we filter infrequently appearing characters from the input corpus A to form a definitive character set. Based on the analysis of the ICDAR2017 (Chiron et al., 2017) and ICDAR2019 (Rigaud et al., 2019) datasets, 1:1 mapping errors, where a single character is incorrectly identified as another character, accounted for 87.9% of all 1:n mapping errors. Here 1:n mapping error refers to cases where a single character be incorrectly recognized as n characters. To enhance the computational efficiency of the implementation proposed in this paper, we focus solely on 1:1 errors. Note that simulations of 1:n $(n > 1)$ errors could be achieved through random insertion.

For each language, we select a set of appropriate fonts F, which includes a variety of historical and modern fonts, and employ a set of vision feature detection algorithms Q to extract and match image features. In our experiments we consider ORB (Rublee et al., 2011), AKAZE (Alcantarilla and Solutions, 2011) and SIFT (Ng and Henikoff, 2003). For each character i in the character set, we calculate its glyph similarity with every other character j by creating their images using fonts f from the font set F. We then calculate their similarity under detector q as:

$$S(i, j, q) = \frac{1}{|F|} \sum_{f \in F} \frac{J(i, j, f, q)}{D(i, j, f, q)}$$

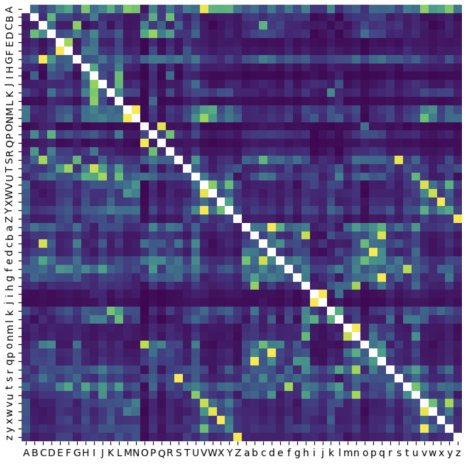

Figure 3: Visualization of a glyph similarity matrix for English-language characters (52 letters only). The saturation of each cell represents the value $S_{\text{norm}}(i, j)$ between each pair of characters i and j.

where $D(i, j, f, q)$ is the average distance between matching point pairs for character i and j, using font f and feature detector q. The degree of overlap for feature matching points $J(i, j, f, q)$ is calculated using the Jaccard Index (Jaccard, 1912). This is defined as the ratio of the number of matching points relative to the combined number of feature points in i and j minus the number of matching points. A number of matching examples are shown in Figure 2. We see that, as glyphs become more similar, J increases and D decreases.

Next, we perform min-max normalization for different mapping types and characters to obtain the normalized score $\in [0, 1]$

$$S_{\text{norm}}(i, j) = \frac{1}{|Q|} \sum_{q \in Q} \frac{S(i, j, q) - \min(S_{iq})}{\max(S_{iq}) - \min(S_{iq})}$$

where S_{iq} is the set of similarity scores between character i and other characters using feature detector q. Figure 3 presents an example of $S_{\text{norm}}(i, j)$ for English-language characters.

The technique of embedding OCR errors by exploiting glyph similarity shares some common aspects with the random injection method described previously. Firstly, we randomly select a target error rate $p \in [0, 15]$ for each chunk from A, where each character i in A has a probability of $\frac{5}{7} \times p$ to be replaced with another character. The choice of replacement is based on $S_{\text{norm}}(i, j)$, with higher-weighted pairs being more likely to be chosen. After replacement, each character has a probability of $\frac{1}{7} \times p$ to be deleted. Any two characters have a probability of $\frac{1}{7} \times p$ of having a random character inserted between them.

4 Datasets

In this paper, we undertake post-OCR experiments involving English, Frisian, German, Icelandic, Irish, Russian, Spanish, and Telugu texts. Clean corpora A for these languages were obtained as follows: data for Icelandic, Irish, and Frisian were sourced from the CC-100 corpus (Conneau et al., 2019). Telugu[2] and Russian[3] datasets were obtained from Kaggle, while English, Spanish, and German were sourced from Project Gutenberg. Summary statistics for the corpora are given in Table 1. The amount of data selected simulates the real-world rarity of these languages to some extent.

Language	Length	Source
English	27,883,394	Gutenberg
Frisian	3,426,499	CC100
German	3,758,352	Gutenberg
Icelandic	3,147,864	CC100
Irish	5,090,436	CC100
Russian	6,613,093	Kaggle
Spanish	7,813,245	Gutenberg
Telugu	5,777,551	Kaggle

Table 1: Corpus character counts and sources.

All training, validation, and test datasets are generated from these clean corpora. The training data is created using the four different methods described in Section 3, with the aim of comparing the synthetic data generation techniques. To implement the method presented in Section 3.3, 20% of the text from each language corpus is used as C for extracting OCR error distributions. The remaining 80% of the corpus is divided in a 8:1:1 ratio to generate training, validation, and test sets.

The test datasets are generated using a method similar to that described in Section 3.2, employing the Tesseract OCR system (Smith, 2007) to convert synthetic images into OCR text. Since the methods described in Sections 3.2 and 3.3 involve using an OCR system to either generate OCR text directly or to extract OCR errors, the Google Vision API OCR system (Fujii et al., 2017) is used when creating training and validation datasets.

5 Models

In our experiments, we compare the performance of various models which have been previously adopted for post-OCR tasks, including models that operate at the subword-level, character-level, and byte-level tokenizers. Additionally, we evaluate current SOTA models which is based on n-gram and majority voting. This comparison encompasses both pre-trained models and those trained from scratch. Training parameters are in Appendix A.

5.1 mT5

The mT5 model (Xue et al., 2020) is an extension of the T5 model (Raffel et al., 2020), which is pre-trained on a multilingual dataset. This model leverages the original T5's text-to-text framework, where every natural language processing task is reframed as a text generation problem, allowing for consistent and flexible handling of a wide range of tasks across languages. The version we use in our experiments is mT5-base.

5.2 mBART

The mBART model (Tang et al., 2020) is a multilingual extension of the BART architecture (Lewis et al., 2019). Developed by Facebook AI, its pre-training approach involves corrupting text with an arbitrary noising function and then learning to reconstruct the original text. mBART is also pre-trained on a large corpus of text in multiple languages, making it adept at both high-resource and low-resource language translation, as well as a variety of other language processing tasks. The version we use in our experiments is mBART-large-50.

5.3 ByT5

The ByT5 model (Xue et al., 2022) is designed to address the limitations of traditional subword tokenization methods by working at the byte level.

[2] https://www.kaggle.com/datasets/sudalairajkumar/telugu-nlp

[3] https://www.kaggle.com/datasets/d0rj3228/russian-literature/data

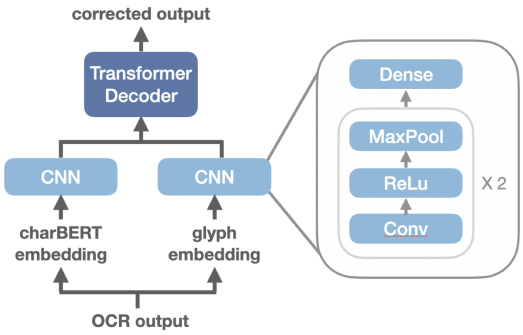

Figure 4: The structure of the CharBERT post-OCR model incorporates glyph embeddings as inputs. It consists of two CNN encoders and one transformer decoder.

This approach allows ByT5 to handle any language or writing system with Unicode representation, making it naturally multilingual and adaptable for handling a wide variety of text data. By operating on bytes, ByT5 avoids the complexities and biases associated with subword vocabularies, enabling more equitable and accurate processing across languages. ByT5 was pre-trained on the same dataset as mT5. The model demonstrates robust performance across a wide range of tasks (Stankevičius et al., 2022; Jentoft, 2023). The version we use in our experiments is ByT5-base.

5.4 CharBERT + Glyph Embedding

To verify the effectiveness of glyph embedding as proposed by Chen and Zhou (2023), we also conducted some experiments using their model. CharBERT (Ma et al., 2020) is a char-level model. In the pre-training phase of CharBERT, one of the tasks involves text denoising which is closely aligns with the objectives of post-OCR correction tasks. Building on the original model, Chen and Zhou (2023) introduced glyph embedding to CharBERT by using a ResNet50 model (He et al., 2016). This supports the extraction of visual information from characters and serves as one of the inputs to the post-OCR correction model. The structure of this model is shown in Figure 4. In our experiments we use the same structure and fine-tuning settings corresponding to the optimal configurations reported in the original work. Since the CharBERT model was pre-trained on the English Wikipedia dataset (Foundation, 2024), we conduct experiments only in English with this model.

5.5 Non-pre-trained Models

The method proposed by Ramirez-Orta et al. (2022) achieved SOTA results in 5 out of 9 languages on the ICDAR2019 dataset. In this study, we also compare this model with other models using the configuration: window type as n-grams, window size of 60, decoding method as beam, weighting as uniform. We refer to as the ENSEMBLE model.

Additionally, we have defined a more conventional Seq2Seq model, referred to as SCRATCH. The SCRATCH model is designed with an embedding size of 512, feed-forward network embeddings of 2048, 4 attention heads, 5 encoder layers, 5 decoder layers, a SentencePiece tokenizer (Kudo and Richardson, 2018), and a vocabulary size of 3000.

6 Experiments

The primary objective of our empirical analysis in this paper is to explore the key factors affecting the performance of models in post-OCR tasks, aiming to identify the most appropriate settings for such tasks. We conduct three different experiments:

- *Experiment 1:* We investigate how data volume impacts on model performance, aiming to identify a generally applicable data augmentation setting that balances performance and training time.
- *Experiment 2:* We seek identify the best method for constructing synthetic data and the best-performing model. Using the findings from Experiment 1, we apply augmentation and generate different datasets using the four methods from Section 3, in combination with multiple post-OCR models.
- *Experiment 3:* We extend the best settings identified in the first two experiments to a wider set of languages for post-OCR tasks, verifying the applicability and effectiveness of these settings in a broader linguistic context.

6.1 Experiment 1

The process of generating OCR synthetic data is highly stochastic, so the same clean text can yield different synthetic text versions when OCR errors are inserted. This variability can potentially enrich the model's learning process. In this experiment, we investigate the impact of data augmentation techniques (by replicating the clean text data multiple times to increase the volume of data) and the quantity of original clean text data used on model performance. We examine whether increasing the diversity of data via this method can enhance the model's ability to correct OCR errors.

		English				Icelandic				Russian				Telugu			
	CER	4.96				10.09				4.13				34.12			
Model	Data	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{2}{3}$	1
mT5	1×	4.56	4.29	3.77	3.42	9.68	9.33	9.02	8.87	3.78	3.33	2.99	2.74	31.44	29.80	28.35	27.70
	2×	4.32	4.10	3.52	3.24	9.44	9.12	8.83	8.49	3.63	3.15	2.63	2.51	29.83	28.69	27.83	26.89
	4×	4.24	3.71	3.13	3.00	9.35	9.00	8.60	8.31	3.54	3.05	2.40	**2.33**	29.31	28.45	27.12	**26.31**
	8×	4.24	3.68	3.05	**2.93**	9.32	8.90	8.54	**8.29**	3.55	3.02	2.39	2.36	29.22	28.44	27.23	26.32
ENSEMBLE	1×	4.84	4.69	4.51	4.46	10.18	10.09	10.00	9.88	4.09	4.00	3.90	3.83	40.23	38.47	38.02	37.71
	2×	4.80	4.63	4.44	4.30	10.09	10.03	9.88	9.74	4.06	3.85	3.80	3.71	38.99	38.65	38.21	36.60
	4×	4.75	4.53	4.23	**4.19**	10.14	9.94	9.83	**9.64**	4.02	3.72	3.69	**3.52**	38.78	38.17	37.43	35.66
	8×	4.79	4.48	4.24	4.22	10.11	9.97	9.80	9.69	4.00	3.74	3.63	3.60	38.65	37.75	36.54	**35.59**

Table 2: Results showing the effects of data volume and augmentation on mT5 and ENSEMBLE model CER on English, Icelandic, and Russian texts.

The experiment is conducted on English, Icelandic, Russian and Telugu texts. These languages were chosen as they include both rich-resource and low-resource languages and belong to different language families, and have different grammatical structures and character sets. This diversity makes them representative for assessing different models and methods for generating synthetic data.

To explore the impact of the original data volume, we experiment with datasets of different sizes: the full dataset size, $\frac{2}{3}$ size, $\frac{1}{3}$ size, and $\frac{1}{6}$ size, as in post-OCR correction tasks, the training data often need to be from the same domain as the text need to be fixed, so the settings of 1, $\frac{2}{3}$, $\frac{1}{3}$, and $\frac{1}{6}$ were designed to simulate the scenarios within different domains under one language environment with varying amounts of clean texts. And to examine the effect of data augmentation techniques, these will be further expanded to 1×, 2×, 4× and 8× their original sizes. This approach will yield multiple versions of dataset A, and since Method ③ is the most commonly used, we will employ it to generate the training data for this experiment. The experiments are conducted using both the pre-trained mT5-base model and ENSEMBLE model.

The results in Table 2 show that mT5 outperforms the ENSEMBLE model across the four languages. Data augmentation enhances model performance with diminishing returns – augmenting from 1× to 4× effectively lowers CER, but gains from 4× to 8× are marginal. In our experiment, training parameters for mT5 included a learning rate of 5e-4, warm-up steps of 250, batch size of 4, dropout rate of 0.2, and 6 epochs on fp32. Training times for 4× augmented datasets on an RTX 4090 were approximately 35, 5, 10 and 8 hours for English, Icelandic, Russian and Telugu, respectively. Overall, the results suggest that 4× augmentation provides sufficient improvement.

6.2 Experiment 2

Next, we explore how different methods of creating synthetic data impact on the performance of different language models. We conduct tests using several popular models: mT5-base (Xue et al., 2020), ByT5-base (Xue et al., 2022), mBART-large (Chipman et al., 2022), SCRATCH, and ENSEMBLE. We also evaluate the approach of Chen and Zhou (2023), which combines CharBERT (Ma et al., 2020) with glyph embedding, and compare these results to CharBERT without glyph embedding.

The results of Experiment 2 are shown in Table 3. It is observed that methods ③ and ④ generally outperform ① and ②. Of the latter two, method ①, which does not rely on additional data, yields slightly higher accuracy. While method ② allows the model to learn from real OCR outputs, it fails to correct sentences with varying degrees of CER. In our experiments, the Google Vision API OCR system was used to generate training data, and Tesseract was used for test data. Since the Vision API OCR system generally has a higher accuracy than Tesseract, models only learn to correct sentences with low CER, which is reflected in the limitations of the method ②.

Method ③ proves more effective, extracting real OCR error distributions and generating training data with various CER values, but requires additional data. With a sufficient volume of data (as in the English language experiments), models trained with this method performed best. Different OCR systems, due to their unique designs, can make various kinds of errors. With insufficient additional data, Method ③ can fail to extract a comprehensive set of OCR errors, leading to distributional shift in the data, which ultimately affects the performance of models that rely on it. In Experiment 2, Method ④ achieved impressive results without

	English				Icelandic				Russian				Telugu			
CER	4.96				10.09				4.13				34.12			
Method	①	②	③	④	①	②	③	④	①	②	③	④	①	②	③	④
mT5	**3.98**	3.42	3.00	3.03	9.48	9.53	**8.31**	8.38	3.36	**2.98**	**2.33**	2.27	30.35	28.24	26.31	25.85
ByT5	4.02	**3.36**	**2.96**	**3.00**	**9.42**	**9.45**	8.39	**8.28**	**3.34**	3.13	2.34	**2.14**	**30.21**	**28.14**	**25.98**	**25.28**
mBART	4.19	4.23	3.63	3.54	9.93	10.26	9.41	9.30	3.41	3.33	2.82	2.78	31.06	28.49	25.66	26.00
CharBERT	4.12	3.60	3.39	3.57	-	-	-	-	-	-	-	-	-	-	-	-
+Glyph	4.11	3.61	3.42	3.54	-	-	-	-	-	-	-	-	-	-	-	-
ENSEMBLE	4.56	4.44	4.19	4.00	10.32	10.65	9.64	9.54	3.77	3.83	3.52	3.34	37.12	34.23	35.66	37.50
SCRATCH	4.79	4.74	4.52	4.62	11.92	9.82	10.43	10.00	3.91	4.06	3.62	3.64	33.56	35.28	34.52	34.92

Table 3: Comparison of CER results across various models and synthetic data generation methods in English, Icelandic, and Russian. The best-performing model for each dataset is highlighted in bold, and the best method for generating synthetic data for each model is underlined.

	English		German		Irish		Icelandic		Frisian		Russian		Spanish		Telugu	
	CER	WER	CER	WER	CER	WER	CER	WER	CER	WER	CER	WER	CER	WER	CER	WER
OCR	4.96	15.43	5.79	19.29	12.57	35.99	10.09	30.06	5.15	16.20	4.13	8.16	6.00	17.38	34.12	90.41
Post-OCR	3.00	7.74	4.27	11.07	11.01	29.97	8.28	24.57	3.55	11.03	2.14	5.88	3.76	8.92	25.28	66.64

Table 4: Performance of the ByT5 model in terms of Character Error Rate (CER) and Word Error Rate (WER) across multiple languages, before and after post-OCR correction.

relying on additional data. By generating its own training data with varying degrees of CER encoded through glyphs alone, it successfully trained models that perform well across diverse languages with different character sets.

Overall, we observe that pre-trained models significantly outperformed smaller Seq2Seq models trained from scratch. mT5 and ByT5 showed similar performance, with ByT5 reducing CER values by 39.5%, 17.9%, 48.2% and 25.9% on English, Icelandic, Russian and Telugu datasets, respectively. CharBERT did not benefit from glyph embedding, consistent with the findings of Amrhein and Clematide (2018).

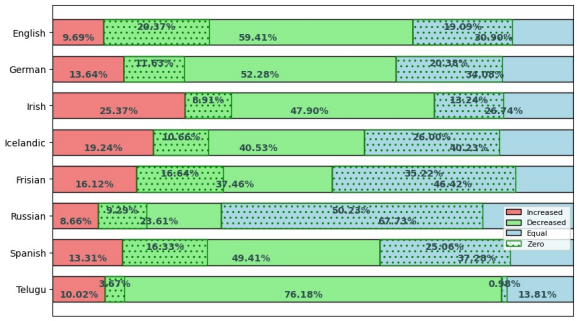

Figure 5: Comparative analysis of CER changes after post-OCR across multiple languages. Each bar represents the distribution of CER changes categorized as Increased (red), Decreased (green), Equal (blue), and Zero (dotted green).

6.3 Experiment 3

In our final experiment, we replicate the original clean text four times for augmentation. Based on our previous experiments, we select Method ④ to generate synthetic data and train the ByT5 model for post-OCR tasks on data from a wider set of languages, so that we can evaluate the chosen configuration in a broader linguistic context.

From the results in Table 4, we see that the model performs well on the English, Russian, and Spanish tasks, with CER reductions of 39.52%, 48.18%, and 37.33%, respectively. This approach also achieves a CER reduction of 31.07% on the low-resource language Frisian. The performance is poorest in the cases of Irish and Icelandic, with only 12.41% and 17.94% reductions in CER, respectively. We attribute this to two main reasons. Firstly, the data for Irish and Icelandic (sourced from the CC100 dataset of web-crawled data) contain numerous proper nouns. These are challenging for conventional NMT models to correct due to their inability to handle out-of-vocabulary words. Secondly, although the ByT5 model was trained on a multilingual dataset, the volume of data for low-resource languages is not substantial. This disparity in data availability can affect the models' performance on post-OCR tasks. Table 4 also indicates that the Telugu test dataset has very high CER and WER, likely due to the complexity of Telugu glyphs (Negi et al., 2001). When Telugu text becomes blurred, it can be difficult for OCR systems to recognize. Thus, post-OCR processing

for Telugu results in a 25.91% reduction in CER.

Figure 5 provides a more detailed sentence-based view of CER across languages. Here we enumerate the following categories for CER changes after post-OCR processing: "Increased" indicates sentences where the CER has increased, "Decreased" indicates the CER has decreased, "Equal" means that CER remained unchanged, and "Zero" means the CER is zero after post-OCR processing (i.e., perfect matching with the ground truth). We observe that only a small number of sentences experienced an increase in CER after post-OCR processing. Specifically, 25.37% of sentences in Irish showed an increase in CER post-OCR, and only 8.66% of sentences in Russian. Notably, 59.52% of sentences in Russian had a CER of zero after post-OCR processing. For Telugu, which had the highest CER, we saw a reduction in CER for 76.18% of sentences after post-OCR processing.

7 Conclusion

Our experiments demonstrated that augmenting data by replicating clean text and constructing synthetic data based on glyph similarity significantly improves model performance for post-OCR correction. Furthermore, we proposed a novel approach to constructing post-OCR synthetic data based on glyph similarity, which outperforms traditional noise injection methods, especially in scenarios with limited data volume, as it does not require additional external data. In our experiments, pre-trained models demonstrated superior performance compared to those without pre-training, with the byte-level ByT5 model achieving the best performance. For rich-resource languages, this approach can reduce the CER by approximately 45%, while for low-resource languages the reduction varies between 12.41% and 31.07%.

Limitations

For the preferred Method ④, the time complexity of calculating the character similarity values is $O(n^2)$. As a result, the calculation of character similarities in languages with a large array of characters, such as Chinese and Japanese, can be quite time consuming.

Acknowledgements

This publication is part of a project that has received funding from (i) the European Research Council (ERC) under the Horizon 2020 research and innovation programme (Grant agreement No. 884951); (ii) Science Foundation Ireland (SFI) to the Insight Centre for Data Analytics under grant No 12/RC/2289 P2.

References

Sina Ahmadi, Milind Agarwal, and Antonios Anastasopoulos. 2023. Pali: A language identification benchmark for Perso-Arabic scripts. arXiv preprint arXiv:2304.01322.

Pablo F Alcantarilla and T Solutions. 2011. Fast explicit diffusion for accelerated features in nonlinear scale spaces. IEEE Trans. Patt. Anal. Mach. Intell, 34(7):1281–1298.

Chantal Amrhein and Simon Clematide. 2018. Supervised OCR error detection and correction using statistical and neural machine translation methods. Journal for Language Technology and Computational Linguistics, 33(1):49–76.

Youssef Bassil and Mohammad Alwani. 2012. OCR post-processing error correction algorithm using google online spelling suggestion. arXiv preprint arXiv:1204.0191.

Guilherme Torresan Bazzo, Gustavo Acauan Lorentz, Danny Suarez Vargas, and Viviane P Moreira. 2020. Assessing the impact of OCR errors in information retrieval. In Advances in Information Retrieval: 42nd European Conference on IR Research (ECIR 2020), pages 102–109. Springer.

Emanuela Boros, Nhu Khoa Nguyen, Gaël Lejeune, and Antoine Doucet. 2022. Assessing the impact of ocr noise on multilingual event detection over digitised documents. International Journal on Digital Libraries, 23(3):241–266.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. arXiv preprint, (2303.12712).

Yung-Hsin Chen and Yuli Zhou. 2023. Enhancing ocr performance through Post-OCR models: Adopting glyph embedding for improved correction. arXiv preprint arXiv:2308.15262.

Hugh A Chipman, Edward I George, Robert E McCulloch, and Thomas S Shively. 2022. mbart: multidimensional monotone bart. Bayesian Analysis, 17(2):515–544.

Guillaume Chiron, Antoine Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. 2017. ICDAR-2017 competition on post-OCR text correction. In Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), volume 1, pages 1423–1428. IEEE.

Eujeong Choi and Chanjun Park. 2023. Dmops: Data management operation and recipes. arXiv preprint arXiv:2301.01228.

Simon Clematide, Lenz Furrer, and Martin Volk. 2016. Crowdsourcing an OCR gold standard for a german and french heritage corpus. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), pages 975–982.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116.

Evgenii Davydkin, Aleksandr Markelov, Egor Iuldashev, Anton Dudkin, and Ivan Krivorotov. 2023. Data generation for Post-OCR correction of Cyrillic handwriting. arXiv preprint arXiv:2311.15896.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint, (1810.04805).

Eva D'hondt, Cyril Grouin, and Brigitte Grau. 2017. Generating a training corpus for OCR post-correction using encoder-decoder model. In Proceedings of the 8th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1006–1014.

Paula Estrella and Pablo Paliza. 2014. Ocr correction of documents generated during argentina's national reorganization process. In Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, pages 119–123.

Wikimedia Foundation. 2024. Wikimedia downloads.

Yasuhisa Fujii, Karel Driesen, Jonathan Baccash, Ash Hurst, and Ashok C Popat. 2017. Sequence-to-label script identification for multilingual OCR. In 2017 14th IAPR international conference on document analysis and recognition (ICDAR), volume 1, pages 161–168. IEEE.

Lenz Furrer and Martin Volk. 2011. Reducing ocr errors in gothic-script documents. In Proceedings of the Workshop on Language Technologies for Digital Humanities and Cultural Heritage, pages 97–103.

Avi Ginsberg and Cui Yu. 2018. Rapid homoglyph prediction and detection. In Proc. 1st International Conference on Data Intelligence and Security (ICDIS), pages 17–23. IEEE.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 252–263.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

Oana Ignat, Jean Maillard, Vishrav Chaudhary, and Francisco Guzmán. 2022. OCR improves machine translation for low-resource languages. arXiv preprint arXiv:2202.13274.

Svanhvít Lilja Ingólfsdóttir, Pétur Orri Ragnarsson, Haukur Páll Jónsson, Haukur Barri Símonarson, Vilhjálmur Þorsteinsson, and Vésteinn Snæbjarnarson. 2023. Byte-level grammatical error correction using synthetic and curated corpora. arXiv preprint arXiv:2305.17906.

Emi Izumi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi, and Hitoshi Isahara. 2003. Automatic error detection in the japanese learners' english spoken data. In The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics, pages 145–148.

Paul Jaccard. 1912. The distribution of flora in the alpine zone. New Phytologist, 11(2):37–50.

Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Synthetic data and artificial neural networks for natural scene text recognition. arXiv preprint arXiv:1406.2227.

Johan Jarlbrink and Pelle Snickars. 2017. Cultural heritage as digital noise: nineteenth century newspapers in the digital archive. Journal of Documentation, 73(6):1228–1243.

Atli Jasonarson, Steinþór Steingrímsson, Einar Freyr Sigurðsson, Árni Davíð Magnússon, and Finnur Ágúst Ingimundarson. 2023. Generating Errors: OCR Post-Processing for Icelandic. In 24th Nordic Conference on Computational Linguistics.

Adam Jatowt, Mickael Coustaty, Nhu-Van Nguyen, Antoine Doucet, et al. 2019. Deep statistical analysis of OCR errors for effective post-OCR processing. In 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL), pages 29–38. IEEE.

Matias Jentoft. 2023. Grammatical error correction with byte-level language models. Master's thesis, University of Oslo.

Kimmo Kettunen. 2016. Keep, change or delete? setting up a low resource ocr post-correction framework for a digitized old finnish newspaper collection. In Digital Libraries on the Move: 11th Italian Research Conference on Digital Libraries, IRCDL 2015, Bolzano, Italy, January 29-30, 2015, Revised Selected Papers 11, pages 95–103. Springer.

Philipp Koehn, Vishrav Chaudhary, Ahmed El-Kishky, Naman Goyal, Peng-Jen Chen, and Francisco Guzmán. 2020. Findings of the wmt 2020 shared task on parallel corpus filtering and alignment. In Proceedings of the Fifth Conference on Machine Translation, pages 726–742.

Caroline Koudoro-Parfait, Gaël Lejeune, and Glenn Roe. 2021. Spatial named entity recognition in literary texts: What is the influence of ocr noise? In Proceedings of the 5th ACM SIGSPATIAL International Workshop on Geospatial Humanities, pages 13–21.

Amrith Krishna, Bodhisattwa Prasad Majumder, Rajesh Shreedhar Bhat, and Pawan Goyal. 2018. Upcycle your ocr: Reusing ocrs for post-ocr text correction in romanised sanskrit. arXiv preprint arXiv:1809.02147.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint, (1808.06226).

Louisa Lam and SY Suen. 1997. Application of majority voting to pattern recognition: an analysis of its behavior and performance. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 27(5):553–568.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

Bohan Li, Yutai Hou, and Wanxiang Che. 2022. Data augmentation approaches in natural language processing: A survey. Ai Open, 3:71–90.

Elvys Linhares Pontes, Ahmed Hamdi, Nicolas Sidere, and Antoine Doucet. 2019. Impact of ocr quality on named entity linking. In Digital Libraries at the Crossroads of Digital Information for the Future: 21st International Conference on Asia-Pacific Digital Libraries, ICADL 2019, Kuala Lumpur, Malaysia, November 4–7, 2019, Proceedings 21, pages 102–115. Springer.

Linlin Liu, Bosheng Ding, Lidong Bing, Shafiq Joty, Luo Si, and Chunyan Miao. 2021. MulDA: A multilingual data augmentation framework for low-resource cross-lingual NER. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 5834–5846.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. Charbert: character-aware pre-trained language model. arXiv preprint arXiv:2011.01513.

Ayush Maheshwari, Nikhil Singh, Amrith Krishna, and Ganesh Ramakrishnan. 2022. A benchmark and dataset for post-OCR text correction in Sanskrit. arXiv preprint arXiv:2211.07980.

Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Alicia Parrish, Hannah Rose Kirk, et al. 2022. Dataperf: Benchmarks for data-centric ai development. arXiv preprint arXiv:2207.10062.

Atul Negi, Chakravarthy Bhagvati, and B Krishna. 2001. An OCR system for Telugu. In Proceedings of 6th International Conference on Document Analysis and Recognition, pages 1110–1114. IEEE.

Pauline C Ng and Steven Henikoff. 2003. Sift: Predicting amino acid changes that affect protein function. Nucleic acids research, 31(13):3812–3814.

Thi Tuyet Hai Nguyen, Adam Jatowt, Nhu-Van Nguyen, Mickael Coustaty, and Antoine Doucet. 2020. Neural machine translation with BERT for post-OCR error detection and correction. In Proceedings of the ACM/IEEE joint conference on digital libraries, pages 333–336.

Raul Puri, Ryan Spring, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. arXiv preprint arXiv:2002.09599.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551.

Juan Antonio Ramirez-Orta, Eduardo Xamena, Ana Maguitman, Evangelos Milios, and Axel J Soto. 2022. Post-ocr document correction with large ensembles of character sequence-to-sequence models. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pages 11192–11199.

Caitlin Richter, Matthew Wickes, Deniz Beser, and Mitch Marcus. 2018. Low-resource post processing of noisy ocr output for historical corpus digitisation. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).

Christophe Rigaud, Antoine Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. 2019. ICDAR-2019 competition on post-OCR text correction. In Proceedings of the 2019 IAPR International Conference on Document Analysis and Recognition (ICDAR), pages 1588–1593. IEEE.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. Orb: An efficient alternative to sift or surf. In 2011 International conference on computer vision, pages 2564–2571. Ieee.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. arXiv preprint arXiv:1511.06709.

Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. Journal of big data, 6(1):1–48.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR), volume 2, pages 629–633. IEEE.

Elizabeth Soper, Stanley Fujimoto, and Yen-Yun Yu. 2021. Bart for post-correction of OCR newspaper text. In Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021), pages 284–290.

Lukas Stankevičius, Mantas Lukoševičius, Jurgita Kapočiūtė-Dzikienė, Monika Briedienė, and Tomas Krilavičius. 2022. Correcting diacritics and typos with a byt5 transformer model. Applied Sciences, 12(5):2636.

Michael Stubbs. 1996. Text and corpus analysis: Computer-assisted studies of language and culture. Blackwell Oxford.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. arXiv preprint, (2008.00401).

Monideepa Tarafdar, Cynthia M Beath, and Jeanne W Ross. 2019. Using ai to enhance business operations. MIT Sloan Management Review, 60(4).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems, 30.

David Wemhoener, Ismet Zeki Yalniz, and R Manmatha. 2013. Creating an improved version using noisy ocr from multiple editions. In 2013 12th International Conference on Document Analysis and Recognition, pages 160–164. IEEE.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. Transactions of the Association for Computational Linguistics, 10:291–306.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. arXiv preprint arXiv:2010.11934.

Ismet Zeki Yalniz and Raghavan Manmatha. 2011. A fast alignment scheme for automatic OCR evaluation of books. In 2011 International Conference on Document Analysis and Recognition, pages 754–758. IEEE.

A Training parameters

Due to the numerous sub-experiments in this study, models were trained on both RTX 4090 24GB and A100 80GB GPUs, without selecting different training parameters for different languages.

The mT5-base, ByT5-base, mBART-large, CharBERT, and ResNet50 models were all sourced from Hugging Face. Parameters for mT5-base, ByT5-base, and mBART-large were identical: dropout rate of 0.2, learning rate of 5e-4, batch size of 4, and 6 epochs. For CharBERT-related models, the dropout rate is 0.2, learning rate is 3e-5, batch size is 8, with 12 epochs. The ENSEMBLE and SCRATCH models have the same parameters: dropout rate of 0.1, learning rate of 1e-4, batch size of 32, and 30 epochs. All models use Adam optimizer and are trained in fp32 precision. The best models are selected based on dev loss.