RoundTripOCR: A Data Generation Technique for Enhancing OCR Error Correction in Low-Resource Devanagari Languages

Anonymous ACL submission

Abstract

Optical Character Recognition (OCR) technology has revolutionized the digitization of printed text, enabling efficient data extraction and analysis across various domains. Just like Machine Translation systems, OCR systems are prone to errors, stemming from factors such as poor image quality, diverse fonts, and language variations. In this work, we address the challenge of data generation and post-OCR error correction, specifically for low-resource languages. We propose a novel approach for synthetic data generation for Devanagari languages, RoundTripOCR, that tackles the scarcity of the OCR Error Correction dataset. In this work, we release a post-OCR text correction dataset for Hindi, Marathi, Bodo, Nepali, Konkani and Sanskrit. We also present a novel approach for OCR error correction by leveraging techniques from machine translation. Our method involves translating the erroneous OCR output into a corrected form by treating the OCR errors as mistranslations in a parallel text corpus. We employ a state-of-the-art pre-trained transformer model, mBART, to learn the mapping from erroneous to correct text pairs, effectively correcting OCR errors. The sample dataset can be accessed using the link[1].

1 Introduction

Devanagari script is the most widely used script in India and other Asian countries. There is a rich collection of ancient Devanagari manuscripts, which is a wealth of knowledge. To make these manuscripts available to people, efforts are being made to digitize these documents. Optical Character Recognition (OCR) technology has revolutionized the digitization and processing of written or printed text by enabling machines to automatically convert scanned documents into editable and searchable text formats. With the proliferation of document digitization efforts across various domains such as finance, healthcare, education, and government, OCR plays a crucial role in enhancing document accessibility, information retrieval, and automation of document-intensive workflows. However, despite significant advancements in OCR technology over the years, the accurate recognition of text from scanned documents remains a challenging task due to inherent complexities in document layouts, font variations, noise, and other distortions.

Traditional OCR systems typically follow a pipeline approach comprising image preprocessing, feature extraction, character segmentation, and recognition stages. While these systems have achieved remarkable success in many applications, they are susceptible to errors, especially when dealing with degraded or low-quality document images. OCR errors can manifest in various forms, including misrecognitions, substitutions, omissions, and insertions, leading to inaccuracies in the recognized text output. These errors not only impede the reliability of OCR systems but also pose significant challenges for downstream tasks such as information extraction, text mining, and content analysis. Addressing OCR errors requires robust error detection and correction mechanisms that can effectively handle a wide range of error patterns and variations.

Our contributions are as follows:

1. **RoundTripOCR**, a novel approach to generate a Post-OCR error correction dataset artificially for Devanagari languages.

2. Post-OCR text correction dataset containing around 13.1 million sentences in Hindi, 1.58 million sentences in Marathi, 2.54 million sentences in Bodo, 2.97 million sentences in Nepali, 1.95 million sentences in Konkani and 8.93 million sentences in Sanskrit.

3. Benchmarks for the Post-OCR error correction task based on the pre-trained Seq2Seq

[1] https://drive.google.com/drive/folders/1EmXGHRHo2-hRqxTN8OwXJK-zdcX-ilYE?usp=sharing

language model for all six languages.

2 Background

Addressing OCR errors is crucial for improving the overall quality and usability of digitized documents.

2.1 Types of OCR Errors

As mentioned by Volk et al. (2011) and Jatowt et al. (2019), OCR systems are prone to various types of errors that can occur during the process of text recognition from scanned documents. Understanding these errors is essential for developing effective error correction techniques and improving the overall accuracy of OCR systems. The most common types of OCR errors include:

- **Substitution Errors**: Substitution errors occur when the OCR system incorrectly recognizes a character and substitutes it with a different character. These errors often result from similarities between characters in terms of shape or visual appearance, making it challenging for the OCR system to distinguish between them accurately. For example, "o" might be substituted for "0" or "l" for "1" in the alphanumeric character set.

- **Omission Errors**: Omission errors occur when the OCR system fails to detect and recognize certain characters or words in the input image. This can happen due to factors such as poor image quality, low resolution, or indistinct boundaries between characters. Omission errors can significantly affect the readability and integrity of the recognized text, especially in documents with dense text or complex layouts.

- **Insertion Errors**: Insertion errors occur when the OCR system mistakenly inserts additional characters or words that are not present in the original image. These errors often arise from misinterpretations of noise or artifacts in the scanned document, leading to spurious insertions in the recognized text. Insertion errors can distort the meaning of the text and introduce inconsistencies in downstream processing tasks.

- **Deletion Errors**: Deletion errors occur when the OCR system erroneously removes or deletes characters or words from the input image. These errors can occur due to segmentation errors, where the OCR system incorrectly identifies boundaries between characters or words, leading to the omission of valid text fragments. Deletion errors can result in loss of information and inaccuracies in the recognized text output.

Over the years, researchers have explored various approaches to mitigate OCR errors, including rule-based post-processing techniques (Khosrobeigi et al., 2020), statistical language models (Mei et al., 2018), and machine learning-based methods (Virk et al., 2021). While these approaches have shown promise in certain scenarios, they often rely on handcrafted rules or linguistic resources, limiting their generalization to diverse document types and languages.
In recent years, there has been growing interest in applying advanced machine learning and natural language processing techniques to address OCR errors effectively. One promising direction is to leverage techniques from machine translation, which aims to automatically translate text from one language to another. By treating OCR errors as mistranslations and modelling the correction process as an automatic post-editing (APE) task, it is possible to harness the power of neural machine translation models to learn the mapping from erroneous to correct OCR text output. This paradigm shift not only enables end-to-end error correction but also facilitates the integration of contextual information and linguistic knowledge into the correction process, leading to more accurate and robust OCR systems.
In the upcoming sections of the paper, we will look at what are the different types of errors that exist in the OCR output and how we will generate the OCR error correction data and use it to mitigate the OCR errors.

2.2 Machine Translation

Machine translation (MT) describes the automatic process of translating text or speech from one language to another utilizing algorithmic methods and technology without the need for human translators. As mentioned by Bhattacharyya (2015) MT's objective is to facilitate communication and understanding amongst multilingual individuals by translating written or spoken text automatically. MT systems typically analyze the input text or speech using natural language processing (NLP) techniques, breaking it down into smaller linguistic units such as words, phrases, or sentences. These units are then processed and translated into the target language based on predefined rules, statistical models, or more advanced methods such as neural networks.

2.3 Automatic Post-editing and OCR Error Correction

Automatic Post-Editing (APE) is a methodology that utilizes various techniques to improve the quality of Machine Translation output automatically, including rule-based, statistical, and neural-based techniques (Vu and Haffari, 2018). APE systems are trained on human-edited translations, allowing them to identify and correct errors in grammar, fluency, and terminology. While MT systems have advanced significantly, they often produce translations that contain errors or lack fluency, especially with complex or domain-specific content. Output generated by a machine translation system is not always perfect and hence requires further editing (Parton et al., 2012). The task of editing a machine-translated text is referred to as post-editing (PE), which is time-consuming and costly. Hence, an efficient and automated system is required for post-editing. APE addresses these limitations by refining MT output, thereby enhancing the overall quality and usability of translations. APE can be easily integrated into existing machine translation workflows, making it a natural extension of the MT process.

While OCR systems play a crucial role in digitizing text, inherent limitations lead to errors in the extracted text. This necessitates post-processing techniques to refine the OCR output and achieve higher accuracy (Nguyen et al., 2021). Viewing this process through the lens of APE offers a valuable framework for developing effective error correction methods. Post-OCR error correction can be considered an Automatic Post Editing task. Similar to a machine translation system generating a translated sentence from a source language, the OCR system produces a "translated" text from the visual information in an image. This "translation" process is prone to errors due to limitations in both OCR systems, image quality, and stylistic variation. Just like an APE system refines a machine-translated sentence to improve fluency and accuracy, the post-OCR correction system aims to refine the text generated by the OCR system to remove errors and achieve a more accurate representation of the original document. Both MT and OCR error correction face common challenges like handling ambiguity, dealing with rare words, and adapting to stylistic variations.

2.4 Round-trip translation

Information that is intentionally manufactured rather than derived from actual events is referred to as synthetic data. Synthetic data generation techniques are generally employed to generate artificial data for training machine learning models and neural networks.

Due to insufficient post-editing data available for the WMT APE 2016 shared task (Bojar et al., 2016) to train neural models, Junczys-Dowmunt and Grundkiewicz (2016), created two phrase-based translation models: English-German and German-English, using provided parallel training data to conduct round-trip translation. Using them in the Round-trip Translation approach resulted in the generation of artificial post-editing triplets <*src, mt, pe*>. This artificial data creation method assisted in resolving the problem of insufficient training data, which frequently arises in NMT-based systems. Inspired by the Round-trip Translation approach, we propose a novel synthetic data generation technique, RoundTripOCR, which we discuss in detail in the following section.

3 RoundTripOCR: Data Generation Technique

The creation of artificial OCR data involves a systematic process aimed at simulating real-world scenarios keeping in consideration the common OCR error types and generating diverse datasets for training and evaluation purposes.

We use monolingual corpora taken from Technology Development for Indian Languages (TDIL)[2] and Maheshwari et al. (2022) to generate sentences of different lengths for generating artificial OCR data. To introduce variability and diversity into the dataset, 50 different Devanagari font combinations were selected from Google Fonts[3]. Each font style offered unique characteristics, such as varying stroke thickness, serif styles, and overall aesthetics as shown in Figure 1. Utilizing the selected Devanagari font combinations, 50 images could potentially be generated from a single sentence. PIL provides a comprehensive set of image processing functionalities, enabling the programmatic creation

[2] https://www.tdil-dc.in

[3] https://fonts.google.com/?subset=devanagari

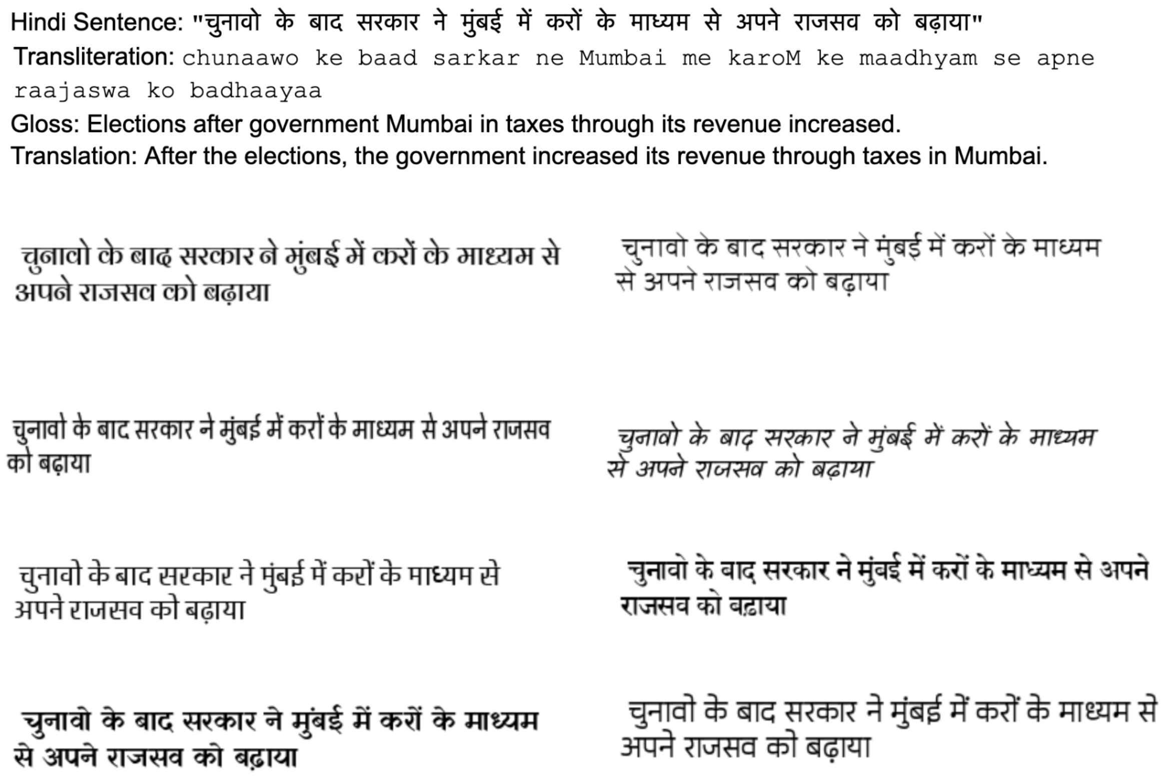

Figure 1: Examples of images generated with different fonts during RoundTripOCR data generation process.

of images with text rendered in specific font styles as described in the Appendix B. The generated images were subjected to optical character recognition (OCR) using the Pytesseract library as described in the Appendix C. Pytesseract is not supported for Bodo, Nepali, and Konkani languages. Thus, we used Pytesseract-Hindi for Bodo and Nepali, and Pytesseract-Marathi for Konkani due to similarities in these languages. We used Pytesseract-Sanskrit for the Sanskrit language. Pytesseract leverages machine-learning algorithms to extract text from images and convert them into machine-readable formats, including the Devanagari texts. The OCR process is aimed at simulating real-world OCR scenarios and generating text outputs from the rendered images. Since we can get 50 <*Text T, OCR output T'*> datapoints from a single sentence <*Text T*>, this approach can be extended to any low-resource language.

By following this methodology, as shown in Figure 2, a comprehensive artificial dataset for OCR error detection and correction was generated, encompassing a diverse range of text passages, font styles, and linguistic variations. This dataset serves as a valuable resource for training and evaluating OCR systems, enabling researchers and practitioners to develop robust OCR algorithms and assess their performance under various conditions.

3.1 Dataset

Leveraging the RoundTripOCR technique, we generate datasets containing around 13.1 million sentences in Hindi, 1.58 million sentences in Marathi, 2.54 million sentences in Bodo, 2.97 million sentences in Nepali, 1.95 million sentences in Konkani and 8.93 million sentences in Sanskrit. To assess the quality of the artificial dataset, we provided 100 data samples from the Hindi and Marathi datasets to language and OCR experts for evaluation. Their feedback indicated that the dataset contains errors very similar to those found in real OCR outputs. We have observed that the quality of training data obtained either from real-world data or by our RoundTripOCR technique is comparable. Thus, we believe that using our technique is a convenient and appropriate way to generate a synthetic dataset to train OCR error correction systems.

Font analysis revealed significant variations in error rates. Specifically, fonts such as Khand-Regular, Rajdhani-Regular, Nirmala, and Biryani exhibited the highest CERs, exceeding 7%. Conversely, fonts like Gargi, Karma-Regular, NotoSans-Regular, and VesperLibre-Regular demonstrated remarkably low CERs, each falling below 1% as shown in Figure

# of Sent.	Hindi	Marathi	Bodo	Nepali	Konkani	Sanskrit
Train dataset	13,129,200	1,581,405	2,541,649	2,970,148	1,950,874	8,935,790
Validation set	10,000	10,000	10,000	10,000	10,000	10,000
Test dataset	10,000	10,000	10,000	10,000	10,000	10,000

Table 1: Dataset distribution for different languages

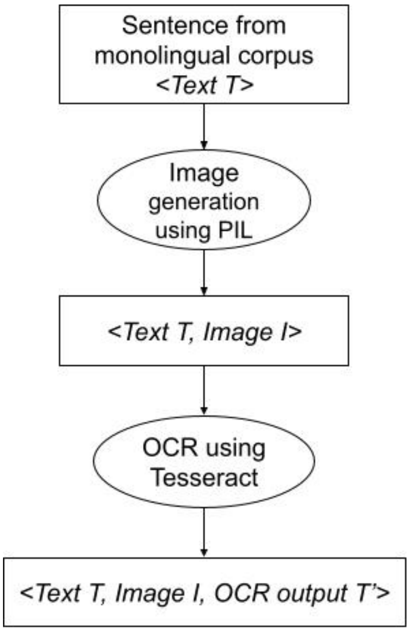

Figure 2: RoundTripOCR: The process of Artificial OCR Data Generation. At the end of the RoundTripOCR process, <*OCR output T'*> will act as OCR output, and <*Text T*> will act as corrected OCR output text.

3. Our findings suggest that models trained on a diverse range of fonts perform more robustly than those trained solely on a single font. This observation underscores the importance of font diversity in enhancing OCR error correction models' performance and resilience. The sample dataset can be accessed using the link[4].

4 Experiments and Results

4.1 Fine-tuning sequence to sequence model using our dataset

In this experiment, we harnessed a vast dataset consisting of 13 million data points, curated through our RoundTripOCR data generation methodology. Leveraging this extensive corpus, we conducted a series of experiments employing a sophisticated sequence-to-sequence model: ***mBART***.

To facilitate effective model training and evaluation, we partitioned the dataset into three distinct sets: training, testing, and validation. Specifically, the testing set contained 10,000 pairs, while the validation set comprised 10,000 pairs. This meticulous partitioning strategy enabled us to assess the performance of our models accurately and reliably across various metrics. We fine-tuned the model for 4 epochs and 3000 max_steps.

mBART (Multilingual BART) represents an extension of the BART architecture tailored specifically for multilingual text processing as explained in the Appendix D.

To facilitate an in-depth investigation, we further curated two distinct datasets. The first dataset encompassed text samples spanning various fonts, providing a rich diversity in font styles. Conversely, the second dataset exclusively features a single font style; particularly we chose the Sumana font as it shows a close to average CER when compared with all the fonts used in the creation of the dataset. This deliberate bifurcation allowed us to explore the potential advantages conferred by employing data with varying font styles, thereby enriching our understanding of the model's performance under different font conditions.

4.2 Results

Results on Hindi test dataset		
Model	CER	WER
Tesseract (baseline)	2.247%	5.833%
mBART (single font)	2.105%	5.817%
mBART (all fonts)	**1.556%**	**3.474%**

Table 2: Results on Hindi test dataset

Results on Marathi test dataset		
Model	CER	WER
Tesseract (baseline)	4.100%	15.372%
mBART (single font)	3.592%	14.472%
mBART (all fonts)	**2.464%**	**9.886%**

Table 3: Results on Marathi test dataset

[4] https://drive.google.com/drive/folders/1EmXGHRHo2-hRqxTN8OwXJK-zdcX-ilYE?usp=sharing

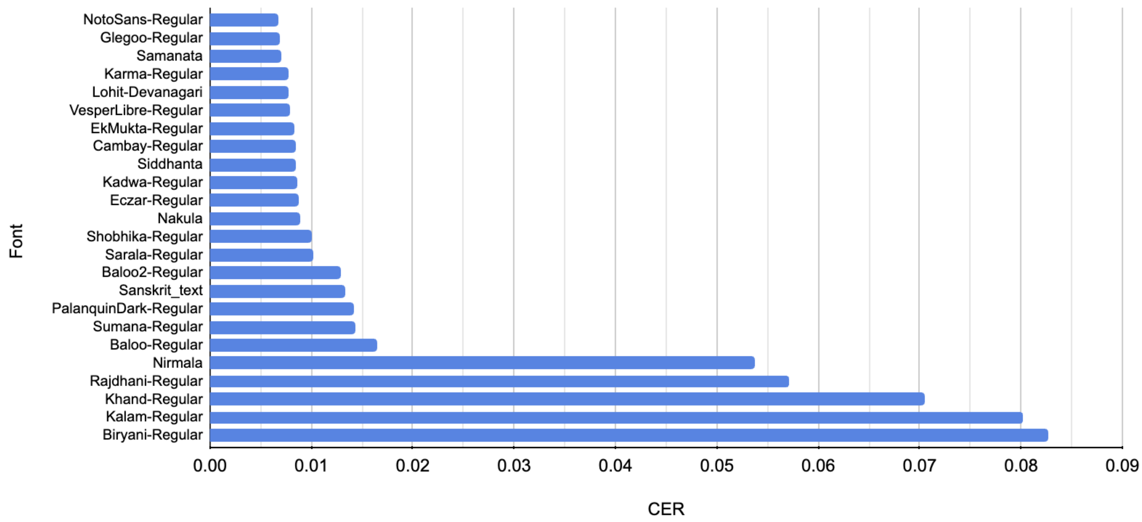

Figure 3: Comparision of different fonts and their CER in the Hindi test dataset

Results on Konkani test dataset		
Model	CER	WER
Tesseract (baseline)	4.222%	16.803%
mBART (single font)	3.279%	13.058%
mBART (all fonts)	**2.265%**	**8.524%**

Table 4: Results on Konkani test dataset

Results on Nepali test dataset		
Model	CER	WER
Tesseract (baseline)	5.783%	24.287%
mBART (single font)	3.189%	14.266%
mBART (all fonts)	**2.387%**	**10.651%**

Table 5: Results on Nepali test dataset

Results on Bodo test dataset		
Model	CER	WER
Tesseract (baseline)	5.893%	24.027%
mBART (single font)	3.677%	13.016%
mBART (all fonts)	**2.359%**	**6.822%**

Table 6: Results on Bodo test dataset

Results on Sanskrit test dataset		
Model	CER	WER
Tesseract (baseline)	8.766%	44.730%
mBART (single font)	6.428%	29.294%
mBART (all fonts)	**5.670%**	**25.501%**

Table 7: Results on Sanskrit test dataset

The evaluation of OCR model performance across multiple Indian languages reveals notable insights. In the Hindi and Marathi test datasets, Tesseract (baseline) exhibits CERs of 2.247% and 4.100%, respectively, with corresponding WERs of 5.833% and 15.372%. Contrastingly, mBART (all fonts) achieves significantly lower error rates, recording CERs of 1.556% and 2.464% and WERs of 3.474% and 9.886% for Hindi and Marathi, respectively. Similar trends are observed in the Konkani and Nepali datasets, where mBART (all fonts) maintains a competitive edge with CERs of 2.265% and 2.387% and WERs of 8.524% and 10.651%. In more challenging languages like Bodo and Sanskrit, mBART (all fonts) continues to demonstrate superiority over the Tesseract baseline, achieving CERs of 2.359% and 5.670% and WERs of 6.822% and 25.501%, respectively. These results underscore the effectiveness of mBART, particularly when trained on diverse font variations, in enhancing OCR accuracy across a range of complex languages.

5 Conclusion and Future Work

In our work, we introduced a novel approach for OCR error correction data generation and created a vast dataset comprising 13.1 million sentences in Hindi, 1.58 million sentences in Marathi, 2.54 million sentences in Bodo, 2.97 million sentences in Nepali, 1.95 million sentences in Konkani and 8.93 million sentences in Sanskrit. Notably, our proposed methodology is versatile and can be extended to other low-resource languages that follow the Devanagari script. By leveraging monolingual corpora, our approach enables the generation of OCR correction datasets, thus addressing the scarcity of data in such languages.

The findings from our experimentation underscore the efficacy of approaches from Machine Translation for the task of OCR output correction, specifically state-of-the-art models like mBART, trained on diverse datasets to substantially enhance OCR accuracy. By improving the accuracy of OCR systems, our research contributes to making textual content more accessible and usable, thereby facilitating broader access to information and knowledge in multilingual societies. Our findings suggest that models trained on a diverse range of fonts perform more robustly than those trained solely on a single font. This observation underscores the importance of font diversity in enhancing OCR error correction models' performance and resilience.

Our findings motivate the exploration of data augmentation techniques utilizing synthetically generated Devanagari script images. By incorporating these images with controlled variations in font styles, noise levels, and image degradations, we can investigate the impact on model generalization and robustness towards real-world document image complexities. This research could delve into generative adversarial networks (GANs) or other image synthesis techniques to create a diverse and realistic training dataset for enhanced OCR performance.

We propose the experimental findings in this work as a baseline, based on which future work can focus on novel and sophisticated techniques for the task of OCR correction, including improvements to the architecture.

6 Limitations

Our work focuses on improving OCR error correction for Devanagari script languages only. Extending this approach to achieve true multilingual OCR is a complex endeavour. Different languages possess unique linguistic characteristics, script variations, and language-specific nuances. Developing a single model capable of handling this vast diversity effectively remains a challenge. Future work should explore techniques for creating language-agnostic or language-adaptive models to address these limitations and achieve broader multilingual OCR applicability.

7 Ethical Statement

This research utilizes datasets that are openly available in the public domain. The data employed for generating artificial data in this study was sourced from publicly accessible repositories, ensuring that there are no privacy or ethical concerns associated with their use. No user information was present in any of the datasets used in the work, protecting the privacy and identity of users. Also, the synthetic data generated as a part of this work will be released under the CC-BY-SA 4.0 license publicly for further research. We understand that every dataset is subject to intrinsic bias and that computational models will inevitably learn biased information from any dataset.

References

Pushpak Bhattacharyya. 2015. *Machine translation*. CRC Press.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Alex Clark et al. 2015. Pillow (pil fork) documentation. *readthedocs*.

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.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. *arXiv preprint arXiv:1605.04800*.

Z. Khosrobeigi, H. Veisi, H.R. Ahmadi, and H. Shabanian. 2020. A rule-based post-processing approach to improve persian ocr performance. *Scientia Iranica*, 27(6):3019–3033.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and

Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Ayush Maheshwari, Nikhil Singh, Amrith Krishna, and Ganesh Ramakrishnan. 2022. A benchmark and dataset for post-OCR text correction in Sanskrit. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6258–6265, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jie Mei, Aminul Islam, Abidalrahman Moh'd, Yajing Wu, and Evangelos Milios. 2018. Statistical learning for ocr error correction. *Information Processing and Management*, 54:Pages 874–887.

Thi Nguyen, Adam Jatowt, Mickaël Coustaty, and Antoine Doucet. 2021. Survey of post-ocr processing approaches. *ACM Computing Surveys*, 54:1–37.

Kristen Parton, Nizar Habash, Kathleen McKeown, Gonzalo Iglesias, and Adrià de Gispert. 2012. Can automatic post-editing make MT more meaningful. In *Proceedings of the 16th Annual conference of the European Association for Machine Translation*, pages 111–118, Trento, Italy. European Association for Machine Translation.

Ray Smith. 2007. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE.

Shafqat Mumtaz Virk, Dana Dannélls, and Azam Sheikh Muhammad. 2021. A novel machine learning based approach for post-OCR error detection. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1463–1470, Held Online. INCOMA Ltd.

Martin Volk, Lenz Furrer, and Rico Sennrich. 2011. Strategies for reducing and correcting ocr errors. In *Language Technology for Cultural Heritage: Selected Papers from the LaTeCH Workshop Series*, pages 3–22. Springer.

Thuy-Trang Vu and Gholamreza Haffari. 2018. Automatic post-editing of machine translation: A neural programmer-interpreter approach. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3048–3053, Brussels, Belgium. Association for Computational Linguistics.

A Appendix

The fonts used to generate images are: EkMukta-Regular, Arya-Regular, YatraOne-Regular, Siddhanta, Sura-Regular, Samanata, Karma-Regular, Nirmala, Asar-Regular, VesperLibre-Regular, Kurale-Regular, MartelSans-Regular, SakalBharati Normal, Biryani-Regular, Sumana-Regular, Sarai, Laila-Regular, Rajdhani-Regular, Nakula, Shobhika-Regular, Baloo-Regular, Lohit-Devanagari, Amiko-Regular, Akshar Unicode, Palanquin-Regular, Eczar-Regular, Glegoo-Regular, Mukta-Regular, Sanskrit2003, PalanquinDark-Regular, Baloo2-Regular, Kalam-Regular, Sanskrit_text, Halant-Regular, Hind-Regular, Cambay-Regular, PragatiNarrow-Regular, Kadwa-Regular, Kokila, Sahadeva, Utsaah, Sahitya-Regular, Khand-Regular, Sarala-Regular, NotoSans-Regular, Jaldi-Regular, RhodiumLibre-Regular, Yantramanav-Regular and Gargi.

Languages which follow devnagari script are: Apabhramsha, Angika, Awadhi, Bajjika, Bhili, Bhojpuri, Boro, Braj, Chhattisgarhi, Dogri, Garhwali, Haryanvi, Hindi, Kashmiri, Khandeshi, Konkani, Kumaoni, Magahi, Maithili, Marathi, Marwari, Mundari, Nagpuri, Newari, Nepali, Pāli, Pahari, Prakrit, Rajasthani, Sanskrit, Santali, Saraiki, Sherpa, Sindhi, Surjapuri, and many more. We can extend our approach to all these languages.

B PIL (Python Image Library)

In today's digital era, the prevalence of digital images is ubiquitous. When working with Python, developers have access to a plethora of image-processing libraries to augment the capabilities of digital images. Among the most widely used libraries are OpenCV, Python Imaging Library (PIL), Scikit-image, and Pillow.

Pillow[5], an extension of PIL (Python Image Library), stands out as a crucial module for image processing in Python. While PIL was once pivotal, it ceased support in 2011 and does not cater to Python 3. In contrast, Pillow offers expanded functionalities, runs seamlessly across major operating systems, and fully supports Python 3. It boasts compatibility with a diverse range of image formats, including "jpeg", "png", "bmp", "gif", "ppm", and "tiff". With Pillow, developers can perform a multitude of operations on digital images, ranging from basic tasks like point operations to advanced functionalities such as image filtering using built-in convolution kernels and colour space conversions.

The Python Imaging Library (Clark et al., 2015), commonly known as PIL, is particularly well-suited for image archival and batch-processing applications. Leveraging the Python Pillow package,

[5]https://pypi.org/project/pillow

developers can seamlessly perform tasks such as creating thumbnails, converting between different image formats, and printing images. The Pillow library encompasses a comprehensive suite of basic image-processing functionalities, including image resizing, rotation, and transformation. Additionally, the histogram method available in the Pillow module facilitates the extraction of statistical data from images, which can then be utilized for statistical analysis and automatic contrast enhancement.

C Tesseract

Tesseract[6] is an open-source OCR Engine designed to extract printed or handwritten text from images. Originally developed by Hewlett-Packard, its development was later taken over by Google (Smith, 2007). Tesseract boasts support for language recognition in over 100 languages straight out of the box. The latest iteration, Tesseract 4.0, features AI integration through LSTM Neural Networks, enhancing its capability to detect and recognize inputs of varying sizes with greater accuracy and efficiency.

Tesseract's versatility lies in its compatibility with various programming languages and frameworks through wrappers like Pytesseract, commonly known as Python-Tesseract. This tool not only serves as an open-source OCR library for Python but also acts as a wrapper for Google's Tesseract OCR Engine. Pytesseract offers the convenience of being a standalone script, enabling direct printing of recognized text without the need to convert it into a separate file. It supports a wide range of image formats, including JPEG, PNG, GIF, BMP, TIFF, and more, making it a popular choice for image-to-text OCR tasks in Python.

Pytesseract played an important role in our work of data generation, where it was used for performing the OCR of the images that we generated using PIL.

D mBART (Multilingual BART)

Bidirectional and Auto-Regressive Transformers (BART) is a sequence-to-sequence model proposed by Lewis et al. (2020) that combines the strengths of bidirectional and auto-regressive transformers. Its architecture consists of an encoder-decoder transformer model with masked self-attention mechanism in the decoder.

[6] https://github.com/tesseract-ocr/tesseract

Encoder: The encoder processes the input sequence bidirectionally using self-attention mechanisms to capture contextual information efficiently. It produces contextualized representations of the input tokens.

Decoder: The decoder generates the output sequence autoregressively, conditioning on the encoder's contextualized representations and previously generated tokens. It employs causal self-attention mechanisms to ensure that each token is generated based only on previously generated tokens, as opposed to looking ahead of the current decoding step.

BART is pre-trained on large-scale text corpora using denoising autoencoding objectives, where corrupted input sequences are reconstructed to their original forms. This pre-training objective encourages the model to learn robust representations of text and enables it to perform well on a wide range of natural language processing tasks.

mBART (Multilingual BART) by Liu et al. (2020) extends the BART architecture to support multilingual text processing. It is designed to handle input sequences in multiple languages and generate output sequences in the corresponding target languages.

Language Embeddings: mBART incorporates language embeddings into its architecture to enable language-specific processing. These embeddings encode information about the source and target languages, allowing the model to adapt its behaviour based on the language of the input and output sequences.

Cross-lingual Pre-training: mBART is pre-trained on multilingual text corpora using cross-lingual objectives, where input sequences from different languages are reconstructed to their original forms. This pre-training objective encourages the model to learn language-agnostic representations of text and enables it to perform effectively on multilingual tasks.