POCO: Low Cost Post-OCR Correction Dataset Construction via Character-Level Probabilistic Simulation

Anonymous ACL submission

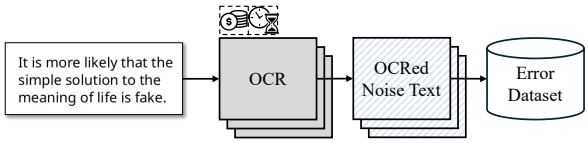
Abstract

Rapid advancements in Large Language Models (LLMs) have significantly enhanced multimodal AI, emphasizing the importance of effectively processing image-based text through Optical Character Recognition (OCR). However, the extensive computational resources required by high-performing LLMs limit their commercial applicability, making smaller OCR models prevalent despite their susceptibility to recognition errors, particularly with low-quality images. Addressing OCR errors through post-correction is essential but constrained by the scarcity of high-quality training datasets. Existing synthetic methods are either resource-intensive or insufficiently realistic. To overcome these limitations, we propose POCO (Post-OCR Correction with Output distributions), a novel low-cost framework leveraging character-level probabilistic simulations to synthetically generate realistic OCR error datasets. Our framework employs a lightweight vision model (ResNet34) to predict character probabilities and inject OCR-like errors into text corpora, effectively creating extensive and high-quality training datasets. Experimental results demonstrate significant improvements in OCR post-correction, validating our framework's practicality and effectiveness.

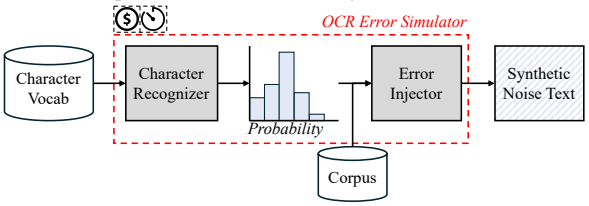
1 Introduction

Recent advances in Large Language Models (LLMs) have accelerated multimodal AI development, highlighting the growing importance of processing image-based text. The increasing prevalence of scanned or photographed text documents, such as reports, papers, signs, and menus, has made Optical Character Recognition (OCR) essential.

OCR is widely used on camera-equipped mobile devices. While powerful LLMs like ChatGPT(OpenAI, 2024) show strong OCR performance, their high computational demands limit widespread commercial use. Consequently, smaller



(a) OCR post-correction datasets generation with OCRs.



(b) OCR post-correction datasets generation with POCO.

Figure 1: OCR post-correction datasets generation

OCR models dominate commercial applications but frequently struggle with errors, especially in low-quality images(Kiss et al., 2019; Kashid and Bhattacharyya, 2024; Vitman et al., 2022).

OCR errors negatively impact downstream AI applications and commercial systems, making error correction crucial. However, the availability of high-quality training datasets for OCR post-correction is limited. Existing dataset construction, such as generating images from ground-truth texts or experimentally extracting error distributions, often yield inconsistent quality and require significant effort (Guan and Greene, 2024; Ignat et al., 2022).

To address these challenges, we propose POCO (Post-OCR Correction with Output distributions), an efficient OCR post-correction dataset generation framework using character-level probabilistic simulations. Our method, illustrated in Figure 1, employs a vision model trained on synthetic character recognition data, termed an OCR Error Simulator, to inject realistic OCR-like errors into text corpora. Experimental results confirm that datasets produced by our approach significantly improve OCR post-correction performance.

The main contributions of this paper are:

- We propose POCO, a simple and cost-

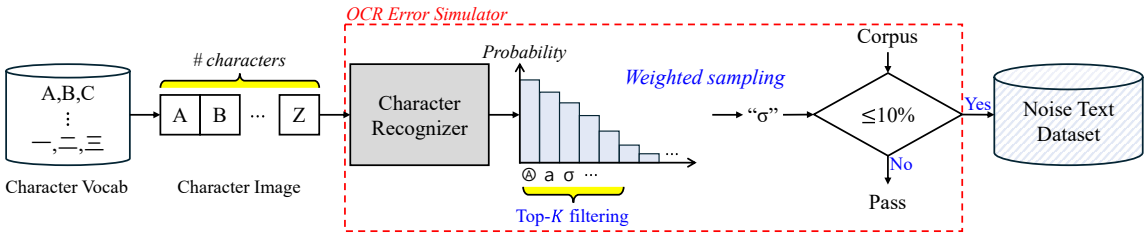


Figure 2: POCO framework pipeline for OCR post-correction dataset geneartion

effective framework for simulating OCR errors at the character level.

- POCO improves OCR post-correction, especially with pretrained models like mT5.
- We validate its effectiveness in handling common OCR errors such as character confusion and spacing.

2 Related Work

Several approaches have been proposed for generating datasets for OCR post-correction:

Random error injection method (D'hondt et al., 2017) inserts, deletes, or substitutes characters randomly into clean text corpora. It is simple, easy to implement, and resource-efficient but often fails to accurately mimic the actual error distribution of OCR systems, limiting its generalization in real-world scenarios.

Image-generation-based method (Boros et al., 2022) renders texts as images, introduces visual noise, and processes them through actual OCR systems to produce realistic error patterns. While this method generates realistic data suitable for image-based training, the type and amount of errors are limited by the OCR system used, and substantial computational resources are required for image generation and processing.

Real error distribution-based insertion method (Grundkiewicz et al., 2019) aligns OCR-processed texts with their original versions to derive probabilities of insertion, deletion, and substitution errors. Errors are then introduced into clean corpora based on these distributions. This method realistically reflects actual OCR errors, but it is resource-intensive due to the additional alignment algorithms and text corpora required, making it challenging to apply to low-resource languages.

Glyph similarity-based insertion method (Guan and Greene, 2024) quantitatively assesses visual similarity between character pairs to simulate substitution errors effectively. It leverages various computer vision techniques, employing Jaccard similarity and average distances to calculate glyph similarity. While effective without additional corpora, it faces computational complexity challenges $O(n^2)$ in languages with large character sets.

3 POCO Framework

As illustrated in Figure 2, we developed a vocabulary specific to each language to create character-level image-text label datasets for training an character recognizer. Subsequently, we used the trained recognizer to inject realistic OCR errors into a text corpus, thus generating datasets for OCR post-correction tasks.

3.1 Vision Model-Based Character Recognizer

We employed a lightweight vision model (non-pretrained ResNet34 (He et al., 2016)) as our character recognizer. The recognizer was trained to predict characters from images, each containing randomly selected sequences of 1 to 8 characters generated via the PIL library. We developed a unified recognizer capable of handling both Chinese and Korean texts by randomly utilizing seven publicly available fonts compatible with both languages.

The training data included 32,546 characters, comprising numerals, common special characters (50), English alphabets (52), Korean characters (11,266), Japanese Hiragana and Katakana (189), and CJK Unified Ideographs (20,989).

3.2 OCR Error Simulator

After training, logits were computed for each character in the vocabulary. For each character, the top five most probable confusion characters (excluding itself) were selected as potential errors, limited within each language or among special characters.

We selected the Chinese-Korean subset from the AIHub multilingual humanities translation corpus (Agency, 2023) to create OCR error-injected

	Chinese	Korean
Label	柏林墙各处的检查站大门同时敞开。	이 부족연맹체 국가는 청나라 옹정제까지 이어가다 1732년에 소멸되었다.
POCO	柏林衛各处的尬查站大门同时儆开。	이 부족연맹체 콕가는 청나라 옹정제까지 이어가다 1?32년에 소멸되었ㅁ.
EasyOCR	柏林墙各处的检查站大门同时敞开。	이 부족연망체 국가는 청나라 용정제까지 이어가다 1732년에 소별되없다.
PaddleOCR	柏林墙各处的检查站大门同时_开。	이 부족연맹체 국가는 청나라 옹정제까지 이어가다 1732년에 소멸되었다_

Figure 3: Simulated OCR error data and actual OCR output data ('_' indicates a missing character)

GT	Top-k highest-probability characters						
	1	2	3	4	5	6	…
B	B	E	R	8	H	b	…
먹	먹	막	댁	멐	덕	악	
痛	痛	痛	癘	癤	痌	癟	

Figure 4: Character recognizer probability sample, Gray cells indicate exclusion due to identity with themselves

	Korean	Chinese
Train	231,069	57,217
Val	30,652	13,674
Test	10,000	10,000

Table 1: Dataset split statistics

datasets. Errors were injected with a 10% probability per character, maintaining approximately a 10% Character Error Rate (CER). The confusion characters were selected based on their logits probabilities, weighted as 40%, 30%, 15%, 10%, and 5%, respectively, to reduce bias from extreme logit differences. Additionally, spacing was randomly inserted (1% probability) or deleted (10% probability) in Korean texts.

Figure 3 compares our synthetically generated errors with results from actual OCR systems (EasyOCR[1] and PaddleOCR (Du et al., 2020)), confirming the simulator's effectiveness in reproducing realistic OCR error patterns, including spacing variations.

Figure 4 shows sample data based on the character recognizer's probability distribution. Top-k candidates often same the ground truth (GT), with the highest-probability candidate frequently being the GT itself. To focus on meaningful variations, identical matches to the GT are excluded, and the top 5 remaining candidates are selected.

4 Experiments

We conducted experiments to evaluate the effectiveness of our proposed OCR post-correction dataset generation framework. Specifically, we aimed to: **1)** Train OCR correction models (XLM-RoBERTa (Conneau et al., 2020) and T5 (Raffel et al., 2020)) on the generated datasets and assess their effectiveness by measuring improvements in CER on real OCR outputs; and **2)** analyze the types of errors produced by actual OCR models compared to those corrected by our framework.

4.1 Experimental Setup

Datasets The experiments utilized Chinese and Korean corpora as detailed in Table 1. As the provided corpus contained only training and validation splits, we randomly sampled 10,000 examples from the training set for testing purposes. We injected errors into the train and validation datasets using the OCR Error Simulator, achieving approximately a 10% CER. The test datasets incorporated real OCR errors obtained using PaddleOCR and EasyOCR.

OCR Error Simulator We utilized non-pretrained model ResNet34 as backbones. The model predicted sequences of 1 to 8 characters, padded appropriately. For dataset generation, only ResNet34 was used. Models were trained for 500 epochs with a batch size of 256 and a learning rate of $5e^{-4}$.

OCR Post-Correction Models We trained both non-pretrained and pretrained OCR correction models using publicly available models from HuggingFace: FacebookAI/xlm-roberta-base (Conneau et al., 2020) and google/mt5-base (Xue et al., 2021). The batch size was set to 128, employing gradient accumulation if memory constraints arose. Models were trained with a learning rate of $5e^{-5}$ and early stopping based on validation CER with a patience of 10 epochs.

4.2 Experimental Results

4.2.1 Effectiveness of OCR Post-Correction Framework

As shown in Table 2, the pretrained mT5 models consistently outperformed others in correct-

[1] https://github.com/JaidedAI/EasyOCR.git

Language	OCR Model		XLM-RoBERTa		mT5		GTP-4o
			Non-Pretrained	Pretrained	Non-Pretrained	Pretrained	
Korean	**EasyOCR**	0.074	0.122	0.111	0.077	**0.063**(+15%)	0.223
Chinese		0.062	0.092	0.059(+5%)	0.095	**0.057**(+8%)	0.139
Korean	**PaddleOCR**	0.040	0.043	0.031(+23%)	**0.021**(+48%)	0.023(+43%)	0.220
Chinese		0.012	0.034	0.013	0.025	**0.012**(-%)	0.134

Table 2: CER Performance of OCR Post-Correction

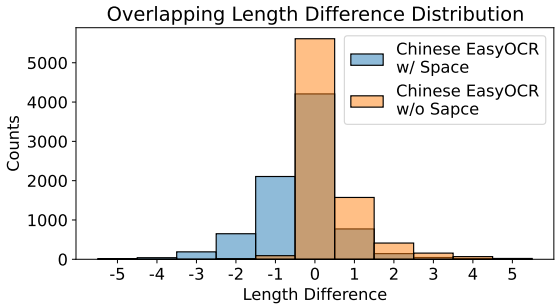


Figure 5: Length distribution of EasyOCR Chinese results based on the presence of whitespace

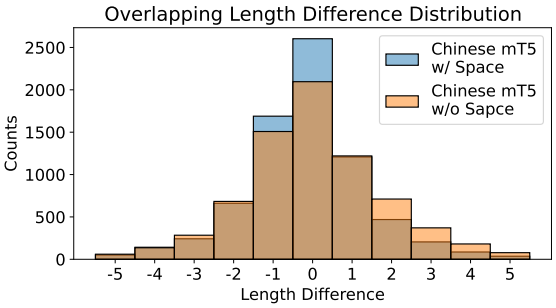


Figure 6: Length distribution of mT5 Chinese post-correction results based on the presence of whitespace

ing OCR errors from EasyOCR and PaddleOCR outputs. The XLM-RoBERTa model also showed improvements in several cases. The Chinese PaddleOCR model had minimal errors, resulting in limited correction potential.

When GPT-4o (ChatGPT) was utilized for correction, the CER increased, likely due to its decoder-based generation approach, which preserves semantic meaning but changes sentence structure, thus raising the error rate. Effective OCR correction requires precise character restoration, highlighting the need for detailed prompting and additional tuning when employing LLMs (Figures 7 and 8).

These results validate the efficacy of our framework, confirming the suitability of encoder-decoder architectures for OCR correction tasks. Encoder-only models demonstrated limited correction capabilities, and decoder-only models often failed to train adequately. Pretrained models consistently showed superior performance.

4.2.2 Error Analysis of OCR and Post-Correction Models

We analyzed the lengths of original and OCR sentences to categorize errors. Positive differences indicated character omissions by models, whereas negative differences suggested unnecessary insertions. Equal lengths indicated balanced insertion-omission errors or misrecognized characters.

Figure 5 illustrates that Chinese EasyOCR results contained significant space insertion errors, as evidenced by length discrepancies that were substantially reduced upon removing spaces. Conversely, the mT5 correction results (Figure 6) demonstrated significantly improved consistency, indicating effective correction of spacing errors by our framework.

5 Conclusion

In this paper, we introduced POCO, a cost-efficient OCR post-correction dataset construction framework utilizing character-level probabilistic simulation. Unlike existing methods, our approach effectively generates realistic OCR errors without requiring intensive computational resources or extensive manual intervention. Through comprehensive experiments, we demonstrated the efficacy of datasets generated by POCO, significantly improving OCR post-correction performance, particularly with pretrained encoder-decoder architectures such as mT5. Our analyses confirmed that POCO effectively addresses typical OCR issues, including erroneous space insertions and character misrecognitions, enhancing overall OCR reliability. Future research directions include further refining the error-injection model for increased adaptability across various languages and OCR systems, ultimately contributing towards robust multimodal text processing in practical applications.

Limitations

Although the proposed POCO framework demonstrates effectiveness in generating realistic OCR error datasets efficiently, several limitations remain:

Character-Level Error Injection Simplification POCO's simulation of OCR errors relies solely on character-level probabilistic modeling, potentially overlooking more complex, contextual OCR errors that frequently occur in real-world OCR processes. Future studies should explore integrating word-level or sentence-level contextual error modeling.

Fixed Error Rate for Dataset Generation The current study primarily uses a fixed error injection rate (10%) across experiments. Real OCR systems often exhibit varying error rates depending on text quality, fonts, and noise conditions. Evaluating different error injection rates would strengthen the generalizability and robustness of the proposed approach.

Limited Evaluation Metrics The evaluation presented in the paper exclusively focuses on Character Error Rate (CER). However, OCR errors significantly impact downstream NLP tasks such as Machine Translation (MT), Named Entity Recognition (NER), and Information Extraction (IE). Additional evaluations using task-specific metrics would provide a clearer picture of POCO's practical impact.

Dataset and Language Generalization Although the framework theoretically supports multilingual capabilities, experiments and evaluations are primarily conducted on Korean and Chinese datasets. While promising for these languages, further validation is required to confirm the framework's effectiveness and generalizability to other languages and scripts, particularly low-resource and morphologically rich languages.

Insufficient Model Variability Analysis The character recognition model utilized (ResNet34) is chosen primarily for its computational efficiency, but deeper insights into how different model architectures or training regimens might influence error realism or dataset quality are not explored. Future work should investigate a broader range of models, potentially including vision transformers or deeper CNN variants, to comprehensively assess performance trade-offs.

References

National Information Society Agency. 2023. Korean-multilingual translation corpus (humanities). https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=data&dataSetSn=71498. Dataset.

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.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.

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.

Yuning Du, Chenxia Li, Ruoyu Guo, Xiaoting Yin, Weiwei Liu, Jun Zhou, Yifan Bai, Zilin Yu, Yehua Yang, Qingqing Dang, and 1 others. 2020. Pp-ocr: A practical ultra lightweight ocr system. *arXiv preprint arXiv:2009.09941*.

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.

Shuhao Guan and Derek Greene. 2024. Advancing post-ocr correction: A comparative study of synthetic data. *arXiv preprint arXiv:2408.02253*.

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. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1164–1174.

Harshvivek Ankush Kashid and Pushpak Bhattacharyya. 2024. Roundtripocr: A data generation technique for enhancing post-ocr error correction in low-resource devanagari languages. In *Proceedings of the 21st International Conference on Natural Language Processing (ICON)*, pages 274–284.

Martin Kiss, Michal Hradis, and Oldrich Kodym. 2019. Brno mobile ocr dataset. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1352–1357. IEEE Computer Society.

OpenAI. 2024. Gpt-4o system card. Accessed: 2025-05-20.

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. *Journal of machine learning research*, 21(140):1–67.

Oxana Vitman, Yevhen Kostiuk, Paul Plachinda, Alisa Zhila, Grigori Sidorov, and Alexander Gelbukh. 2022. Evaluating the impact of ocr quality on short texts classification task. In *Mexican International Conference on Artificial Intelligence*, pages 163–177. Springer.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498.

A Prompts

Figures 7 and 8 illustrate system and user prompts for OCR Post-Correction. We used ChatGPT for experiments inference and translation.

System Prompt

role: system,
content:
You are an expert in correcting grammar errors in texts extracted via Optical Charecter Recognition(OCR).

Your job is to correct each sentence into fluent and natural Chinese without changing the original meaning.

Return only the corrected texts in the same order.
Do not explain anything

Figure 7: System prompt for OCR Post-Correction.

User Prompt

role: user,
content:
{Sentence 1}
{Sentence 2}
…
—
Return a list of the same length with each sentence cleaned.

Figure 8: User prompt for OCR Post-Correction

B Additional Experimental Results

Figures 9 and 10 visualize the distribution of sentence lengths before and after processing with EasyOCR and PaddleOCR, comparing them to the original sentences. To examine the impact of whitespace, we present two versions: one including spaces and one without.

Figures 11 and 12 visualize the distribution of sentence lengths before and after processing with XLM Roberta, comparing them to the original sentences. To examine the impact of whitespace, we present two versions: one including spaces and one without.

Figures 13 and 14 visualize the distribution of sentence lengths before and after processing with mT5, comparing them to the original sentences. To examine the impact of whitespace, we present two versions: one including spaces and one without.

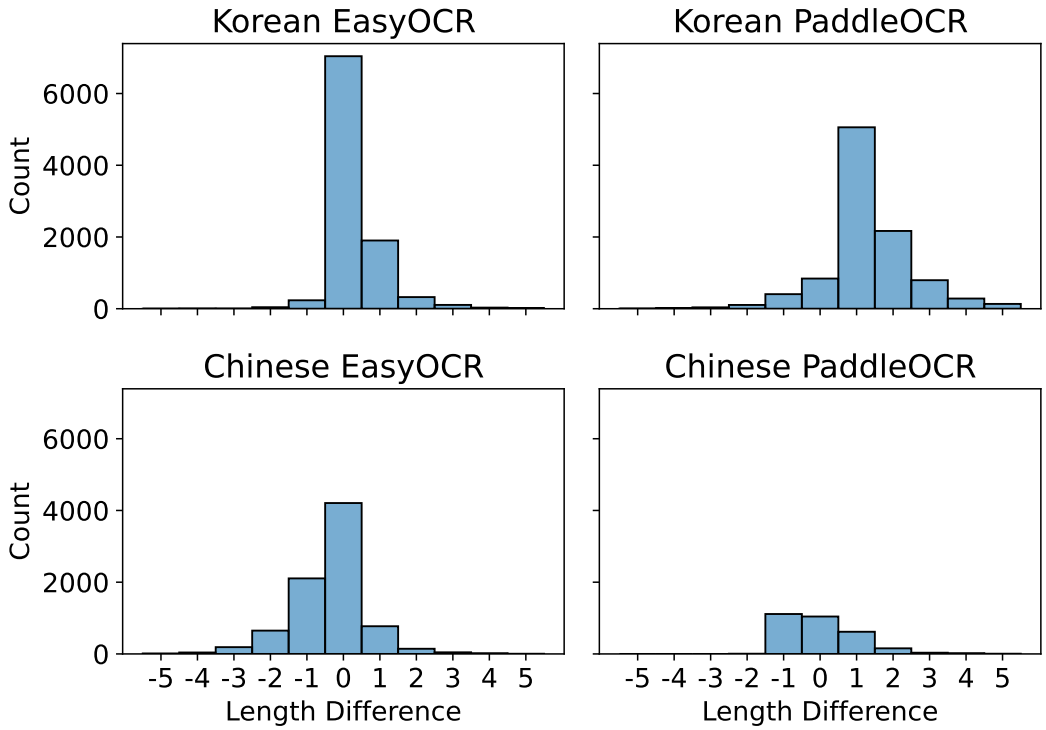


Figure 9: Length distribution of OCR results

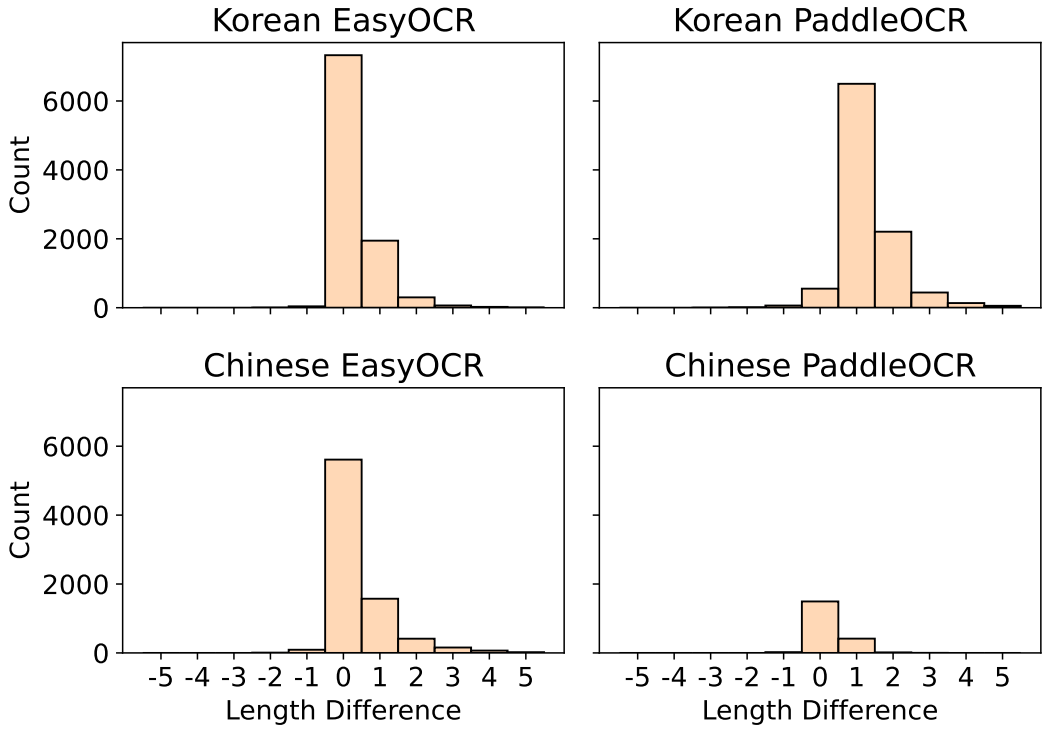


Figure 10: Length distribution of OCR results without whitespace

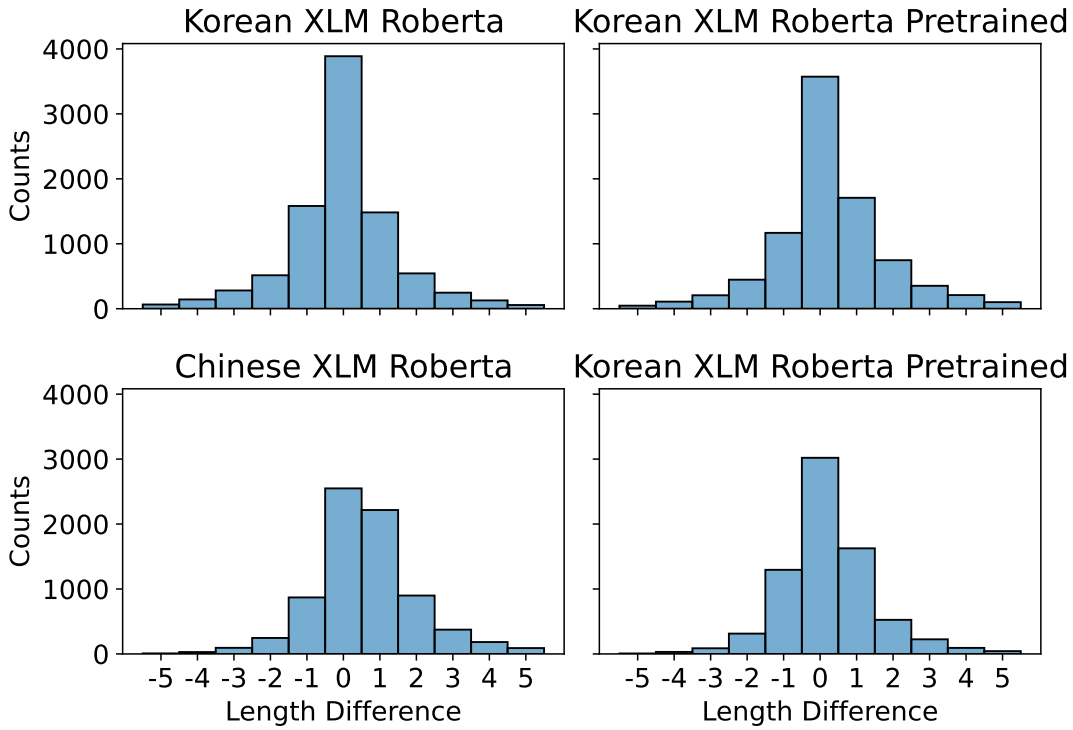


Figure 11: Length distribution of XLM Roberta model results

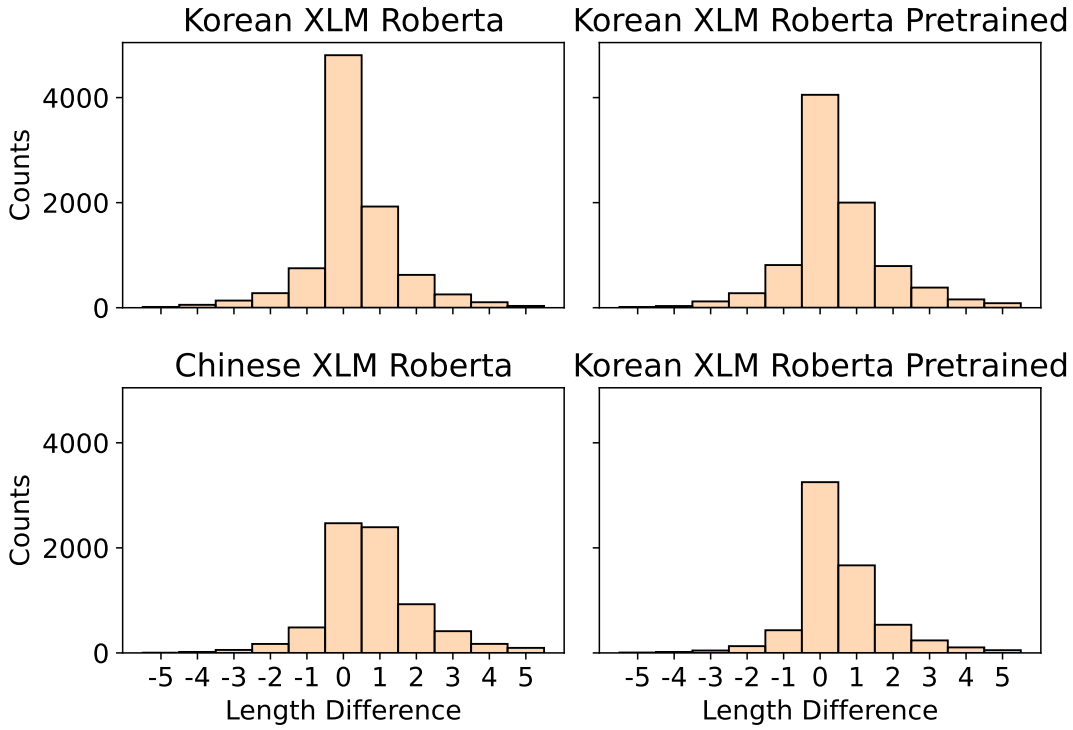


Figure 12: Length distribution of XLM Roberta model results without whitespace

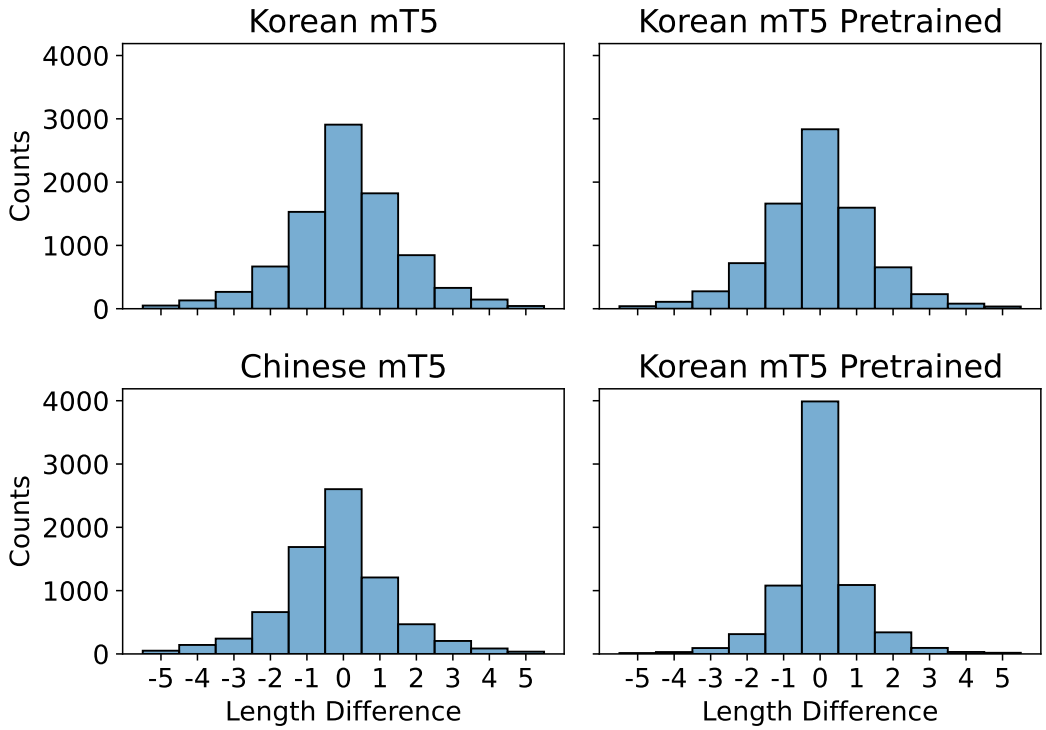


Figure 13: Length distribution of mT5 model results

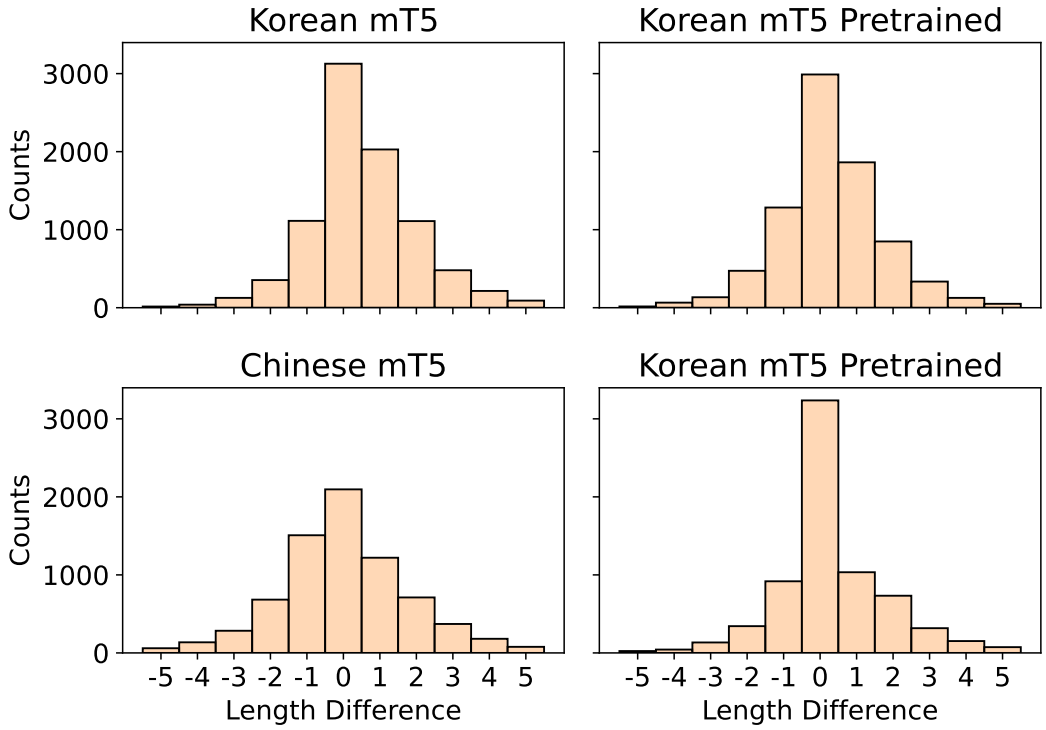


Figure 14: Length distribution of mT5 model results without whitespace