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

Anonymous ACL submission

Abstract

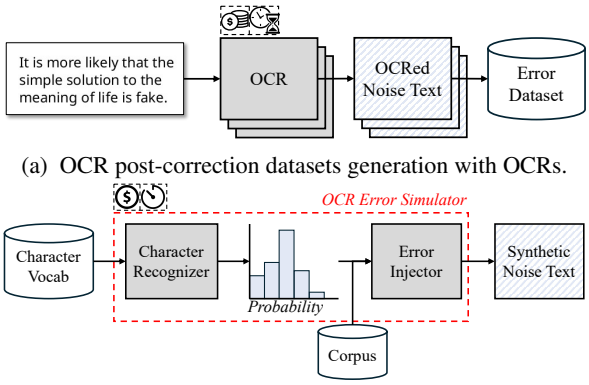
Rapid advancements in Large Language Models (LLMs) have significantly improved multimodal AI, highlighting the importance of accurately processing image-based text through Optical Character Recognition (OCR). However, the high computational cost of powerful LLMs limits their commercial use, leading to the widespread adoption of smaller OCR models that are more error-prone, especially with low-quality images. Post-correction is essential for improving OCR outputs but is hindered by the lack of high-quality training data. Existing synthetic approaches are often computationally expensive or fail to produce realistic errors. To address this, we propose POCO (Post-OCR Correction with Output distributions), a simple and low-cost framework that uses character-level probabilistic simulations to generate realistic OCR error datasets. POCO employs a lightweight vision model (ResNet34) to predict character probabilities and inject OCR-like errors into text corpora, enabling the creation of high-quality training data at scale. Experiments show that POCO significantly improves OCR post-correction performance, demonstrating its practicality and effectiveness. *The code will be made publicly available on GitHub.*

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 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).



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

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

Figure 1: OCR post-correction datasets generation

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-effective** framework for simulating OCR errors at the character level.
- POCO improves OCR post-correction, especially with pretrained models like mT5.

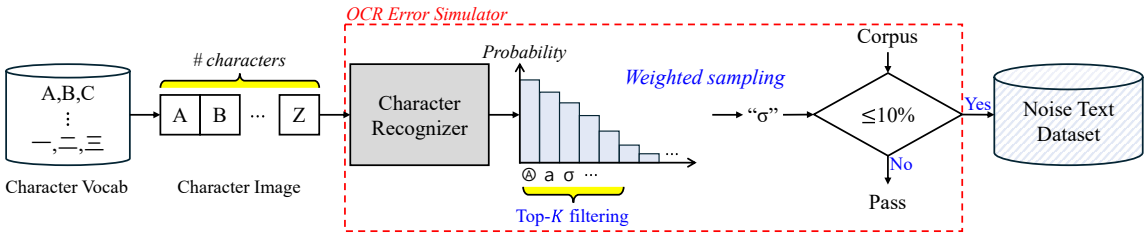


Figure 2: POCO framework pipeline for OCR post-correction dataset geneartion

- We validate its effectiveness in handling OCR errors such as character confusion and spacing.

2 Related Work

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

Random error injection (D'hondt et al., 2017) randomly inserts, deletes, or substitutes characters in clean text. It is simple and efficient but does not accurately reflect real OCR error patterns, limiting its generalization.

Image-generation-based approach (Boros et al., 2022) renders text as images, adds visual noise, and applies OCR to produce realistic errors. While suitable for image-based training, it depends heavily on the OCR system used and is computationally expensive.

Real error distribution-based method (Grundkiewicz et al., 2019) aligns OCR outputs with ground truth to estimate error probabilities, then injects errors based on these distributions. Though realistic, it requires extensive alignment and large corpora, making it less suitable for low-resource languages.

Glyph similarity-based method (Guan and Greene, 2024) simulates substitution errors based on visual similarity between characters, using Jaccard similarity and average distance metrics. It requires no extra corpora but faces $O(n^2)$ complexity 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[1] to create OCR error-injected 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%) or deleted (10%) in Korean texts.

Figure 3 compares our synthetically generated errors with results from actual OCR (EasyOCR[2] and PaddleOCR (Du et al., 2020)), confirming the simulator's effectiveness in reproducing realistic OCR error patterns, including spacing variations.

[1] https://aihub.or.kr/aihubdata/data/view.do?dataSetSn=71498
[2] https://github.com/JaidedAI/EasyOCR.git

	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 쿡가는 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	癗	癪	痌	癬	…

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

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[3] 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.

[3]https://github.com/jitsi/jiwer

OCR Error Simulator We utilized non-pretrained ResNet34 as backbones. The model predicted sequences of 1 to 8 characters, padded appropriately. The model were trained for 500 epochs with a batch size of 256 and a learning rate of $5e^{-4}$. The analysis of other models is provided in Appendix A.

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. 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, our method generally resulted in better CER compared to the random corruption baseline. The encoder-only pretrained model, XLM-RoBERTa, showed improvements in several cases, particularly when combined with the encoder-decoder model mT5. In the EasyOCR setting, our method consistently outperformed the random baseline, achieving up to 15% and 8% improvement over the original outputs in Korean and Chinese, respectively.

The Chinese PaddleOCR showed minimal baseline errors, leaving little room for improvement. In contrast, the Korean PaddleOCR achieved large performance gains. As shown in Appendix B, this is due to a high number of deletion errors, particularly involving spaces, commas, and periods.

Language	OCR Model	Original	XLM-RoBERTa				mT5				GPT-4o
			Non-Pretrained		Pretrained		Non-Pretrained		Pretrained		
			Random	Ours	Random	Ours	Random	Ours	Random	Ours	
Korean	EasyOCR	0.074	0.131	0.122	0.13	0.111	0.088	0.077	0.069	**0.063**(+15%)	0.223
Chinese		0.062	0.084	0.092	0.069	0.059(+5%)	0.095	0.095	0.072	**0.057**(+8%)	0.139
Korean	PaddleOCR	0.04	0.038	0.043	0.316	0.031(+23%)	**0.018**(+55%)	0.021(+48%)	**0.018**(+55%)	0.023(+43%)	0.220
Chinese		0.012	0.037	0.034	0.014	0.013	0.026	0.025	**0.011**(+8%)	0.012	0.134

Table 2: CER Performance of OCR Post-Correction

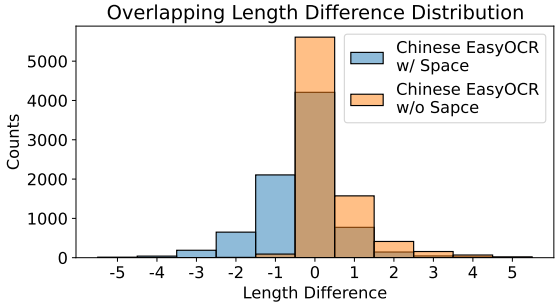


Figure 5: Length distribution of EasyOCR Chinese results based on the presence of whitespace

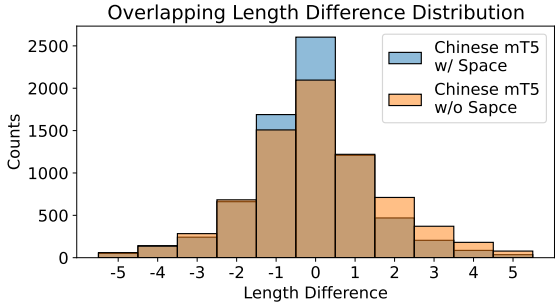


Figure 6: Length distribution of mT5 Chinese post-correction results based on the presence of whitespace

These characters are common in Korean writing and are often misrecognized by PaddleOCR, making post-correction especially effective. Although the random strategy occasionally outperformed our method in the low-error PaddleOCR setting, our approach still delivered competitive results, demonstrating robustness across varying OCR qualities.

When GPT-4o was used for correction, the CER increased, likely due to its decoder-only architecture, which preserves meaning but alters sentence structure. Since OCR correction requires precise character restoration, this highlights the need for careful prompting and possible fine-tuning, as shown in Appendix C.

These results validate the effectiveness of POCO and confirm the suitability of encoder-decoder architectures for OCR correction tasks. Encoder-only models exhibited limited correction capabilities, while decoder-only models often failed to train adequately. Across all settings, pretrained models consistently achieved 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

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.

**We use ChatGPT for translation.*

A OCR Error Simulator Models Analysis

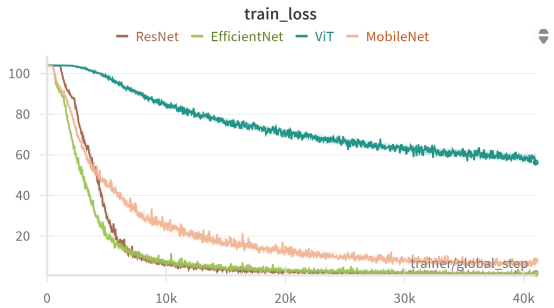


Figure 7: 500 epochs train loss of each simulators.

MobileNet	EfficientNet	ViT	ResNet34
25s	49s	73s	146s

Table 3: Average epoch time for each simulator model.

While ResNet34 was used as the simulator in our main experiments, we also trained EfficientNet, ViT, and MobileNet to explore alternative architectures. As shown in Figure 7, ViT and MobileNet showed higher training loss compared to ResNet34. Table 3 summarizes the training speed, with MobileNet being the fastest, followed by EfficientNet, ViT, and ResNet34. These results suggest that EfficientNet is a alternative to ResNet34, offering a good balance between accuracy and efficiency.

Lang-OCR	Replace	Deletion	Insertion	Complex
ko-easy	29,923	2,431	1,752	2,428
ko-paddle	882	**17,597**	1,660	1,714
ch-easy	10,043	1,223	1,880	3,159
ch-paddle	1,755	701	1,176	447

Table 4: Error type statistics for Korean and Chinese OCR outputs.

B Error Type Statistics

Table 4 presents the distribution of OCR error types by language and model. The Korean PaddleOCR output shows an unusually high number of deletions, mainly caused by missing spaces, commas, and periods commonly found in formal Korean writing. This explains the significant post-correction gains in this setting, as restoring these characters effectively lowers the error rate.

C Prompts

Figures 8 illustrate prompt for OCR Post-Correction.

Prompt for OCR Post-Correction

System:
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

User:
{Sentence 1}
…
{Sentence N}
—
Return a list of the same length with each sentence cleaned.

Figure 8: Prompt for OCR Post-Correction.

D Length Distribution Visualization

Figures 9 and 10 show sentence length distributions before and after OCR, with and without spaces, compared to the original.

Figures 11 and 12 show sentence length distributions before and after processing with XLM-RoBERTa, compared to the original. To assess the effect of whitespace, we provide versions with and without spaces.

Figures 13 and 14 show sentence length distributions before and after processing with mT5, compared to the original. To assess the effect of whitespace, we provide versions with and without spaces.

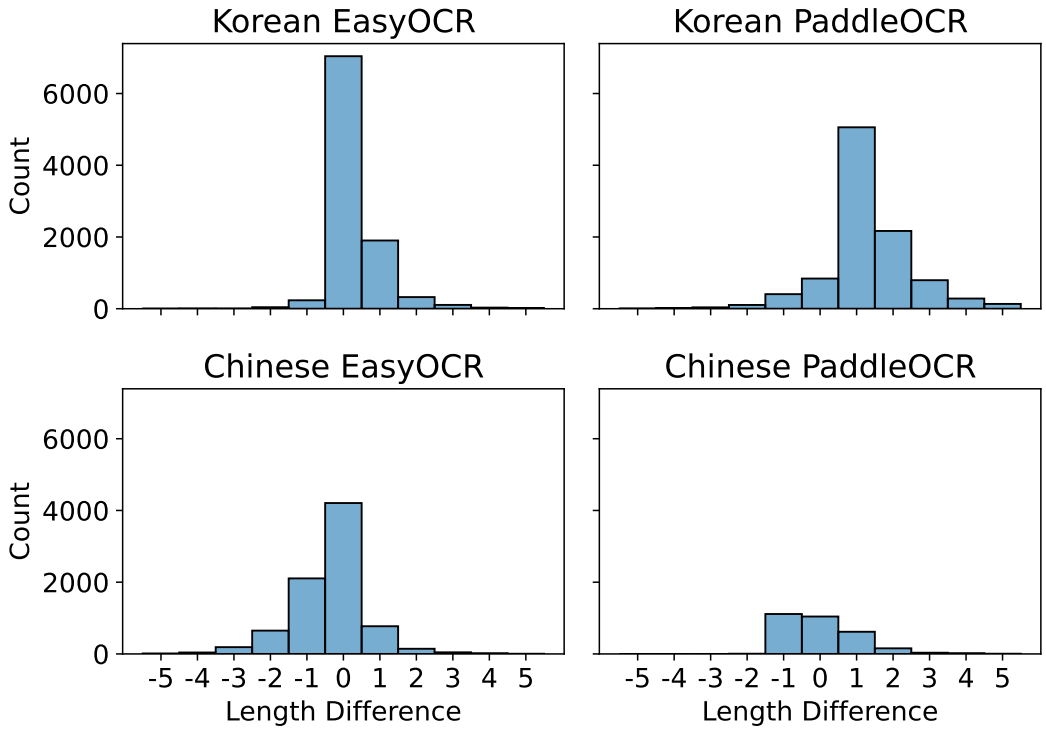


Figure 9: Length distribution of OCR results.

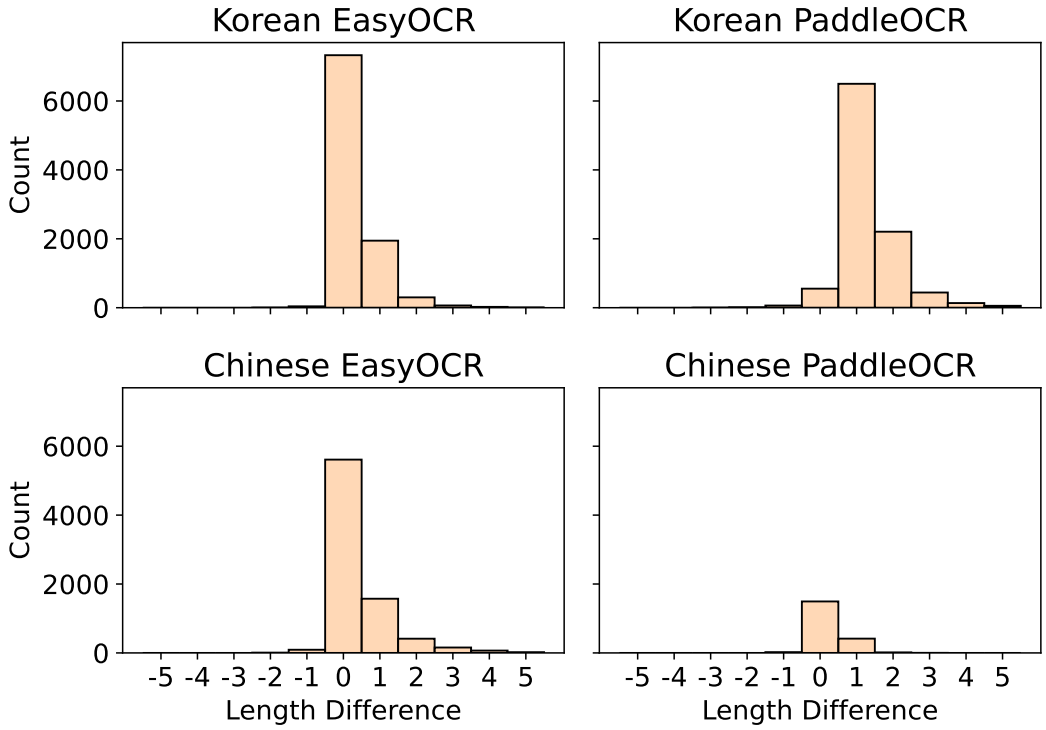


Figure 10: Length distribution of OCR results without whitespace.

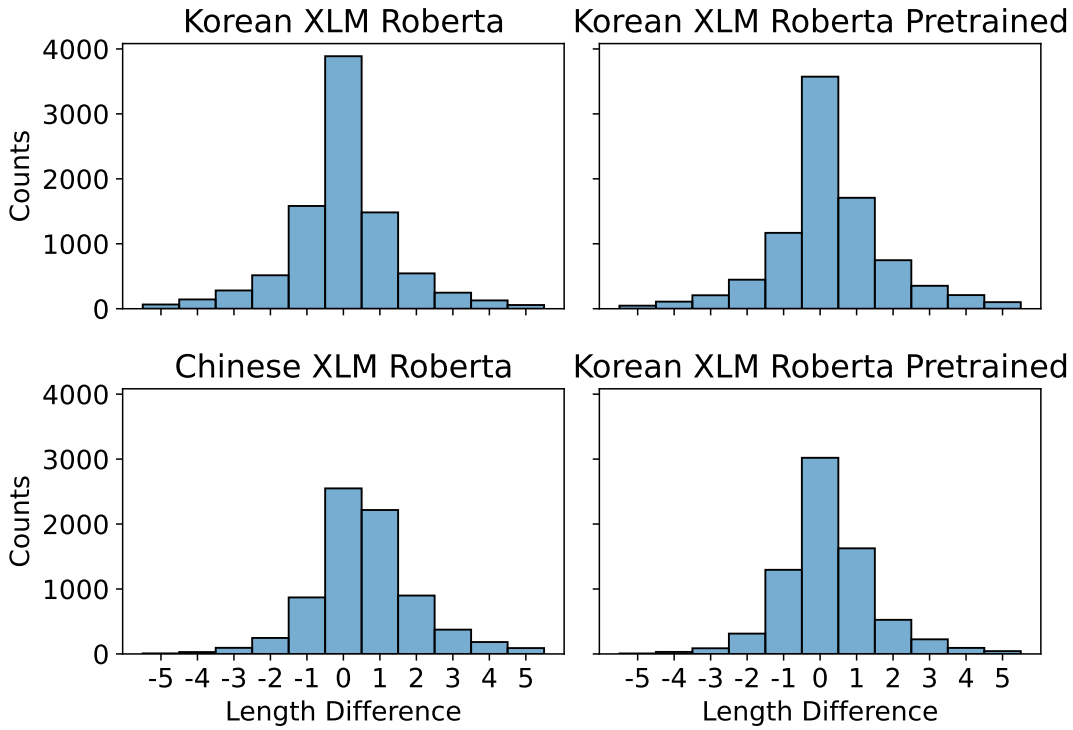


Figure 11: Length distribution of XLM Roberta model results.

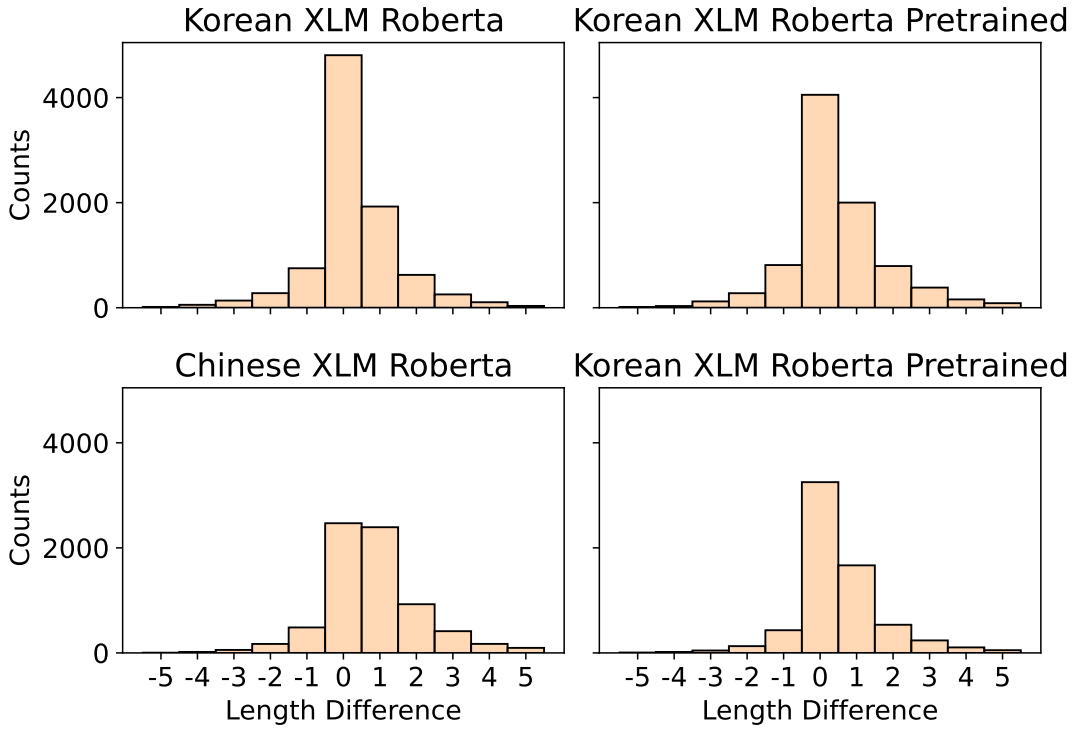


Figure 12: Length distribution of XLM Roberta model results without whitespace.

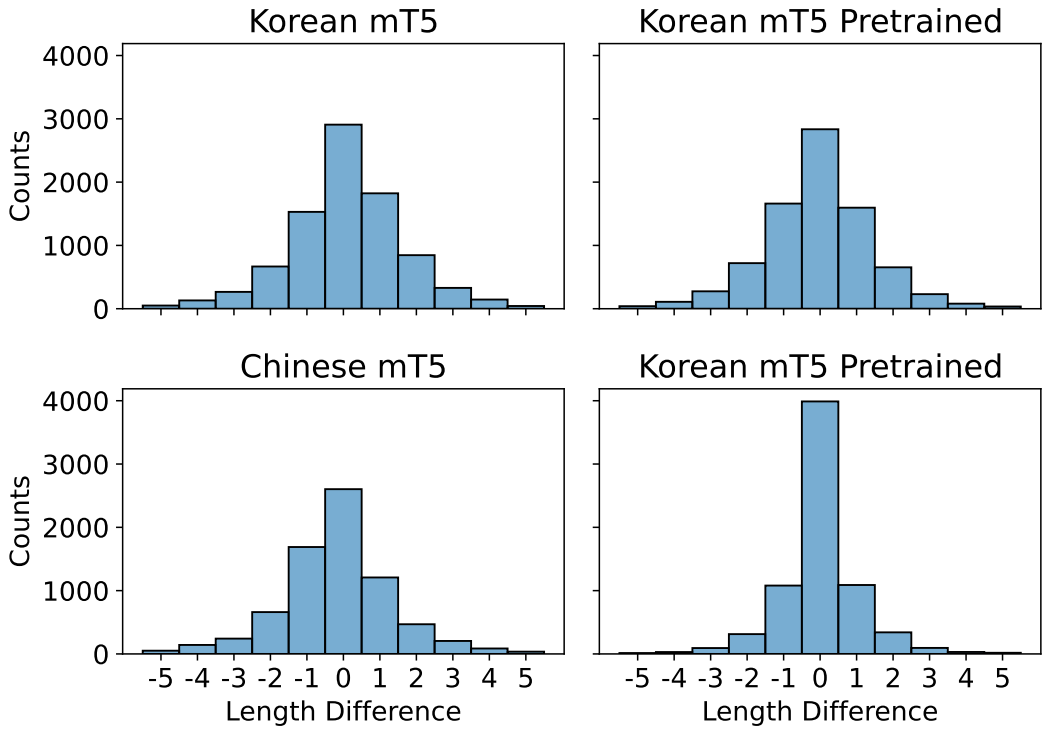


Figure 13: Length distribution of mT5 model results.

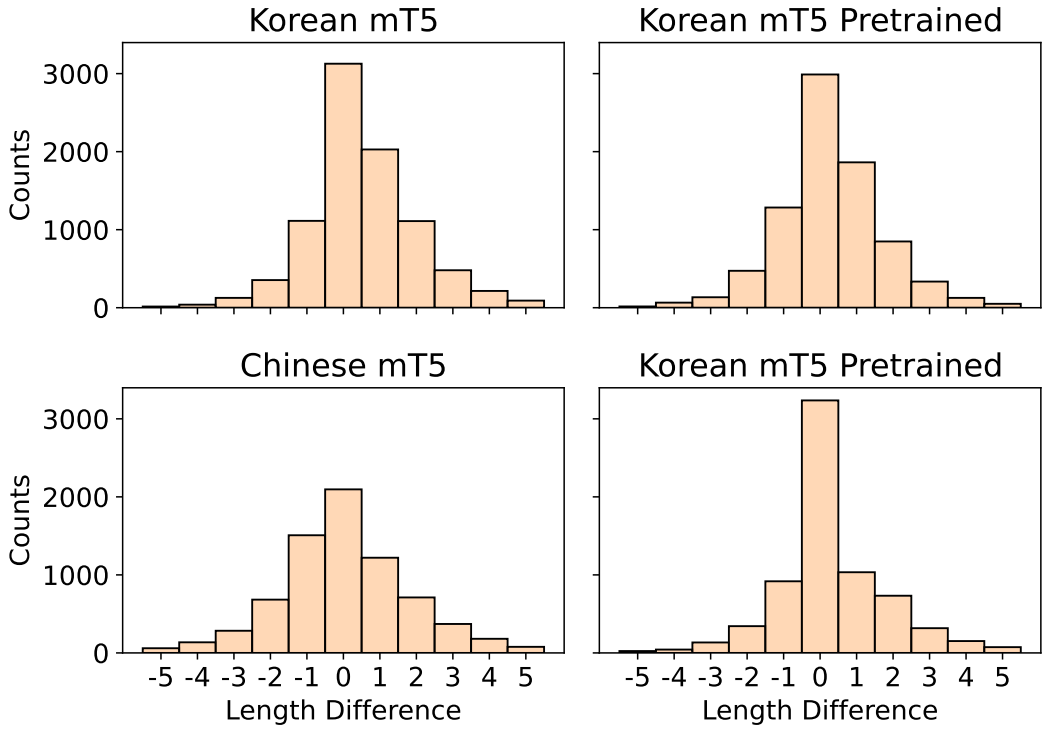


Figure 14: Length distribution of mT5 model results without whitespace.