



Towards Robust Text Recognition: Benchmarking AI Models and Designing a High Accuracy Alternative

Soumya Karunakar Shetty¹, Dr. Abuzar Ansari²

¹(Data Science, SIES College of Arts, Science and Commerce, Sion (West)

Email: shetty26.soumya@gmail.com)

²(Head of Data Science Department, SIES College of Arts, Science and Commerce, Sion (West)

Email: abuzara@sies.edu.in)

How to Cite this Article:

Shetty, S. K. (2026). Towards Robust Text Recognition: Benchmarking AI Models and Designing a High Accuracy Alternative. International Journal of Creative and Open Research in Engineering and Management, <i>02</i>(10), 1-9.
<https://doi.org/10.55041/ijcope.v2i10.120>

License:

This article is published under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

© The Author(s). Published by International Journal of Creative and Open Research in Engineering and Management.



<https://doi.org/10.55041/ijcope.v2i10.120>

Abstract—Chatbots that accept images are now a common way to pull text out of photographs, but little is known about how well they do it when the text is small, crowded or handwritten. We tested 14 chatbots on 50 images prepared for this purpose, ten each of small print, dense pages, handwriting, script fonts and numbers. Every answer was scored against the true text with character error rate (CER) and word error rate (WER). Mean CER ran from 2.73% for Grok to 17.24% for HuggingChat, and mean WER from 6.06% for Claude to 39.80% for HuggingChat. Dense pages were the hardest category (mean CER 23.4%), and only 5% of the handwritten transcriptions matched the reference exactly. The errors fell into a few recurring types: output that stopped early, fluent but wrong words, look-alike characters swapped for one another, text copied from another image in the same batch, and added commentary. To deal with these, we designed a pipeline that finds text with DBNet, crops each line and reads it with a CNN-BiLSTM-CTC recogniser trained from scratch, then orders the lines and applies a constrained correction. Because every detected line is decoded on its own, we expect fewer omissions and fewer invented words than in the chatbots, and we set a target CER below the 2.73% of the best chatbot. Measured results for the pipeline are left for a follow-up study.

Keywords—*optical character recognition, handwriting recognition, AI chatbots, character error rate, word error rate, DBNet, CRNN*

I. INTRODUCTION

Getting text out of an image used to mean running a dedicated OCR program. Today many people simply upload the picture to a chatbot and ask for the text. The reply comes back as clean, confident prose, and that is where the trouble starts. In everyday use, ChatGPT, Gemini and similar tools misread very small print and some handwriting, and nothing in the reply tells the reader that a word is wrong. A mistaken digit in an invoice number looks exactly like a correct one. We could not find a careful comparison of how the common chatbots behave in these situations, so users have little basis for choosing one, and developers have little guidance on what to fix. General-purpose chatbots were also never built as text-extraction systems, which leaves room for a purpose-built alternative. This paper therefore has two parts. In the first, we measure how accurately 14 chatbots transcribe small, dense and handwritten text, using images whose true text is known. In the second, we describe a recognition pipeline designed around the weaknesses we found. Our objectives were (1) to build a benchmark with verified ground truth and score the chatbots on it with CER, WER, exact-match rate and rates of missing and extra words; (2) to sort the errors into types and see how they change with text size, density and handwriting; and (3) to design a pipeline that addresses those errors and to state what we expect it to achieve.



II. LITERATURE REVIEW

Several recent benchmarks ask how well large multimodal models read text. OCRBench reported that such models struggle with handwritten and non-semantic text [1], and OCRBench v2 found that most models scored below 50 out of 100, with weak spots in less common text, fine-grained perception and layout [2]. The pattern repeats in narrower settings: low-resource Pashto text [3], video frames, where models often beat EasyOCR and RapidOCR but hallucinated and failed on occluded or stylised text [4], and Nigerian licence plates, where models differed clearly in character error rate [5]. For Devanagari, ten systems performed alike on clean synthetic text, yet nine degraded sharply on 300 real scans [6].

A recurring concern is that these models do not always transcribe what is on the page. Lee et al. found that vision-language models often rewrite imperfect text into a more plausible form [7], He et al. built a benchmark for OCR hallucination under degraded inputs such as ID cards and invoices [8], and a study of Ancient Greek editions described fluent but wrong substitutions, in contrast to the local errors of traditional OCR [9]. These observations are close to what we saw in the chatbots we tested.

Handwriting has been examined mostly on historical material. Crosilla et al. found that proprietary multimodal models, Claude 3.5 Sonnet in particular, beat open-source ones in zero-shot tests and performed about as well as Transkribus, but rarely corrected their own mistakes [10]. Related studies tested Gemini on historical handwriting [11], reported that GPT-4o and Claude Sonnet 3.5 outperformed EasyOCR, Keras, Pytesseract and TrOCR on handwritten records [12], and proposed prompting strategies for multi-page documents [13].

Text density and image quality matter too. FiCo finds that OCR error rises sharply as text becomes denser or resolution drops, even for OCR-specialised models [14]. Smaller dedicated models are one response: LightOnOCR is a 1B-parameter end-to-end model [15], DeepSeek-OCR recovers about 97% of text below 10x compression but only about 60% at 20x [16], and PaddleOCR-VL uses a 0.9B model for document parsing [17].

Language models are also used to correct OCR output. Bourne reported that the best of seven models cut character error rate by more than 60% on one newspaper dataset [18], but the results are not uniform. Correction helped for English but not for Finnish [19], and a comparison of thirteen digitisation pipelines found no consistent benefit from post-correction, which often made accurate extractions worse [20].

Most of this work evaluates models on documents or domains chosen by the authors, and few studies put small print, dense pages, handwriting and numbers side by side for the consumer chatbots that people actually use. Our benchmark does that, and we use its error patterns to motivate a pipeline in which correction is constrained, given the mixed record of post-correction in [18]–[20].

III. METHODOLOGY

A. Benchmark and Data Collection

The benchmark has 50 images, ten in each of five categories. Small-text images are 6 to 10 pt. Dense-text images are full pages at 9 to 13 pt, from about 8,000 to nearly 33,000 characters. The handwriting set has ten samples, and the script-font set has ten decorative typefaces at 44 to 72 pt. The numbers set (7 to 46 pt) holds currency values, timestamps, nine strings of 70 digits, alphanumeric codes such as EZW-8432-JF31, and a 7 pt grid of six-digit groups. The typed images were made by entering the text in Microsoft Word and saving the page as an image, so the content, typeface and size of each were known exactly. The handwriting samples were collected from friends of the author and therefore cover several writing styles. The ground-truth transcription of every image was prepared by the author, and crossed-out words in the handwriting samples were left out of the reference.

B. Chatbots and Data Collection

We tested 14 chatbots on 22 August 2026, using their free or default tiers: ChatGPT, Gemini, Claude, Microsoft Copilot, Grok, DeepSeek, Meta AI, Le Chat, Qwen, Kimi, Poe, HuggingChat, GitHub Copilot and NotebookLM. Images were uploaded in batches of ten, and each image was submitted once to each chatbot. The same prompt was used throughout: “Transcribe the text exactly as visible. Do not correct spelling, grammar, capitalization, punctuation, or formatting. Do not summarize or interpret the text. If a character or word cannot be read confidently, write [unclear]. Do not omit small text. Process every uploaded image and keep the results in filename order.” All images were saved in one folder. We ran a Python script, gave it the location of that folder, and it generated the extracted text for every image in it automatically.



C. Evaluation Metrics

Character error rate is the Levenshtein distance between the output and the reference, measured in characters and divided by the number of characters in the reference. Word error rate is the same calculation on words. With S, D and I for substitutions, deletions and insertions, and N for the reference length, both are $(S + D + I) / N$, and either can exceed 100% when a chatbot adds text. We also report missing words (D/N), extra words (I/N), substituted words (S/N) and the number of exact matches. Before scoring we lower-cased the text, collapsed whitespace, treated quotation marks and dashes of different styles as equal and removed Markdown. Punctuation, numbering style, spelling and expanded abbreviations were left as they were, so they count as errors. Scores are averaged per image. Two runs returned nothing (NotebookLM and HuggingChat, one dense image each) and are left out of the means, and two Meta AI replies that read only “No output” are scored as failures.

Our first scoring workbook contained two mistakes that we corrected. A header row had been read as an image with a three-character reference, which added a 200% error to every chatbot, and a rule that removed trailing commentary had cut 15 long outputs short, most of them the answers of thirteen chatbots on one dense page. All results below were recomputed from the original outputs.

D. Proposed Recognition Pipeline

The pipeline splits the job into stages that can be checked one at a time (Fig. 1). The input image is first resized, denoised, contrast-adjusted and deskewed. DBNet then finds the text regions and returns bounding boxes, and each box is cropped and divided into lines or regions. A CNN-BiLSTM-CTC recogniser, trained from scratch, reads each line and outputs a character sequence. Low-confidence results are filtered out, the remaining lines are put back in reading order using their coordinates, and a constrained correction step produces the final text.

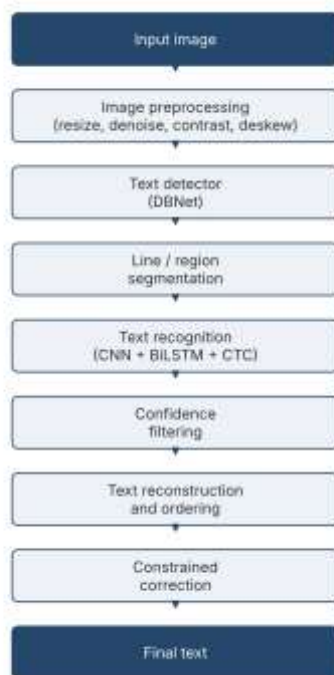


Fig. 1. Architecture of the proposed text-recognition pipeline.

IV. RESULTS

A. Overall Comparison

Table I gives the results over all 50 images. Grok had the lowest mean CER (2.73%) and Claude the lowest mean WER (6.06%). The six best systems by CER, namely Grok, Claude, Microsoft Copilot, ChatGPT, Kimi and Gemini, lie between 2.73% and 4.66%, a spread smaller than the variation between images. Le Chat, GitHub Copilot, Poe and HuggingChat form a weaker group, with CER from 13.60% to 17.24%. Of the 698 scored runs, 353 (50.6%) matched the reference exactly and 46 (6.6%) had a CER of 50% or more.



TABLE I
MEAN CER, WER AND WORD-LEVEL ERROR RATES PER CHATBOT (%)

Chatbot	Images scored	Mean CER (%)	Mean WER (%)	Missing words (%)	Extra words (%)	Substituted words (%)	Exact matches
Grok	50	2.73	7.30	1.10	2.06	4.14	31
Claude	50	3.00	6.06	2.36	1.01	2.69	30
Microsoft Copilot	50	3.93	8.26	2.89	0.94	4.43	27
ChatGPT	50	4.17	9.92	0.71	1.47	7.73	29
Kimi	50	4.23	9.61	0.73	2.87	6.01	28
Gemini	50	4.66	7.98	2.84	1.46	3.68	28
DeepSeek	50	7.15	12.77	1.49	3.77	7.52	26
NotebookLM	49	7.70	16.65	2.13	7.74	6.78	17
Qwen	50	8.51	14.85	4.12	3.19	7.54	28
Meta AI	50	8.78	12.25	4.81	2.38	5.06	26
Le Chat	50	13.60	17.39	11.78	1.14	4.47	28
GitHub Copilot	50	16.12	26.93	5.14	3.45	18.34	19
Poe	50	16.33	32.08	7.81	6.97	17.30	17
HuggingChat	49	17.24	39.80	12.13	9.76	17.91	19
All chatbots	698	8.43	15.81	4.28	3.43	8.10	353

B. Results by Category

Accuracy depended heavily on the category (Table II). Script fonts caused almost no trouble: 98.6% of runs were exact, and the one real failure was a HuggingChat answer with an annotation added after a correct transcription. Most chatbots also read the small print without error. The category means for Gemini, Qwen, NotebookLM, GitHub Copilot and Poe were raised by one or two images, described below. Dense text was the hardest category, with a mean CER of 23.4%. Grok (6.08%), Kimi (8.25%) and NotebookLM (9.20%) did best on it, and Le Chat (56.00%), HuggingChat (50.43%) and GitHub Copilot (48.58%) did worst. For handwriting the mean WER was 26.2%, only 5% of runs were exact, and Claude (4.45% CER), Grok (4.85%) and Gemini (4.94%) were the most accurate. On numbers, most chatbots were nearly perfect, but Meta AI (24.79%) and HuggingChat (21.79%) failed badly. No single chatbot was best in every category.

TABLE II
MEAN CER (% , LEFT) AND MEAN WER (% , RIGHT) BY CHATBOT AND IMAGE CATEGORY

Chatbot	Small text	Dense text	Real handwriting	Script font	Numbers
Grok	0.50	6.08	4.85	0.00	2.22
Claude	0.00	10.17	4.45	0.00	0.36
Microsoft Copilot	0.00	11.77	6.12	0.00	1.74
ChatGPT	0.00	14.35	5.91	0.00	0.58
Kimi	0.00	8.25	10.06	0.00	2.86
Gemini	6.85	11.45	4.94	0.00	0.04
DeepSeek	0.10	24.22	6.53	0.00	4.88
NotebookLM	12.90	9.20	16.15	0.00	0.40
Qwen	7.12	26.76	7.26	0.00	1.42
Meta AI	0.61	11.59	6.92	0.00	24.79
Le Chat	3.45	56.00	8.50	0.00	0.04
GitHub Copilot	15.07	48.58	9.89	0.00	7.05
Poe	15.43	38.40	20.01	0.06	7.74
HuggingChat	0.17	50.43	8.21	8.91	21.79
Category mean	4.44	23.37	8.56	0.64	5.42

Chatbot	Small text	Dense text	Real handwriting	Script font	Numbers
Grok	0.34	7.21	21.21	0.00	7.73
Claude	0.00	10.36	18.07	0.00	1.88
Microsoft Copilot	0.00	12.35	22.84	0.00	5.99
ChatGPT	0.00	26.41	20.32	0.00	2.85
Kimi	0.00	10.38	27.36	0.00	10.33
Gemini	8.65	12.92	18.09	0.00	0.26
DeepSeek	0.12	28.09	22.96	0.00	12.69
NotebookLM	26.88	11.71	41.75	0.00	2.42
Qwen	9.55	31.27	27.49	0.00	5.96
Meta AI	1.18	14.22	16.81	0.00	29.04
Le Chat	4.75	58.24	23.63	0.00	0.31
GitHub Copilot	18.81	59.86	27.88	0.00	28.09
Poe	21.00	45.10	46.78	0.34	47.16
HuggingChat	0.22	54.08	31.73	9.23	105.16
Category mean	6.54	27.30	26.22	0.68	18.56

C. Error Analysis

Long outputs were often cut short. On the largest dense page (32,765 characters after normalisation), Claude returned 2,089 characters and Microsoft Copilot 1,694. Le Chat returned only 1,150 to 1,250 characters for three pages of 11,000 to 19,000 characters, which gives it an average of 54% missing words on dense text. The text that was returned was usually accurate, so these are failures of completeness rather than of reading. At 9 pt some answers fell apart: ChatGPT produced “Theses depamet pate abot tezt at acoso 230 dota” for “The design department repaired the laboratory freezer at a cost of 1,250 dollars”. Other errors were fluent substitutions, such as “repaired” read as “repainted” (Le Chat), “first” as “final” (Meta AI), “406” as “400” (Meta AI, on two images) and “9:45” as “4.5” (Poe).



Character confusions showed up in the numbers set. In the alphanumeric codes, ChatGPT and Poe both read “EZW-8432-JF31” as “EZW-8432-IF31” and “WYP-5823-NJ20” as “WYP-5823-NI20”, and GitHub Copilot read “LGJ-4566-EK47” as “LGI-4566-EK47”, so the letter J became I. Kimi read an E as F, and Microsoft Copilot swapped V and Y. Among digits, Claude read “141953” as “143953”, ChatGPT read “923110” as “923130”, and Poe and GitHub Copilot changed values such as 7,542 to 7,392 in a handwriting-style typeface. Claude, DeepSeek and Kimi each dropped one digit from a 70-digit string. We did not find a lower-case “i” written in place of the digit “1” in the non-dense images. The nearest cases are the J-to-I confusion and the digit substitutions just described. Most of the digit errors came from one image, the 7 pt grid of six-digit groups, which accounts for 211 of the 246 word-level substitutions in the category. The handwriting errors were smaller but widespread. Several outputs changed a list number such as “8)” to “8.”, thirteen chatbots wrote “require” for “requires” on one line, an identifier written as “studentrj” came back in twelve different forms, and the margin heading “Imp for ML” was read as “EM”, “Time for ML” or “EMT”. Because images went in ten at a time, some chatbots also transcribed the wrong image. Gemini returned, word for word, the reference text of a different image (CER 68.5%), GitHub Copilot did the same on one image and nearly so on another, and Qwen and Poe returned text from other images. NotebookLM gave 706 characters for a 324-character image by adding sentences from elsewhere. The prompt told the chatbots to write [unclear] for illegible characters, yet HuggingChat appended “[unclear] — a faint small horizontal mark...” after a correct transcription of a script-font image, and Meta AI gave no output at all on two pages of digits.

V. DISCUSSION

A. Why the Chatbots Fall Short

The vendors do not publish full architectures, so what follows is a general account that we could not test. A chatbot with vision usually encodes the uploaded image as patch tokens, often after resizing or tiling it to fit a fixed budget, and a language model then writes the transcription one token at a time. Several of the errors we saw follow from this. A fixed budget means that small characters are represented with few pixels, and tiling can split a line in the wrong place. The decoder prefers likely text, so an unreadable word may be replaced by a plausible one when it should be left blank, which matches the fluent substitutions. Similar rewriting of imperfect text has been reported for other vision-language models [7], [9]. Output limits explain the truncated pages. Since transcription is handled as a conversation, answers can pick up commentary or text from neighbouring images. No step locates each line of text, so the chatbot cannot say what it failed to read.

The ranking of the chatbots needs cautious reading. A gap of one or two points among the top six is smaller than the effect of a single failed image, each chatbot was run once, and we did not run significance tests. The category results say more. The chatbots did well on large, clean, short text, and they got worse when the text was near the limit of resolution, when the answer was long, or when the writing was ambiguous.

B. Expected Performance of the Proposed Pipeline

We have not yet measured the pipeline on this benchmark, so what follows is what its design leads us to expect, not a result. Cropping each detected region and reading it separately should hand small characters to the recogniser at a usable scale and leave a record of every region found, so lines should not be skipped silently. CTC decoding is not autoregressive. Each line is decoded in one forward pass, so the model should not run past the end of a line or finish it with plausible words that are not in the image, and the per-character confidence lets uncertain output be flagged. Constrained correction limits the language prior to cases where the image supports it. We therefore expect fewer omissions and fewer invented or substituted words than the chatbots showed, particularly on small and dense text, and we set the target for overall CER below the 2.73% of the best chatbot. Handwriting is difficult for any recogniser and depends on how well the training data covers different writers, so we expect gains there to be smaller and less certain. We also expect the pipeline to be faster. A line-level CNN-BiLSTM runs locally and needs one forward pass per crop, whereas a chatbot has to upload the image, encode it and generate every output token in turn. Neither accuracy nor speed has been measured in this study, and chatbot response times were not recorded in Phase 1, so these expectations should be tested directly.

C. Limitations

Each image was submitted once to each chatbot, so run-to-run variation is unknown, and the chatbots change as they are updated. We used free tiers, and paid tiers may behave differently. Ten images per category give limited statistical power. The typed images were made in Word and may not reflect photographs with blur, perspective or uneven light, and the



handwriting set is small. The batch upload of ten images produced cross-image errors that may not occur when images are sent one at a time. The Phase 1 benchmark is in English only, so it does not test the multilingual capability of the pipeline.

VI. CONCLUSION

Across 50 images and 14 chatbots, mean CER ranged from 2.73% to 17.24% and mean WER from 6.06% to 39.80%. In some cases the models worked well, especially on script fonts and short passages of small print. In others they worked badly. Dense pages were truncated or garbled, handwriting rarely matched the reference exactly, and single characters were swapped, as when J was read as I or one digit was read as another. Because the wrong answers were usually fluent, they were easy to miss. These findings led us to a pipeline of DBNet detection, line-level CTC recognition and constrained correction. From its design we expect fewer omissions and invented words than the chatbots produced, with a target CER below 2.73%, and faster processing, although both expectations remain to be measured. Future work should evaluate the pipeline on this benchmark, add photographed documents, collect handwriting from more writers, repeat the chatbot runs and test the multilingual character set.

REFERENCES

- [1] Y. Liu *et al.*, “OCRBench: On the hidden mystery of OCR in large multimodal models,” arXiv:2305.07895, 2023. [Online]. Available: <https://arxiv.org/abs/2305.07895>
- [2] L. Fu *et al.*, “OCRBench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning,” arXiv:2501.00321, 2024. [Online]. Available: <https://arxiv.org/abs/2501.00321>
- [3] I. Haq *et al.*, “PsOCR: Benchmarking large multimodal models for optical character recognition in low-resource Pashto language,” arXiv:2505.10055, 2025. [Online]. Available: <https://arxiv.org/abs/2505.10055>
- [4] S. Nagaonkar, A. Sharma, A. Choithani, and A. Trivedi, “Benchmarking vision-language models on optical character recognition in dynamic video environments,” arXiv:2502.06445, 2025. [Online]. Available: <https://arxiv.org/abs/2502.06445>
- [5] I. I. Tijjani, A. A. Mustapaha, S. I. Muhammad, and M. B. Aliyu, “Evaluating vision-language models as a zero-shot learning alternative to You Only Look Once and optical character recognition for Nigerian license plate recognition,” arXiv:2607.02025, 2026. [Online]. Available: <https://arxiv.org/abs/2607.02025>
- [6] A. P. Singh, “Can OCR-VLMs read Devanagari? A stress-test benchmark and post-correction study,” arXiv:2606.29213, 2026. [Online]. Available: <https://arxiv.org/abs/2606.29213>
- [7] G. G. Lee, K. E. Ak, J. Mohta, Y. Xu, and D. Dimitriadis, “Do VLMs read or rewrite? On transcription faithfulness in vision-language models,” arXiv:2607.21617, 2026. [Online]. Available: <https://arxiv.org/abs/2607.21617>
- [8] Z. He *et al.*, “Seeing is believing? Mitigating OCR hallucinations in multimodal large language models,” NeurIPS 2025; arXiv:2506.20168, 2025. [Online]. Available: <https://arxiv.org/abs/2506.20168>
- [9] A. Karamolegkou, N. Anglraud, B. Sagot, and T. Clérice, “Reading or guessing? Visual grounding failures of vision-language models for OCR in Ancient Greek editions,” arXiv:2605.27750, 2026. [Online]. Available: <https://arxiv.org/abs/2605.27750>
- [10] G. Crosilla, L. Klic, and G. Colavizza, “Benchmarking large language models for handwritten text recognition,” arXiv:2503.15195, 2025. [Online]. Available: <https://arxiv.org/abs/2503.15195>
- [11] L. Li, “Handwriting recognition in historical documents with multimodal LLM,” arXiv:2410.24034, 2024. [Online]. Available: <https://arxiv.org/abs/2410.24034>
- [12] S. Kim, J. Baudru, W. Ryckbosch, H. Bersini, and V. Ginis, “Early evidence of how LLMs outperform traditional systems on OCR/HTR tasks for historical records,” arXiv:2501.11623, 2025. [Online]. Available: <https://arxiv.org/abs/2501.11623>
- [13] B. Gutteridge, M. T. Jackson, T. Kukurin, and X. Dong, “Judge a book by its cover: Investigating multimodal LLMs for multi-page handwritten document transcription,” arXiv:2502.20295, 2025. [Online]. Available: <https://arxiv.org/abs/2502.20295>



- [14] J. Tu, N. Crispino, K. Montgomery, C. Wang, and D. Song, “FiCo: Evaluating vision-language models under visual fidelity and compression at scale,” in *Findings of the Association for Computational Linguistics: ACL 2026*, San Diego, CA, USA, 2026, pp. 35261–35277. [Online]. Available: <https://aclanthology.org/2026.findings-acl.1758.pdf>
- [15] S. Taghadouini, A. Cavaillès, and B. Aubertin, “LightOnOCR: A 1B end-to-end multilingual vision-language model for state-of-the-art OCR,” arXiv:2601.14251, 2026. [Online]. Available: <https://arxiv.org/abs/2601.14251>
- [16] H. Wei, Y. Sun, and Y. Li, “DeepSeek-OCR: Contexts optical compression,” arXiv:2510.18234, 2025. [Online]. Available: <https://arxiv.org/abs/2510.18234>
- [17] C. Cui *et al.*, “PaddleOCR-VL: Boosting multilingual document parsing via a 0.9B ultra-compact vision-language model,” arXiv:2510.14528, 2025. [Online]. Available: <https://arxiv.org/abs/2510.14528>
- [18] J. Bourne, “CLOCR-C: Context leveraging OCR correction with pre-trained language models,” arXiv:2408.17428, 2024. [Online]. Available: <https://arxiv.org/abs/2408.17428>
- [19] J. Kanerva, C. Ledins, S. Kåpyaho, and F. Ginter, “OCR error post-correction with LLMs in historical documents: No free lunches,” in *Proc. Third Workshop on Resources and Representations for Under-Resourced Languages and Domains (RESOURCEFUL-2025)*, 2025. [Online]. Available: <https://arxiv.org/abs/2502.01205>
- [20] M. Gómez Rey, P. Callejo, M. Muñoz-Organero, and C. Alario-Hoyos, “A comparative evaluation of digitization pipelines for historiographical sources,” arXiv:2608.24976, 2026. [Online]. Available: <https://arxiv.org/abs/2608.24976>