

The OCR Boundary for Tiny Local Document Intelligence

Chen, Ho Yiing (norika)
Independent Researcher, Taiwan
ORCID: 0009-0006-6816-9891
Zenodo DOI: 10.5281/zenodo.20728820
norika at charenix.com

June 17, 2026

Abstract

Small document-intelligence systems often fail for a boring reason: the language runtime is judged on content the OCR layer never saw. I report an Iseu Receipt JP experiment on a 500-image realistic-hard receipt set. An Apple Vision OCR path produced 0.0 exact match and about 0.51–0.57 field accuracy on measured hard subsets after the OCR text was passed into the receipt JSON runtime. A single multilingual Tesseract path reached 0.724 exact match and 0.9507 field accuracy on 500 images. A merged local frontend using Japanese/English OCR with two page-segmentation modes, followed by the same receipt runtime and repair policies, reached 0.944 exact match and 0.9918 field accuracy on the same 500-image set. The pack folder is about 15 MB, excluding the host Tesseract binary. This is not a final vision model. It is a product bridge and a sharper boundary: image quality, OCR mode, and structured-field gates must be measured before a tiny LLM is blamed or shipped.

1 The Problem

When a receipt system returns the wrong total, the visible failure is JSON. The actual failure may have happened earlier. The image may be blurred. The receipt may be outside the frame. The OCR layer may have dropped the registration number. The extractor then receives a broken transcript and is asked to behave like a magician.

That is not a fair test. It is also not a product.

For a small local system, I define the pipeline as:

$$I \xrightarrow{Q} I' \xrightarrow{O_m} T_m \xrightarrow{M} T \xrightarrow{R} \hat{Y} \xrightarrow{G} \text{ship or retake}.$$

Here I is the captured image, Q is a capture-quality gate, O_m is one OCR mode, T_m is the transcript from that mode, M is transcript merge, R is the receipt JSON runtime, and G is the final field gate. If $Q = 0$, the app should ask for another photo. If O_m fails to see the field, R should not be credited with an impossible mistake.

2 Capture Gate

The capture rule is not decoration. It is part of the model boundary.

Let:

$$q(I) = \mathbf{1}[a(I) \geq a_0] \mathbf{1}[b(I) \leq b_0] \mathbf{1}[s(I) \leq s_0] \mathbf{1}[c(I) \geq c_0] \mathbf{1}[r(I) = 1],$$

where $a(I)$ is the document area share, $b(I)$ is blur, $s(I)$ is skew, $c(I)$ is contrast, and $r(I)$ means the target stays inside the guide frame. A production capture should continue only when:

$$q(I) = 1.$$

This is the rule the user feels as “put the receipt inside the box and hold still.” It is not user-hostile. It is the price of making a 15–50 MB local pipeline compete with a heavier cloud path.

3 OCR Mode Fusion

The frontend used here runs local OCR in multiple page-segmentation modes. Let $m \in \{4, 6\}$ represent the two modes used in the current receipt bridge. Each mode yields a transcript:

$$T_m = O_m(I').$$

The merge step forms:

$$T = M(T_4, T_6).$$

The merge is not meant to be beautiful prose. It is meant to preserve enough field evidence for the structured runtime. A repeated invoice number is less harmful than a missing invoice number. A noisy tax line can still carry the digits that a field policy needs.

4 Measured Results

The dataset is:

`var/product/iseeu-llm/receipt-jp/photo-degradation-realistic-hard500/manifest.json`

It contains 500 realistic-hard generated receipt photos with no raw private documents.

The measured paths are:

Path	Images	Exact match	Field accuracy
Apple Vision OCR path	20	0.000	0.5667
Tesseract English-only path	20	0.000	0.5333
Tesseract multilingual, single path	500	0.724	0.9507
Iseeu OCR frontend, merged psm 4+6	500	0.944	0.9918

Table 1: Image-to-JSON receipt results. The Apple Vision and English-only figures are small measured subsets. The 500-image rows are the main receipt frontend comparison.

The single-path multilingual run matters because it shows that language data alone is not enough:

$$A_{\text{exact}}(\text{single}) = 0.724.$$

The merged frontend matters because it moves the same receipt task to:

$$A_{\text{exact}}(\text{merged}) = 0.944.$$

The absolute exact-match gain is:

$$\Delta_{\text{exact}} = 0.944 - 0.724 = 0.220.$$

The field gain is:

$$\Delta_{\text{field}} = 0.9918 - 0.9507 = 0.0411.$$

That looks smaller, but it is exactly the hard part. At high field accuracy, the remaining errors are concentrated in brittle fields.

Field	Accuracy
merchant_name	1.000
invoice_registration_number	0.974
date	0.992
total	0.988
tax_8_percent	0.992
tax_10_percent	0.994
tax_total	0.986
payment_method	1.000
unsupported_reason	1.000

Table 2: Per-field accuracy for the merged 500-image receipt frontend.

5 Field Gate

For receipt JSON, exact match is strict. It is useful, but it hides where the remaining risk lives. The 500-image merged frontend produced these field figures:

Let F be the set of required fields. A field gate is:

$$g_f = \mathbf{1}[A_f \geq \tau_f].$$

The product gate is:

$$G_{\text{receipt}} = \prod_{f \in F} g_f.$$

If $\tau_f = 0.99$ for every field, the current frontend fails because:

$$A_{\text{invoice}} = 0.974, \quad A_{\text{total}} = 0.988, \quad A_{\text{tax total}} = 0.986.$$

This is not bad news. It tells us exactly where the remaining work is. It also prevents a near-good demo from being sold as finished.

6 Failure Bank

The failure bank contains:

$$138/500$$

rows from the single multilingual path. The errors are concentrated around tax totals, tax-rate lines, registration numbers, merchant names, dates, totals, and payment methods. After the merged path, only:

$$28/500$$

exact failures remain. That is the set that matters now.

I use a simple priority score:

$$P_f = (1 - A_f)w_f,$$

where w_f is the business cost of field f . For Japanese qualified invoices, the registration number deserves a high w_f even when its raw error rate is not the largest. A wrong registration number is not just a typo. It can break accounting trust.

7 Relation to Existing OCR and Document Work

Apple’s Vision framework exposes image text recognition through `VNRecognizeTextRequest` [1]. Tesseract is a mature open OCR engine with a long history and modern LSTM recognition paths [2, 3]. PP-OCR shows that lightweight OCR can be intentionally engineered for small deployment footprints [4]. LayoutLM and Donut represent two important document-understanding directions: OCR-plus-layout modeling and OCR-free document parsing [5, 6]. Qwen2.5-VL is useful here as a teacher for hard visual cases, not as the final small phone artifact [7].

The contribution of this note is narrower. It is a measured boundary for a tiny local receipt route:

guided capture → local OCR fusion → JSON runtime → field gate.

The point is not to worship one OCR engine. The point is to refuse an invalid test: a tiny extractor should not be judged on text that never reached it.

8 Limitations

The Apple Vision result in the table is a measured small subset and a local hard-case baseline, not a universal statement about Apple Vision. The merged frontend result is stronger because it covers the 500-image set, but those images are still generated from source ghosts rather than raw private receipts. The current runtime also depends on a host Tesseract binary, so the 15 MB pack folder is not the entire installed system. The next paper-quality step is a private real-photo holdout with the same formulas and no content leakage.

9 Conclusion

The third lesson is the bluntest one. A small document LLM is not an OCR miracle. It needs a capture box, an OCR boundary, a merge policy, and a field gate. In this receipt experiment, that boundary changed the outcome from unusable OCR-to-JSON results to 0.944 exact match and 0.9918 field accuracy on 500 hard receipt images. The remaining failures are no longer vague. They are registration numbers, totals, and tax fields with measurable costs.

Data and Artifacts

The local evidence used in this note is stored under:

```
dist/iseeu/iseeu-ocr-frontend-receipt-jp-v1/README.md
var/product/iseeu-llm/bench/
var/product/iseeu-llm/ocr-frontend-v1/
var/product/iseeu-llm/receipt-jp/photo-degradation-realistic-hard500/
```

Acknowledgements

This work was carried out as independent research using personally maintained local machines and opportunistic off-hours compute access. Draft preparation and code editing were assisted by AI coding tools under the author’s direction.

References

- [1] Apple. Recognizing Text in Images. Apple Developer Documentation, accessed 2026. <https://developer.apple.com/documentation/vision/recognizing-text-in-images>.

- [2] Tesseract OCR. Tesseract Open Source OCR Engine. Software repository, accessed 2026. <https://github.com/tesseract-ocr/tesseract>.
- [3] Smith, Ray. An Overview of the Tesseract OCR Engine. Ninth International Conference on Document Analysis and Recognition, 2007. <https://research.google.com/pubs/archive/33418.pdf>.
- [4] Du, Yuning; Li, Chenxia; Guo, Ruoyu; Yin, Xiaoting; Liu, Weiwei; Zhou, Jun; Bai, Yifan; Yu, Zilin; Yang, Yehua; Dang, Qingqing; and Wang, Haixiang. PP-OCR: A Practical Ultra Lightweight OCR System. arXiv preprint arXiv:2009.09941, 2020. <https://arxiv.org/abs/2009.09941>.
- [5] Xu, Yiheng; Li, Minghao; Cui, Lei; Huang, Shaohan; Wei, Furu; and Zhou, Ming. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. KDD, 2020. <https://arxiv.org/abs/1912.13318>.
- [6] Kim, Geewook; Hong, Teakgyu; Yim, Moonbin; Nam, Jeongyeon; Park, Jinyoung; Yim, Jinyeong; Hwang, Wonseok; Yun, Sangdoo; Han, Dongyoon; and Park, Seunghyun. OCR-free Document Understanding Transformer. ECCV, 2022. <https://arxiv.org/abs/2111.15664>.
- [7] Qwen Team. Qwen2.5-VL Technical Report. arXiv preprint arXiv:2502.13923, 2025. <https://arxiv.org/abs/2502.13923>.