ENGINEERING FIELD GUIDE | SANGAMPANDEY.INFO | ~18 MIN READ

CHOOSING A DOCUMENT-UNDERSTANDING STACK

How to actually choose an OCR model

Scope it with a client, measure what actually matters, and don’t ship the wrong one. The guide you open *before* you commit to a vendor.

Accurate as of **July 2026**. Pricing, licenses & benchmarks move fast; re-verify before you rely on any figure

Most teams choose an OCR model the way they’d pick a JSON library. They read a leaderboard, grab the top name, move on. Then three months later the RAG answers are subtly wrong, the extraction pipeline mislabels a field, and nobody can say why. The reason is almost always the same: the model was chosen in the abstract, and its quality was never *measured* on the documents it would actually face.

This guide is the opposite of a leaderboard. It’s the process a team runs, from the first client conversation to the production regression harness, to make a defensible OCR decision and be able to prove it. It assumes you’ve skimmed the companion landscape report (what exists); here we care about *how you decide* and, above all, *how you measure*.

THE ONE LAW

You don't choose an OCR model. You choose it for a specific set of documents and a specific job, and you can only choose well if you can measure well.

Everything below is downstream of that sentence. The model name is the *last* decision you make, not the first.

PART I · THE REFRAME

The model is the last decision, not the first

Before you compare a single vendor, you owe two answers: which documents, and to do what.

An OCR model does not have a quality. It has a quality *on a distribution of documents, for a task.* The same model that reads your clean digital invoices at 99% will collapse to 60% on a stack of handwritten delivery notes, and neither number is "the model's accuracy." Both are true, for different inputs.

So the first work of choosing an OCR is not evaluation. It's **scoping**: pin down the document distribution you have to serve and the downstream job you're serving it for. The job matters as much as the documents, because it decides *which part of the landscape you even need.* "Highlight the exact line an answer came from" points at word-level bounding boxes. "Clean markdown for a RAG pipeline" points at structure. "Three fields off an invoice" might not need OCR at all. Same documents, three different correct answers.

> Choose for a distribution and a job. The model falls out of the measurement.

PART II · THE FORK MOST TEAMS MISS

Is this even an OCR problem?

The cheapest, most accurate OCR is the one you never run.

Before the quadrant, before any vendor, there's a fork that isn't on anyone's slide: **does the document already carry a real text layer?** A born-digital Word file, PowerPoint, HTML page, or clean digital PDF already contains its text and often its structure. Running OCR on it is slower, costlier, and *less* accurate than reading what's there.

This is where a tool like **Microsoft MarkItDown**[17] belongs, and where people misuse it. MarkItDown converts Office/HTML/structured files into clean Markdown for LLMs, and it's excellent for that. But it is *not* an OCR engine: point it at a scanned image and it returns nothing useful, because there's no text to extract and no recognition model underneath. Same for **Docling**, **Unstructured**, and plain `pypdf`. They're *converters*, one step *before* the OCR decision.

THE PRE-FORK, STATED PLAINLY

Has a reliable text layer? → Use a converter (MarkItDown / Docling / pypdf). Cheaper, faster, lossless. *No OCR.*

Image, scan, photo, or broken encoding? → *Now* you enter the quadrant and choose a real OCR model.

So MarkItDown is neither “in the list” nor “a bad thing.” It’s in a *different* box, and it’s the right first reach when the input is already digital. Most real corpora are *mixed*, so a mature pipeline routes per page: converter where there’s a text layer, OCR where there isn’t.

PART III · THE OPTION SPACE

Four quadrants, plus one

Two questions collapse a chaotic market into a map you can reason about.

The landscape reduces to two axes: **text blocks ↔ full structure** (what output you need) and **API ↔ self-host** (who runs it). That yields four quadrants, plus in 2026 a fifth cluster the original grid didn’t have: frontier LLMs used directly as OCR.

Fig. 1: The OCR quadrant. Full rosters, pricing and benchmarks in the companion landscape report. Sources: 1–6, 14.

The map tells you *where to shortlist*, not what to pick. Picking still comes from measurement. Shortlist 2–4 candidates from the report, then bring them here to be judged.

PART IV · THE HEART OF IT

What you can't measure, you can't

choose

OCR quality is invisible until it fails three steps downstream. The whole discipline is making it visible before then.

There is no “is this OCR good?” There are three different questions, and most teams only ever ask the first, which is why they ship the wrong model. You judge OCR at three levels, and each answers something the others can’t.

Fig. 2: Character → structure → task. Level 3 decides; Levels 1–2 diagnose *why* a model lost.

1

CHARACTER FIDELITY

Did it read the words right?

Metrics: character/word error rate, edit distance against a reference. Full transcripts are expensive, so the practical trick is **fact assertions**: strings that *must appear* (“the total `$4,281.50`”) and strings that *must not* (a footer you’re stripping). Five to twenty checkable facts per page beats one hand-typed transcript.

Hides: importance. A wrong digit in a dollar amount scores the same as a stray space.

2

STRUCTURE FIDELITY

Did it carry the layout, reading order and tables?

Separate from Level 1, and the one people forget. A model can get *every character right* and still destroy the meaning: two columns interleaved across the gutter, table cells scrambled. Metrics: TEDS for tables, pairwise reading-order checks. The fastest human check: **overlay the bounding boxes on the page image**. Dropped regions, boxes on empty space, and wrong column order jump out in seconds.

Hides: nothing you can see in the raw text, which is exactly why you look at the boxes.

3

TASK FIDELITY · THE NORTH STAR

Is the LLM answering better after the OCR?

OCR is a means to an end. If the app is RAG or extraction, you care whether it gives the *right answer*, not the CER. Build a set of questions and fields with **known answers**, run the *whole* pipeline with each candidate, and measure end-to-end accuracy. The OCR that makes the app answer best wins, *even if it's not top of any leaderboard*, because this metric automatically weights what your task depends on and ignores what it doesn't.

The catch: it's noisier, because it mixes OCR error with retrieval and LLM error. Use Level 3 to pick the winner, Levels 1–2 to diagnose why.

> Level 3 decides. Levels 1 and 2 explain.

Five principles that separate a real eval from a vibe check

1. Measure what you'll ship, not a proxy. If you'll do RAG, score RAG answers; if you extract five fields, score those five. A benchmark measures someone else's documents for someone else's task.

2. Stratify or lie. One aggregate number averages your clean invoices and your handwriting into a score that describes neither. Report a *matrix*: accuracy per

document profile. “98% on digital invoices, 60% on handwriting” is the truth, and it doubles as your scope boundary.

3. A model is a point in three dimensions. Never rank on quality alone; every candidate is *(quality × cost-per-1k × latency)*. “1,000 pages a second, all wrong” is worthless; so is “99% at $500/1k and 40 s/page” for a live product.

4. Hallucination is a different, and worse, failure than error. Traditional OCR fails *loudly* (garbage you catch); a VLM fails *silently* (a plausible date on a blank field). A fabricated total is a liability; a blank is a caught exception. Measure it separately with blank/noise pages, and prefer models that abstain, that flag low confidence instead of inventing.

5. “Good enough” is set by the cost of an error, not a percentage. 95% is excellent for search and catastrophic for drug dosages. The bar comes *from the business*, and it’s set by the tail, not the median. The worst 5% of documents define the experience and the liability.

METRICS HONESTY: WHAT EACH ONE HIDES

CER/WER: blind to importance. **TEDS:** penalizes valid alternative table linearizations. **Edit distance on markdown:** punishes formatting *choices* as if they were reading *errors*. **Any leaderboard:** distribution shift, contamination, and saturation. An audit of a top OCR benchmark found **~53% of the “errors” at the top were the benchmark’s own noise, not the model’s.**[7] Triangulate across metrics; end every eval with human eyes on the worst outputs.

PART V · THE ASSET

Build the golden set

The eval set is the real deliverable. Build it once, and the model choice (and every future re-choice) falls out of it.

You don’t need perfect transcripts of thousands of pages. You need a small, *representative*, *stratified* set with *tiered, cheap* ground truth. Fifty to a hundred pages is plenty, and this set becomes a durable asset: your bake-off harness today, your regression suite forever after.

1. **Sample from reality, not the demo folder.** Pull pages from the actual pipeline the model will face; over-sample the ugly tail.
2. **Stratify and tag every page.** Clean / average / worst-that-still-must-work, each tagged by profile (digital · scan · photo · handwriting · table · multi-column · language). The tags are what produce the per-profile matrix.
3. **Seed hallucination bait.** A handful of blank, near-blank, and pure-noise pages. They catch the silent fabrication no clean page will reveal.
4. **Write tiered ground truth.** Per page: a few must-appear facts, a few must-not-appear, the order of two spans, key field values, table row/column counts. Cheap to author, machine-checkable.
5. **Don’t let a candidate grade its own test.** If you bootstrap labels with a strong model, make sure it isn’t one you’re evaluating. That’s contamination.

WHY THIS IS THE HIGHEST-LEVERAGE HOUR YOU’LL SPEND

Vendors change models under you, new open weights drop monthly, and your document mix drifts. A frozen golden set turns every one of those events from “re-litigate the decision” into “re-run the harness.”

PART VI · THE BAKE-OFF

Run the comparison

A day for a first read, a week for a decision you’d stake a contract on.

1

SCOPE

Define “correct”

Write down what the output must be: raw text, markdown, tables as HTML/JSON, specific fields, reading order, grounding boxes. This decides what you diff.

2

SHORTLIST

Pick 2–4 candidates across quadrants

One cheap cloud, one open VLM, one frontier LLM. Cross-quadrant on purpose. You’re testing the *approach*, not just the model.

3

RUN

Same golden set through every candidate

Log the full triple for each: accuracy, cost per 1k, p95 latency.

4

DIFF

Compare text, bucket by error type

Missing, hallucinated, wrong reading order, table breakage, math. Hunt the silent failures specifically.

5

SEE

Visualize the bounding boxes

Overlay each model’s boxes on the page image. Catches in seconds what a text diff hides.

6

READ

Look at the worst 10–15 outputs, by eye

Metrics rank; eyes explain. You don’t hand-grade millions; you spot-read the tail.

7

DECIDE & FREEZE

Pick the winner on *your* matrix; keep the harness

Choose on your stratified scorecard at acceptable cost and latency, not the leaderboard. Then freeze the set and re-run on every model or version change.

PART VII · THE DECISION

Read the scorecard, then commit

The winner is the model that wins on your documents, for your job, at a cost and latency you can live with. Nothing more.

By now the choice is nearly mechanical, because the scorecard encodes everything that matters: a quality number *per profile*, a cost per thousand pages, a p95 latency. You're not after the highest single number. You're after the model whose *weakest relevant profile* clears your bar at a price and speed the product can bear.

Two forces settle it. **The job picks the axis**. Need grounding boxes? That rules out most frontier LLMs. Need clean structure for RAG? That rules out plain text-block clouds unless you build a post-processor. **Scale and constraints pick the hosting**. Chasing product-market fit, or under a few million pages a month? Use an API. Regulated data that can't leave your walls, or huge steady batch volume with an ML team on payroll? That's where self-host earns its keep.

THE DEFAULT THAT'S RIGHT MOST OF THE TIME

Start on an API. Earn the right to self-host.

An API buys ease, support, and someone to call when a document breaks; self-host buys control and, only at real batch scale, lower cost. In the source talk, Joe Barrow estimates fewer than 5% of teams should self-host[13]. Treat that as a prior, not a law. The real test is the decision criteria above: hosting is justified by *steady batch-shaped volume + an in-house ML team + a residency constraint*, not by a percentage.

Whatever you pick, resist two traps: don't optimize cost or latency at the expense of the quality your whole app is built on, and don't let a benchmark overrule your own scorecard. The model that read someone else's clean PDFs best is not the model that will read your customer's crumpled ones best.

PART VIII · THE MONEY

API vs self-host, honestly

The sticker price is not the price. Lock-in and engineer-time are the terms that actually decide.

The real comparison isn't "$1.50/1k vs $0.20/1k." It's (**`sticker × lock-in`**) for an API against (**`GPU cost + amortized engineer time`**) for self-host, and that engineering term is what kills self-host for most teams. Self-host *barely* beats the cheap plain-text clouds (already floored near GPU cost) until millions of pages a month; it beats the *expensive* structured tiers at a fraction of that.

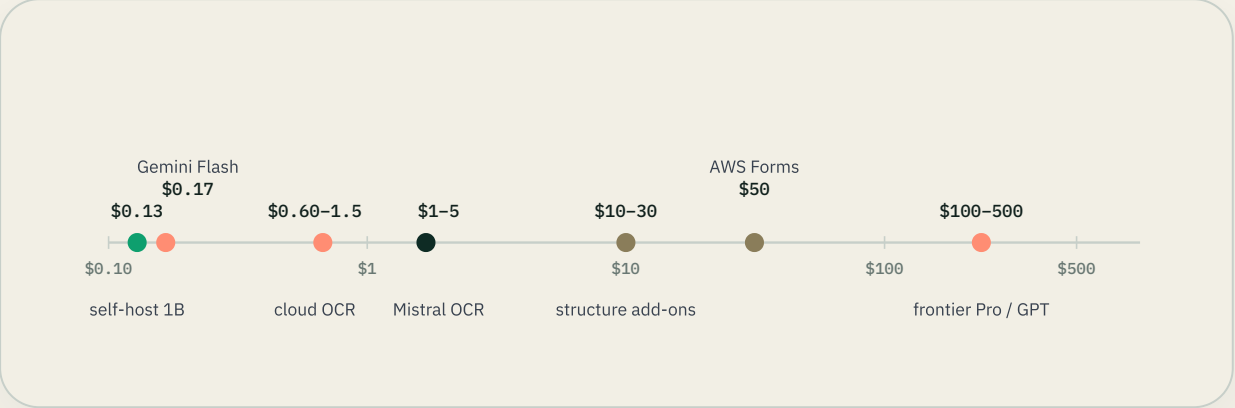

Fig. 3: Cost per 1,000 pages, log scale. A single page spans a ~3,800× range by quadrant and feature. Sources: 1–6, 11.

YOU'RE BEATING…	API $/1K	SELF-HOST $/1K	BREAK-EVEN VOLU
Cheap plain OCR (Textract/Google/Azure base)[1]	$0.60–1.50	~$0.20	~3–5M pages/mo
Structured tier (Azure Layout, prebuilt)[3]	$10	~$0.20	~400K pages/mo
Tables/forms (Reducto, Google Form Parser)[6]	$15–30	~$0.20	~135–270K pages/m
Premium extraction (AWS Forms, Nanonets, ABBYY)[1]	$50–300	~$0.20	~15–80K pages/mo

† Break-even volumes are a **modeled illustration**, not a quoted figure. They are computed from the API prices cited, a self-host marginal cost of ~$0.20/1k on a saturated H100, and ~$4K/mo of ops overhead. Re-run the math with your own rates. A common community rule of thumb is **four used RTX 3090s ≈ one H100** for batched inference[12]; verify against your workload.

The cloud wins on *idle*: a GPU you're not saturating is money on fire, which is why spiky, low-latency, user-facing OCR usually belongs on an API even when batch OCR of the same documents belongs on your own metal.

PART IX · PRODUCTION

Ship it without getting burned

The decision isn't "which model." It's an architecture that survives the model changing under you.

OCR is *sticky*: once your app is built on one model's output format, reading-order behavior, and box coordinate system, ripping it out is painful. The fix is to refuse the coupling in the first place.

BUILD THIS

- **An abstraction layer over OCR.** Define *your own* internal schema: canonical reading order, normalized 0–1 boxes, your own block types. Every backend gets an adapter; nothing downstream touches a vendor format. Highest-leverage defense, and it doubles as your license kill-switch.
- **A golden-set regression harness** on every model/version bump, because vendors push new weights silently.
- **Dual-run shadow eval** for migrations: run the candidate beside production, diff, score, then cut over.
- **Confidence-gated human review**: route low-confidence pages to a person, not to blind trust.
- **Validation guardrails**: cross-check numbers, enforce format rules, adopt “no box, no claim.”

NEVER DO THIS

- × Wire a vendor’s raw JSON schema straight into downstream code.
- × Assume this month’s weights are next month’s weights.
- × Ship a big-bang model swap with no shadow period.
- × Trust a VLM on blank or degraded pages without a hallucination check.
- × Treat a passing offline eval as proof it works in production.

Monitor the same three levels in production, sampled continuously: confidence distributions (drifting?), structure sanity (tables still parsing?), and task accuracy on a rolling labeled sample. When any moves, your frozen golden set tells you whether the documents changed or the model did.

PART X · THE LANDMINES

Licensing, residency, lock-in

The failure that shows up in no metric: the model was great, and you weren't allowed to use it.

"Open" is three questions, not one. Is the *code* open? Are the *weights* open? Is the *base model* they were trained on open? Most traps live in the gaps: Apache-licensed code with revenue-capped weights, or an Apache tag on weights fine-tuned from a non-commercial base[18], inheriting the restriction whether the card admits it or not.

- **Broadly permissive** (Apache/MIT weights, no revenue cap): LightOnOCR[19], PaddleOCR-VL, GLM-OCR, dots.ocr, DeepSeek-OCR, Granite-Docling, olmOCR.
- **Revenue-gated:** Datalab's Chandra (free under $2M funding/revenue)[9] and Surya (under $5M)[10]. Different thresholds, plus a ban on competing with their API. Free to prototype, a bill the day you cross the line.
- **Non-commercial, full stop:** MonkeyOCR, Nougat, anything inheriting Qwen's research license[18]. Demo only.

READ THIS BEFORE YOU WRITE "SHIP ANYTHING"

A permissive weights license is *necessary but not sufficient.* It does **not** erase base-model restrictions, **trademark** constraints on names/branding, **patents**, **export-control** obligations, or the **training-dataset** license the weights inherit. "Apache/MIT" means you've cleared one gate, not all of them. **Verify commercial use with counsel** before you build a product on it.

Data residency is the third axis: OCR ingests exactly your highest-sensitivity documents. Sending them to a hosted API can trigger HIPAA, which requires a signed Business Associate Agreement before any PHI[15], and GDPR, which requires a data-processing agreement plus in-region processing[16]. *Self-hosted + permissively-licensed weights* is often the cleanest way to satisfy both, which is a real reason regulated teams land in the open-VLM quadrant. But it is **not automatically the only compliant option**: a hosted vendor with a signed BAA/DPA and in-region processing can also qualify. These are legal determinations. **Confirm the specific requirement with your security and compliance team**, not a blog.

THE MOVE THAT DE-RISKS ALL OF IT

The abstraction layer from Part IX *is* your insurance. Verify the license before you build, not after your Series A. Keep a permissive backend ready behind the same adapter, so the day a license turns against you, the swap is a sprint, not a rewrite.

PART XI · THE CLIENT

What to ask, and what to promise

You scope an OCR project from a representative pile of the client's actual documents and a written agreement on "done," never from a brief.

Yes. You ask the client for documents, and you get explicit confirmation. The single most important request: *"Send me 50–100 of your real documents: your most common, your ugliest, and your weird edge cases. Not your cleanest demo file."* Everything downstream is built on that sample.

Then you run a short **calibration** and report back: "Here are the document profiles I found; here's what I'll validate for; here's what's explicitly out of scope." That report is a deliverable, and getting it signed protects both sides.

> Scope is a contract. Out-of-scope profiles are a feature you write down, not a failure you discover in production.

Set expectations from the cost of an error, in the client's language: "On your digital invoices we'll be ~99%; on the handwritten annotations, expect ~60% and human review. Acceptable, or do we route those separately?" A client who agreed to the tail up front is a partner; one who meets it in production is a dispute.

THE SHORT VERSION

Dos & don'ts

DO

- Ask "is this even an OCR problem?" first, and use a converter for digital files.
- Get 50–100 *real*, representative documents before choosing anything.
- Define "correct" as the downstream job, in writing.
- Measure at all three levels; let task fidelity decide.
- Report a per-profile matrix, never one aggregate number.
- Log quality, cost, and latency together, always the triple.
- Seed blank/noise pages to catch silent hallucination.
- Prefer models that abstain over models that confidently lie.
- Start on an API; freeze a golden set as a regression harness.
- Build an abstraction layer; verify licenses with counsel before you build.

DON'T

- × Pick from a leaderboard, because it measures someone else's documents.
- × Trust one aggregate accuracy number.
- × Rank on quality alone, or optimize cost/latency into garbage.
- × Point MarkItDown/pypdf at scans and expect OCR.
- × Treat CER as proof, because a perfect-character page can be meaningless.

- ✕ Assume a benchmark leader wins on your data.
- ✕ Ship a self-hosted model whose license you didn't read.
- ✕ Send regulated documents to an API without a BAA/DPA.
- ✕ Hard-wire a vendor's output schema into your app.
- ✕ Do not promise accuracy based on your own test corpus instead of the client's documents.

APPENDIX · COPY-PASTE & DOWNLOAD

The templates

Everything above, reduced to the four artifacts you'll reuse. Each downloads as a real file.

A · CLIENT INTAKE QUESTIONNAIRE

- ☐ **Document profiles & rough mix.**
  "Invoices 60%, contracts 30%, handwritten notes 10%," with a real sample of each.
- ☐ **Source of each type.**
  Born-digital, scanned, phone-photographed, faxed? (Decides converter-vs-OCR per type.)
- ☐ **Volume & shape.**
  Pages/month, peak, batch vs live? (Decides API vs self-host and the cost model.)
- ☐ **Languages & scripts.**
- ☐ **Structure needed.**
  Tables? forms? handwriting? math? checkboxes? signatures?
- ☐ **The downstream job.**

Search, field extraction, Q&A, full re-typesetting? (Decides text-blocks vs structure.)

- [ ] **Grounding requirement.**
  Highlight the exact source on the page? (Decides bounding-box need.)

- [ ] **Sensitivity & residency.**
  PII/PHI/PCI? On-prem or residency constraints? (Rules API in or out.)

- [ ] **Accuracy bar & cost of an error.**
  What's "good enough," and what does one wrong field cost?

- [ ] **SLA.**
  Latency, throughput, uptime.

- [ ] **Scale trajectory & budget.**
  Also decides which licensing revenue caps you'll hit.

- [ ] **The 50–100 ugliest documents.**
  The single most important ask on this list.

B · GOLDEN-SET SPEC

```
golden_set:
  size: 50-100 pages          # sampled from the REAL pipeline, not the demo
folder
  buckets:
    clean:   ~30%             # best case users send
    average: ~40%             # typical real document
    worst:   ~25%             # blurry photos, skew, stamps, handwriting,
                              # merged-cell tables, multi-column, faint scans
    bait:     ~5%             # blank / near-blank / pure-noise
(hallucination trap)
  tags_per_page:              # what produces the per-profile matrix
    - source: [digital, scan, photo]
    - handwriting: bool
    - tables: bool
    - multi_column: bool
    - language: str
    - bucket: [clean, average, worst, bait]
```

```
  ground_truth:              # tiered, NOT a full transcript
    must_appear:     ["$4,281.50", "Invoice #A-102"]
    must_not_appear: ["CONFIDENTIAL"]
    order:           ["header before body", "col-1 before col-2"]
    fields:          {total: "$4,281.50", date: "2026-03-11"}
    table:           {rows: 7, cols: 4}
  rule: never let a model you are evaluating generate its own ground truth
```

C · THE SCORECARD (A POINT IN 3D, PER PROFILE)

CANDIDATE	DIGITAL	SCAN	HANDWRITING	TABLES	$/1K	P9
Cloud (Textract)	–	–	–	–	$1.50	–
Open VLM (PaddleOCR-VL)	–	–	–	–	$0.20	–
Frontier (Gemini Flash)	–	–	–	–	$0.17	–

Fill quality cells from your golden set (task accuracy where you can). Winner = highest *weakest-relevant-profile* at acceptable cost/latency, not the highest single cell.

D · FACT-ASSERTION EVAL FORMAT

```
{
  "page_id": "invoices/2026-03/acme-102.pdf#p1",
  "profile": {"source": "scan", "tables": true, "handwriting": false, "lang":
"en", "bucket": "worst"},
  "assertions": {
    "must_appear":     ["Acme Corp", "$4,281.50", "Net 30"],
    "must_not_appear": ["CONFIDENTIAL"],
    "order":           [["invoice_number", "line_items"]],
    "fields":          {"total": "$4,281.50", "due_date": "2026-04-10"},
    "table":           {"rows": 7, "cols": 4}
  }
}
```

```
Score per assertion type → per-profile matrix. Run on every model/version
change.
```

SOURCES

References

All claims accurate as of July 2026. Pricing, model licenses and benchmark scores change frequently. Re-verify before you rely on any figure.

1. AWS Textract pricing: base OCR and à-la-carte tables/forms/queries; volume tiers.
2. Google Cloud Document AI pricing: Enterprise OCR, Layout Parser, Form Parser.
3. Azure AI Document Intelligence pricing: Read, Layout, prebuilt, custom.
4. Mistral OCR pricing: OCR 3 / OCR 4 / Document AI per-page rates.
5. LlamaParse pricing: fast → agentic credit ladder.
6. Reducto pricing: parse vs extract credit model.
7. LlamaIndex: olmOCR-Bench review, insights & pitfalls. The audit finding ~53% of top-model "errors" are benchmark/annotation noise.
8. "When Good OCR Is Not Enough: Benchmarking OCR Robustness for RAG". The 82.9% OCR → 53.0% RAG "blind spot."
9. Datalab: Chandra. Modified OpenRAIL-M weights license, free under $2M funding/revenue, no competitive use.
10. Datalab: Surya-OCR-2 model card. Modified AI-Pubs OpenRAIL-M, free under $5M funding/revenue.
11. GPU cloud pricing comparison (2026): H100 hourly rates across providers (basis for $/1k self-host math).
12. Cloud GPU TCO vs self-hosted LLM: self-host break-even & the multi-3090 ≈ H100 rule of thumb (community estimate; verify).
13. Joe Barrow, "How to choose an OCR model". Source talk; the "fewer than 5% should self-host" estimate is the speaker's opinion.
14. OmniDocBench & PaddleOCR-VL 1.5 report: sub-1B open models (GLM-OCR ~94.6, PaddleOCR-VL ~94.5) topping the leaderboard.

15 U.S. HHS: HIPAA Business Associate Agreement provisions. BAA requirement before a vendor handles PHI.

16 GDPR Article 28, Processor. Data-processing-agreement requirement for third-party processing.

17 Microsoft MarkItDown: file-to-Markdown converter (not an OCR engine).

18 Nanonets-OCR license discussion: an Apache-tagged model inheriting the non-commercial Qwen research license from its base.

19 LightOnOCR-2: Apache-2.0 weights and dataset.

20 **Modeled figures note.** Break-even volumes in Part VIII are illustrative, computed from the cited API prices, ~$0.20/1k self-host marginal cost, and ~$4K/mo ops overhead, not vendor quotes. Substitute your own numbers.

Sangam Pandey · sangampandey.info · engineering field guide · 2026-07-26.
Built from Joe Barrow's "How to choose an OCR model."

Principle to leave with: **you can't choose what you can't measure, so build the measurement first, and let the model fall out of it.**