

## LESSONS FROM PRODUCTION

# 15 things I learned building AI agents that cross-analyze hundreds of unstructured documents

Hard-won lessons from building a system that runs 4 parallel analysis workstreams across messy PDFs, Word docs, Excel files, and images — then connects the findings across domains.

These patterns apply to any system doing cross-document analysis at scale: contract review, compliance, research synthesis, knowledge management.



Zohar Babin

Chief Strategy Officer, Kaltura

## 01

## LESSON 1

# Extraction is harder than analysis.

## By a lot.

Everyone focuses on the LLM prompts. But 80% of the real engineering is getting clean, reliable text out of messy documents.

1

**Pre-inspect the PDF**

First page <100 chars? Skip text extraction entirely — go straight to OCR



2

**Try fast extraction (pymupdf → pdftotext)**

6 quality gates per method: min chars, printable ratio, density, readability, watermark, corruption



3

**Falls through? OCR chain (Tesseract → GLM-OCR)**

Vision-language model catches what traditional OCR misses



4

**Last resort: Claude vision (expensive, 120s timeout)**

Confidence: 0.65. Only for truly unreadable files

**KEY INSIGHT**

**Confidence scales with method quality.** pymupdf gets 0.9 base, pdftotext 0.7, OCR 0.65. Then scaled further by file-size-to-text ratio. Sparse extractions get penalized automatically.

## 02

## LESSON 2

# Entity resolution is your **invisible foundation**.

"IBM", "International Business Machines", and their subsidiary "Red Hat" — are these the same entity in your analysis? If your system can't resolve that, every downstream finding is wrong.

1

**Exact match**

Confidence: 1.0 — cheapest, most reliable



2

**Normalized match (strip suffixes, lowercase)**

"Inc", "Corp", "LLC", "Ltd" removed before comparison



3

**Alias & abbreviation expansion**

Config-driven: "IBM" → "International Business Machines"



4

**Fuzzy match (rapidfuzz token sort ratio)**

Threshold varies by name length — longer names need higher similarity



5

**TF-IDF cosine similarity**

Catches semantic similarity that character-level matching misses



6

**Parent-child / learned matches**

Cache learns from confirmed matches across runs

**COUNTER-INTUITIVE**

**Names ≤5 characters are blocked from fuzzy matching.** Without this, "Inc." matches random entities. Simple guards prevent expensive false positives downstream.

# 03

LESSON 3

## Documents aren't independent. They form a hierarchy.

An MSA governs every Order Form beneath it — until an Order Form says *"notwithstanding anything to the contrary..."* and the child overrides the parent. Precedence isn't always top-down.

```
# 4 documents, 4 contradictory termination clauses

MSA §4.2: Mutual termination, 30-day notice
  ↓ AMENDS
Amendment #2: No termination during first 12 months
  ↓ WAIVES (notwithstanding...)
Order Form #3: Customer may terminate at any time
  ↓ CONDITIONS
Order Form #3 §8: No refund of prepaid annual fees upon early termination
```

12

Document types  
(MSA, OF, SOW, DPA...)

7

Relationship types  
(governs, amends, waives...)

50ms

CoC impact query  
across 200 contracts

KEY INSIGHT

**LLMs extract what each clause says. The graph determines how clauses interact.**  
Amendment chains, cascade impact, conflict detection — all deterministic graph traversal, not LLM calls.

## LESSON 4

# Don't dump everything into one context.

## Map-merge-resolve.

A 200-page master agreement might have the deal-killer on page 147. You can't skip large files. But you can't just dump them into one giant context either — accuracy drops from 95% to 74%.

**MAP****Chunk at page boundaries, analyze independently**

Same question asked per chunk. Focused context = higher accuracy per chunk. 150K chars, 15% overlap.

**MERGE****Combine answers with priority logic**

YES beats NO. Specific beats generic. "Termination right on page 147" survives even if other chunks said NOT\_ADDRESSED.

**RESOLVE****LLM arbitration — only when chunks disagree**

Chunk 1 says YES, chunk 3 says NO? A synthesis pass resolves the conflict. Only triggered on disagreement — saves cost.

**VALIDATE****Second pass on NOT\_ADDRESSED answers**

Catches things that fell exactly on a chunk boundary. The overlap minimizes this, but doesn't eliminate it.

**74%**

Full-document  
context accuracy

**95%**

Focused chunks  
+ clause-aware prompts

**+21pt**

Entirely  
engineering

**MAPREDUCE-INSPIRED, NOT CLASSICAL**

The reduce phase is smarter than simple aggregation — it's priority-based merging with conditional LLM arbitration. **This is an accuracy pattern, not a scaling pattern.** You get better answers by analyzing focused chunks than by analyzing everything at once.

## 05

## LESSON 5

# Hallucination is an engineering problem, not a model problem.

No single defense works. You need layers — like security, defense in depth.

1

**Structured output via Pydantic**

Every response validated against a schema. Malformed output rejected immediately.

2

**Mandatory citation verification**

Every claim must include file\_path, page, section, exact\_quote. Verified against source.

3

**"Not found" escape valve**

Explicit instruction to write gaps instead of fabricating. Without this, models invent clauses.

4

**Adversarial Judge review**

Separate agent spot-checks findings. P0: 100%, P1: 20%, P2: 10%. Accusatory framing.

5

**6-layer deterministic numerical audit**

Re-derives every number. Cross-checks sources. Catches what LLMs miss.

## THE BIG ONE

**Layer 3 changed everything.** When you tell the model "if you can't find this clause, say NOT\_FOUND" — hallucination drops dramatically. Without the escape valve, models fabricate rather than admit ignorance.

## 06

## LESSON 6

# Know when to stop using LLMs.

We had an LLM agent doing validation and report synthesis. We replaced it with deterministic Python. Quality went up. Cost went down.

**BEFORE: LLM AGENT**

- Non-deterministic output
- Occasionally skipped checks
- Expensive per run
- Hard to debug failures
- "Felt smart" but unreliable

**AFTER: DETERMINISTIC PYTHON**

- 30 DoD checks, every time
- 6-layer numerical re-derivation
- Zero cost, instant execution
- Every failure has a stack trace
- Boring, but always correct

**RULE OF THUMB**

**Use LLMs for analysis and synthesis. Use Python for validation, dedup, and audit.** If you can write the logic as deterministic code, do it. LLMs are for tasks where the rules can't be fully specified.

# 07

LESSON 7

## Self-verification improves accuracy by 9.2%.

After agents produce findings, a follow-up pass challenges them on high-severity claims. The framing matters.

DOESN'T WORK

- "Please review your finding"
- "Are you sure about this?"
- "Double-check the above"
- *Models confirm their own output*

WORKS: ACCUSATORY FRAMING

- "This finding appears fabricated"
- "The cited clause doesn't exist"
- "Prove this with a direct quote"
- *Forces the model to re-examine evidence*

FROM THE RESEARCH

Addleshaw Goddard found that **accusatory follow-up prompts** produced a 9.2% accuracy improvement across 510 contracts. Polite verification prompts had near-zero effect. The model needs to feel challenged, not confirmed.

Source: Addleshaw Goddard RAG Report, 2024



## 08

## LESSON 8

# Cross-agent dedup is a different problem than you think.

When 4 agents analyze the same document, they find the same issue — but describe it completely differently.

```
# Legal agent:
"Change of control termination right in Section 12.3"

# Finance agent:
"Revenue at risk: $2.4M ARR subject to CoC clause"

# Same customer. Same clause. Different framing.
# Semantic similarity: 0.82 (above threshold)
```

Three rules that emerged from production:

1

**Never dedup within the same agent**

Two similar findings from Legal are intentionally distinct. Trust the agent.

2

**Only dedup across agents on the same document**

Similar findings on different documents are different findings.

3

**Keep contributing agent metadata**

When you merge, record which agents found it. Explains severity escalation.

## 09

## LESSON 9

# Context window engineering is a first-class discipline.

It's not just about fitting data in. It's about what goes where.

```
# 200K token context window. Budget: 80K (40% margin)
```

```
START: Critical instructions, output schema  
(highest recall zone)
```

```
MIDDLE: Document content, customer data  
(lowest recall — ~40% worse)
```

```
END: Reminders, constraints, format rules  
(second-highest recall zone)
```

```
RESERVED: ~115K tokens for tool calls & reasoning
```

Other hard-won numbers:

**150K**

chars per chunk  
(page-boundary aligned)

**20**

max customers  
per agent batch

**120KB**

file size threshold  
(Grep instead of Read)

## 10

## LESSON 10

# Quality gates and agent guardrails must be **blocking**. Not advisory.

If validation just logs a warning, nobody reads it. If it halts the pipeline, quality is non-negotiable. Same for agents — without hard boundaries, they go rogue.

## 5 blocking quality gates

**HALT**

Extraction quality, coverage validation, numerical audit, citation verification, Definition of Done. Pipeline halts on failure.

**KILL**

### Agent turn limits: soft at 200, hard kill at 3x

Agents get a grace period, then forcibly terminated. Prevents runaway sessions from spiraling costs.

**BLOCK**

### Path guard: agents can only write under `_dd/`

Symlink resolution prevents escape. No writes outside the project directory, ever.

**BLOCK**

### Bash guard: 24 blocked patterns + 5 regex rules

No `rm -rf`, no `sudo`, no pipe-to-shell, no package installs. Agents have tools, not shell freedom.

**PRINCIPLE**

**Better to produce nothing than unreliable output. Better to kill an agent than let it run wild.** Fail-closed, not fail-open. Hard boundaries, not soft suggestions.

## 11

## LESSON 11

# Every claim must be traceable to source.

If an AI says "termination right in Section 4.2" — prove it. Citation verification is what separates analysis from hallucination.

1

**Scope 1: Exact page match**

Search the cited page for the exact quote. Normalized whitespace handles PDF layout artifacts.

↓

2

**Scope 2: Adjacent pages ±1**

Catches cross-page quotes and off-by-one page errors from extraction.

↓

3

**Scope 3: Full document fuzzy match (80%+)**

OCR'd text never matches exactly. Fuzzy threshold catches reformatted quotes.

↓

4

**Scope 4: Cross-file search**

Quote not in the cited file? Search ALL files for the customer. Auto-corrects file misattribution.

**THE INDEX STRUCTURE IS THE FOUNDATION**

Extracted text, file inventories, and customer registries are persisted in a **structured folder hierarchy with index files** — not just for pipeline performance, but so any tool (Claude Code, grep, custom scripts) can explore and cross-reference the same dataset. The structure is the API.

## 12

## LESSON 12

# Most of what AI finds is **noise**.

Run 4 agents across hundreds of documents and you'll get thousands of findings. Most are useless. Without classification, the real risks drown in a sea of low-value alerts.

1

**Noise filter (15 patterns)**

"Cannot assess: extraction failed", "binary file", "unreadable document" — artifacts of the extraction process, not real findings.



2

**Data quality filter (14 patterns)**

"Data unavailable", "cannot verify financial data", "records unavailable" — real gaps, but not material risks. Routed to a separate appendix.



3

**Material findings**

Everything that survives both filters. These are the findings that actually matter for deal decisions.

**PLUS: SEVERITY RECALIBRATION**

5 rules that auto-adjust severity based on context. Example: a change-of-control clause that only applies to competitors — not the buyer — gets **downgraded from P0 to P3** automatically. Without this, every CoC clause looks like a deal-killer.

## 13

## LESSON 13

# Every API call is a deal cost.

Cross-analyzing thousands of documents across hundreds of customers, jurisdictions, partnerships, IP portfolios — costs grow fast. In M&A, pipeline costs show up on someone's fee schedule. Cost optimization is an architecture decision, not an afterthought.

## NAIVE APPROACH

- Best model for everything
- No per-agent tracking
- No budget limits
- Costs spiral, nobody notices until the invoice

## OPTIMIZED APPROACH

- 3 model profiles: economy, standard, premium
- Per-agent + per-step cost tracking
- Hard budget limits that halt the pipeline
- Right model for right task

```
# Not every task needs the best model
Extraction & triage: Haiku   ($0.80/M input)
Specialist analysis: Sonnet ($3/M input)
Executive synthesis: Opus   ($15/M input) — only where it matters
```

## WHY THIS MATTERS FOR M&A

With intelligent cost management at every layer, **the pipeline cost becomes manageable even for large-scale deals** — making the tool invaluable alongside expensive advisors, not additive to the fee burden.

## 14

## LESSON 14

# Make every run smarter than the last.

Inspired by Andrej Karpathy's "LLM Wiki" pattern. Instead of starting from scratch each time, compound knowledge across runs.

KB

## Persistent knowledge base

Entity profiles, clause summaries, contradictions, insights — atomic writes, auto-indexed

↗

## Finding lineage via SHA-256 fingerprinting

Track findings across runs even when wording changes. 5-state lifecycle: active → resolved → recurring

⊕

## Graph-based relationship intelligence

NetworkX knowledge graph with 11 typed edge types. Cycle detection. Path queries. Contradiction detection.

↑

## Agent context enrichment

Next run's prompts include prior entity profiles, known contradictions, finding history

### WHY THIS MATTERS

Run 1 finds the risk. Run 2 knows the context, catches what changed, and flags new contradictions. **The system develops institutional memory** — like an analyst who remembers every deal they've ever worked on.

## 15

## LESSON 15

# Same clause. Different deal. Different severity.

Your AI doesn't know what matters — the deal structure decides. Without that context, every flag looks critical.

**ASSET PURCHASE**

- **P0** — Anti-assignment clause blocks contract transfer. Buyer must get consent or lose the customer.
- **P1** — Shared services agreement must be replicated. No transition plan exists.

**STOCK PURCHASE**

- **P3** — Same clause, but the legal entity doesn't change. No assignment occurs. Routine notation.
- **P3** — Intercompany obligations eliminated at closing. Not a risk.

**The twist:** some contracts treat ownership change as "deemed assignment" — so anti-assignment triggers even in stock deals. The system disaggregates CoC into 5 subtypes to catch this.

L1

**Prompt-time: inject deal-type rules into agent instructions**

Agents reason differently about findings based on transaction structure

L2

**Post-hoc: deterministic rules downgrade known false positives**

TfC capped at P2. Competitor-only CoC → P3. Speculative language → P2.

L3

**Executive synthesis: professional judgment overrides**

Full audit trail: original severity → recalibrated → executive override + rationale

**KEY INSIGHT**

**One calibration layer isn't enough.** Prompt-time rules alone still produce false P0s. Deterministic rules alone miss nuance. You need both — plus a judgment layer on top. Domain context must flow through the entire pipeline.



THE TAKEAWAY

The gap between a demo and a system people can trust is entirely engineering.

74% → 95% accuracy. No model change. Just better extraction, chunking, validation, verification, and architecture.

These patterns aren't specific to M&A. They apply to any system doing cross-document analysis at scale — legal review, compliance, research synthesis, knowledge management.

|            |  |
|------------|--|
| PROJECT    | Due Diligence Agents                                 |
| INSTALL    | <code>pip install dd-agents</code>                   |
| GITHUB     | <code>zoharbabin/due-diligence-agents</code>         |
| LICENSE    | <code>Apache 2.0 • 3,000+ tests • mypy strict</code> |
| BUILT WITH | Claude Agent SDK                                     |

