LinkTec LABS

RESEARCH NOTE N°01 · APPLIED AI & DATA

Beyond RAG: the rise of **Context Architecture**

Engineering the context layer for autonomous agents.

ABSTRACT

Retrieval-Augmented Generation made large language models useful on private knowledge. But the single-shot pipeline — embed, search, stuff, generate — was designed for one human question, not for autonomous agents that issue thousands of calls and accumulate state. As agents reach production, the binding constraint is no longer the model or the vector index; it is the *context layer*: what reaches the model, when, in what form, and at what cost. This note reframes RAG as one component of a broader discipline — *context architecture* — and surveys the techniques that now define the state of the art: hybrid and contextual retrieval, late-interaction models, agentic (self-correcting) retrieval loops, graph-structured reasoning, context-budget management, and the evaluation methods that keep all of it honest. We close with the reference architecture LinkTec Labs deploys for organisations moving from proof-of-concept to production.

01 — THE SHIFT

From clever prompts to engineered context

Retrieval-Augmented Generation[1] was the idea that unlocked enterprise adoption of language models: instead of fine-tuning a model on private data, you retrieve the relevant passages at query time and let the model read them. It worked. It also quietly set a ceiling. The canonical pipeline — chunk the corpus, embed it, run a nearest-neighbour search on the user's question, concatenate the top results, and generate — assumes a single, well-formed human query and a single pass. That assumption breaks the moment the caller is an agent.

Autonomous agents do not ask one question. They plan, call tools, read results, revise, and call again — emitting orders of magnitude more retrievals than a person, and carrying an ever-growing working state between steps. The scarce resource shifts from *model quality* to *context quality*: the limited window must be filled, at every step, with exactly the information and tools the next decision requires — and nothing else. Andrej Karpathy framed it memorably: the LLM is a new kind of operating system, and the context window is its RAM[2]. The job of the engineer is memory management.

This is why the field has moved, in the space of two years, from *prompt engineering* to what is now widely called *context engineering* — and, at production scale, to *context architecture*: the systematic design of the pipelines, indices, controllers and budgets that decide what the model sees. Anthropic has published explicit guidance on the practice[3]; analysts now describe it as the discipline that supersedes prompt engineering. The thesis of this note is simple and, we believe, consequential:

> The bottleneck in applied AI is no longer the model. It is the architecture of the context that surrounds it.

For a model vendor, that is a footnote. For an organisation trying to put agents into production on its own messy data, it is the whole game — and it is buildable, measurable, and ownable. The remainder of this note lays out the terrain.

02 — THE LONG-CONTEXT ILLUSION

Why "just paste everything" fails

The first reflex when context matters is to make the window bigger. Frontier models now advertise windows of hundreds of thousands — even millions — of tokens, and a tempting conclusion follows: if everything fits, retrieval is obsolete. The evidence says otherwise. Long windows are necessary but not sufficient, because models do not use long contexts uniformly.

Two failure modes are now well documented. The first is positional: Liu et al. showed that performance on multi-document tasks follows a U-shaped curve — accuracy is highest when relevant information

sits at the very beginning or end of the input and drops by more than thirty percent when the same information is buried in the middle[4]. The phenomenon — "lost in the middle" — replicates across model families. The second is cumulative: Chroma's *context rot* study demonstrates that as raw input grows, performance degrades non-uniformly and often well before the advertised limit, as attention is diluted across more tokens and the model is more easily lured by coherent-but-irrelevant material[5].

>30%

Accuracy drop when relevant facts sit mid-context rather than at the edges ("lost in the middle")[4].

~32k

Token range beyond which many frontier models fall sharply below their short-context accuracy in independent testing[5].

NIAH

"Needle in a haystack" measures lexical recall only — it overstates true long-context reasoning[5].

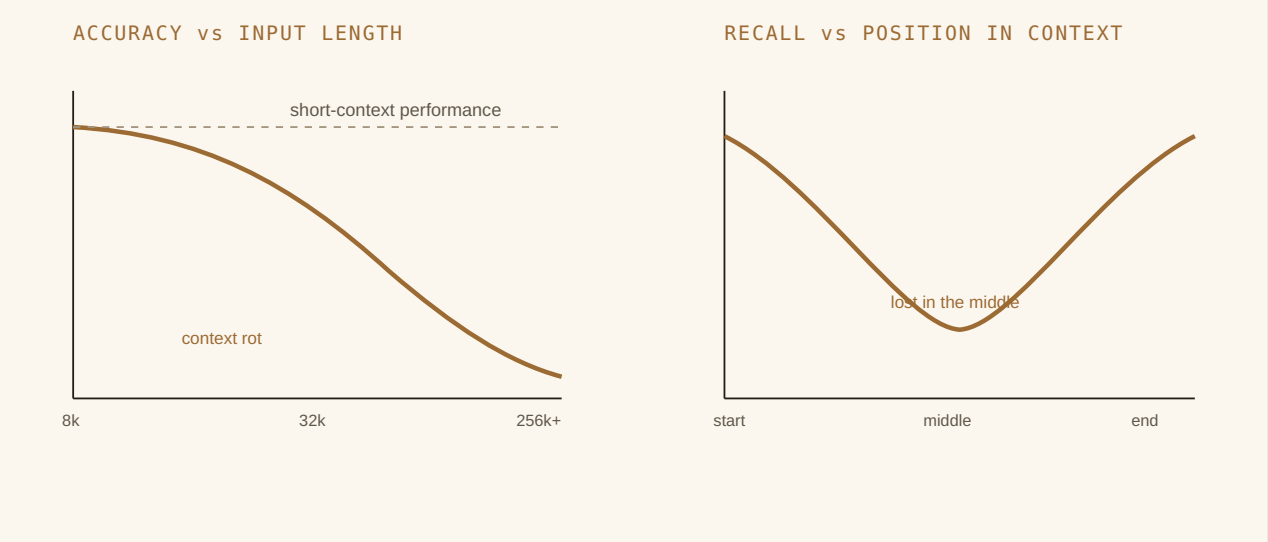

Fig. 1 — Two empirical limits of long context. Left: aggregate accuracy decays as input grows ("context rot")[5]. Right: recall depends on *where* a fact sits in the window[4]. Schematic, after the cited studies.

The operational consequence is decisive. A bigger window does not remove the need to choose what goes into it — it raises the stakes of choosing badly. Retrieval is not a workaround for small windows; it is the mechanism by which we keep the *signal-to-token ratio* high regardless of window size. Context architecture begins here.

03 — THE RETRIEVAL SUBSTRATE, DONE RIGHT

Hybrid, contextual, reranked, late-interaction

If retrieval is permanent, it must be excellent. The naive baseline — dense vectors over fixed-size chunks — leaves a great deal of accuracy on the table. Four upgrades now constitute the production standard.

Hybrid retrieval and rank fusion

Dense embeddings capture semantic similarity but miss exact terms — identifiers, error codes, names, rare tokens — that sparse lexical methods such as BM25 catch reliably. The robust default is to run

both and merge their results with Reciprocal Rank Fusion, then deduplicate. Hybrid retrieval is now the recommended starting point for essentially all production systems[6].

Contextual retrieval

Chunking destroys context: a paragraph that says "the margin fell to 3.2%" loses the entity and period it refers to. Anthropic's *contextual retrieval* prepends a short, model-generated description situating each chunk in its document before indexing — to both the embedding and the BM25 index. The measured effect is large: contextual embeddings cut top-20 retrieval failures by 35%; adding contextual BM25 reaches 49%; adding a reranker reaches 67%[6].

49%

Fewer failed retrievals with contextual embeddings + contextual BM25 vs. a standard baseline[6].

67%

Fewer failed retrievals once a reranking stage is added on top[6].

50–100

Candidates to retrieve before reranking with a cross-encoder, a common production funnel.

Reranking

First-stage retrieval optimises for recall; it returns many plausible candidates. A cross-encoder reranker — which reads query and passage jointly rather than comparing pre-computed vectors — then reorders them for precision. The pattern is to over-retrieve (50–100 candidates) and rerank down to the handful that actually enter the prompt, trading a small latency cost for a substantial precision gain. The trade-off between how many candidates to rerank and the added latency is the central tuning knob[6].

Late interaction

Between single-vector dense retrieval and full cross-encoding sits *late interaction*: ColBERT represents query and document as *sets* of token-level vectors and scores them with a fine-grained MaxSim operator, capturing nuance a single pooled vector loses[7]. Once academic, late-interaction models are now firmly in production, with tooling (RAGatouille, PyLate) and millions of monthly downloads; ColPali extends the idea to documents-as-images, retrieving over rendered pages — tables, figures, layout — without a brittle OCR pipeline[8]. For visually complex corpora this is a step change.

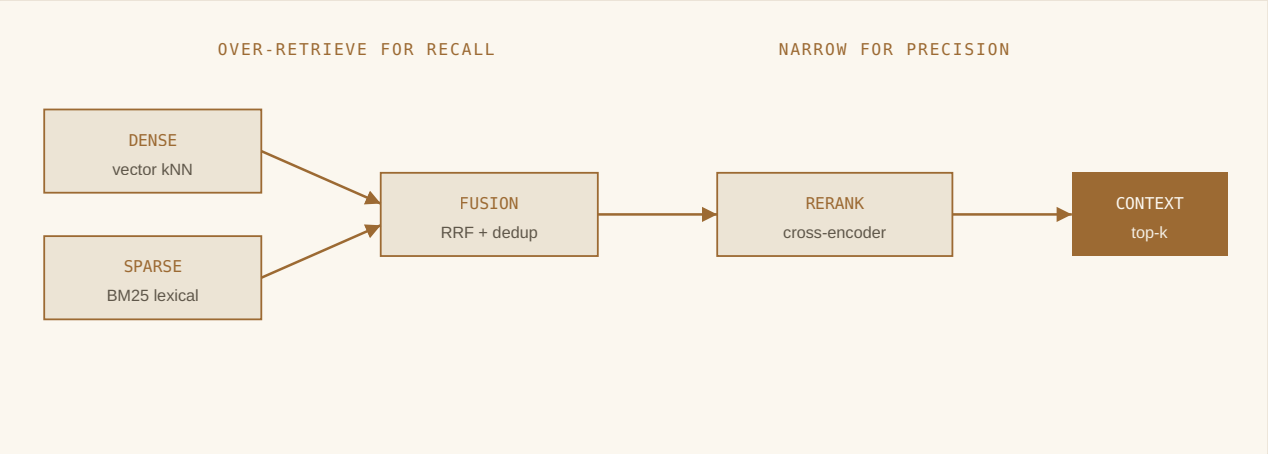

Fig. 2 — The production retrieval funnel: hybrid first-stage recall (dense + sparse), reciprocal-rank fusion, then cross-encoder reranking to a small, high-precision top-k. Contextual indexing[6] and late interaction[7][8] upgrade individual stages.

04 — FROM PIPELINES TO AGENTIC RETRIEVAL

Retrieval as a control loop

Even an excellent funnel runs once. The deeper shift is to let the agent *control* retrieval: decide whether to retrieve at all, judge whether what came back is sufficient, and act to fix it if not. Four named patterns from the literature now compose most production systems.

- → **Self-RAG** trains the model to emit reflection tokens — deciding on demand when to retrieve and critiquing whether its own draft is supported by the evidence[9].
- → **Corrective RAG (CRAG)** adds a lightweight retrieval evaluator that grades the relevance of retrieved documents and, on a low score, triggers a corrective action — query rewriting or a web fallback — rather than answering from poor context[10].
- → **FLARE** retrieves *actively*, generating forward and pausing to fetch evidence whenever its next-token confidence drops — retrieval driven by the model's own uncertainty[11].
- → **Adaptive-RAG** routes by query difficulty: a small classifier sends easy questions straight to generation and reserves the expensive multi-step loop for genuinely hard ones — controlling cost, not just accuracy[12].

The unifying picture is a loop — *route → retrieve → grade → act → generate → reflect* — in which retrieval is a tool the agent invokes under a policy, not a fixed prelude. The accuracy gains on the queries that matter most (multi-hop, compositional, ambiguous) are large relative to single-shot RAG, precisely because the agent can recover from a bad first retrieval instead of confidently answering from it[9][10][12].

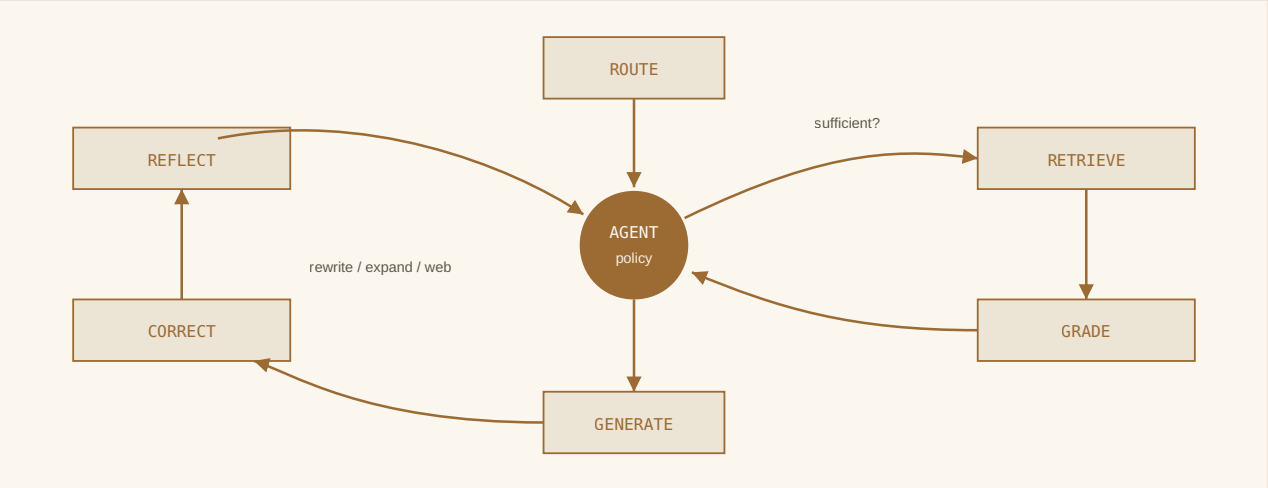

Fig. 3 — Agentic retrieval as a control loop. The agent decides whether to retrieve, grades sufficiency, and takes corrective action before generating — the shared structure behind Self-RAG[9], CRAG[10], FLARE[11] and Adaptive-RAG[12].

Graph-structured retrieval for multi-hop reasoning

Some questions are not about finding a passage but about traversing relationships — "what did our top three competitors do last quarter that affected our European customers." Flat chunk retrieval cannot compose that answer. GraphRAG builds a knowledge graph of entities and relations during ingestion and retrieves structured subgraphs, enabling genuine multi-hop reasoning and global summarisation; Microsoft Research reports comprehensiveness gains of roughly 50–70% over vector RAG on complex summarisation tasks[13]. The mature pattern is not graph *instead of* vectors but graph *alongside* them: hybrid retrieval as the default, with selective graph enrichment for known multi-hop query classes.

05 — THE CONTEXT BUDGET

Managing the window as a memory hierarchy

Retrieval decides what *could* enter the window. The context budget decides what *actually* does, turn after turn, as an agent runs for tens or hundreds of steps. This is where most production agents quietly fail: not on a single answer, but on the fortieth, when the window has filled with tool transcripts and the model loses the thread. Treating the window as a managed memory hierarchy — not a bucket — is the difference.

Three techniques carry most of the load. **Context editing** prunes low-signal content (stale tool output, superseded drafts) by rule before it accumulates; in Anthropic's hundred-turn evaluation, context editing cut token consumption by 84% and allowed workflows to complete that would otherwise have exhausted the window[3]. **Compaction** periodically summarises history into a compact state object, preserving decisions and discarding transcript. **Sub-agent isolation** gives each sub-task its own clean window and returns only a condensed summary — typically one to two thousand tokens — to the orchestrator, so the lead agent's context never absorbs the full working state of its delegates[3].

> DESIGN PRINCIPLE
>
> Token budget is managed *before* content enters the window, not after. Every token spent on low-signal material is paid twice — once in cost, and again in the accuracy lost to context rot.

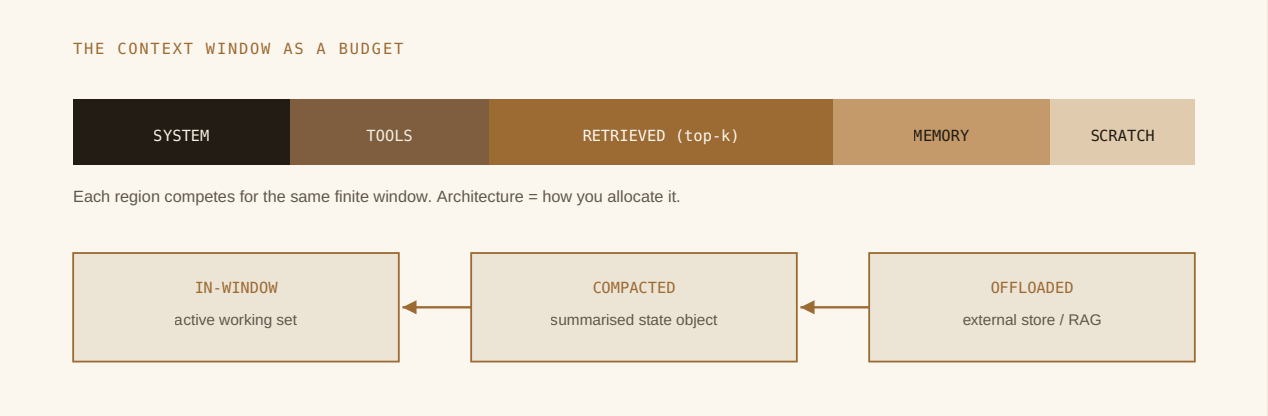

Fig. 4 — The window as a memory hierarchy. Active working set stays in-window; older state is compacted into summaries; the rest is offloaded to external stores and retrieved on demand — the agentic analogue of paging[3][16].

RAG, CAG, or long context? A decision, not a religion

Not every problem needs retrieval. When a knowledge base is small, stable, and fits the window, *cache-augmented generation* (CAG) preloads the whole corpus and precomputes its key-value cache once, eliminating per-query retrieval latency and retrieval errors entirely[14]. The engineering judgement is to match mechanism to corpus:

APPROACH	BEST WHEN	WATCH-OUTS
RAG / Agentic RAG	Large, changing, or multi-tenant corpora; need for citations and freshness	Retrieval quality is now your accuracy ceiling — invest in the funnel
CAG (cached context)	Small, stable corpus that fits the window; latency-critical	Re-cache on change; bounded by window size[14]
Long-context (no retrieval)	One-off documents; exploratory reading	Context rot & lost-in-the-middle still apply[4][5]
GraphRAG (alongside)	Multi-hop, relationship & global-summary questions	Graph construction cost; use selectively[13]

06 — MEASURING WHAT MATTERS

Evaluation as the control system

None of the above is safe to deploy on intuition. Context architecture is an empirical discipline, and its instrument is evaluation. The community has converged on a small, decomposable metric set —

operationalised by RAGAS and equivalents — that separates *retrieval* quality from *generation* quality[15]:

- → **Context precision & recall** — are the retrieved chunks relevant, and do they actually contain the answer? These isolate the retriever from the generator.
- → **Faithfulness** — decomposed as the share of the answer's atomic claims that are directly supported by the retrieved context. The front-line metric against hallucination[15].
- → **Answer relevance** — does the response actually address the question asked?

Alongside these, classic ranking metrics — Precision@k and Recall@k — track the funnel directly; reasonable starting targets for narrow-domain systems are Precision@5 ≥ 0.7 and Recall@20 ≥ 0.8. The point is not the specific numbers but the loop: instrument retrieval and generation separately, set targets, and let measured faithfulness and context precision — not vibes — drive every change to chunking, indexing, reranking and prompting. A system you cannot measure, you cannot improve; a system you measure separates a retrieval bug from a generation bug in minutes instead of weeks.

> WHY THIS MATTERS FOR TRUST
>
> Faithfulness scoring gives a defensible, auditable answer to the question every regulated buyer asks: "how do you know the model isn't making this up?" The answer is a number, tracked over time — not a promise.

07 — A REFERENCE ARCHITECTURE

The context layer, as LinkTec Labs builds it

The techniques above are not a menu to pick from at random; they compose into a layer. The architecture below is the one we deploy for organisations moving from proof-of-concept to production — deliberately sized for real, mixed, mid-scale corpora rather than for leaderboard conditions.

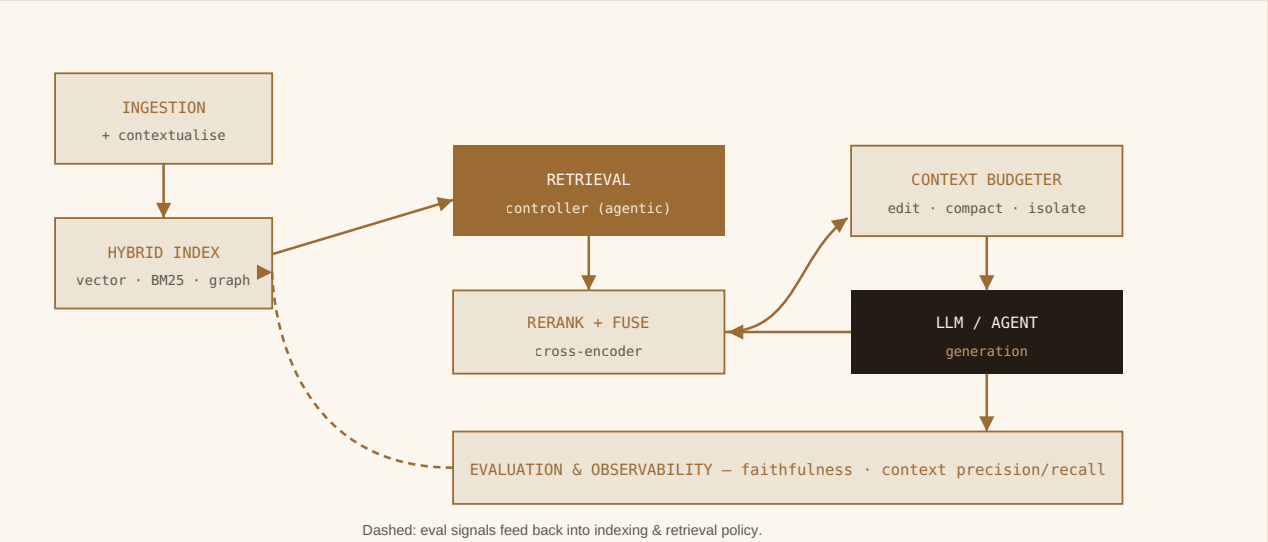

Fig. 5 — LinkTec Labs reference context architecture. Contextualised ingestion feeds a hybrid (vector + BM25 + selective graph) index; an agentic controller governs retrieval, reranking and fusion; a context budgeter manages the window; and an evaluation layer closes the loop. Sovereign-deployable, with full audit trace.

Four principles govern it. **Retrieval quality is the product** — contextual indexing and reranking are not optional polish but the accuracy floor. **The controller is explicit** — retrieval is a governed tool with routing and grading, never a blind prelude. **The budget is engineered** — the window is allocated, edited and compacted on a policy, not left to fill. And **everything is measured** — every change is justified by a movement in faithfulness or context precision, with a full, exportable audit trace from query to answer. That last property is what makes the architecture deployable in regulated, sovereignty-sensitive environments — the subject of a forthcoming note in this series.

08 — PRACTITIONER'S PLAYBOOK

What to do on Monday

- → **Stop pasting; start curating.** A larger window is not a retrieval strategy. Keep the signal-to-token ratio high at every step[4][5].
- → **Make hybrid the default.** Dense + BM25 + RRF before anything exotic; it is the highest-return first move[6].
- → **Contextualise chunks before indexing.** A one-line situating description per chunk is among the cheapest large accuracy gains available[6].
- → **Over-retrieve, then rerank.** 50–100 candidates into a cross-encoder, down to a precise top-k[6].
- → **Give the agent a retrieval policy.** Route by difficulty, grade sufficiency, correct on failure — don't answer from bad context[9][10][12].
- → **Budget the window like memory.** Edit, compact, and isolate sub-agents; treat tokens as a managed resource[3].

→ **Match mechanism to corpus.** RAG, CAG, long-context and graph are tools, not allegiances[13] [14].

→ **Instrument before you optimise.** Faithfulness and context precision/recall, tracked over time, or you are guessing[15].

The model is a commodity. The context around it is the architecture — and the architecture is where the advantage lives.

LinkTec Labs designs, builds and measures context architectures for organisations moving from proof-of-concept to production — sovereign-deployable, audit-ready, and sized for real data. This is the first in our 2026 research series. **linktec.fr / labs**

REFERENCES

Primary & technical sources

[1] Lewis et al. **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.** NeurIPS 2020. arXiv: 2005.11401 — https://arxiv.org/abs/2005.11401

[2] LangChain. **Context Engineering for Agents** (incl. A. Karpathy's "LLM as OS / context as RAM" framing). 2025 — https://www.langchain.com/blog/context-engineering-for-agents

[3] Anthropic. **Effective Context Engineering for AI Agents.** Anthropic Engineering, 2025 — https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents

[4] Liu et al. **Lost in the Middle: How Language Models Use Long Contexts.** TACL 2024. arXiv:2307.03172 — https://arxiv.org/abs/2307.03172

[5] Chroma Research. **Context Rot: How Increasing Input Tokens Impacts LLM Performance.** 2025 — https://www.trychroma.com/research/context-rot

[6] Anthropic. **Introducing Contextual Retrieval.** 2024 — https://www.anthropic.com/news/contextual-retrieval

[7] Khattab & Zaharia. **ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT.** SIGIR 2020. arXiv:2004.12832 — https://arxiv.org/abs/2004.12832

[8] Faysse et al. **ColPali: Efficient Document Retrieval with Vision Language Models.** ICLR 2025. arXiv:2407.01449 — https://arxiv.org/abs/2407.01449

[9] Asai et al. **Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection.** ICLR 2024. arXiv: 2310.11511 — https://arxiv.org/abs/2310.11511

[10] Yan et al. **Corrective Retrieval-Augmented Generation (CRAG).** 2024. arXiv:2401.15884 — https://arxiv.org/abs/2401.15884

[11] Jiang et al. **Active Retrieval-Augmented Generation (FLARE).** EMNLP 2023. arXiv:2305.06983 — https://arxiv.org/abs/2305.06983

[12] Jeong et al. **Adaptive-RAG: Learning to Adapt Retrieval-Augmented LLMs through Question Complexity.** NAACL 2024. arXiv:2403.14403 — https://arxiv.org/abs/2403.14403

[13] Edge et al. **From Local to Global: A Graph RAG Approach to Query-Focused Summarization.** Microsoft Research, 2024. arXiv:2404.16130 — https://arxiv.org/abs/2404.16130

[14] Chan et al. **Don't Do RAG: When Cache-Augmented Generation is All You Need.** 2024. arXiv:2412.15605 — https://arxiv.org/abs/2412.15605

[15] Es et al. **RAGAS: Automated Evaluation of Retrieval-Augmented Generation.** 2023. arXiv:2309.15217 — https://arxiv.org/abs/2309.15217

[16] Packer et al. **MemGPT: Towards LLMs as Operating Systems.** 2023. arXiv:2310.08560 — https://arxiv.org/abs/2310.08560