**LinkTec** LABS

NOTE DE RECHERCHE N°01 · IA & DONNÉES APPLIQUÉES

# Au-delà du RAG : l'essor de **l'architecture de contexte**

*Concevoir la couche de contexte des agents autonomes.*

RÉSUMÉ

Le RAG (Retrieval-Augmented Generation) a rendu les grands modèles de langage utiles sur des données privées. Mais le pipeline en une passe — embed, search, stuff, generate — a été conçu pour une question humaine unique, pas pour des agents autonomes qui émettent des milliers d'appels et accumulent un état. À mesure que les agents passent en production, le verrou n'est plus le modèle ni l'index vectoriel ; c'est la *couche de contexte* : ce qui atteint le modèle, quand, sous quelle forme, et à quel coût. Cette note repositionne le RAG comme un composant d'une discipline plus large — l'*architecture de contexte* — et passe en revue les techniques qui définissent désormais l'état de l'art : retrieval hybride et contextuel, modèles à late interaction, boucles de retrieval agentique (auto-correctives), raisonnement structuré par graphe, gestion du budget de contexte, et les méthodes d'évaluation qui maintiennent l'ensemble honnête. Nous terminons par l'architecture de référence que LinkTec Labs déploie pour les organisations qui passent du POC à la production.

01 — LA BASCULE

## Du prompt malin au contexte conçu

Le RAG — Retrieval-Augmented Generation[1] — fut l'idée qui a débloqué l'adoption des modèles de langage en entreprise : plutôt que de fine-tuner un modèle sur des données privées, on récupère les passages pertinents au moment de la requête et on laisse le modèle les lire. Cela a fonctionné. Cela a aussi, discrètement, posé un plafond. Le pipeline canonique — découper le corpus, l'embedder, lancer une recherche du plus proche voisin sur la question de l'utilisateur, concaténer les meilleurs résultats, puis générer — suppose une question humaine unique et bien formée, et une seule passe. Cette supposition s'effondre dès que l'appelant est un agent.

Les agents autonomes ne posent pas une question. Ils planifient, appellent des outils, lisent les résultats, révisent, et rappellent — émettant des ordres de grandeur plus de requêtes qu'un humain, et transportant entre chaque étape un état de travail qui ne cesse de croître. La ressource rare se déplace de la *qualité du modèle* vers la *qualité du contexte* : la fenêtre limitée doit être remplie, à chaque étape, avec exactement l'information et les outils que la décision suivante exige — et rien d'autre. Andrej Karpathy l'a résumé : le LLM est un nouveau type de système d'exploitation, et la fenêtre de contexte en est la RAM[2]. Le travail de l'ingénieur, c'est la gestion de mémoire.

C'est pourquoi le champ est passé, en l'espace de deux ans, du *prompt engineering* à ce qu'on appelle désormais le *context engineering* — et, à l'échelle de la production, à l'*architecture de contexte* : la conception systématique des pipelines, index, contrôleurs et budgets qui décident de ce que voit le modèle. Anthropic a publié des recommandations explicites sur la pratique[3] ; les analystes la décrivent désormais comme la discipline qui succède au prompt engineering. La thèse de cette note est simple et, nous le croyons, lourde de conséquences :

> Le verrou de l'IA appliquée n'est plus le modèle. C'est l'architecture du contexte qui l'entoure.

Pour un éditeur de modèles, c'est une note de bas de page. Pour une organisation qui tente de mettre des agents en production sur ses propres données, souvent désordonnées, c'est tout l'enjeu — et il est constructible, mesurable, et appropriable. La suite de cette note en dresse le terrain.

02 — L'ILLUSION DU LONG CONTEXTE

## Pourquoi « tout coller » échoue

Le premier réflexe, quand le contexte compte, est d'agrandir la fenêtre. Les modèles de pointe affichent désormais des fenêtres de centaines de milliers — voire de millions — de tokens, et une conclusion tentante s'ensuit : si tout tient, le retrieval est obsolète. Les faits disent le contraire. Les longues

fenêtres sont nécessaires mais pas suffisantes, car les modèles n'exploitent pas un long contexte de façon uniforme.

Deux modes de défaillance sont aujourd'hui bien documentés. Le premier est positionnel : Liu et al. ont montré que la performance sur les tâches multi-documents suit une courbe en U — la précision est maximale lorsque l'information pertinente se trouve tout au début ou tout à la fin de l'entrée, et chute de plus de trente pour cent lorsque cette même information est enfouie au milieu[4]. Le phénomène — « lost in the middle » — se reproduit d'une famille de modèles à l'autre. Le second est cumulatif : l'étude *context rot* de Chroma démontre qu'à mesure que l'entrée brute grossit, la performance se dégrade de façon non uniforme, souvent bien avant la limite annoncée, l'attention se diluant sur davantage de tokens et le modèle étant plus facilement attiré par du matériel cohérent mais hors-sujet[5].

**>30 %**

Chute de précision quand les faits pertinents sont au milieu plutôt qu'aux extrémités (« lost in the middle »)[4].

**~32k**

Seuil de tokens au-delà duquel de nombreux modèles de pointe tombent nettement sous leur précision court-contexte, en tests indépendants[5].

**NIAH**

Le « needle in a haystack » ne mesure que le rappel lexical — il surestime le vrai raisonnement long-contexte[5].

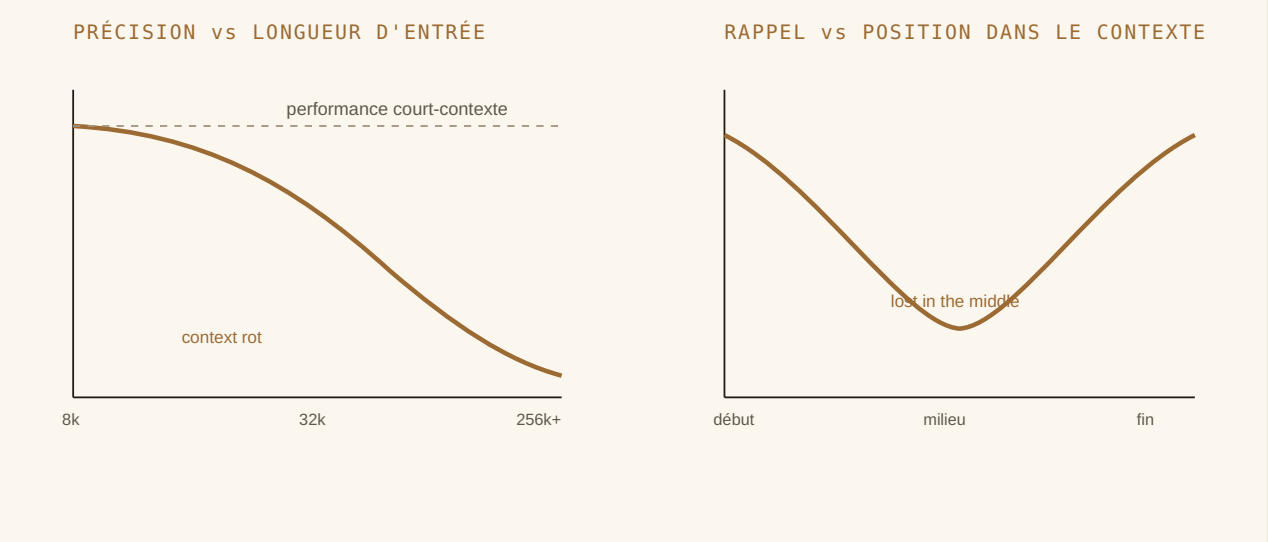


Fig. 1 — Deux limites empiriques du long contexte. À gauche : la précision agrégée décline à mesure que l'entrée grossit (« context rot »)[5]. À droite : le rappel dépend de *l'endroit* où se situe un fait dans la fenêtre[4]. Schéma, d'après les études citées.

La conséquence opérationnelle est décisive. Une fenêtre plus grande ne supprime pas le besoin de choisir ce qu'on y met — elle augmente le coût d'un mauvais choix. Le retrieval n'est pas un palliatif aux petites fenêtres ; c'est le mécanisme par lequel on maintient un *ratio signal/token* élevé, quelle que soit la taille de la fenêtre. L'architecture de contexte commence ici.

03 — LE SUBSTRAT DE RETRIEVAL, BIEN FAIT

# Hybride, contextuel, reranké, late-interaction

Si le retrieval est permanent, il doit être excellent. La base naïve — vecteurs denses sur des chunks de taille fixe — laisse énormément de précision sur la table. Quatre améliorations constituent désormais le standard de production.

### Retrieval hybride et rank fusion

Les embeddings denses capturent la similarité sémantique mais ratent les termes exacts — identifiants, codes d'erreur, noms propres, tokens rares — que les méthodes lexicales (sparse) comme BM25 attrapent de façon fiable. Le réglage robuste par défaut est d'exécuter les deux et de fusionner leurs résultats par Reciprocal Rank Fusion, puis de dédupliquer. Le retrieval hybride est aujourd'hui le point de départ recommandé pour pratiquement tout système de production[6].

### Contextual retrieval

Le chunking détruit le contexte : un paragraphe qui dit « la marge est tombée à 3,2 % » perd l'entité et la période auxquelles il se réfère. Le *contextual retrieval* d'Anthropic préfixe à chaque chunk une courte description, générée par le modèle, qui le situe dans son document, avant l'indexation — à la fois pour l'embedding et pour l'index BM25. L'effet mesuré est important : les contextual embeddings réduisent les échecs de retrieval (top-20) de 35 % ; l'ajout de contextual BM25 atteint 49 % ; l'ajout d'un reranker atteint 67 %[6].

**49 %**

D'échecs de retrieval en moins avec contextual embeddings + contextual BM25 vs. une base standard[6].

**67 %**

D'échecs de retrieval en moins une fois une étape de reranking ajoutée[6].

**50–100**

Candidats à récupérer avant reranking par cross-encoder — un entonnoir de production courant.

### Reranking

Le retrieval de premier étage optimise le rappel ; il renvoie de nombreux candidats plausibles. Un reranker de type cross-encoder — qui lit conjointement la requête et le passage, plutôt que de comparer des vecteurs pré-calculés — les réordonne ensuite pour la précision. Le motif consiste à sur-récupérer (50–100 candidats) puis à reranker jusqu'à la poignée qui entre réellement dans le prompt, échangeant un faible surcoût de latence contre un gain de précision substantiel. L'arbitrage entre le nombre de candidats rerankés et la latence ajoutée est le bouton de réglage central[6].

### Late interaction

Entre le dense retrieval à vecteur unique et le cross-encoding complet se trouve la *late interaction* : ColBERT représente requête et document comme des *ensembles* de vecteurs au niveau token et les score via un opérateur MaxSim fin, capturant une nuance qu'un vecteur agrégé unique perd[7]. Jadis académiques, les modèles à late interaction sont aujourd'hui solidement en production, avec un outillage (RAGatouille, PyLate) et des millions de téléchargements mensuels ; ColPali étend l'idée aux

documents-images, en récupérant sur des pages rendues — tableaux, figures, mise en page — sans pipeline OCR fragile[8]. Pour les corpus visuellement complexes, c'est un saut qualitatif.

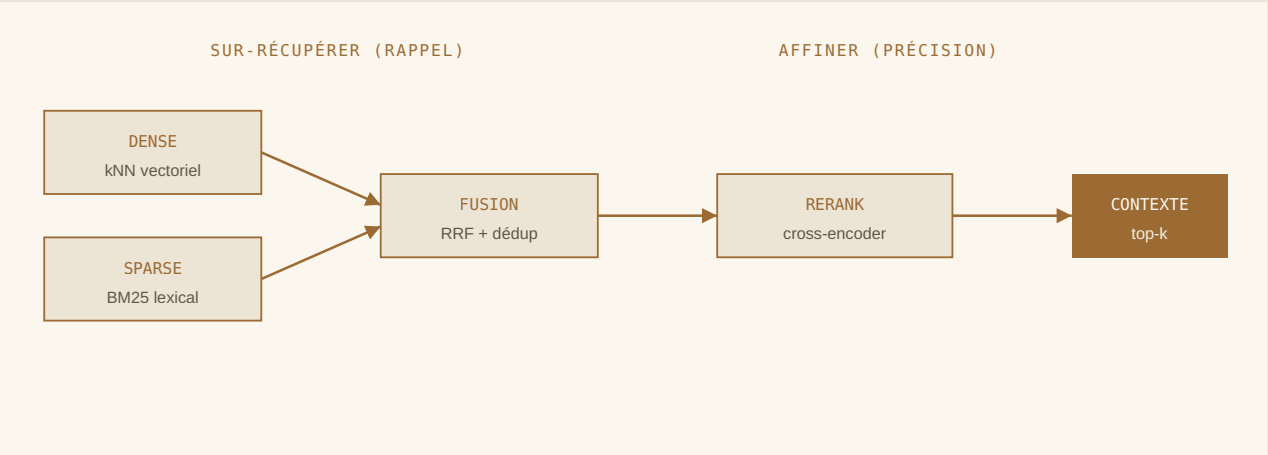


Fig. 2 — L'entonnoir de retrieval en production : rappel hybride au premier étage (dense + sparse), fusion par rang réciproque (RRF), puis reranking par cross-encoder vers un top-k réduit et précis. L'indexation contextuelle[6] et la late interaction[7][8] améliorent chaque étage.

## 04 — DES PIPELINES AU RETRIEVAL AGENTIQUE

# Le retrieval comme boucle de contrôle

Même un excellent entonnoir ne s'exécute qu'une fois. La bascule plus profonde consiste à laisser l'agent *contrôler* le retrieval : décider s'il faut récupérer, juger si ce qui revient est suffisant, et agir pour corriger sinon. Quatre motifs nommés, issus de la littérature, composent aujourd'hui la plupart des systèmes de production.

- → **Self-RAG** entraîne le modèle à émettre des *reflection tokens* — décidant à la demande quand récupérer, et critiquant si son propre brouillon est étayé par les preuves[9].
- → **Corrective RAG (CRAG)** ajoute un évaluateur léger qui note la pertinence des documents récupérés et, sur un score faible, déclenche une action corrective — réécriture de requête ou repli web — plutôt que de répondre sur un contexte médiocre[10].
- → **FLARE** récupère de façon *active*, en générant vers l'avant et en s'arrêtant pour chercher des preuves dès que sa confiance sur le prochain token baisse — un retrieval piloté par l'incertitude du modèle[11].
- → **Adaptive-RAG** route selon la difficulté : un petit classifieur envoie les questions faciles directement à la génération et réserve la boucle multi-étapes coûteuse aux questions réellement difficiles — pilotant le coût, pas seulement la précision[12].

L'image unificatrice est une boucle — *router → récupérer → noter → agir → générer → réfléchir* — dans laquelle le retrieval est un outil que l'agent invoque sous une politique, et non un préambule figé. Les gains de précision sur les requêtes qui comptent le plus (multi-hop, compositionnelles, ambiguës) sont

importants face au RAG en une passe, précisément parce que l'agent peut se rattraper d'un mauvais premier retrieval au lieu d'y répondre avec assurance[9][10][12].

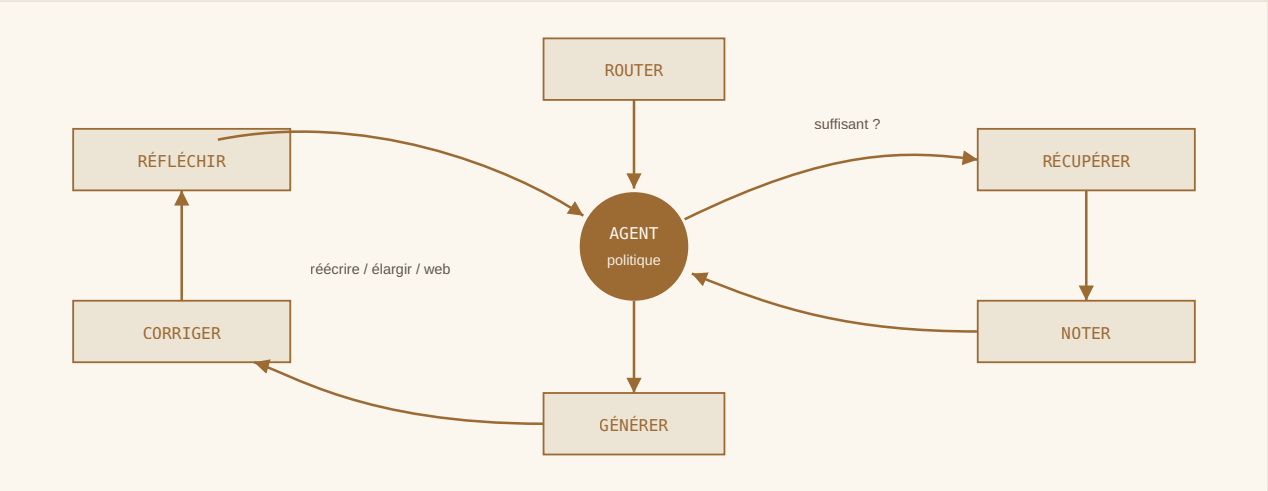


Fig. 3 — Le retrieval agentique comme boucle de contrôle. L'agent décide s'il faut récupérer, note la suffisance, et prend une action corrective avant de générer — la structure commune derrière Self-RAG[9], CRAG[10], FLARE[11] et Adaptive-RAG[12].

### Retrieval structuré par graphe pour le raisonnement multi-hop

Certaines questions ne consistent pas à trouver un passage mais à parcourir des relations — « qu'ont fait nos trois principaux concurrents le trimestre dernier qui ait affecté nos clients européens ». Le retrieval de chunks plats ne peut pas composer cette réponse. GraphRAG construit, à l'ingestion, un graphe de connaissances d'entités et de relations, et récupère des sous-graphes structurés, permettant un véritable raisonnement multi-hop et une synthèse globale ; Microsoft Research rapporte des gains d'exhaustivité de l'ordre de 50 à 70 % par rapport au RAG vectoriel sur des tâches de synthèse complexes[13]. Le motif mûr n'est pas le graphe *à la place* des vecteurs, mais le graphe *aux côtés* des vecteurs : le retrieval hybride par défaut, avec un enrichissement graphe sélectif pour les classes de requêtes multi-hop connues.

## 05 — LE BUDGET DE CONTEXTE

# Gérer la fenêtre comme une hiérarchie mémoire

Le retrieval décide de ce qui *pourrait* entrer dans la fenêtre. Le budget de contexte décide de ce qui y entre *réellement*, tour après tour, à mesure qu'un agent s'exécute sur des dizaines, voire des centaines d'étapes. C'est là que la plupart des agents de production échouent en silence : non pas sur une réponse isolée, mais à la quarantième, quand la fenêtre s'est remplie de transcripts d'outils et que le modèle perd le fil. Traiter la fenêtre comme une hiérarchie mémoire gérée — et non comme un seau — fait toute la différence.

Trois techniques portent l'essentiel de la charge. Le **context editing** élague le contenu à faible signal (sorties d'outils périmées, brouillons remplacés) par règle, avant qu'il ne s'accumule ; dans l'évaluation à cent tours d'Anthropic, le context editing a réduit la consommation de tokens de 84 % et permis à des

workflows d'aboutir là où ils auraient autrement épuisé la fenêtre[3]. La **compaction** résume périodiquement l'historique en un state object compact, préservant les décisions et écartant le transcript. L'**isolation par sous-agents** donne à chaque sous-tâche sa propre fenêtre propre et ne renvoie qu'un résumé condensé — typiquement un à deux mille tokens — à l'orchestrateur, de sorte que le contexte de l'agent principal n'absorbe jamais l'état de travail complet de ses délégués[3].

> PRINCIPE DE CONCEPTION
>
> Le budget de tokens se gère *avant* que le contenu n'entre dans la fenêtre, pas après. Chaque token dépensé en matériel à faible signal est payé deux fois — une fois en coût, et une autre en précision perdue au context rot.

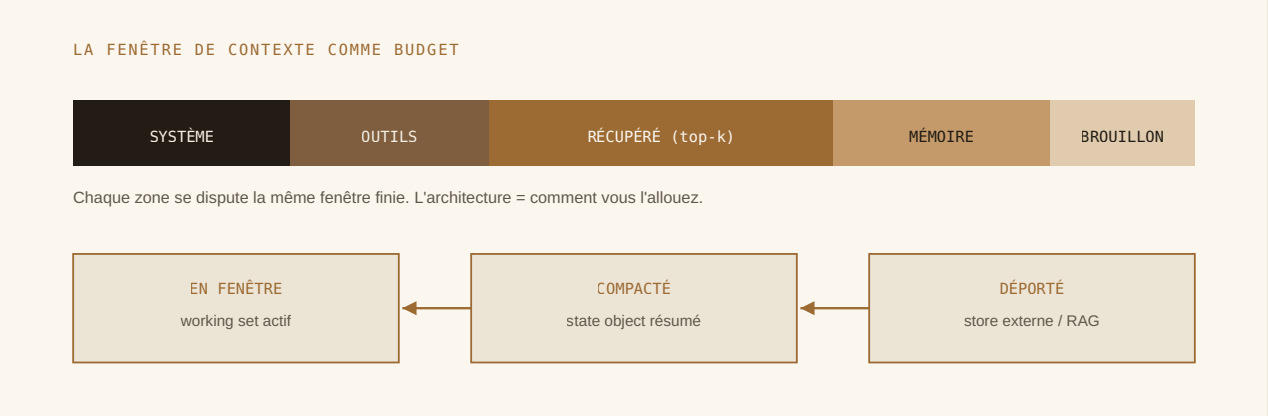


Fig. 4 — La fenêtre comme hiérarchie mémoire. Le working set actif reste en fenêtre ; l'état plus ancien est compacté en résumés ; le reste est déporté vers des stores externes et récupéré à la demande — l'analogue agentique de la pagination[3][16].

## RAG, CAG ou long contexte ? Une décision, pas une religion

Tous les problèmes n'exigent pas de retrieval. Quand une base de connaissances est petite, stable, et tient dans la fenêtre, la *cache-augmented generation* (CAG) précharge l'ensemble du corpus et précalcule son cache clé-valeur une fois, éliminant entièrement la latence de retrieval par requête et les erreurs de retrieval[14]. Le jugement d'ingénierie consiste à apparier le mécanisme au corpus :

| APPROCHE | IDÉALE QUAND | POINTS DE VIGILANCE |
|---|---|---|
| **RAG / RAG agentique** | Corpus volumineux, changeant ou multi-tenant ; besoin de citations et de fraîcheur | La qualité du retrieval devient votre plafond de précision — investissez l'entonnoir |
| **CAG (contexte caché)** | Petit corpus stable tenant dans la fenêtre ; latence critique | Re-cacher à chaque changement ; borné par la taille de la fenêtre [14] |
| **Long contexte (sans retrieval)** | Documents ponctuels ; lecture exploratoire | Context rot & lost-in-the-middle s'appliquent toujours [4][5] |
| **GraphRAG (en complément)** | Questions multi-hop, relationnelles & de synthèse globale | Coût de construction du graphe ; à utiliser de façon sélective [13] |

06 — MESURER CE QUI COMPTE

## L'évaluation comme système de contrôle

Rien de ce qui précède n'est sûr à déployer à l'intuition. L'architecture de contexte est une discipline empirique, et son instrument est l'évaluation. La communauté a convergé vers un petit ensemble de métriques décomposables — opérationnalisé par RAGAS et ses équivalents — qui sépare la qualité du *retrieval* de celle de la *génération* [15] :

- → **Context precision & recall** — les chunks récupérés sont-ils pertinents, et contiennent-ils effectivement la réponse ? Ces métriques isolent le retriever du générateur.
- → **Faithfulness** — décomposée comme la part des affirmations atomiques de la réponse qui sont directement étayées par le contexte récupéré. La métrique de première ligne contre l'hallucination [15].
- → **Answer relevance** — la réponse traite-t-elle réellement la question posée ?

À côté de celles-ci, les métriques de ranking classiques — Precision@k et Recall@k — suivent l'entonnoir directement ; des cibles de départ raisonnables pour des systèmes à domaine étroit sont Precision@5 ≥ 0,7 et Recall@20 ≥ 0,8. L'important n'est pas les chiffres précis mais la boucle : instrumenter retrieval et génération séparément, fixer des cibles, et laisser la faithfulness et la context precision mesurées — pas l'intuition — piloter chaque changement de chunking, d'indexation, de reranking et de prompting. Un système qu'on ne peut pas mesurer, on ne peut pas l'améliorer ; un système qu'on mesure sépare un bug de retrieval d'un bug de génération en minutes plutôt qu'en semaines.

POURQUOI CELA FONDE LA CONFIANCE

Le score de faithfulness apporte une réponse défendable et auditable à la question que pose tout acheteur régulé : « comment savez-vous que le modèle n'invente pas ? ». La réponse est un nombre, suivi dans le temps — pas une promesse.

07 — UNE ARCHITECTURE DE RÉFÉRENCE

## La couche de contexte, telle que LinkTec Labs la construit

Les techniques ci-dessus ne sont pas un menu où piocher au hasard ; elles se composent en une couche. L'architecture ci-dessous est celle que nous déployons pour les organisations qui passent du POC à la production — délibérément dimensionnée pour des corpus réels, mixtes et de taille moyenne, plutôt que pour des conditions de leaderboard.

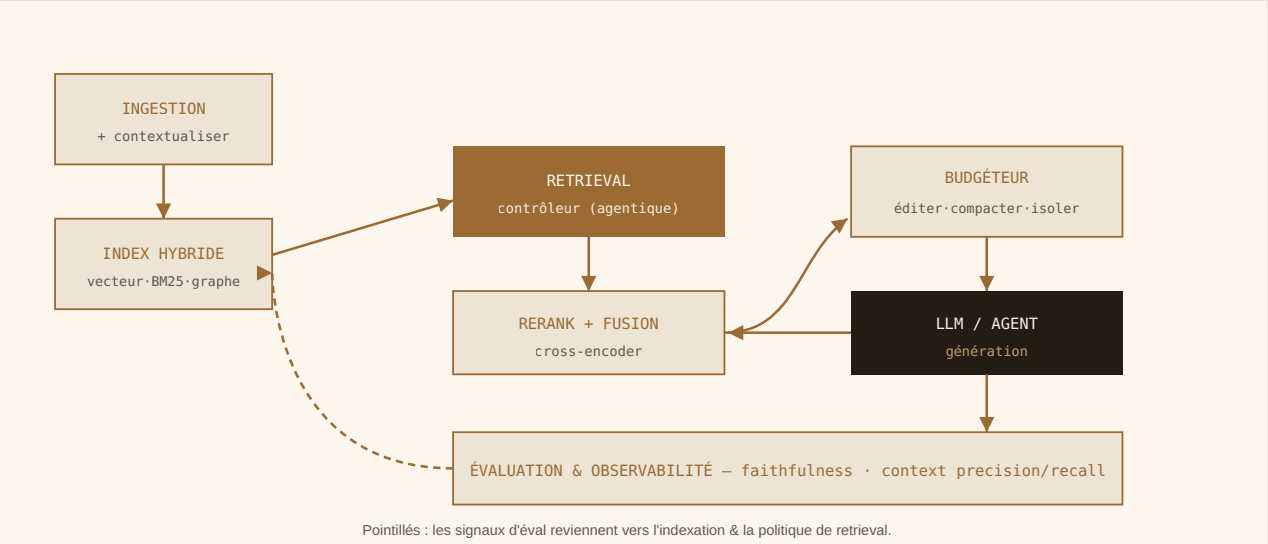


Fig. 5 — Architecture de contexte de référence LinkTec Labs. Une ingestion contextualisée alimente un index hybride (vecteur + BM25 + graphe sélectif) ; un contrôleur agentique gouverne retrieval, reranking et fusion ; un budgéteur de contexte gère la fenêtre ; et une couche d'évaluation boucle l'ensemble. Déployable en souveraineté, avec trace d'audit complète.

Quatre principes la gouvernent. **La qualité du retrieval est le produit** — l'indexation contextuelle et le reranking ne sont pas un vernis optionnel mais le plancher de précision. **Le contrôleur est explicite** — le retrieval est un outil gouverné, avec routage et notation, jamais un préambule aveugle. **Le budget est conçu** — la fenêtre est allouée, éditée et compactée selon une politique, et non laissée se remplir. Et **tout est mesuré** — chaque changement est justifié par une variation de faithfulness ou de context precision, avec une trace d'audit complète et exportable, de la requête à la réponse. Cette dernière propriété est ce qui rend l'architecture déployable dans des environnements régulés et sensibles à la souveraineté — le sujet d'une prochaine note de cette série.

08 — PLAYBOOK DU PRATICIEN

# Quoi faire dès lundi

- → **Arrêter de coller ; commencer à curer.** Une fenêtre plus grande n'est pas une stratégie de retrieval. Gardez un ratio signal/token élevé à chaque étape[4][5].
- → **Faire de l'hybride le défaut.** Dense + BM25 + RRF avant toute exotisme ; c'est le coup d'ouverture au meilleur rendement[6].
- → **Contextualiser les chunks avant l'indexation.** Une ligne de description situant chaque chunk est parmi les gains de précision les moins chers disponibles[6].
- → **Sur-récupérer, puis reranker.** 50–100 candidats dans un cross-encoder, jusqu'à un top-k précis[6].
- → **Donner à l'agent une politique de retrieval.** Router par difficulté, noter la suffisance, corriger sur échec — ne répondez pas sur un mauvais contexte[9][10][12].
- → **Budgéter la fenêtre comme de la mémoire.** Éditer, compacter, isoler les sous-agents ; traiter les tokens comme une ressource gérée[3].
- → **Apparier le mécanisme au corpus.** RAG, CAG, long contexte et graphe sont des outils, pas des allégeances[13][14].
- → **Instrumenter avant d'optimiser.** Faithfulness et context precision/recall, suivies dans le temps, ou vous devinez[15].

---

Le modèle est une commodité. Le contexte qui l'entoure est l'architecture — et l'architecture est là où vit l'avantage.

LinkTec Labs conçoit, construit et mesure des architectures de contexte pour les organisations qui passent du POC à la production — déployables en souveraineté, prêtes pour l'audit, et dimensionnées pour des données réelles. Première note de notre série de recherche 2026. **linktec.fr / labs**

RÉFÉRENCES

# Sources primaires & techniques


[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** (avec le cadre « LLM as OS / context as RAM » d'A. Karpathy). 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