LinkTec LABS

NOTE DE RECHERCHE N°02 · IA & DONNÉES APPLIQUÉES

La mémoire des agents : **se souvenir sans exploser le budget**

État persistant pour les agents longue durée.

RÉSUMÉ

Une fenêtre de contexte plus grande résout la *capacité* au sein d'une session ; elle ne fait rien pour la *continuité* entre les sessions. Or les agents qui atteignent la production s'exécutent sur des centaines d'étapes et reviennent jour après jour — et la réponse naïve, tout garder dans la fenêtre, est à la fois ruineuse et discrètement inexacte. Cette note traite la mémoire comme une discipline d'ingénierie à part entière : un système typé de mémoires de travail, épisodique, sémantique et procédurale, doté d'un cycle de vie explicite — encoder, consolider, récupérer, oublier — gouverné par une politique plutôt que laissé croître. Nous passons en revue les architectures qui définissent l'état de l'art (hiérarchies mémoire façon OS, stores externes par extraction, mémoire fichier et context editing), la mécanique de coûts qui les rend viables, les benchmarks qui les maintiennent honnêtes, et les modes de défaillance — croissance non bornée, faits périmés, vie privée — qui décident de leur déployabilité. Nous terminons par l'architecture mémoire gouvernée et prête pour la souveraineté que construit LinkTec Labs.

01 — CAPACITÉ N'EST PAS CONTINUITÉ

Pourquoi une mémoire, pas seulement une fenêtre plus grande

L'erreur la plus répandue dans la conception d'agents est de traiter la mémoire comme un problème de longueur de contexte. Ce n'en est pas un. Une longue fenêtre de contexte résout la *capacité* — combien un agent peut lire d'un coup. La mémoire résout la *continuité* — ce qu'un agent reporte d'une étape, d'une session, d'une semaine à la suivante. Les deux sont orthogonales, et les confondre produit des agents à la fois coûteux et amnésiques.

Le symptôme est familier à quiconque en a mis un en production. L'agent est brillant sur la première douzaine de tours, puis se dégrade : à la quarantième étape, la fenêtre est saturée de transcripts d'outils et de décisions antérieures, et le modèle perd le fil[8]. Le réflexe — agrandir la fenêtre et tout y reverser — échoue deux fois. C'est coûteux : le coût d'attention croît de façon super-linéaire avec la longueur de séquence, donc chaque token conservé est payé à chaque tour suivant. Et c'est inexact : comme nous l'avons soutenu dans la Note de recherche N°01, les modèles n'exploitent pas un long contexte de façon uniforme, donc une fenêtre plus pleine est souvent *moins* fiable. La continuité achetée par la force brute de la capacité est une continuité qui s'érode précisément quand on en a le plus besoin.

La mémoire est l'alternative : un système délibéré qui décide quoi garder, sous quelle forme, où le placer, et quand le laisser partir — de sorte que la fenêtre de travail reste petite et nette tandis que la *connaissance* qu'a l'agent de sa tâche, de son utilisateur et de son historique persiste à l'extérieur. Le cadre qui s'est stabilisé dans le champ est exactement celui-ci : le long contexte traite la capacité ; la mémoire traite la continuité[6].

TERMES CLÉS

Un vocabulaire commun pour la suite. Les spécialistes peuvent passer.

Fenêtre de contexte	Les tokens qu'un modèle peut traiter en une passe — sa surface de travail, pas sa mémoire.
Mémoire de travail	Le petit état en contexte pour la tâche courante ; analogue à la RAM. Volatile, rapide, rare.
Mémoire à long terme	Information persistée *hors* de la fenêtre et récupérée à la demande ; survit aux sessions.
Mémoire épisodique	Traces d'événements passés précis — quoi, quand, dans quel ordre (le journal de l'agent).
Mémoire sémantique	Faits et préférences stables, dé-contextualisés, distillés de nombreux épisodes.
Mémoire procédurale	Compétences et workflows appris — comment faire les choses, schémas d'usage d'outils.

Consolidation	Compresser des épisodes bruts en mémoire durable de plus haut niveau (réflexion, résumé).
Compaction	Résumer l'historique en fenêtre en un state object compact pour récupérer des tokens.
Récupération (retrieval)	Sélectionner et réinjecter les quelques mémoires pertinentes au moment de la décision.
Oubli / décroissance	Retirer ou dévaluer délibérément les mémoires périmées ou faibles pour borner la croissance.

02 — UNE TAXONOMIE DE TRAVAIL

Quatre mémoires, un agent

Les systèmes de mémoire utiles ne sont pas monolithiques. Le cadre académique canonique, CoALA, adapte les sciences cognitives aux agents de langage et distingue la mémoire *de travail* de trois stores à long terme — *épisodique*, *sémantique* et *procédurale*[2]. La distinction n'est pas pédante : chaque store a une politique d'écriture, un schéma de récupération et un taux de décroissance différents, et les confondre est une cause majeure de mémoires gonflées et peu fiables.

La mémoire épisodique est le journal de l'agent — « mardi, l'utilisateur a rejeté le fournisseur A sur le prix ». La mémoire sémantique est ce que ce journal *distille* — « l'utilisateur est sensible au prix sur l'infrastructure ». La mémoire procédurale est la compétence qui se cristallise par répétition — « pour réconcilier des factures, appeler l'outil X puis Y ». La trajectoire d'un agent qui mûrit est la promotion régulière des épisodes en sémantique et en procédures : l'expérience qui devient expertise[3].

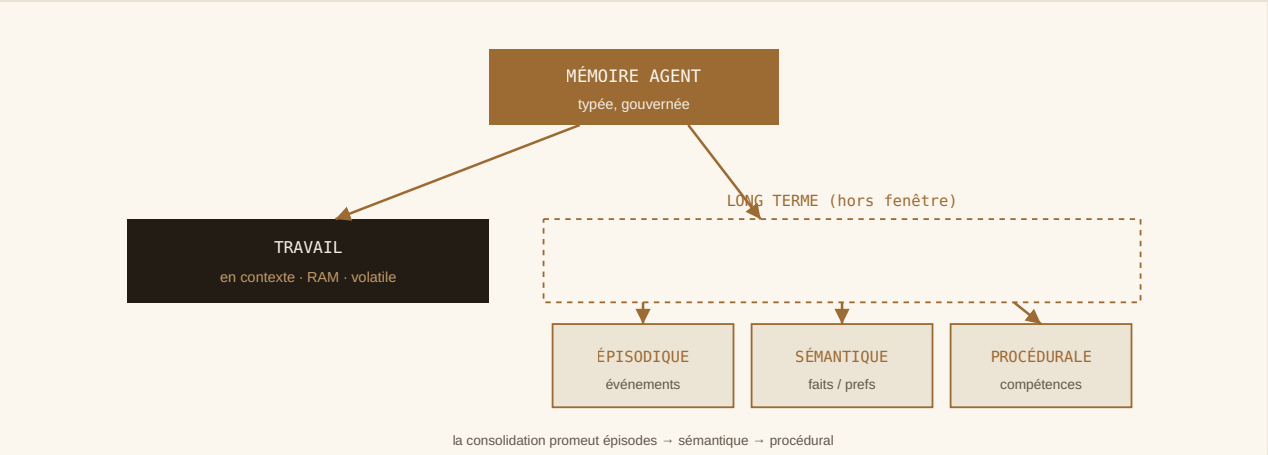

Fig. 1 — Une taxonomie de travail d'après CoALA[2]. La mémoire de travail vit dans la fenêtre ; la mémoire à long terme est typée en épisodique, sémantique et procédurale, chacune avec sa politique d'écriture, de récupération et de décroissance.

03 — LE CYCLE DE VIE DE LA MÉMOIRE

Encoder, consolider, récupérer, oublier

La mémoire est une boucle, pas une base de données. Quatre opérations la définissent, et chacune est une décision de conception avec ses conséquences de coût et de qualité.

Encoder (écrire). Après chaque interaction, l'agent doit décider de ce qui — s'il y a lieu — mérite d'être gardé, et l'extraire en une entrée structurée plutôt qu'un transcript brut. Ce chemin d'écriture est lui-même un coût d'inférence ; les systèmes récents le réduisent drastiquement. L'extraction en une passe, « ADD-only », de Mem0 diminue les appels au modèle à l'écriture de 60–70 % sans perte de qualité notable[10]. **Consolider.** Périodiquement, les épisodes bruts sont synthétisés en insight durable — le mécanisme de « réflexion » popularisé par Generative Agents, qui transforme un flux d'événements en mémoire sémantique de plus haut niveau[3]. **Récupérer (lire).** Au moment de la décision, un petit ensemble de mémoires pertinentes est sélectionné et injecté — par récence, importance et pertinence — gardant la fenêtre de travail légère. **Oublier.** L'opération la moins développée et la plus nécessaire : sans décroissance, les stores croissent sans borne et la qualité de récupération s'effondre sous son propre poids[11].

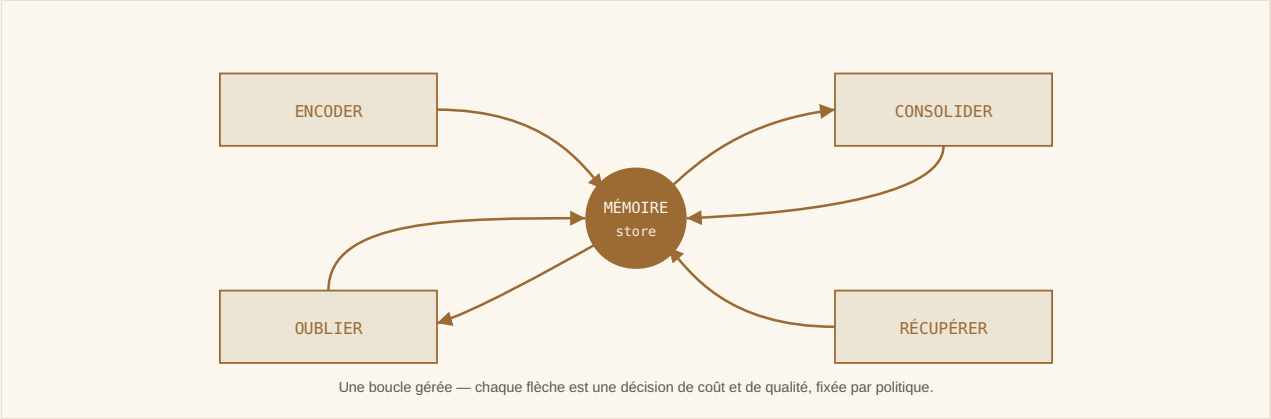

Fig. 2 — Le cycle de vie de la mémoire. L'encodage est une extraction structurée, pas un dépôt de transcript[10] ; la consolidation distille les épisodes en insight[3] ; la récupération garde la fenêtre légère ; l'oubli borne la croissance[11].

04 — PATRONS D'ARCHITECTURE

Trois façons de tenir l'état

Trois architectures dominent la production, et elles sont complémentaires plutôt que concurrentes.

La hiérarchie façon OS (mémoire auto-éditée)

MemGPT — désormais le framework Letta — décrit le LLM comme un processus sur un système d'exploitation à mémoire contrainte, avec une hiérarchie en paliers : mémoire *cœur* dans la fenêtre (RAM), mémoire de *rappel* pour l'historique consultable (un cache disque), et mémoire d'*archives* pour le stockage froid[1]. Surtout, l'agent *édite sa propre mémoire* via des appels d'outils dans sa boucle normale : quand l'utilisateur dit « appelez-moi Alice », le modèle l'écrit lui-même dans son bloc cœur. La mémoire devient une action, pas un effet de bord.

Le store externe par extraction

Mem0 prend le parti inverse : une couche mémoire dédiée qui extrait les faits saillants de chaque échange, les consolide en un store structuré, et en récupère un ensemble compact à la demande — gardant le contexte par appel bien en deçà d'une approche pleine-historique[6]. C'est le patron lorsque la mémoire doit passer à l'échelle sur de nombreux utilisateurs et sessions à coût prévisible.

Mémoire fichier et context editing

La primitive de production la plus récente traite la mémoire comme un système de fichiers que le modèle peut lire et écrire hors de la fenêtre. Le memory tool d'Anthropic persiste l'information entre conversations via des fichiers dans l'infrastructure du développeur, tandis que le *context editing* efface automatiquement les résultats d'outils périmés à mesure que la fenêtre se remplit — ensemble, ils mènent à terme des workflows longs qui épuiseraient autrement le contexte[7].

Fig. 3 — Trois patrons complémentaires : une hiérarchie en paliers façon OS avec blocs auto-édités[1] ; un store externe par extraction[6] ; et une mémoire fichier avec context editing[7].

05 — L'ÉQUATION DE COÛT

Se souvenir sans exploser le budget

La promesse de la mémoire n'est pas seulement la précision ; c'est de faire mieux *et* moins cher que d'entasser l'historique dans le contexte. Les chiffres sont désormais sans appel. Face à une base pleine-contexte, Mem0 rapporte une **latence p95 inférieure de 91 %** et **plus de 90 % d'économies de coût en tokens**, tout en *améliorant* la qualité des réponses de 26 % (LLM-as-judge) par rapport à une mémoire commerciale de premier plan[6]. Sur les grands benchmarks de mémoire longue, la mémoire structurée tient en moyenne sous ~7 000 tokens par récupération là où les approches pleine-contexte en consomment 25 000 ou plus — une réduction de 70 à 85 % du contexte actif[10].

91 %

De latence p95 en moins vs. traiter tout l'historique de conversation[6].

>90 %

D'économies de tokens vs. pleine-contexte, avec une meilleure qualité de réponse[6].

~5×

Moins de compute au test-time à précision égale quand le travail mémoire est fait hors-pic (« sleep-time »)[9].

Le levier le plus profond est le *moment* où le travail a lieu. Les opérations de mémoire — extraction, consolidation, ré-indexation — n'ont pas à se trouver sur le chemin critique de l'utilisateur. Le *sleep-time compute* les déplace vers les périodes d'inactivité : l'agent « réfléchit » hors-ligne, transformant le contexte brut en contexte appris, réduisant le compute au test-time pour une précision égale d'environ 5× et le coût moyen par requête de ~2,5× lorsqu'il est amorti sur des requêtes apparentées[9]. La mémoire, bien faite, n'est pas une taxe à chaque tour ; c'est un investissement réalisé quand le compteur est au plus bas.

STRATÉGIE	CONTINUITÉ	COÛT PAR TOUR	IDÉALE QUAND
Historique complet en contexte	Élevée (jusqu'au context rot)	Croît à chaque tour	Tâches courtes seulement
Compaction / context editing	Moyenne	Bornée[7]	Longues sessions uniques
Store par extraction + retrieval	Élevée, inter-sessions	Faible, plat[6]	Nombreux utilisateurs / sessions
Consolidation sleep-time	Élevée, croissante	Hors chemin critique[9]	Usage récurrent, prévisible

06 — MESURER LA MÉMOIRE

Benchmarks et les capacités qui comptent

Les revendications de mémoire ne coûtent rien ; la mémoire mesurée, si. Trois benchmarks ancrent le champ. LoCoMo évalue la mémoire conversationnelle à très long terme sur des dialogues d'environ 300 tours et jusqu'à 35 sessions[4]. LongMemEval isole cinq capacités distinctes — extraction d'information, raisonnement multi-sessions, raisonnement temporel, mises à jour de connaissances, et abstention — et constate que les assistants commerciaux et les modèles long-contexte subissent une chute de précision d'environ 30 % sur l'interaction prolongée[5]. BEAM pousse à l'extrême, sondant la mémoire sur des conversations allant jusqu'à dix millions de tokens[10].

> LES DEUX CAPACITÉS QUE L'ON SOUS-ESTIME
>
> Les **mises à jour de connaissances** — réécrire un fait quand il change (« elle a changé de poste ») — et l'**abstention** — savoir quand la réponse n'a jamais été stockée. Les systèmes de pur rappel échouent silencieusement aux deux : ils ressortent des faits périmés et fabulent au lieu d'admettre l'ignorance. Ce sont elles, pas le rappel brut, qui séparent un jouet d'un agent digne de confiance[5].

07 — DÉFAILLANCES & GOUVERNANCE

Là où la mémoire devient un passif

La mémoire introduit des modes de défaillance que le retrieval seul n'avait pas — et ce sont précisément ceux que tout acheteur sérieux interroge. **Croissance non bornée :** la réflexion améliore la cohérence mais ne régule pas la taille du store ; sans politique de décroissance explicite, les traces s'accumulent jusqu'à dégrader la récupération[11]. **Mémoire périmée et conflictuelle :** un vieux fait qui aurait dû être réécrit ressurgit en erreur assurée. **Empoisonnement de mémoire :** une seule mauvaise écriture — accidentelle ou adversariale — persiste et contamine chaque session future. Et, décisif pour l'entreprise : la **vie privée.** Une mémoire qui se souvient d'un client se souvient de ses données personnelles, indéfiniment, à travers les sessions.

C'est là que la mémoire cesse d'être une fonctionnalité de performance pour devenir une question de gouvernance. Un store persistant d'informations utilisateur est une donnée régulée ; sous le RGPD européen, il porte un droit de rectification et d'effacement. Une mémoire d'agent qui ne peut être inspectée, corrigée et supprimée sur demande n'est pas prête pour la production en Europe — c'est un passif. L'exigence est concrète : chaque mémoire doit être attribuable à sa source, éditable et oubliable, avec une trace d'audit. Mémoire et souveraineté sont la même conversation — le sujet de la Note de recherche N°03.

> Un agent qui ne peut pas oublier sur demande ne peut pas être déployé sur de vraies données clients. L'oubli n'est pas une lacune — c'est une primitive de conformité.

08 — UNE ARCHITECTURE DE RÉFÉRENCE

Mémoire gouvernée, telle que LinkTec Labs la construit

Les patrons ci-dessus se composent en une couche gouvernée unique. La conception ci-dessous est celle que nous déployons — typée par classe de mémoire, explicite à chaque opération, et auditable de bout en bout.

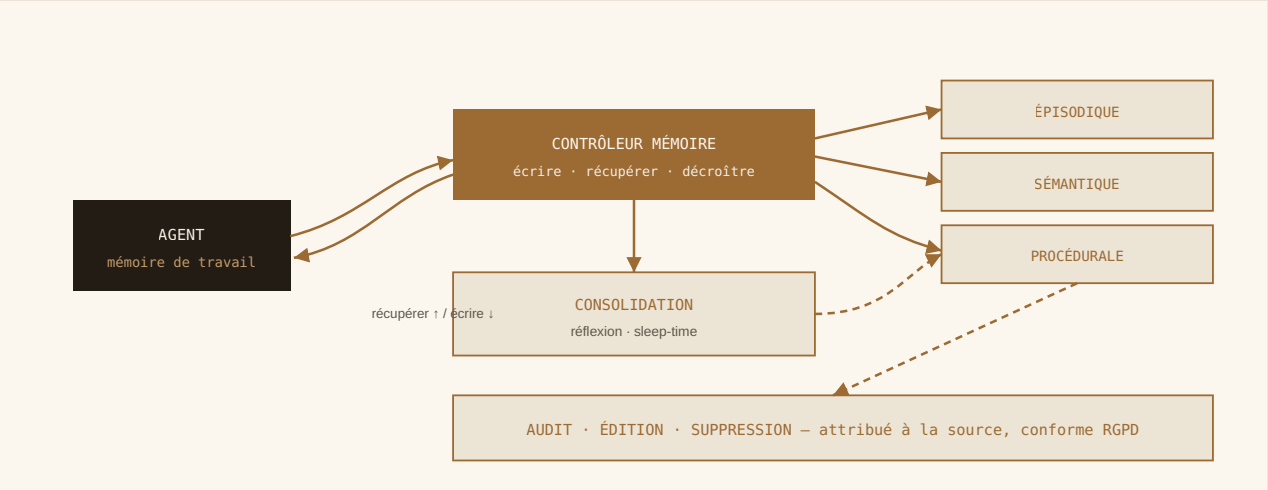

Fig. 4 — Architecture mémoire gouvernée LinkTec Labs. Un contrôleur mémoire arbitre chaque écriture, récupération et décroissance ; la consolidation tourne hors-pic ; les stores sont typés par classe ; et une couche d'audit rend chaque mémoire attribuable, éditable et effaçable. Déployable en souveraineté par conception.

Quatre principes la gouvernent. **La mémoire est typée** — les stores épisodique, sémantique et procédural ont des politiques d'écriture et de décroissance distinctes ; un magma indifférencié est la route vers le gonflement. **Chaque opération est explicite** — écrire, récupérer et oublier sont gouvernés par politique et instrumentés, jamais émergents. **Le budget est la contrainte** — la récupération renvoie un petit working set classé, et la consolidation tourne hors du chemin critique. Et **tout est gouverné** — chaque mémoire est attribuée à sa source, éditable et effaçable, avec une trace d'audit. Les trois premiers achètent la performance ; le quatrième est ce qui rend le système déployable sur de vraies données clients.

09 — PLAYBOOK DU PRATICIEN

Quoi faire dès lundi

- → **Séparer capacité et continuité.** Ne résolvez pas un problème de mémoire par une fenêtre plus grande [6].
- → **Typer votre mémoire.** Travail, épisodique, sémantique, procédurale — chacune avec ses règles d'écriture et de décroissance [2].
- → **Écrire structuré, pas brut.** Extraire les faits saillants ; l'extraction en une passe réduit le coût d'écriture de 60–70 % [10].
- → **Consolider hors-pic.** Réfléchir et ré-indexer en période creuse ; le sleep-time compute se rentabilise [3][9].
- → **Récupérer un petit ensemble.** Classer par récence, importance et pertinence ; garder la fenêtre légère [3].

- → **Faire de l'oubli une fonctionnalité.** Une politique de décroissance borne la croissance ; la suppression explicite satisfait la loi[11].
- → **Tester mises à jour et abstention,** pas seulement le rappel — c'est là que la confiance se gagne ou se perd[5].
- → **Auditer chaque mémoire.** Source, édition, suppression — sinon vous ne pouvez pas livrer sur des données clients.

La capacité permet à un agent de lire. La mémoire lui permet d'apprendre. La discipline consiste à empêcher la seconde de dévorer le budget — ou la confiance.

LinkTec Labs conçoit des couches de mémoire gouvernées et prêtes pour la souveraineté, pour des agents qui tournent longtemps et reviennent souvent — auditables, éditables, et dimensionnées pour le coût réel. Deuxième note de notre série de recherche 2026 ; la première, *Au-delà du RAG : l'architecture de contexte*, en est le compagnon naturel. **linktec.fr / labs**

RÉFÉRENCES

Sources primaires & techniques

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

[2] Sumers, Yao, Narasimhan & Griffiths. **Cognitive Architectures for Language Agents (CoALA).** TMLR 2024. arXiv:2309.02427 — https://arxiv.org/abs/2309.02427

[3] Park et al. **Generative Agents: Interactive Simulacra of Human Behavior.** UIST 2023. arXiv:2304.03442 — https://arxiv.org/abs/2304.03442

[4] Maharana et al. **Evaluating Very Long-Term Conversational Memory of LLM Agents (LoCoMo).** ACL 2024. arXiv:2402.17753 — https://arxiv.org/abs/2402.17753

[5] Wu et al. **LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory.** ICLR 2025. arXiv:2410.10813 — https://arxiv.org/abs/2410.10813

[6] Chhikara et al. **Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory.** 2025. arXiv:2504.19413 — https://arxiv.org/abs/2504.19413

[7] Anthropic. **Managing context on the Claude Developer Platform** (memory tool & context editing). 2025 — https://www.anthropic.com/news/context-management

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

[9] Lin et al. **Sleep-time Compute: Beyond Inference Scaling at Test-time.** 2025. arXiv:2504.13171 — https://arxiv.org/abs/2504.13171

[10] Mem0. **AI Memory Benchmarks 2026 : LoCoMo, LongMemEval & BEAM** (chiffres inter-benchmarks ; source secondaire). 2026 — https://mem0.ai/blog/ai-memory-benchmarks-in-2026

[11] Survey. **Memory for Autonomous LLM Agents: Mechanisms, Evaluation, and Emerging Frontiers.** 2026. arXiv:2603.07670 — https://arxiv.org/abs/2603.07670