iaetbibliotheques.fr /2026/03/bibliotheques-et-agents-ia-le-risque-de-linvisibilisation

Bibliothèques et agents IA : le risque de l'invisibilisation

Géraldine Geoffroy ⋮

To be (in the IA Agent Context) or not to be

L'année 2026 sera l'année des agents IA… C'était annoncé, et effectivement depuis le début de l'année nous assistons à la diffusion et à la montée en puissance de deux grandes familles d'outils agentiques d'un nouveau type : d'une part des assistants orientés coding comme Claude Code, Codex, Gemini CLI, Opencode etc., et d'autre part des frameworks de création, de configuration et d'orchestration d'agents permettant l'automatisation de workflows via des canaux de communication (Slack, Discord, messagerie…) comme OpenClaw et ses multiples dérivés.

L'émergence de ces outils marque une forme de rupture dans la manière dont les interactions avec les LLMs sont architecturées, et pour en comprendre l'importance, revenons brièvement sur l'évolution de l'usage des LLMS :

1) Le chatbot "historique" dans le cadre duquel le modèle reçoit une question et produit une réponse. L'essentiel du travail consistait à améliorer les prompts (prompt engineering), maintenir l'historique des conversations ou à enrichir le modèle avec des bases de connaissances externes via des systèmes de type RAG, tout ceci apparaissant presque déjà comme la préhistoire de l'applicatif IA.

2) Les frameworks agentiques "classiques", tels que CrewAI ou SmolAgents qui ont posé les bases de ce qu'on appelle l'IA agentique, à savoir l'idée qu'un modèle de langage ne se contente pas de générer du texte mais est capable, quand il a été entraîné pour le "function calling", d'agir dans un environnement en appelant des outils externes, interrogeant des bases de données, exécutant des scripts, et en planifiant des étapes successives pour atteindre un objectif. Pour ce faire, un agent fonctionne généralement selon le paradigme ReAct (Reasoning + Acting) c'est-à dire selon un cycle du type : planifier la tâche → choisir un outil → exécuter l'action → analyser le résultat → ajuster le plan → recommencer, et ces frameworks orientés code ont précisément permis d'outiller et d'orchestrer un ou plusieurs agents pour la réalisation d'une tâche complexe.

▸ **Exemple d'agent avec le framework CrewAI**

3) La nouvelle génération des agents "dans l'infrastructure", qui ne reposent plus sur des briques à assembler programmatiquement, mais proposent des agents déjà opérationnels :

- à installer en dur sur l'infrastructure (PC ou serveur) ;
- livrés avec des outils nativement intégrés intégrés (built-in tools) avec accès au système de fichiers, exécution de code, recherche web, gestion de tâches planifiées, cron jobs… ;
- activables via des commandes préconstruites (`/model`, `/init`, `/resume`…)

- dans le terminal, sans autre interface utilisateur.

Une différence importante entre les premiers frameworks agentiques et cette nouvelle génération d'agents réside dans le fait que, dans le premier cas, l'agent est en grande partie hardcodé, c'est-à dire programmé explicitement dans le code avec ses outils, son rôle, ses objectifs... tandis que dans l'autre, le comportement de l'agent est défini par sa configuration et sa documentation, qui déterminent l'environnement dans lequel celui-ci apprend à agir.

Notons que le parti-pris CLI-first, s'il a entraîné une très forte adoption dans le monde des développeurs, constitue peut-être encore pour l'instant un frein à leur diffusion auprès du grand public, mais gageons que des interfaces web apparaîtront rapidement pour faciliter leur prise en main, à l'instar de claude-code-studio pour Claude Code.

Globalement, l'écosystème IA tend ainsi à évoluer vers des infrastructures du type LLM + Tools + Contexte + Runtime, réunies dans des applications agentiques formant une stack complète

```
Apps
│
├─ UI
│
├─ Agent runtime
│ (OpenClaw / Nanobot / LangGraph)
│
├─ Agents
│ (Codex / Claude Code)
│
├─ Tools / Protocols / Skills
│ (MCP / APIs / SKILL.md)
│
└─ Models
   (LLM / SLM)
```

Dans cet environnement, les trois éléments fondamentaux différenciants de ces infrastructures sont :

- le contexte ;
- le protocole MCP, qui permet au modèle de disposer d'outils pour "parler" au monde extérieur ;
- et ce que l'on appelle les skills ("compétences"), des sortes de playbooks réutilisables regroupant l'ensemble des connaissances nécessaires pour accomplir une tâche, et qui sont injectées dans le contexte du modèle.

Du prompt engineering au context engineering

Ce changement de paradigme s'accompagne en effet d'un déplacement conceptuel important : pour

permettre aux agents IA de gagner en autonomie, la notion de contexte devient centrale et le `prompt engineering` (qui consiste à trouver la bonne formulation pour obtenir la bonne réponse) cède progressivement la place au `context engineering` qui s'articule autour de la production d'un environnement structuré décrivant les outils disponibles, les procédures à suivre, les sources de données accessibles et les règles de fonctionnement de l'agent.

En fin de compte, plutôt que d'essayer de tout faire tenir dans un prompt, on dote le modèle de toutes les informations nécessaires pour prendre des décisions, planifier des actions et agir en conséquence, charge à lui de choisir lesquelles mobiliser au sein de toute cette documentation opérationnelle souvent rédigée en Markdown, lisible donc à la fois par le LLM et par les humains.

Formellement, ce principe général de l'augmentation des capacités du modèle et de son pilotage par du contexte se décline en différents types de fichiers selon les outils utilisés (MEMORY.md, TOOLS.md, etc.), des formats ouverts étant rapidement arrivés pour standardiser ces pratiques, comme AGENTS.md qui joue pour les LLM un rôle comparable à celui des README pour les humains.

▸ **Exemple de personnalisation (CLAUDE.md)**

Un aspect central du context engineering concerne la gestion de la mémoire des agents : contrairement aux chatbots classiques dont le contexte disparaît une fois la conversation terminée, les agents présentent la particularité de pouvoir maintenir, toujours grâce à ce système d'instructions textuelles, une mémoire persistante leur permettant d'accumuler des connaissances (préférences utilisateur, procédures validées, erreurs déjà rencontrées...), de capitaliser sur leurs expériences et d'améliorer progressivement leur efficacité .

Cette mémoire est généralement stockée dans des fichiers dédiés (par exemple MEMORY.md) auto-alimentés par l'agent au fil des interactions grâce à des consignes très simples du type "Before starting, review your memory. After completing a task, update your memory with what you learned."

▸ **Exemple de fichier MEMORY.md**

MCP : rendre les données AI-friendly

Le protocole MCP (Model Context Protocol) a été conçu pour lever la barrière de la base de connaissances fermée des LLM (qui les limite à leurs données d'entraînement) en proposant une norme ouverte et interopérable d'interaction des modèles avec des services ou des sources de données externes via une architecture client/serveur standardisée. Un serveur MCP agit donc comme un contrat documenté d'exposition des outils et des données qui est interprétable côté client par des applications agentiques, et techniquement on peut le voir comme une forme d'OpenAPI pour les agents IA par le biais duquel

- le serveur annonce les outils disponibles ;
- leur documentation permet à l'agent de choisir lequel utiliser (et comment) ;
- le modèle appelle la fonction avec des arguments structurés ;
- le serveur exécute la requête et renvoie le résultat.

Ainsi, du point de vue de la panoplie des protocoles d'accès et de dissémination des métadonnées bibliographiques déjà utilisés dans l'écosystème documentaire (API Rest, SRU, OAI-PMH…), les serveurs MCP doivent être vus comme un moyen supplémentaire de rendre les données accessibles grâce une simple couche d'interopérabilité supplémentaire à destination des LLMs et des agents IA.

J'ai déjà exposé dans un précédent billet le fonctionnement, les tenants et les aboutissants des MCP ainsi que la grande quantité de serveurs MCP déjà développés et découvrables dans des registres collaboratifs, en mettant l'accent sur la nécessité pour les bibliothèques de consolider leur rôle de fournisseuses de données de qualité en rendant leurs (méta)données intégrables dans les architectures mêmes des modèles de langage. L'enjeu est clair en réalité, **si nos catalogues nationaux (Sudoc, data.bnf, IdRef…) voire locaux, nos archives ouvertes, nos référentiels, nos entrepôts de données de recherche ou nos bibliothèques numériques ne sont pas exposés via des MCP, ils seront absents des environnements de travail des agents IA, et donc absents des pratiques de leurs utilisateurs, chercheurs y compris**.

L'exemple du serveur MCP connecté à la plateforme data.gouv.fr et mis à disposition par la DINUM est à ce titre intéressant et inspirant, car il illustre les possibilités de recherche, exploration, interrogation ou analyse des jeux de données publics en langage naturel juste en branchant ce MCP dans un environnement agentique.

Les skills : exporter l'expertise documentaire

En complément des MCP qui répondent à la question de l'accès aux données, les skills sont des élements contextuels majeurs car elles servent à documenter des tâches spécifiques et à décrire pour un agent comment les accomplir, tout ceci dans le cadre d'un simple fichier markdown `SKILL.md` (possiblement augmenté de scripts python ou de tout autre ressource nécessaire) chargé automatiquement par l'agent et appliqué chaque fois que cela est pertinent.

Une skill bien formée (conforme par exemple au format https://agentskills.io/) regroupe ainsi l'ensemble des définitions, instructions textuelles, exemples, appels d'API avec paramètres, structuration des résultats attendue ou scripts nécessaires pour qu'un agent puisse accomplir une tâche de manière autonome, et se structure généralement comme ceci

```
my-skill/
├── SKILL.md
├── scripts/
├── references/
└── assets/
```

Formellement, le principe est simple : on documente une fois l'ensemble des connaissances et procédures métier (schémas de raisonnement, flux de travail…) et on stocke les dossiers de skills dans le workspace de l'application agentique (.claude/skills pour Claude Code, .codex/skills pour Codex, nanobot/skills pour nanobot…) afin qu'elles puissent être réutilisées dans différents workflows.

Pour rendre cela plus concret, voici un exemple de skill écrite dans le cadre de mon projet de hub de skills documentaires

▸ **Skill de stratégie de recherche documentaire**

ou encore cette skill de recherche dans le SRU du Sudoc avec ce résultat en l'activant dans Codex en réponse à la question : "Cherche des ouvrages dont le titre contient jardins japonais, en français et publiés après 2000."

```
• Je vais utiliser le skill sudoc-sru et lancer la recherche via cli.py avec un filtre titre (mti), langue
  française et année > 2000. Je commence par localiser le script puis j'exécute la requête avec le préfixe
  UV_CACHE_DIR=/root/.cache/uv comme demandé.

• Explored
  └ List sudoc-sru

• La requête a retourné 12 notices brutes, dont une qui ne contient pas l'expression exacte dans le titre.
  Je filtre maintenant les résultats pour ne garder que les titres contenant précisément jardins japonais
  (insensible à la casse).

✔You approved codex to run UV_CACHE_DIR=/root/.cache/uv uv run /root/.codex/skills/sudoc-sru/cli.py sear...
  this time
```

```
• Résultat Sudoc (ouvrages doc-type=b, français lan=fre, publiés après 2000 year-from=2001, titre contenant
  jardins japonais) : 10 notices.

  1. La mystérieuse beauté des jardins japonais… (2015) – Berthier, François – PPN 190477008 –
     https://www.sudoc.fr/190477008
  2. Le Japon des jardins : comprendre les jardins japonais… (2012) – Peeters, Francis – PPN 165216603 –
     https://www.sudoc.fr/165216603
  3. Jardins japonais (2011) – Klecka, Virginie – Rustica éd. – PPN 152148523 –
     https://www.sudoc.fr/152148523
  4. Jardins japonais (2010) – Elisseeff, Danielle – PPN 147293790 – https://www.sudoc.fr/147293790
  5. Jardins japonais (2009) – Ketchell, Robert – Artémis éd. – PPN 134058062 –
     https://www.sudoc.fr/134058062
  6. Jardins japonais contemporains (2003) – Rico Nosé, Michiko – Flammarion – PPN 072286105 –
     https://www.sudoc.fr/072286105
  7. La mystérieuse beauté des jardins japonais (2025) – Berthier, François – PPN 293346003 –
     https://www.sudoc.fr/293346003
  8. Jardins japonais : conseils pour créer son jardin zen (2016) – Quéant, Virginie – PPN 196635055 –
     https://www.sudoc.fr/196635055
  9. Niwaki et jardins japonais : créer des reflets de nature (2016) – Dumas, Frédérique – Eyrolles – PPN 1
     96724120 – https://www.sudoc.fr/196724120
  10. Les jardins japonais traditionnels et contemporains… (2011) – Celestin/Richert/Labrid – LR presse – P
     PN 154354961 – https://www.sudoc.fr/154354961
```

On trouve des skills prêtes à l'emploi, portables et partageables dans différents hubs ou registres de skills (https://skills.sh/, https://clawhub.ai/, https://github.com/K-Dense-AI/claude-scientific-skills...) pour effectuer toutes sortes de tâches, et il me paraît évident que, tout comme pour les MCP à propos de l'ouverture des données, les bibliothèques auraient tout intérêt à s'investir dans la rédaction de skills pour les agents IA afin d'intégrer les pratiques documentaires dans ces nouveaux outils, tout simplement car en mettant à disposition des skills de référence (recherche documentaire, les revues de littérature, la rédaction de plans de gestion de données, la gestion des références bibliographiques...), il y a là une **véritable opportunité de redéploiement des compétences, outils et services documentaires dans les systèmes IA en les intégrant directement dans les environnements numériques où les utilisateurs sont amenés à de plus en plus produire et exploiter l'information.**.

Idéalement même, un marketplace communautaire de MCP, de skills et d'agents pour l'ESR serait sans doute particulièrement utile.

Si l'on prend un peu de recul, nous sommes en réalité face à une double transformation :

- d'une part une transformation progressive de la technicité du code vers une **ingénierie de l'orchestration et de la formalisation structurée du savoir-faire et de l'expertise dans une documentation compréhensible par des LLM** ;
- et d'autre part une logique d'intégration dans des frameworks d'agents au sein desquels les interactions jusque là classiques deviennent des briques modulaires d'une architecture plus large. Je pense notamment au RAG qui peut désormais être intégré via un MCP dans un système plus englobant, voire remplacé par des techniques avancées de gestion du contexte (compression de contexte, récupération hiérarchique, etc.) rendant potentiellement inutile le maintien d'une base vectorielle parallèle.

Dans ce contexte, les bibliothèques disposent déjà des ressources les plus précieuses dans l'écosystème de l'IA, à savoir des métadonnées ouvertes, fiables et structurées, doublées et accompagnées d'un capital informationnel et méthodologique très riche sur leur exploitation en tant que corpus. La question est donc de savoir comment rendre ces données et l'expertise associée exploitables dans les environnements IA, désormais caractérisés par des architectures où les agents deviennent progressivement les nouveaux intermédiaires d'accès à l'information via des données exposées en MCP et des procédures métier formalisées sous forme de skills.

Pour 2026 donc, peut-être un peu moins de discours sur l'IA, et un peu plus de travail concret et d'expérimentations sur les données, les protocoles, les outils, les agents et modèles spécialisés...

Tags: agent

→

Le Steering, ou comment comment modifier le comportement d'un LLM sans fine-tuning »