MEthodology in Data Science

Systèmes agentiques

Olivier Schwander
<olivier.schwander@sorbonne-universite.fr>

Master MIND
Sorbonne Université



2026-2027

Continuum agentique

Plusieurs niveaux

- Niveau 0 : modèle brut
- Niveau 1 : chatbot
- Niveau 2 : outils
- Niveau 3 : agent
- Niveau 4 : système multi-agents

Modèle de base

Entraînement : complétion

- Autosupervision
- Objectif : prédire le token suivant
- Fonction de coût : cross-entropy (log-vraisemblance négative)
- Échelle : des milliards de tokens de texte
- Ce qui émerge : grammaire, connaissances génériques, styles, in-context learning

Inférence

- Distribution de probabilité sur le prochain token
- Décodage glouton (argmax) ou échantillonnage (température)
- Pour le moment : complétion, pas dialogue

Capacités

Ce qu'on apprend

- Grammaire, culture générale, style, etc
- Plausibilité
- `Louis XIV est né en` $\rightarrow$ `1638`

Ce qu'on n'apprend pas

- Vérité
- Aucune distinction entre « vrai » et « souvent écrit »
- Pas de notion de date, de source, de certitude

Besoin

- Connaissances externes

Tokenisation

Tokenizer OpenAI GPT-5

```
Louis XIV est né en 1638
2+2 = 4
Est-ce que 5324523 + 988812333 = 654234234 ?
```

Conséquences sur la véracité

- Découpage arbitraire
- Sans considération pour la sémantique des nombres
- Conséquence : l'arithmétique n'est pas traitée en base 10
- Parfois juste : reconnaissance de motifs, mais pas de calcul

Besoin

- Capacités externes

Attention et contexte

Attention

- Chaque token est relié à tous les autres
- Coût en calcul : quadratique en la longueur de séquence

Améliorations

- Mémoire linéaire mais calcul quadratique
- Pas de bonne façon d'y échapper

Contexte

- Fenêtre de contexte finie
- Trop de contexte : coût, latence et dilution de l'attention

Besoin

- Autre source pour les connaissances

Vers les chatbots

Fine-tuning supervisé

- Dialogue comme complétion spécialisée
- Question → Réponse
- Tokens spécialisés

```
<|im_start|>system
Tu es un assistant littéraire concis.<|im_end|>
<|im_start|>user
Qui a écrit L'Étranger ?<|im_end|>
<|im_start|>assistant
Albert Camus - roman publié en 1942.<|im_end|>
<|im_start|>user
Et en quelle année a-t-il reçu le Nobel ?<|im_end|>
<|im_start|>assistant
```

Alignement

Algorithmes

- RLHF : apprentissage par renforcement à partir de préférences humaines
- DPO : optimisation directe (classification)
- Étiquetage humain
- Récompenses mécaniques : test des programmes, vérification des preuves mathématiques, comparaison

Préférences humaines

- Safety : éviter le contenu toxique (harcèlement, racisme, sexisme, piratage, etc)
- Style de réponse : façon de parler, avertissements, leçons de morale

Limites structurelles

Hallucinations

- Sortie fluide, confiante, factuellement fausse
- Universel en ML : sortie confiante $\neq$ sortie juste
- Causes mécaniques : maximum de vraisemblance, pression à répondre, conformité aux attentes utilisateur
- Conséquence de l'objectif d'entraînement

Connaissances figées

- Date de coupure des données d'entraînement
- Manques de connaissances métier ou privées
- Connaissances rares ou récentes : sous-représentées, donc mal apprises

Limites à dépasser

Calcul

- Opérations arithmétiques, même simples : plusieurs étapes
- Chaque étape risque d'échouer, les erreurs se propagent
- Tout ce qui est algorithmique : tri, comparaison, etc

Action

- Besoin de plus que la génération de texte
- Appeler une calculatrice, un interpréteur Python, une commande shell, une base de données, une API externe
- Lecture ou écriture : des actions

Continuum agentique

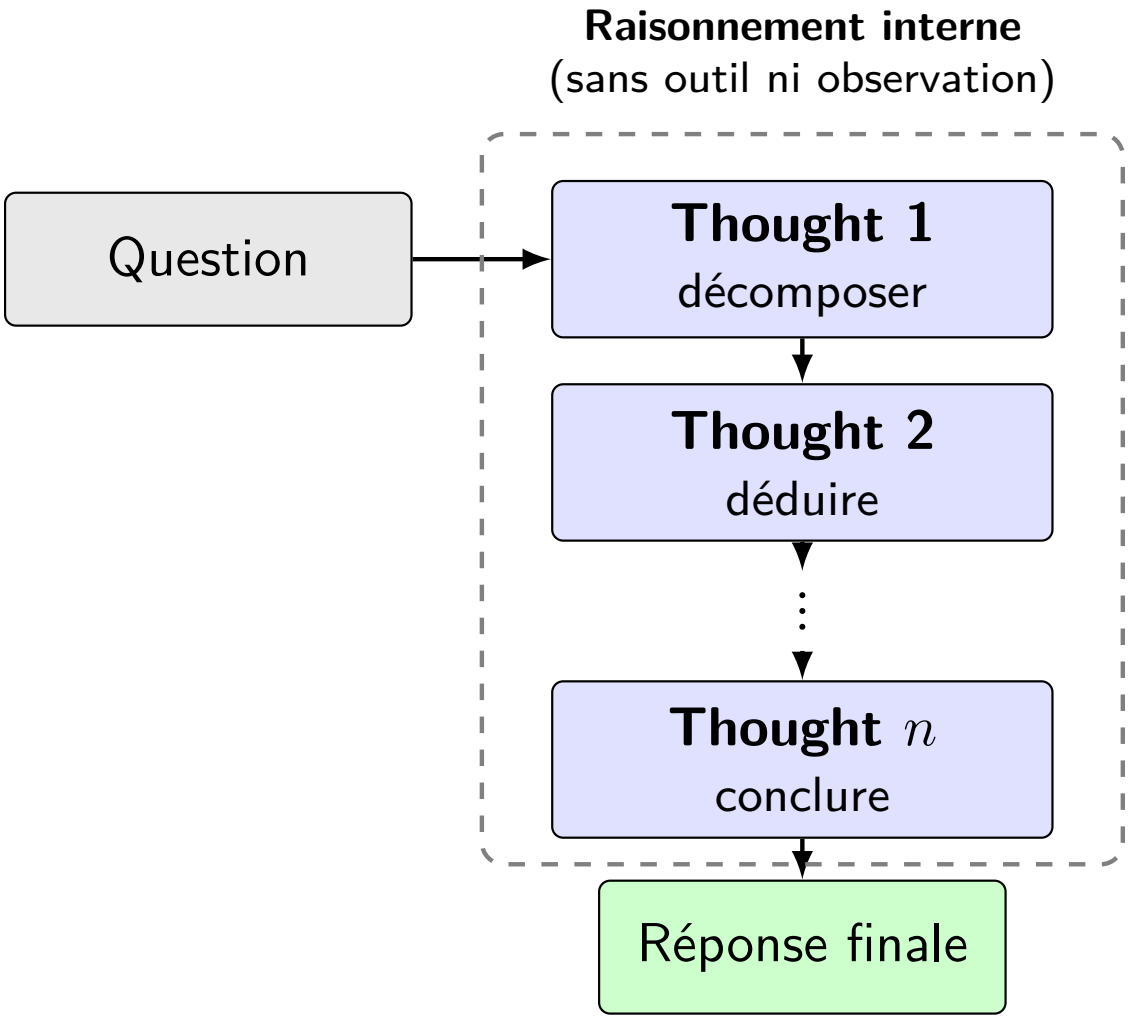
Plusieurs niveaux

- Niveau 0 : modèle brut (complétion)
- Niveau 1 : chatbot (modèle + template de conversation)
- Niveau 2 : modèle + outils (le modèle peut demander une exécution)
- Niveau 3 : agent (boucle : décide, agit, observe, recommence)
- Niveau 4 : système multi-agents (plusieurs boucles coordonnées)

Complexité croissante

- Maturité décroissante

Chain of thought



Éléments de l'agent

Modèle

- LLM avec entraînement spécifique pour le raisonnement et l'appel d'outils
- Planification, appels d'outils, reprise après erreur, sortie finale

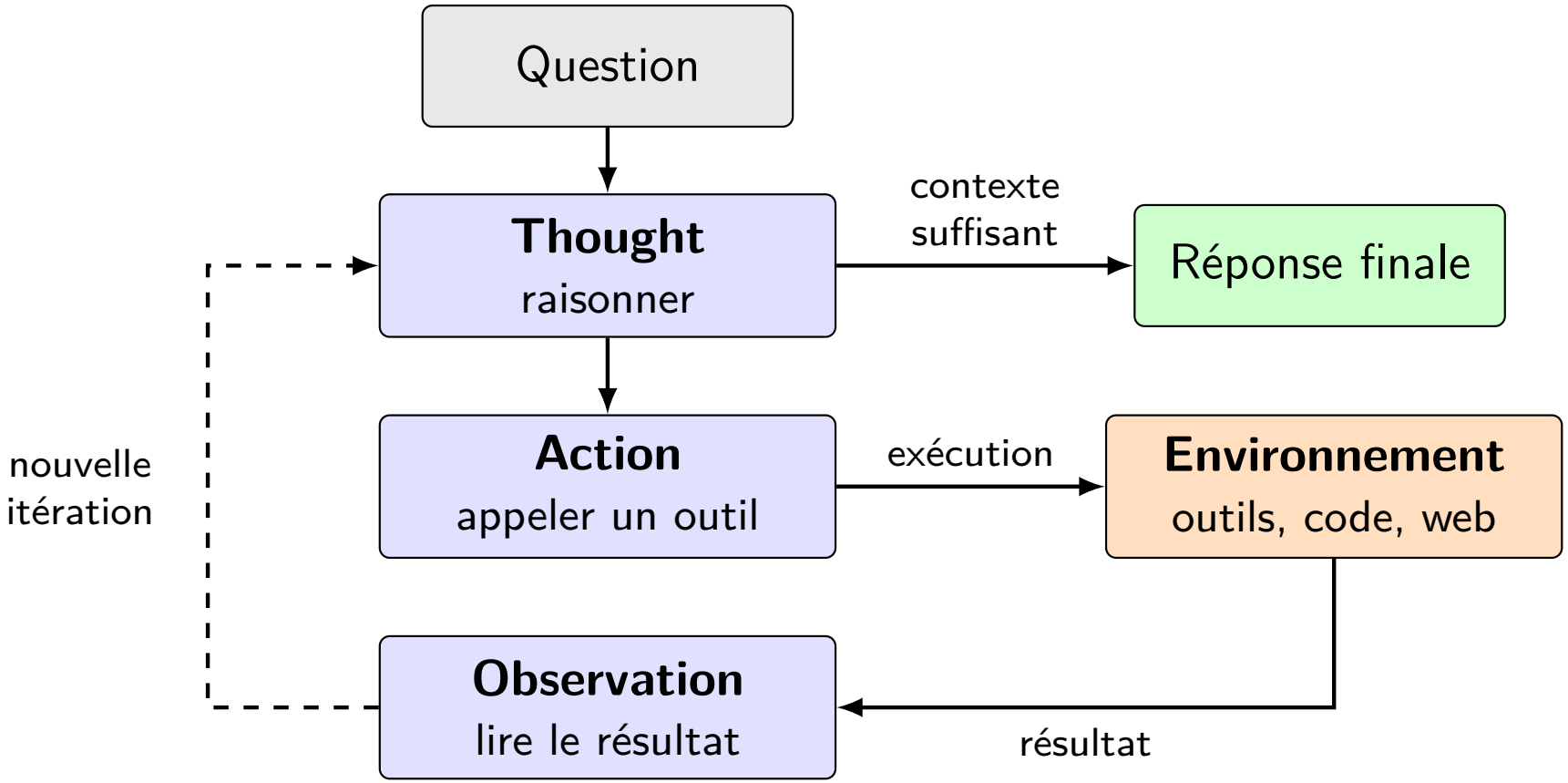
Outils

- Déterministes : lecture, calcul, action
- Non-déterministes : autres agents

Boucle

- Gère l'enchaînement
- Critique

Boucle ReAct



Évolution récente

Entraînement spécifique

- SFT et alignement plus efficace
- Appel d'outils dans les données d'entraînement
- Meilleure gestion des boucles
- Respect des formats

Meilleurs modèles

- Sortie structurée fiable
- Contextes plus longs

Standardisation

- Besoin de savoir comment marchent les outils
- Model Context Protocol
- Skills

Génération auto-régressive

Décodage probabiliste

- Token par token $p(x_t|x_{t'<t})$
- Arg-max $\rightarrow$ déterministe
- Température : renormalisation
  - $t = 0$ déterministe
  - $t = 1$ pas de changement
  - $t > 1$ plus de variété
- Top-k : conserver uniquement les tokens les plus probables pour éviter la queue de distribution

Appel d'outil

- Token spécialisé
- Parmi tous les autres
- Le moteur d'inférence détecte le token et fait ce qu'il faut

Convention d'appel

Tokens spéciaux

- Frontières (<|im_start|>, <|tool_call|>, ...)
- Dépend du modèle
- Doit être compris par le moteur d'inférence (Ollama, vLLM)

Description des outils disponibles

- Nom, argument
- Et surtout motivation pour les utiliser
- Documentation en langue naturelle

En apprentissage

Fine-tuning

- Supervisé : exemples d'appels d'outils
- Préférence : évaluation automatique

Objectifs

- Appeler le bon outil, au bon moment, avec les bons arguments
- Génération d'une sortie au bon format
- La difficulté augmente avec le nombre d'outils disponibles

En inférence

Zero-shot (in-context learning)

```
{ "type": "function",
  "function": {
    "name": "calculator",
    "description": "Performs mathematical calculations",
    "parameters": {
      "type": "object",
      "properties": {
        "operation": {
          "type": "string",
          "description": "The operation to be done in pyt
        } },
      "required": [ "operation" ]
    }
  }
```

Décodage contraint

Structure de données

- Au format JSON
- Aucune approximation tolérée
- Impossible de continuer en cas d'erreur de syntaxe

Grammaire formelle

- Seuls les tokens valides sont autorisés lors du décodage
- Garantie de la validité syntaxique de la sortie
- Pas de garantie sur la qualité sémantique

En cas d'erreur

- Besoin de pouvoir reprendre : hallucination d'outils, mauvais arguments, mauvais types
- En partie au moteur : gestion des exceptions, etc
- En partie au modèle : générer un nouvel appel

Exemple complet

https://docs.mistral.ai/resources/cookbooks/concept-deep-dive-tokenization-tool_calling

```
<s> [AVAILABLE_TOOLS]
 [{"type": "function", "function": {"name": "calculator",
"description": "Performs mathematical calculations", "paran
[/AVAILABLE_TOOLS]
[INST] What's 2+2? [/INST]
[TOOL_CALLS]
 [{"name": "calculator", "arguments": {"operation": "2+2"}
[TOOL_RESULTS]
 {"content": 4, "call_id": "VvvODy9mT"}
[/TOOL_RESULTS]
 2+2=4</>s
```

Conséquences de la boucle agentique

Taille du contexte

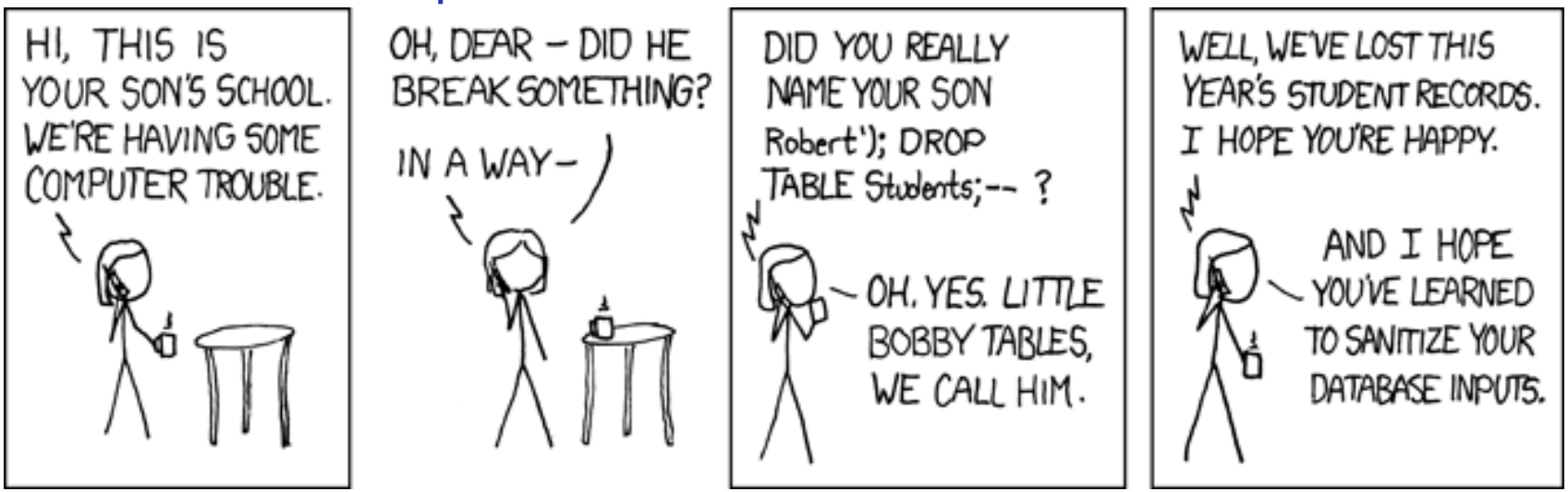
- Encore pire que d'habitude
- La doc des outils
- Leur sortie
- Empire avec le nombre d'outils

Sécurité

- Sans agent : réponse fausse ou hors sujet, atteinte à l'image de marque, ridicule
- Avec agent : conséquence sur le monde, fuite de données, suppressions
- Besoin d'isolation : mécanismes de sandbox (VM, containers, capabilities Linux, proxy filtrants, etc)

Injections

XKCD 327 — Exploits of a Mom



Avant 2022, j'étais persuadé de ne jamais remontrer cette figure en cours...

Des attaques mais pas que

- Contenu malicieux forgé par un attaquant
- Agent qui se met à faire n'importe quoi

RAG — Retrieval Augmented Generation

Historiquement

- Pas vraiment un outil
- Pré-LLM et chatbots

Sans condition

1. Entrée → Requête
2. Résultats → Prompt
3. Prompt → Sortie

Requête ?

- Base de données SQL ou non : data2sql
- Recherche web
- Base de données vectorielle : embeddings dans un espace latent
- RI classique : tf-idf et cie

RAG moderne

Un outil comme un autre

- Appelé si nécessaire
- Parmi d'autres outils

Toujours utile

- Pas de réentraînement
- Connaissances métier ou privées
- Besoin de limiter l'usage du contexte

Besoin de standardisation

M modèles × N outils

- M×N intégrations à écrire et maintenir
- Éviter les formats d'appel propriétaire

MCP : Model Context Protocol

- Standard ouvert
- Architecture : les serveurs MCP exposent des outils, ressources, prompts
- Découverte dynamique : l'agent découvre les outils au début
- Protocole uniquement : exécution totalement libre

Model Context Protocol

Isolation

- Étanchéité entre agent et serveur
- Uniquement les échanges prévus
- Accès stdio ou http

En pratique

- Fonctions dans le langage de programmation
- Bibliothèque d'emballage : récupérer le nom de la fonction, ses arguments, sa documentation
- Tous les outils et leur documentation sont injectés dans le contexte

Skills

Description d'un savoir faire

- Pas seulement un catalogue d'outils

Découverte hiérarchique

- Phrase courte de description pour donner l'intérêt de la skill
- Chargement dynamique si le modèle juge la description courte pertinente
- Version longue injectée dans le contexte : description précise de plusieurs tâches, exemples, contre-exemples

Sécurité

- Souvent : appel de commandes, on peut faire n'importe quoi
- Agent de code : on valide à chaque fois
- Agent autonome : besoin d'une sandbox **très très** étanche

Pourquoi plusieurs agents ?

Différentes architectures

- Parallélisme : plusieurs sous-tâches indépendantes en même temps
- Contexte partitionné : une fenêtre par agent
- Spécialisation : prompts, outils et rôles dédiés par sous-tâche
- Adversarialité : un agent critique un autre agent

Conséquences

- Difficulté de coordination
- Coût multiplié
- Erreurs multipliées

Architecture orchestrator-worker

Parallélisation

- Orchestrateur : décompose et délègue
- Workers : explorent en parallèle
- Orchestrateur : synthétise

Exemple : recherche d'information

- Analyse de la requête
- Recherche et résumé des documents
- Synthèse de la réponse

Architecture planner / executor / critic

Boucle ReAct généralisée

- Planner : décompose le problème en étapes
- Executor : exécute les étapes (avec outils)
- Critic : évalue, critique, déclenche des révisions

Exemple : discussion scientifique

- Suggestions d'idées
- Analyse et synthèse
- Boucle

Exemple : développement logiciel complexe

- Planification, suggestion, discussion, élimination des mauvaises idées

Architecture pipeline

Séquence

- Enchaînement de modèles spécialisés
- Reprise après erreur
- Étapes substituables, human-in-the-loop

Exemple : rédaction d'article

- Idée → Littérature → Plan → Code → Figures → Rédaction → Revue

Navier-Stokes — Septembre 2026

Équations de Navier-Stokes

- Base de la simulation numérique de la mécanique des fluides
- Existence d'une solution régulière ? Question mathématique difficile

Résultat annoncé par OpenAI

- Existence d'une solution non-régulière
- Intérêt physique assez faible : peu d'importance pour la simulation numérique
- Intérêt mathématique discutable et discuté : démarche souvent plus intéressante que le résultat
- Intérêt IA indiscutable : preuve d'un résultat majeur (si confirmé...)

Architecture telle que décrite

Multi-agent massif

- Coordination par un modèle interne non publié
- 10 000 agents en parallèle, répartis en groupes
- Trois boucles imbriquées : locale (ReAct), de groupe (diversité), globale (sélection/réallocation des ressources)

Preuve formalisée et vérifiée en Lean

- Assistant de preuve
- Sortie : pas juste du LaTeX
- Preuve vérifiable mécaniquement, par un outil de confiance
- (Anecdote : c'est ça qu'on appelait « IA » dans les années 60)

Garantie

- Agents probabilistes $\rightarrow$ vérification déterministe

Denario

Pipeline Data 2 Article

- Assistant de recherche multi-agents modulaire
- Chaque module : un système agentique autonome, utilisable seul ou dans la chaîne
- Chaîne : idée → vérification de nouveauté → plan → code → figures → rédaction → review

Communication par fichiers structurés

- idea.md, methods.md, results.md, etc
- Reprise après erreur
- Intervention humaine

Intérêt scientifique

- Discutable et discuté
- Démarche vs résultat

Modules Denario

- **Idea maker / Idea hater** : génération + critique systématique — le motif adversarial
- **Novelty search** : bibliographie
- **Plan** : méthodologie structurée
- **Code** : écrit et exécute du Python
- **Plots** : figures
- **Paper** : .tex/.bib/.pdf
- **Peer review** : double aveugle SIMULÉ par multi-agents (clarté, nouveauté, rigueur scientifique)

Conclusion

Agents indispensables

- Agir sur le monde, injecter des connaissances, améliorer la véracité

Sécurité

- Injections inévitables : instructions et données entremêlées
- Sandbox obligatoire : isolation des entrées-sorties, minimisation des outils accessibles, données limitées au strict nécessaire
- Conséquences graves...

Architectures complexes

- Rarement justifiables pour le moment
- Coût élevé