CONSTRUIRE UN AGENT LLM

UN LIVRE DE

PATRICE HUETZ

Construire un Agent LLM

Guide pratique

Patrice Huetz

patrice-huetz.fr

patrice-huetz.fr · contact@patrice-huetz.fr

Avant-propos

Le Declic

Decembre 2023, 23h47. Je fixe mon terminal depuis trois heures.

L'agent IA que j'avais construit — celui qui devait revolutionner mon workflow — venait de supprimer mon fichier de configuration. Encore. La troisieme fois cette semaine.

"Il suffit de lui dire de ne pas le faire," m'avait repondu un collegue.

Mais c'etait plus profond que ca. Mon agent ne *comprenait* pas ce qu'il faisait. Il executait des commandes sans contexte, sans memoire, sans raisonnement. Un perroquet avec acces root.

Ce jour-la, j'ai decide de comprendre. Vraiment comprendre. Comment construire un agent qui *pense* avant d'agir ?

Douze mois plus tard, apres $847 de factures API, des dizaines de fichiers supprimes par erreur, et des centaines d'heures de debugging — j'ai les reponses. Ce livre, c'est tout ce que j'aurais voulu savoir avant de commencer.

Ce Livre Est Pour Vous Si...

- Vous avez deja utilise ChatGPT ou Claude et voulez aller **beaucoup** plus loin
- Vous etes developpeur et voulez construire vos propres agents autonomes
- Vous voulez comprendre la recherche recente (ToT, MCTS, RAG avance...)
- Vous etes frustre par les limites des chatbots actuels

- Vous voulez eviter les erreurs qui m'ont coute des semaines et des centaines de dollars

Ce livre n'est PAS pour vous si vous cherchez une introduction aux Transformers ou l'histoire de l'IA. Ces informations sont gratuites sur YouTube, ChatGPT ou Wikipedia.

Ce Que Vous Allez Construire

A travers ce livre, nous allons construire **Code Buddy** ensemble — un agent IA de terminal complet :

Fonctionnalite	Chapitre	Ce que vous obtenez
Agent fonctionnel	1	30 minutes, 50 lignes
Raisonnement Tree-of-Thought	4	Explore plusieurs solutions
Monte-Carlo Tree Search	5	Recherche optimale
Auto-reparation	6	40% vs 15% de succes
RAG avec graphe de dependances	7-8	Contexte complet automatique
Cache semantique	12	68% de reduction d'appels
Model routing FrugalGPT	13	-70% de couts
Memoire persistante	14	Agent qui apprend
Securite complete	16	Sandbox, permissions, audit

Le code complet : ~27,000 lignes, 45+ outils, 200+ tests, open-source.

Ce Que Ce Livre Va Vous Eviter

La Facture de $847

Un de mes agents est parti en boucle infinie. 6 heures de tokens GPT-4. J'ai decouvert la facture le lundi matin.

Chapitre 3 : limites d'iterations, budget tokens, et garde-fous.

L'Agent Qui Supprime Vos Fichiers

23h47, un mardi. Configuration supprimee. La troisieme fois cette semaine. Mon agent ne comprenait pas la difference entre "nettoyer" et "detruire".

Chapitre 16 : systeme de permissions, confirmations, et sandbox.

Le RAG Qui Retourne N'importe Quoi

Mon RAG trouvait les bons fichiers mais pas les dependances. L'agent editait `user-service.ts` sans connaitre `database.ts` qu'il importait. Resultat : du code qui compile mais qui crash.

Chapitre 8 : Dependency-Aware RAG avec graphe AST.

Le Modele 10x Trop Cher

Tout sur GPT-4 alors que 73% des requetes marchent avec un modele 10x moins cher. J'ai brule $300 en une semaine avant de comprendre.

Chapitre 13 : FrugalGPT et model routing automatique.

Comment Ce Livre Est Different

Approche classique	Ce livre
"L'attention est un mecanisme qui..."	**Le code qui marche** (TypeScript, teste)
"Considerons l'architecture..."	**Le probleme -> La solution**
"Les Transformers ont revolutionne..."	**Comment economiser 70% sur l'API**
Exercice : "Reflechissez a..."	Exercice : **Implementez X en 30 min**
Theorie d'abord, pratique peut-etre	**Code d'abord**, theorie en annexe

Prerequis

Obligatoire

- **TypeScript/JavaScript** : niveau intermediaire (async/await, classes, types)
- **Terminal** : a l'aise en ligne de commande (navigation, commandes de base)
- **Compte API** : xAI (Grok), OpenAI, ou Anthropic
- **Budget** : ~$20 de credits pour les exercices

Ce Que Vous N'Avez PAS Besoin de Savoir

- Machine Learning / Deep Learning
- Les maths derriere les Transformers
- Comment entrainer un LLM

- Python (tout est en TypeScript)

Ce livre est sur l'**utilisation** des LLM pour construire des agents, pas sur leur fonctionnement interne. La theorie est en **Annexe A** si vous etes curieux.

Structure du Livre

Partie I : Demarrage Rapide (Ch. 1-3)

Objectif : Un agent fonctionnel en 2 heures

Partie II : Raisonnement Avance (Ch. 4-6)

Objectif : Un agent qui reflechit avant d'agir

Partie III : Memoire Intelligente (Ch. 7-9)

Objectif : Un agent qui comprend le contexte complet

Partie IV : Action (Ch. 10-11)

Objectif : Un agent qui agit efficacement (45+ outils)

Partie V : Optimisation (Ch. 12-13)

Objectif : -70% de couts, +3x de vitesse

Partie VI : Production (Ch. 14-19)

Objectif : Un agent deployable, securise, evolutif

Comment Lire ce Livre

Option 1 : Lineaire

Suivez les chapitres dans l'ordre. Chaque chapitre construit sur le precedent.

Option 2 : Par Besoin

Sautez directement au chapitre qui repond a votre probleme : - "Mon agent est trop cher" -> Chapitres 12-13 - "Mon RAG ne trouve pas le bon contexte" -> Chapitres 7-8 - "Mon agent fait des erreurs betes" -> Chapitres 4-6 - "Mon agent est dangereux" -> Chapitre 16

Option 3 : Code First

Clonez le repo, lancez l'agent, puis lisez les chapitres pour comprendre comment ca marche.

Le Code Source

Tout le code est disponible et teste :

```
git clone https://github.com/phuetz/code-buddy.git
cd code-buddy
npm install
export GROK_API_KEY=your_key
npm run dev
```

Statistique	Valeur
Lignes de code	~27,000
Outils integres	45+
Tests	200+
Couverture	85%+

Remerciements

Ce livre n'existerait pas sans les travaux de chercheurs brillants que je cite tout au long des chapitres : Yao et al. (Tree-of-Thought), Lewis et al. (RAG), Browne et al. (MCTS), et tant d'autres. Merci aussi a la communaute open-source qui partage ses decouvertes.

Commencez Maintenant

Passez directement au **Chapitre 1** pour avoir un agent fonctionnel en 30 minutes.

La theorie sur les Transformers ? Elle est en **Annexe A** si vous etes curieux. Mais vous n'en avez pas besoin pour construire.

Pret a construire un agent qui pense ? Tournez la page.

Patrice Huetz Decembre 2024

CHAPITRE 1

Votre Premier Agent en 30 Minutes

Ce Que Vous Allez Obtenir

Temps	Résultat
10 min	Un agent qui répond à vos questions
20 min	Un agent qui lit et modifie vos fichiers
30 min	Un agent avec garde-fous (pas de facture $847)

À la fin de ce chapitre, vous aurez un agent fonctionnel qui peut : - Lire votre code - Proposer des modifications - Exécuter des commandes (de manière sécurisée)

1.1 Le Test des 5 Minutes : Agent ou Simple Prompt ?

Avant de coder, clarifiions ce qui distingue un **agent** d'un simple appel API.

Simple Prompt	Agent
Une question → Une réponse	Un objectif → N actions automatiques
"Explique ce code"	"Corrige ce bug et vérifie que les tests passent"
Pas de mémoire entre appels	Contexte maintenu sur plusieurs itérations
Pas d'outils	Lit fichiers, exécute commandes, appelle APIs

Le test : Si accomplir la tâche nécessite plusieurs étapes que vous devriez faire vous-même entre les appels LLM, vous avez besoin d'un agent.

1.2 Les 3 Erreurs Fatales (et Comment les Éviter)

Erreur #1 : L'Agent Sans Limite → $847

```
// ❌ CE CODE VA VOUS RUINER
async function dangerousAgent(goal: string) {
  while (true) {  // Boucle infinie !
    const response = await llm.chat(messages);
    if (response.includes("DONE")) break;
    // ... actions
  }
}
```

Ce qui s'est passé : Un de mes premiers agents est parti en boucle infinie pendant 6 heures. Il n'arrivait jamais à résoudre le problème, mais continuait d'essayer. Facture : $847.

```
// ✅ VERSION SÉCURISÉE
const MAX_ITERATIONS = 15;
const MAX_TOKENS_PER_SESSION = 100_000;

async function safeAgent(goal: string) {
  let iterations = 0;
  let totalTokens = 0;

  while (iterations < MAX_ITERATIONS && totalTokens < MAX_TO
KENS_PER_SESSION) {
    const response = await llm.chat(messages);
    totalTokens += response.usage.total_tokens;
    iterations++;

    if (response.includes("DONE")) break;
  }

  if (iterations >= MAX_ITERATIONS) {
    console.warn("⚠️ Limite d'itérations atteinte");
  }
}
```

Erreur #2 : L'Agent Qui Supprime Vos Fichiers

```
// ❌ DANGER : L'AGENT PEUT TOUT FAIRE
const tools = [
  { name: "run_command", execute: (cmd) => exec(cmd) }  // r
m -rf * ?
];
```