



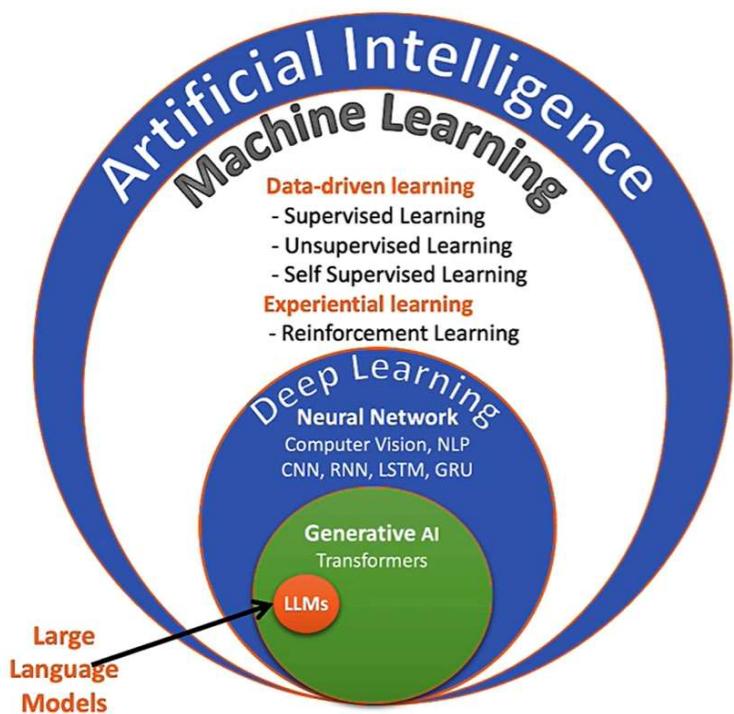
Systèmes Intelligents pour la Santé Numérique

Partie III: Generative AI and Prompt Engineering

ENIS - GI3 - 2025-2026

• • • •

Introduction



L'**Intelligence Artificielle générative** est un type d'intelligence artificielle capable de créer de nouvelles données à partir d'un ensemble d'exemples d'entraînement.

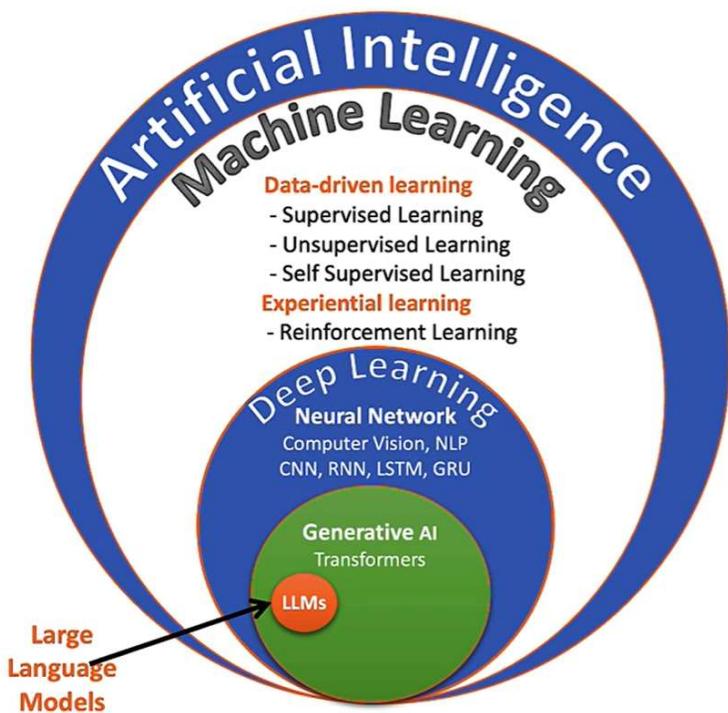
Objectif :

- Simuler la créativité humaine en produisant du contenu original cohérent et réaliste.

Domaines d'application :

- Création d'images et d'art numérique
- Rédaction de texte et chatbots
- Musique, design, jeux vidéo, code...

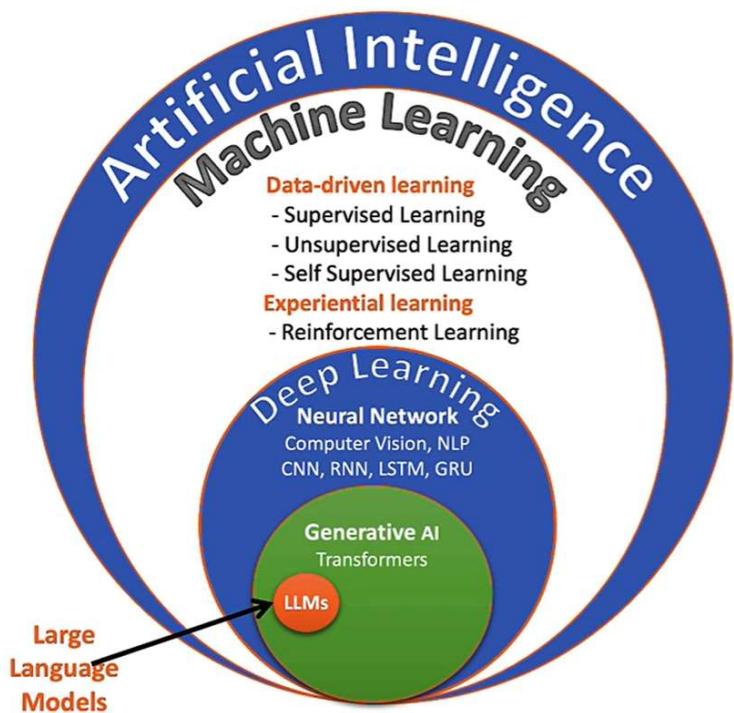
Introduction



Accélération des recherches : Plusieurs facteurs ont contribué à une augmentation significative des recherches et du développement dans ce domaine.

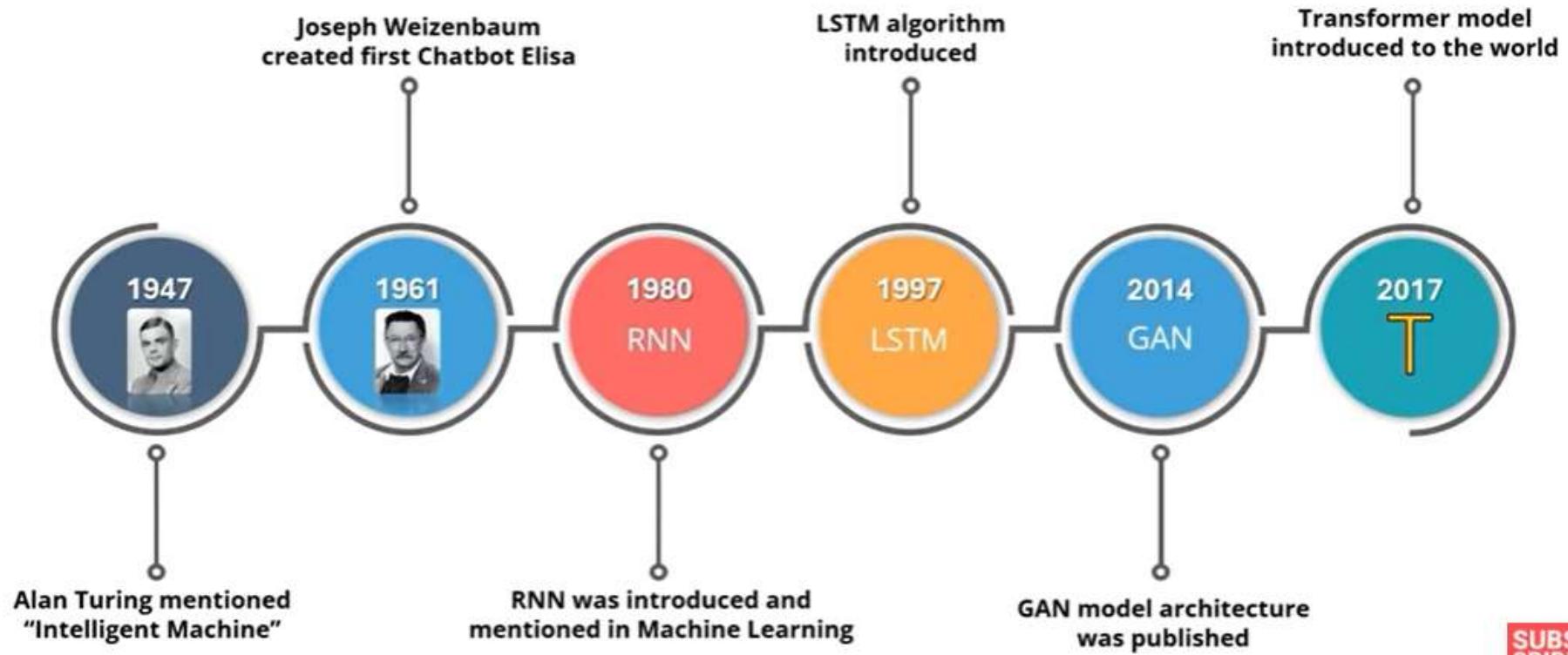
- **Modèles LLMs puissants**
- **Disponibilité massive de données**
- **Avancées technologiques**

Introduction



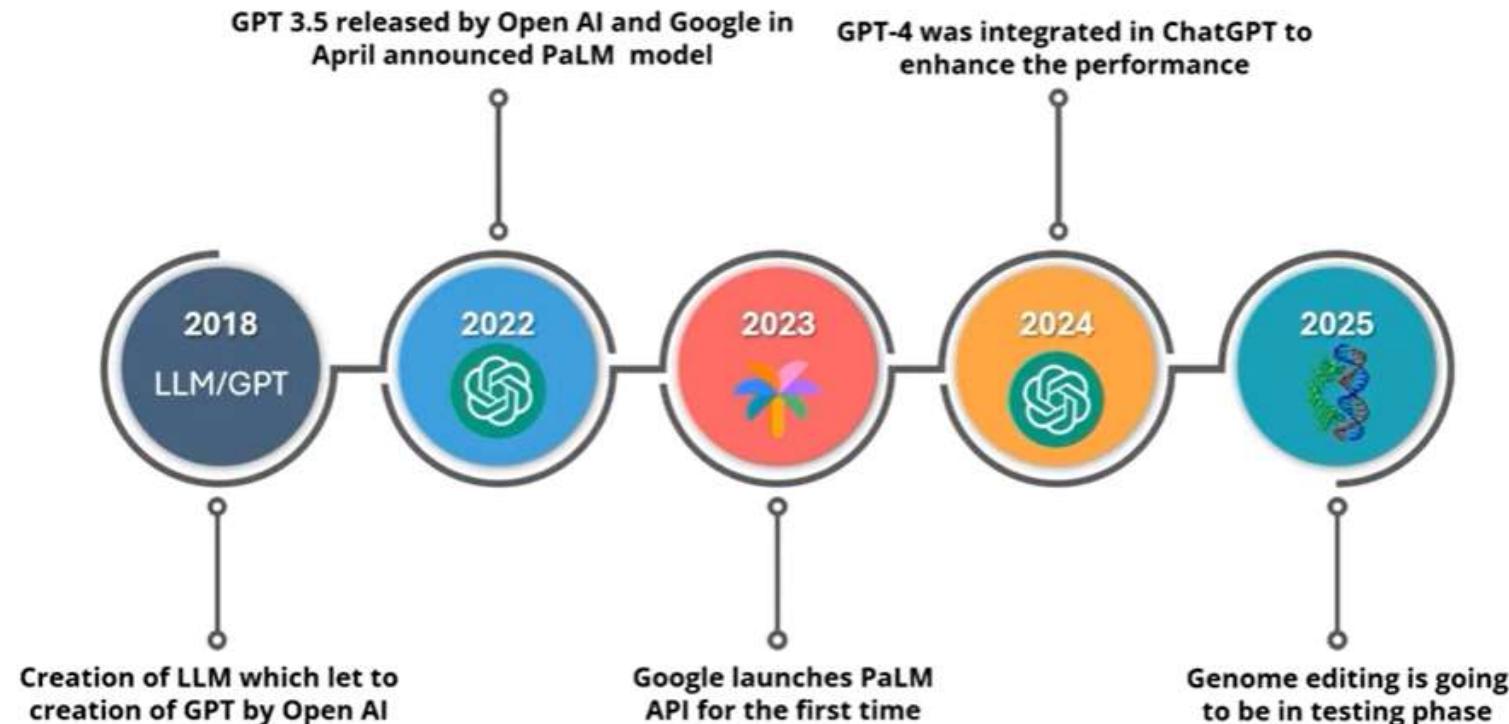
- Generative Adversarial Networks (GANs) (Ian Goodfellow)
- Break through in NLP : **Transformers** (google)
Attention is All You need (<https://arxiv.org/pdf/1706.03762.pdf>)
- Advent of Large Models :
 - BERT (Bidirectional Encoder Representations from Transformers), Google
 - GPT (Generative Pre-trained Transformers) (Open AI)
- Advanced GANs – Style GANs
- GPT-3 (Premier LLM)(Open AI)
- Vision Transformers – CLIP, DALL-E : Images (OpenAI)
- Chat GPT : GPT – 3.5 (Text), DALL-E 2(Images)
- BARD (Google), GPT- 4 (Open AI)
• Stable Diffusion (Images)
• Llama3, Code Llama, Github Copilot, Mistral, Mixtral

Introduction

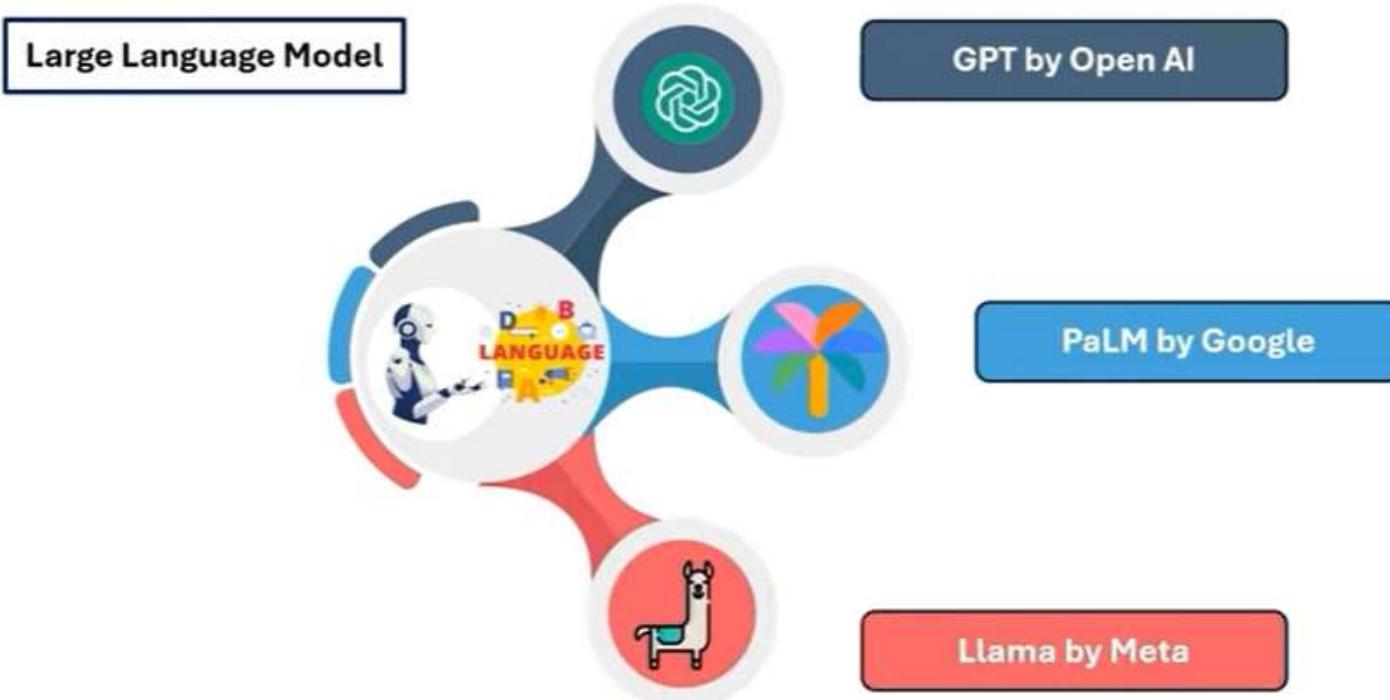


SUBSCRIBE

Introduction

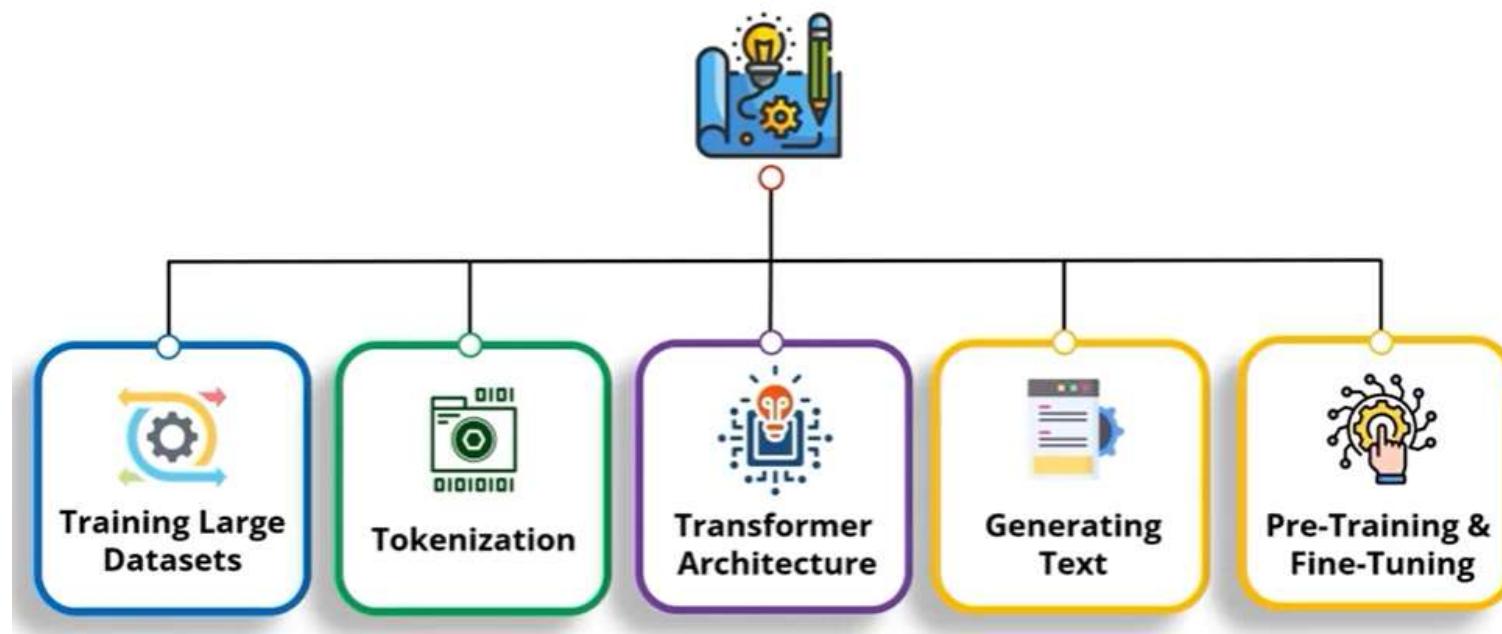


Introduction



Introduction

How LLM works?



Vers une IA Générative Spécialisée

Objectif : Produire un contenu plus pertinent et contextuel capable de répondre à des besoins précis, Enrichir, spécialiser et personnaliser la génération.

Techniques clés :

🔧 **Fine-Tuning** : spécialiser un modèle sur un domaine précis

- Ajustement d'un modèle pré-entraîné sur un **jeu de données spécifique**.
- Exemple : Adapter GPT pour rédiger des rapports médicaux ou analyser des contrats.

⌚ **Few-Shot Learning** : apprendre efficacement avec peu d'exemples

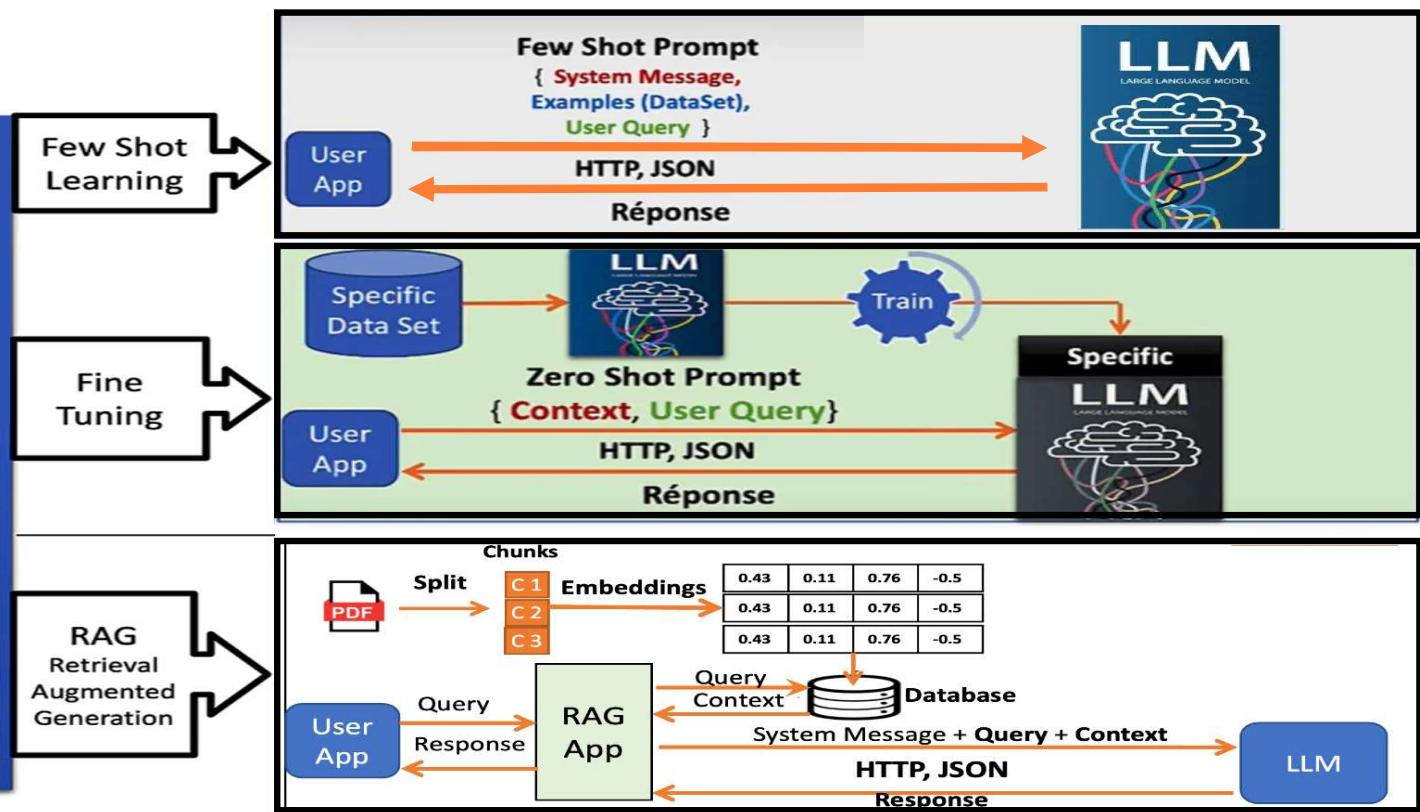
- Apprendre à **comprendre un style ou une tâche** à partir de **quelques exemples seulement**.
- Exemple : Fournir trois résumés d'articles pour que l'IA en rédige un quatrième dans le même style.

⚠️ **RAG (Retrieval-Augmented Generation)** : combiner génération et recherche d'informations

- Combine la recherche d'informations externes avec la génération.
- Exemple : Génération de rapports avec des données réelles issues d'une base interne.

Vers une IA Générative Spécialisée

Comment adapter IA Générative



Vers une IA Générative Maîtrisée

Objectif :

Améliorer la qualité, la cohérence et la pertinence des contenus générés par l'IA.

Techniques clés :

Prompt Engineering :

- Concevoir soigneusement les instructions données au modèle.
- Orienter la production pour obtenir des résultats plus précis et adaptés au contexte.

Bénéfices :

- Contrôle de la création.
- Pertinence et cohérence accrues.
- Production de contenus clairs, esthétiques et conformes aux attentes.

Vers une IA Générative Maîtrisée

Prompt :

Ensemble spécifique d'instructions envoyé à un modèle de langage (LLM) pour accomplir une tâche.

Prompt Engineering (Ingénierie du Prompt) :

Processus de conception, d'évaluation et de déploiement du prompt pour des tâches spécifiques

- **Définir la tâche et les métriques** : Identifier clairement l'objectif et les critères de succès pour la tâche.
- **Préparer les données** : Rassembler des datasets pertinents pour guider le modèle.
- **Concevoir et évaluer le prompt** : Créer différentes formulations et tester leurs performances pour sélectionner la meilleure.
- **Intégrer le prompt** : Connecter le prompt aux API, bases de données ou applications nécessaires pour l'utilisation réelle.
- **Déployer et surveiller** : Mettre en production et suivre les résultats pour ajuster ou améliorer le prompt si nécessaire.

Vers une IA Générative Maîtrisée

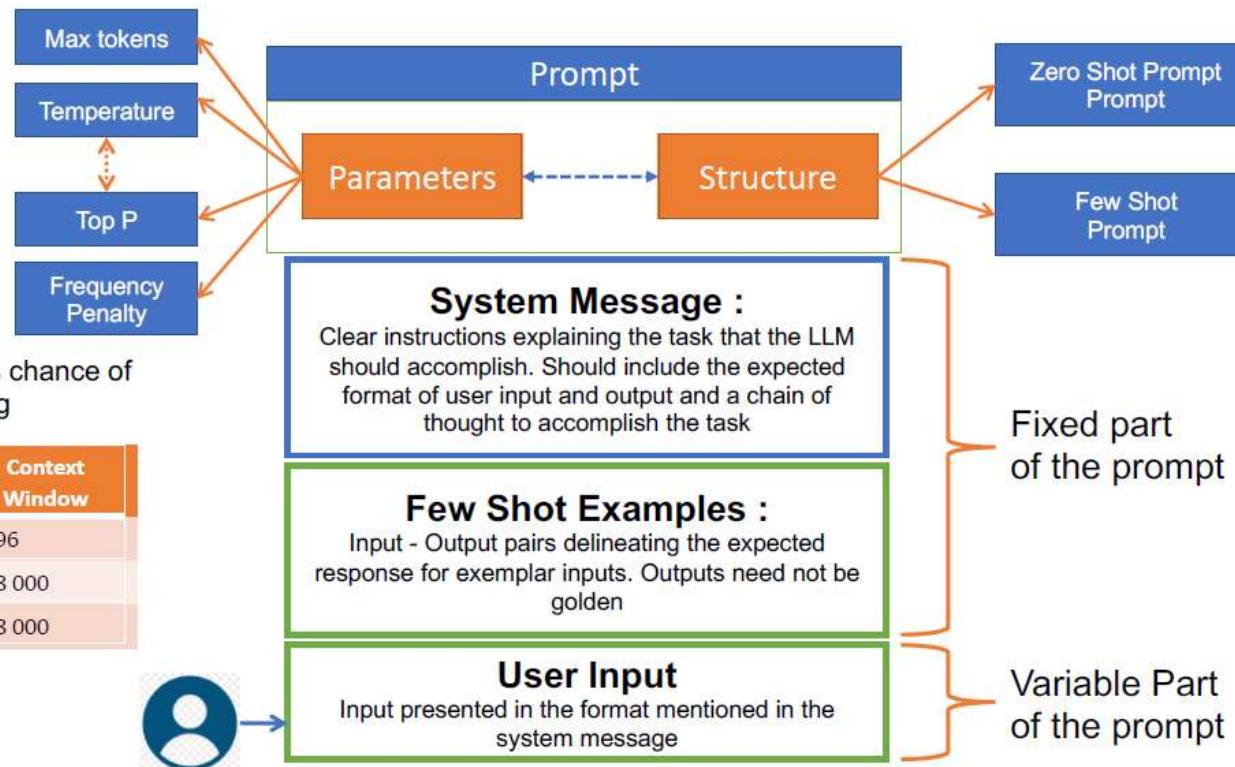
Length of input + output

More temperature = More randomness in response

More Top P = More tokens selected for completion

More FP = Less chance of tokens repeating

| Model | Context Window |
|---------------|----------------|
| Gpt-3.5-turbo | 4096 |
| Gpt-4-tubo | 128 000 |
| Mixtral-8x7B | 128 000 |



<https://platform.openai.com/tokenizer>

Vers une IA Générative Maîtrisée

Top-p est un paramètre de génération dans les modèles de langage. Il sélectionne les tokens dont la probabilité cumulée atteint p, assurant précision et cohérence.

Exemple Prompt:

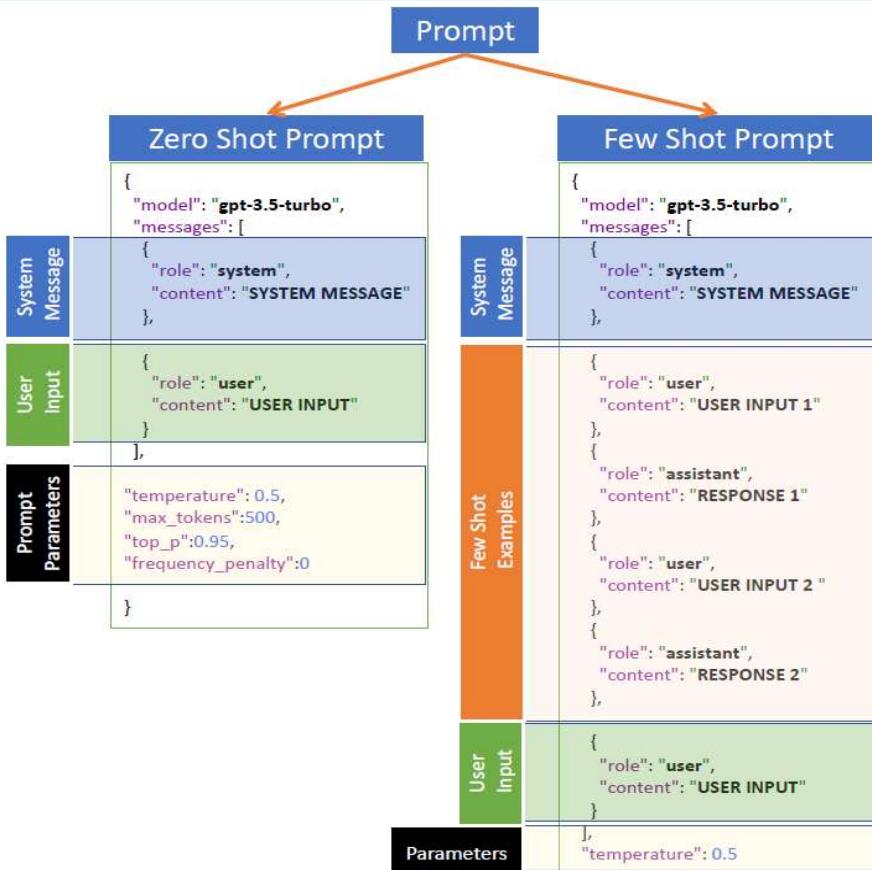
»Rédige un résumé concis d'un dossier médical sur un patient atteint de diabète de type 2 »

Top-p Résultat

0.5 « Le patient présente un diabète de type 2. Il suit un traitement par metformine. »

0.9 « Le patient, âgé de 55 ans, souffre de diabète de type 2 depuis 5 ans. Il suit un traitement à base de metformine et un régime alimentaire équilibré. »

Vers une IA Générative Maîtrisée



There are four common patterns of prompts :

- **Instruction prompt**
- **Reasoning prompt**
- **Induction prompt**
- **Paraphrasing prompt**
- **Chain-of-Thought Prompt (Cot prompt)**

Vers une IA Générative Maîtrisée

| Type de prompt | Objectif | Exemple |
|------------------|--------------------------------------|---|
| Instruction | Action directe | "Liste les 5 avantages du solaire" |
| Reasoning | Raisonnement étape par étape | "Explique pourquoi le solaire est durable..." |
| Induction | Déduire la règle à partir d'exemples | "Pomme → Fruit, Orange → ?" |
| Paraphrasing | Reformuler un texte | "Réécris cette phrase simplement" |
| Chain-of-Thought | Raisonnement détaillé avant réponse | "Résous ce problème étape par étape..." |

Vers une IA Générative Maîtrisée

Chain-of-Thought Prompting : CoT Prompt

- Pour l'invite CoT, nous ajoutons des instructions détaillées étape par étape au message système demandant au modèle de réfléchir attentivement avant de décider du résultat à générer
- En dehors de cet ajout, il n'y a aucun autre changement par rapport à prompt standard

Standard Prompt

- Q : Ahmed has 5 tennis balls, he buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
- A : The answer is 11
- Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more. How many apples do they have?

The cafeteria now has 9 apples.

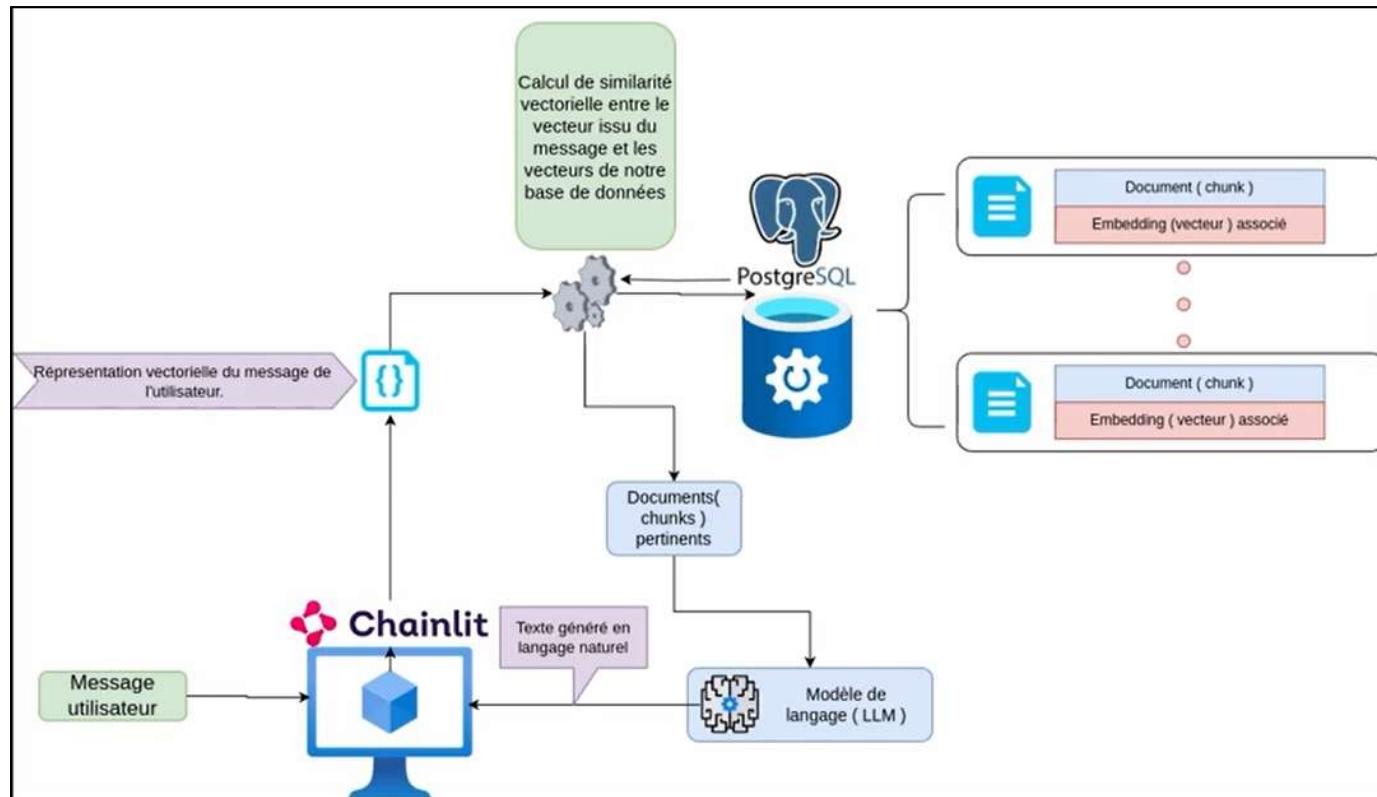
CoT Prompt

- Q : Ahmed has 5 tennis balls, he buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
- A : Ahmed started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5+6=11$. The answer is 11
- Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more. How many apples do they have?

The cafeteria started with 23 apples. They used 20, so they had $23-20=3$ apples left. Then they bought 6 more, so they now have $3+6=9$ apples.



Simple RAG (Retrieval Augmented Generation)



Implémentation d'un chatbot à travers une architecture RAG

- Connexion à votre compte Github
- Création d'un nouveau projet Chatbot-RAG
- Ouvrir vs code et cloner le projet en local
- Modifier la description du fichier readme et faire un commit puis push
- Vérifier l'état du git à travers le terminal
- -rajouter les fichiers modifiés au repository local
- Sauvegarder en local puis en ligne
- Créer des sous-dossiers dans le projet : <data> <notebook> <src>
- Créer sous <notebook> un fichier prototypage.ipynb
- Créer sous <src> .env .gitignore
- Créer un envrinomment virtuel et activer le
- Installer numpy et pandas via le terminal
- Sauvegarder tous les requirements dans un fichier requirement.txt
- Faire un push tout en ignorant venv et .env
- Installer postgresql et s'y connecter à travers VScode
- Créer la base rag_chatbot via Vscode

- Installer psycopg pour interagir avec la base
<https://www.psycopg.org/psycopg3>
- Dans le notebook, prototypage.ipynb créer une connexion à la base créée , insérer une table ayant ID comme clé primaire, corpus text et embedding float, faire un import psycopg
- Corpus téléchargeable via ce lien
https://www.info.univ-tours.fr/~antoine/parole_publique/Accueil_UBS/index.html
- Utiliser un modèle embedding openai / Gemini/Ollama et faire les installations nécessaires
- Installer pgvector (vérifier l'installation de C++ et n'oubliez pas d'ajouter le path à nmake à la variable d'environnement)
<https://github.com/pgvector/pgvector>
- Ajouter une nouvelle requête à la base CREATE extension vector
- Introduire une requête
- Introduire un modèle de langage pour introduire une réponse

Limite du RAG avec bases vectorielles

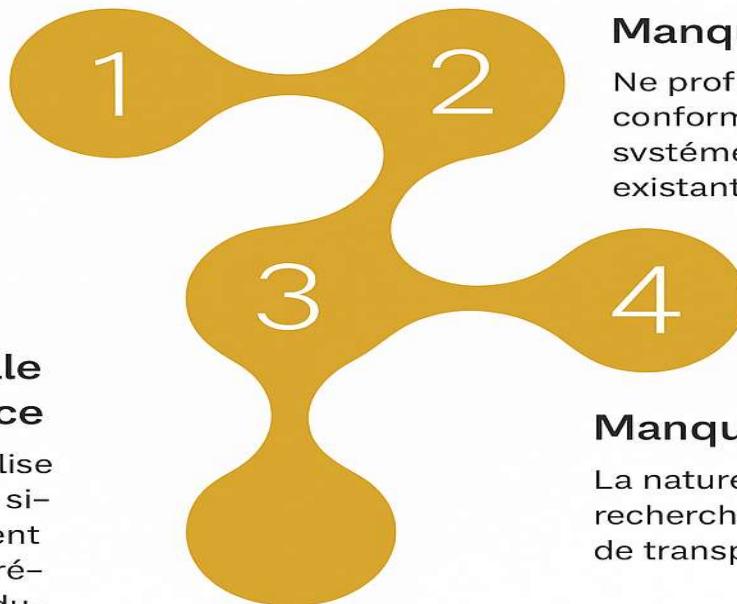
La similarité seule n'est pas suffisante pour un raisonnement riche et une explicabilité

Ne tire qu'une fraction de vos données

Au-delà des simples «métadonneés», les bases vectorielles S capturent pas les relations de données structurées.

Similarité vectorielle ≠ Pertinence

La recherche vectorielle utilise une mesure incomplète de similité. Se fier uniquement à cela peut produire des résultats non pertinents ou dupliqués. Il faut prendre le contexte en compte.



Manque de maturité

Ne profite pas de la sécurité, conformité et robustesse des systèmes de bases de données existants.

Manque d'explicabilité

La nature «boîte noire» de la recherche vectorielle manque de transparence et d'explicabilité.

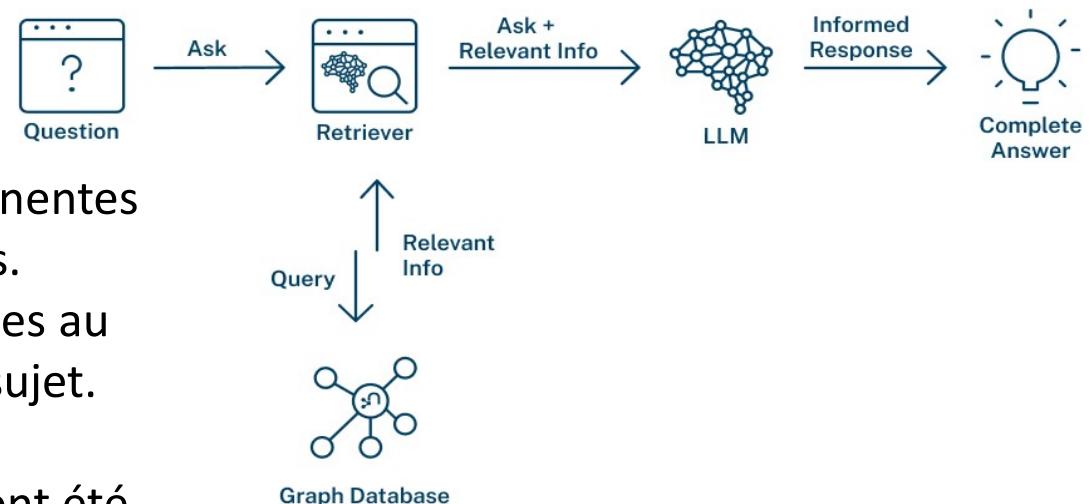
LLMs & Knowledge Graphs



<https://www.youtube.com/watch?v=XNneheyPg-6>

De RAG vers GraphRAG

Un graphe de connaissances Neo4j combiné avec des LLM offre des améliorations uniques :



Pertinence : Obtenez des réponses plus pertinentes comparées aux seules recherches vectorielles.

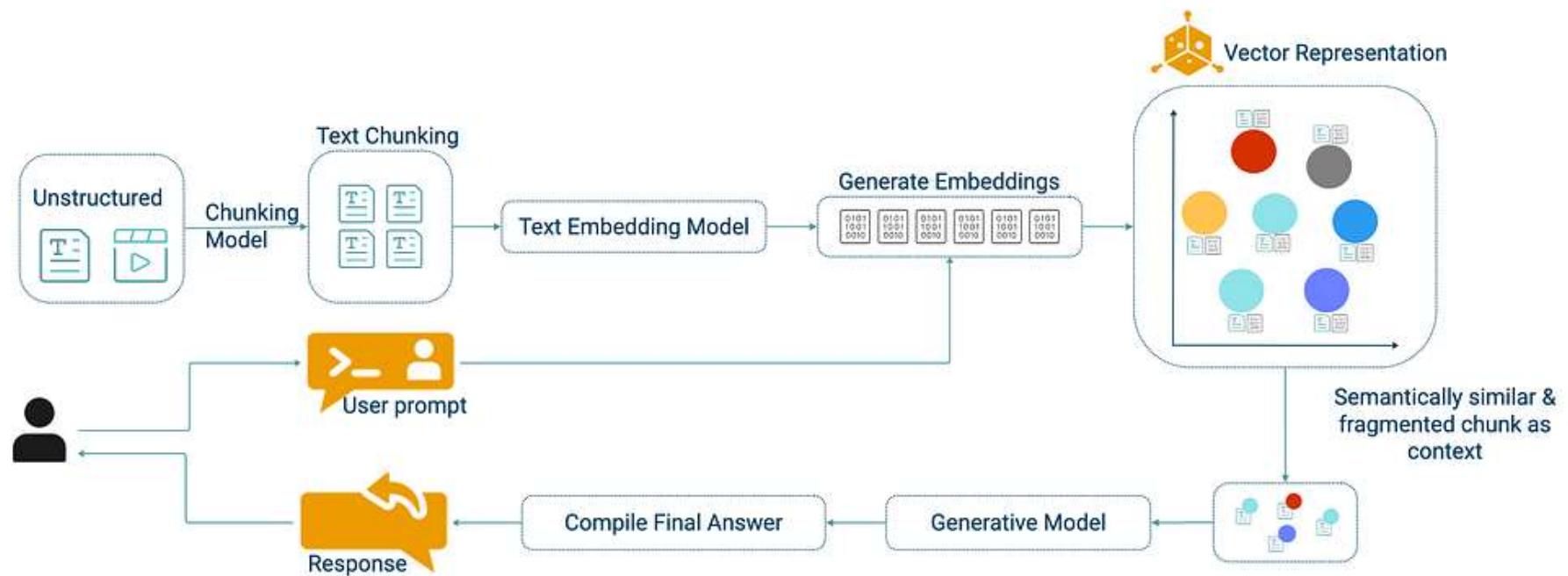
Contexte : Inclut des connaissances spécifiques au domaine, factuelles et structurées sur votre sujet.

Explicabilité : Fournit à l'utilisateur plus d'explications sur la façon dont les résultats ont été obtenus.

Sécurité : Contrôle d'accès basé sur les rôles.

<https://www.youtube.com/watch?v=XNneheyPg-6>

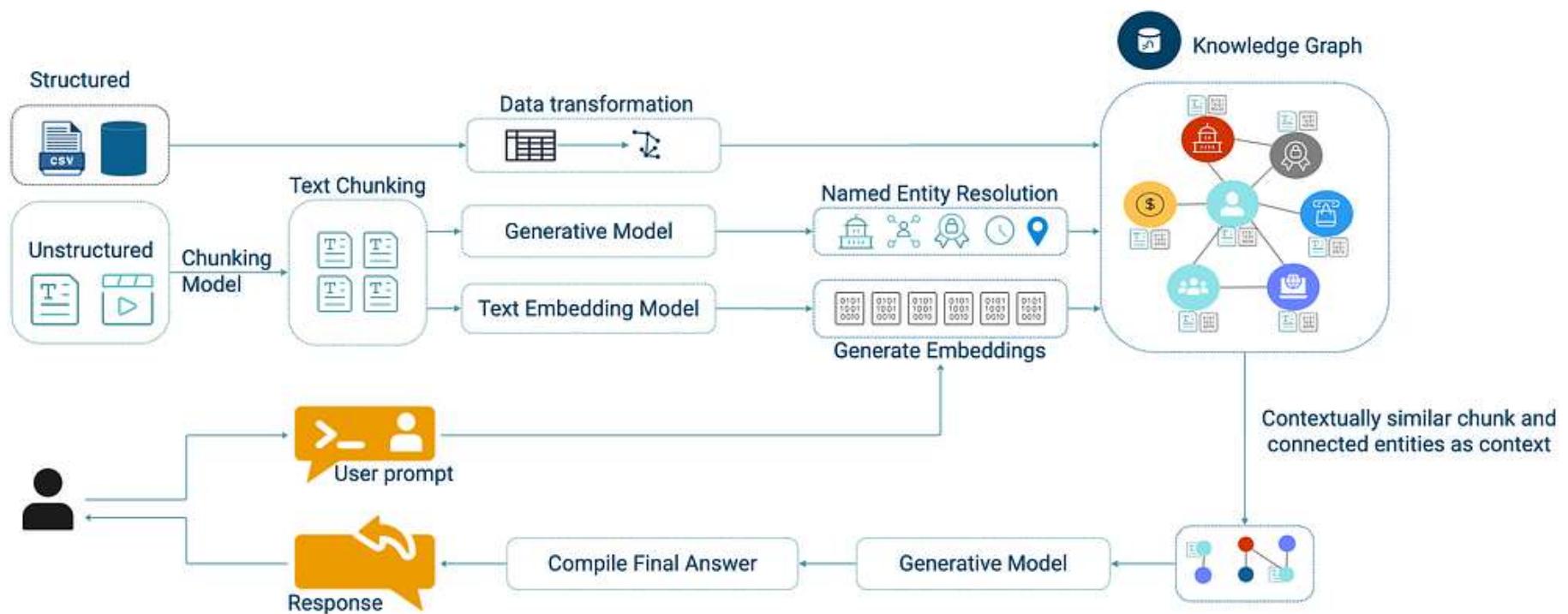
Traditional RAG architecture



<https://neo4j.com/blog/developer/graphrag-and-agentic-architecture-with-neoconverse>

/

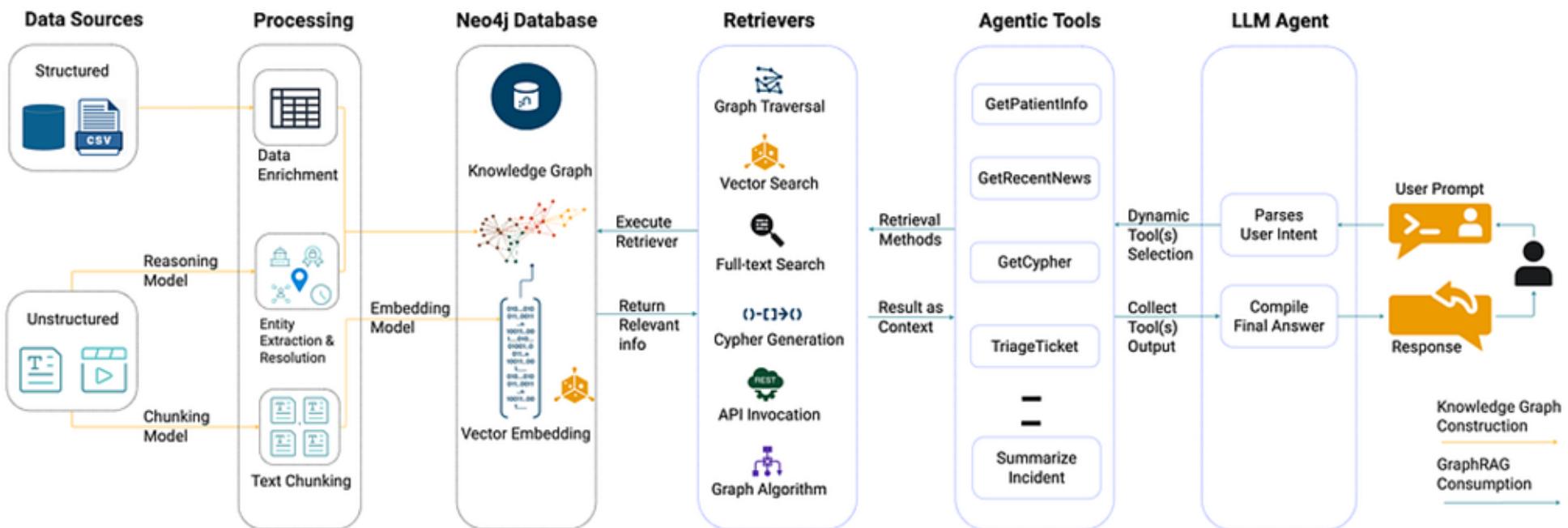
GraphRAG architecture



<https://neo4j.com/blog/developer/graphrag-and-agentic-architecture-with-neoconverse>

/

Agentic GraphRAG architecture



<https://neo4j.com/blog/developer/graphrag-and-agentic-architecture-with-neoconverse>

Ecosystème d'outils pour agents IA

Off the shelf tools



Focused on specific domains
Easy to use
Not customizable

Low-code / No-code tools



Can expand to multiple domains
Accessible
Customizable to a degree

AI agent frameworks



Scalable to all domains
Higher barrier to entry
Highly customizable

<https://www.youtube.com/watch?v=VAeQWMaPJk8>