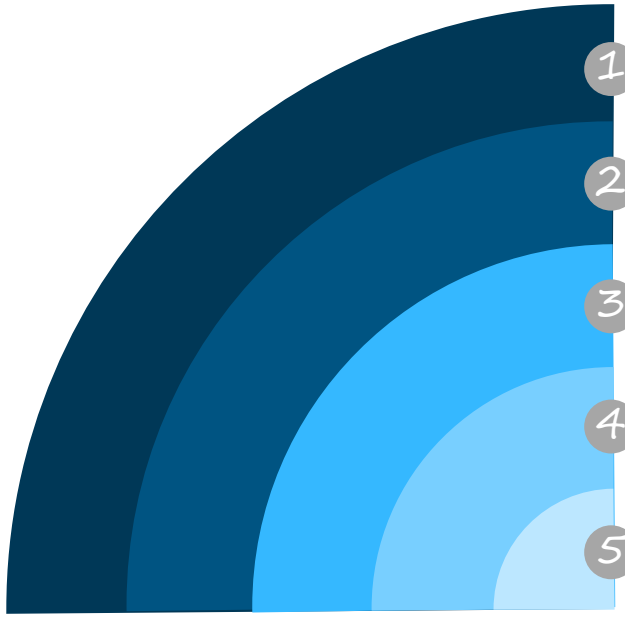
IBI2025_GenAI :

Customiser son modèle de langage

What is Generative AI ?

Generative AI refers to AI models capable of creating new content, like text, images, or code, based on patterns learned from training data



1 Artificial Intelligence	The science that enables machines to imitate human thinking
2 Machine Learning	Computer systems that learn from data to improve task performance without explicit instructions
3 Generative AI	AI techniques that lean form data about existing artifacts and use this to generate new artifacts
4 Foundation Models	Large ML models that are trained on unlabeled data and fine-tuned to wide range of application
5 ChatGPT, Dall-e, Gemini	Generative AI tools based on large models that transform prompts in new content (text, image, etc.)

TEXT

General writing Help
Sales emails
Marketing content
Support via chat/email

IMAGE

Design prototyping
Asset generation
Adhoc customization
3D models generation

VIDEO

Video Asset Curation
Video Generation
See Your Product

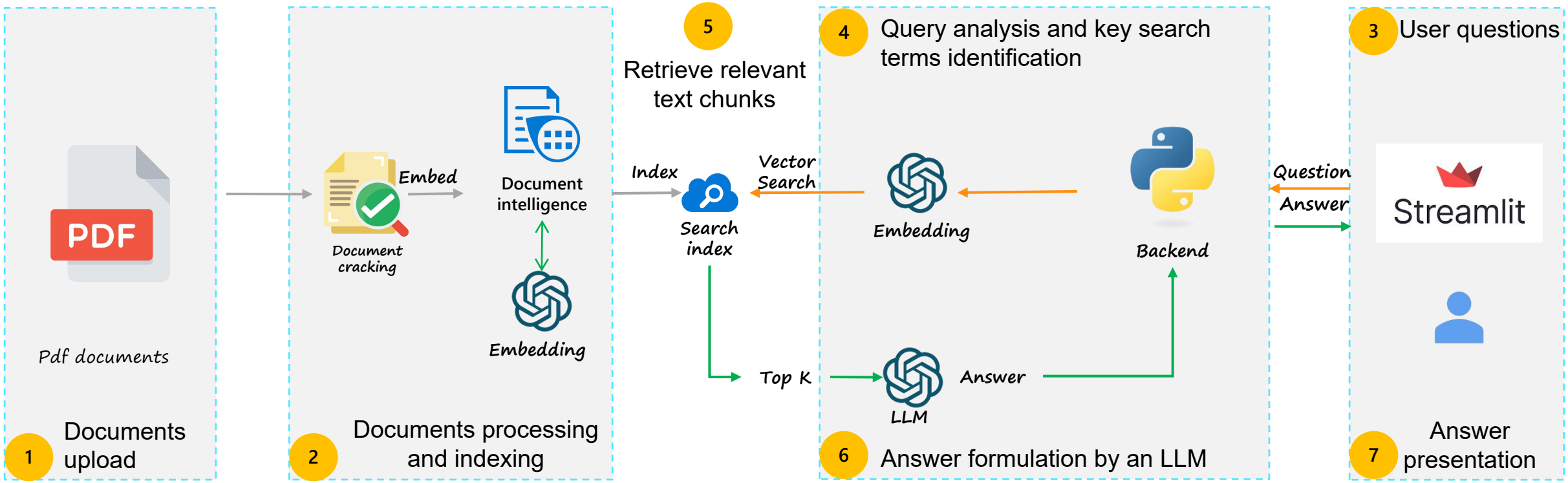
SPEECH

Voice Synthesis
Accent Cloning
Speech Understanding
Unscripted IVR

CODE

Programming Assist
Code documentation
Text to App builders
Codebase Translation

GenAI-based research assistant



Technologies à utiliser

Chroma

LangChain

Gemini

Google colab

Streamlit

FOUNDATION MODELS

Les modèles de langage les plus répandus sont des « modèles de fondation » (*foundation models*).

Ils sont caractérisés par :

- des capacités généralistes
- des connaissances limitées aux données fournies lors de l'entrainement initial
- un coût élevé en termes de ressources pour cet entrainement

OpenAI

GPT

Anthropic

Claude

Amazon

Titan

Hugging Face

BLOOM

Google

PALM

Meta

LLaMa

Mistral

Mistral

POURQUOI CUSTOMISER UN LLM ?

On peut être amené à nécessiter d'un LLM qu'il…

- Soit alimenté par des informations **privées**
- Soit alimenté par des informations en temps réel ou **actualisées** régulièrement
- Possède des connaissances spécifiques à un **domaine métier**
- Soit capable d'effectuer une **tâche** sous-représentée dans son corpus d'entrainement

3 STRATÉGIES DE CUSTOMISATION D'UN LLM

Basse complexité → Haute complexité

Prompt Engineering

Structurer son prompt pour inclure :
- du contexte
- des instructions
- des exemples
- spécifier un format de sortie attendu
- …

Retrieval Augmented Generation (RAG)

Fournir des données supplémentaires dans son prompt au travers d'un mécanisme de recherche préalable dans une base de connaissances.

Fine-tuning

Réentrainer quelques couches d'un LLM existant pour adapter le modèle à une tâche spécialisée.

STRATÉGIE PROMPT ENGINEERING

Conception de prompts robustes pour utiliser efficacement les LLM.

- Structure générale d'un prompt
  - Contexte — *Tu es en train de rédiger un résumé pour une présentation sur les énergies renouvelables.*
  - Instruction — *Résume le contenu du paragraphe suivant en une phrase.*
  - Données d'entrée — *« Les énergies renouvelables sont des sources d'énergie propres et durables, telles que… »*
  - Format de sortie — *Le résultat doit être un JSON au format* `{'reponse': '…' }`

Il existe de nombreux guides autour du prompt engineering :
https://www.promptingguide.ai/fr

PROMPT ENGINEERING

PROS

- Facilité de mise en place : aucune infrastructure supplémentaire nécessaire
- Possibilité d'apprendre au modèle des tâches simples (via few-shot prompting)

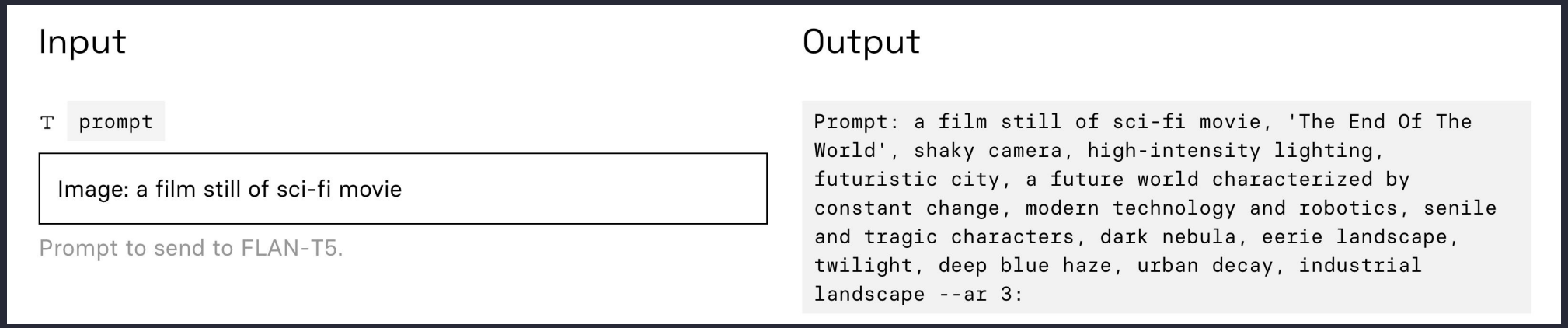
CONS

- Un prompt plus long impacte les performances de l'inférence
- Un prompt plus long sera facturé plus cher par les services SaaS

STRATÉGIE FINE-TUNING

On souhaite apprendre une nouvelle tâche au LLM. Cela demande un entrainement de certaines couches du LLM à l'aide d'un jeu de données montrant des exemples de l'attendu, une opération gourmande en ressources.

Ex de tâche à apprendre : Extrapoler une description simple en une instruction raffinée pour un générateur d'image



FINE-TUNING

PROS

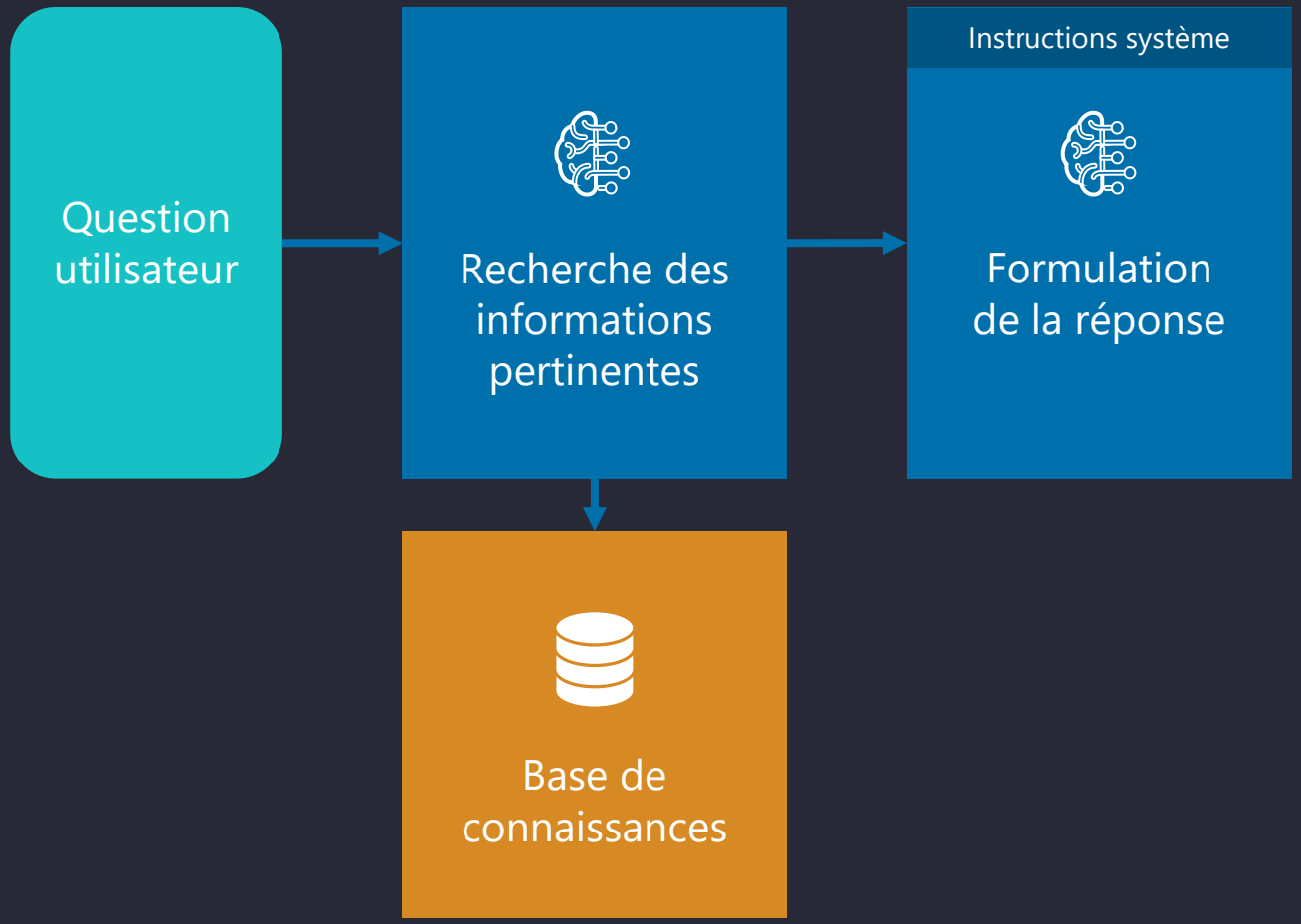
- Optimisation du modèle sur une tâche précise, potentiellement complexe et spécialisée
- Performances accrues étant donné que le prompt peut être très succinct

CONS

- Besoin important en ressources (GPU) pour effectuer le fine-tuning
- Expertise en machine learning requise selon la complexité de la tâche à apprendre
- Facturation importante de l'hébergement des modèles fine-tunés par les hyperscalers

STRATÉGIE RAG

Un RAG (Retrieval-Augmented Generation) est un modèle de génération de texte qui utilise une étape de récupération d'informations pour améliorer la qualité de la génération.



Prompt final

Instructions système :
Vous êtes un assistant pour les chercheurs. Votre tâche est de répondre aux questions des utilisateurs.

Question utilisateur :
Quelle est la définition de l'IA générative ?

Eléments de contexte trouvés en base de connaissances :
«Generative AI refers to AI models capable of creating new content, like text, images, or code, based on patterns learned from training data»

RETRIEVAL-AUGMENTED GENERATION

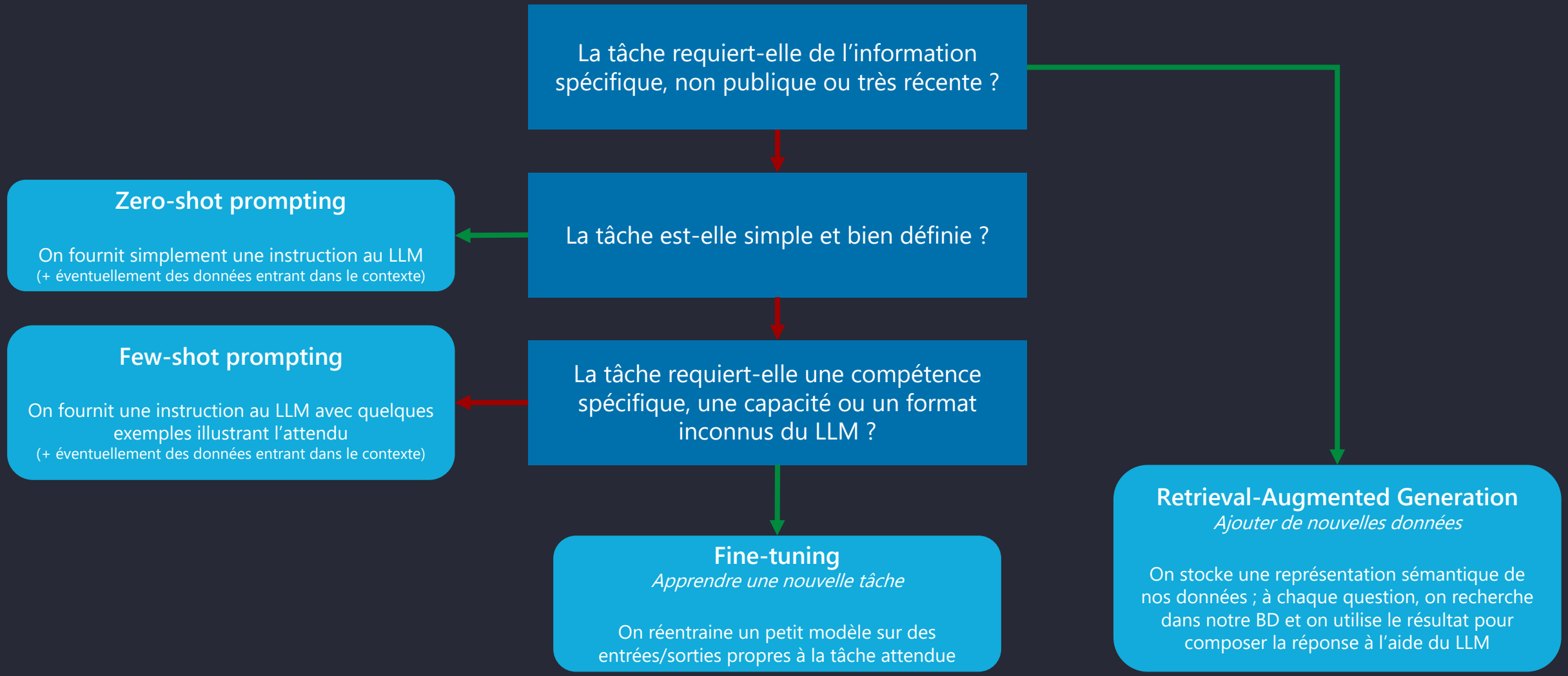
PROS

- Capacité à prendre en compte une documentation externe d'une volumétrie importante
  (là où le prompt engineering serait limité par la taille de la fenêtre de contexte maximale du LLM)
- Facilité de mise à jour des données exploitées par le LLM
  (là où le fine-tuning coûterait trop cher à mettre à jour régulièrement)

CONS

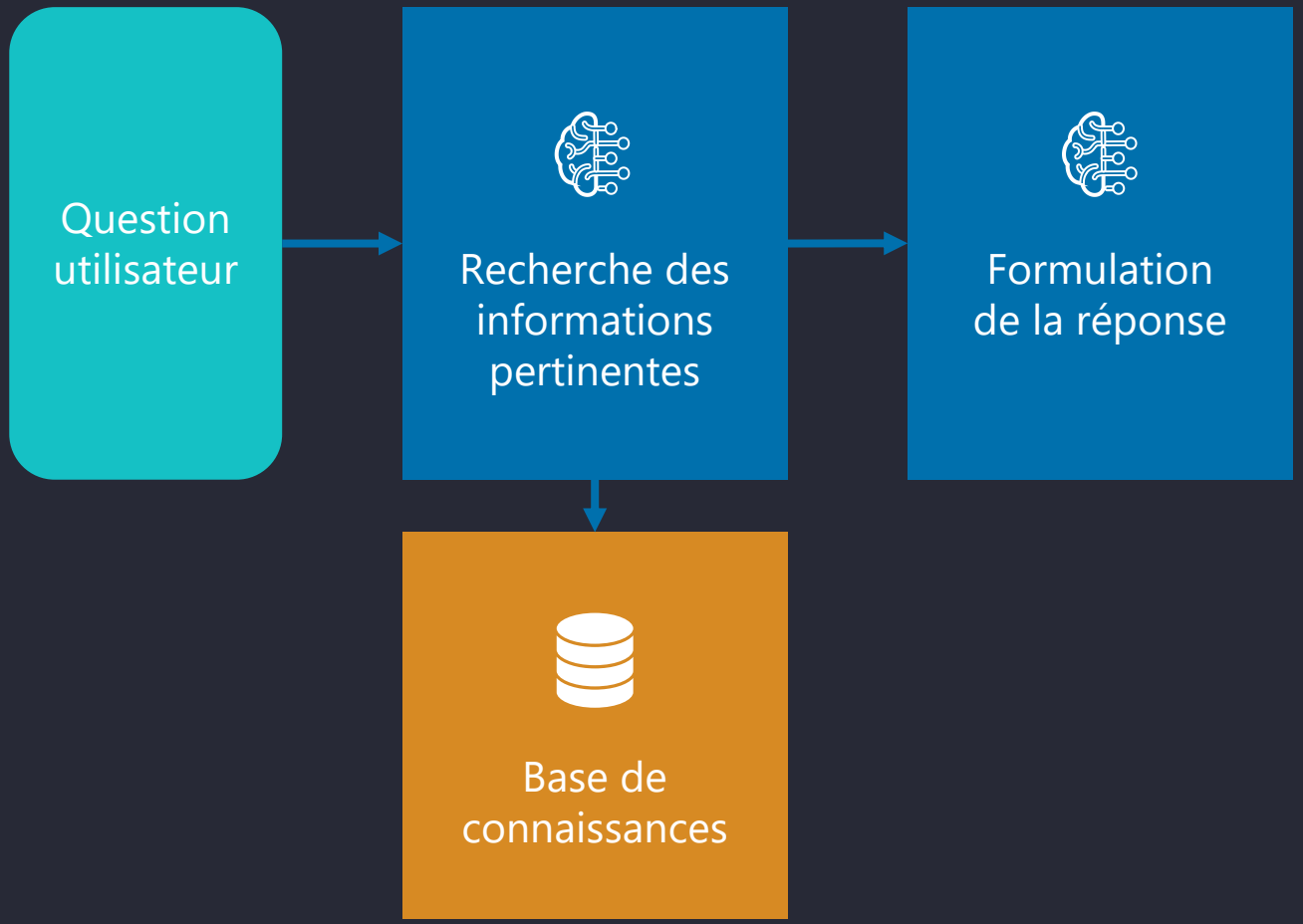
- Performances dépendantes de l'étape supplémentaire de recherche
- Le modèle n'est pas aussi bon que la qualité de la base de connaissances sous-jacente

STRATÉGIES D'UTILISATION D'UN LLM



Retrieval-Augmented Generation : Comment ça marche ?

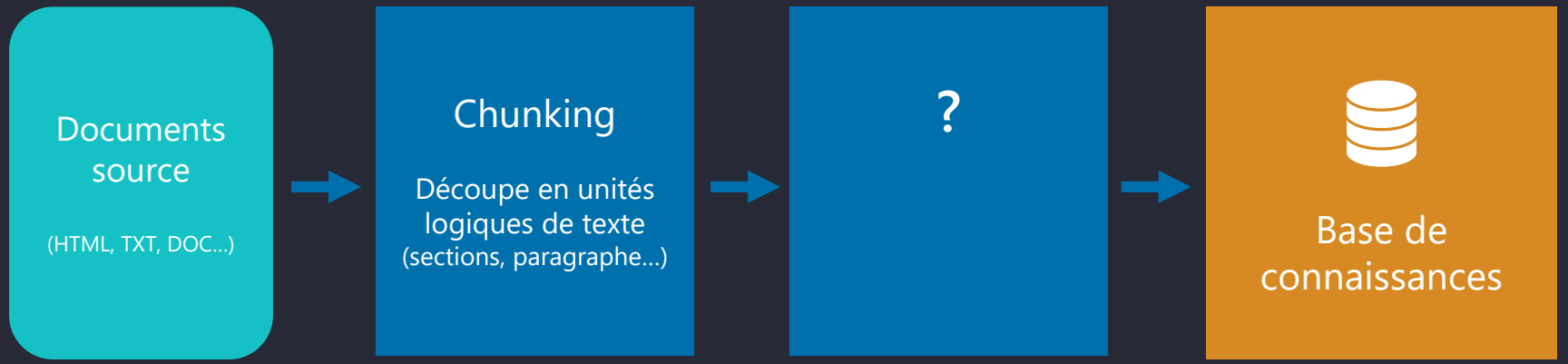
RETRIEVAL-AUGMENTED GENERATION (RAG)



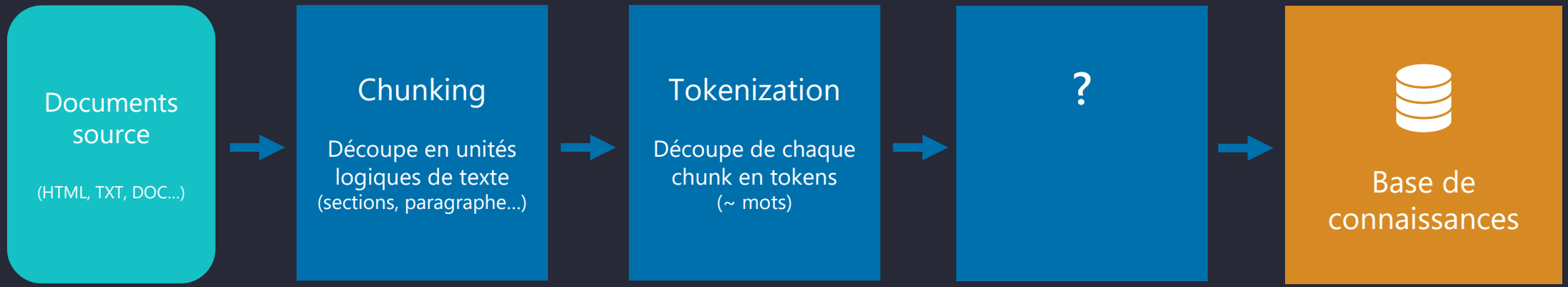
RETRIEVAL-AUGMENTED GENERATION (RAG)

RETRIEVAL-AUGMENTED GENERATION (RAG)

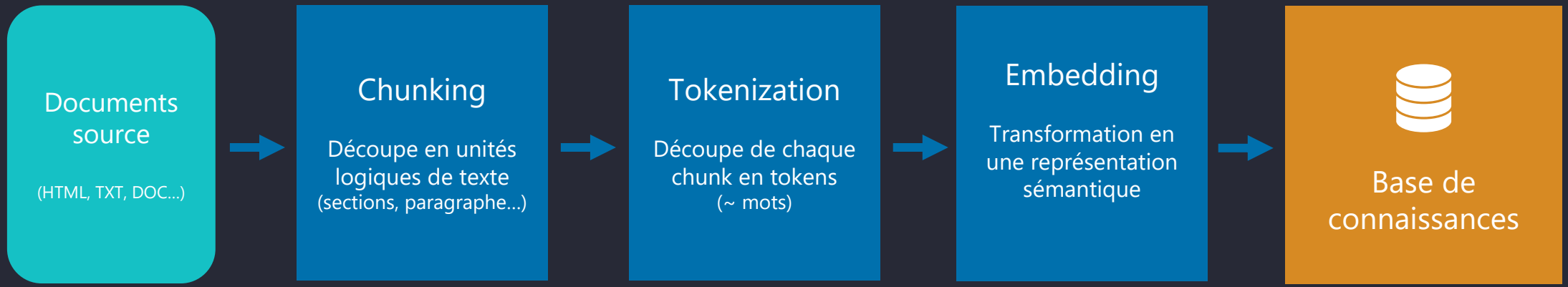
RETRIEVAL-AUGMENTED GENERATION (RAG)



RETRIEVAL-AUGMENTED GENERATION (RAG)



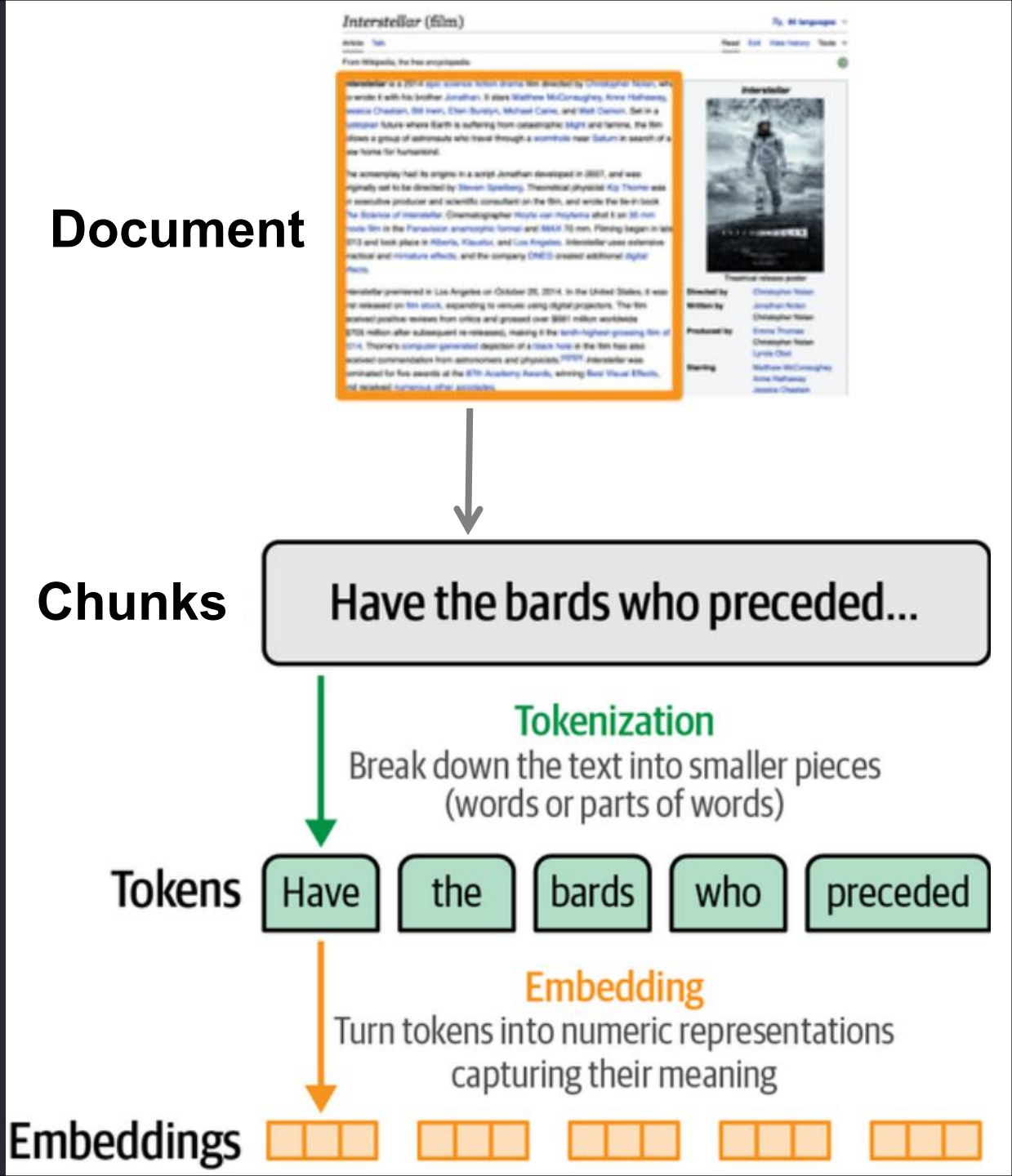
RETRIEVAL-AUGMENTED GENERATION (RAG)



DU TEXT VERS DES EMBEDDINGS

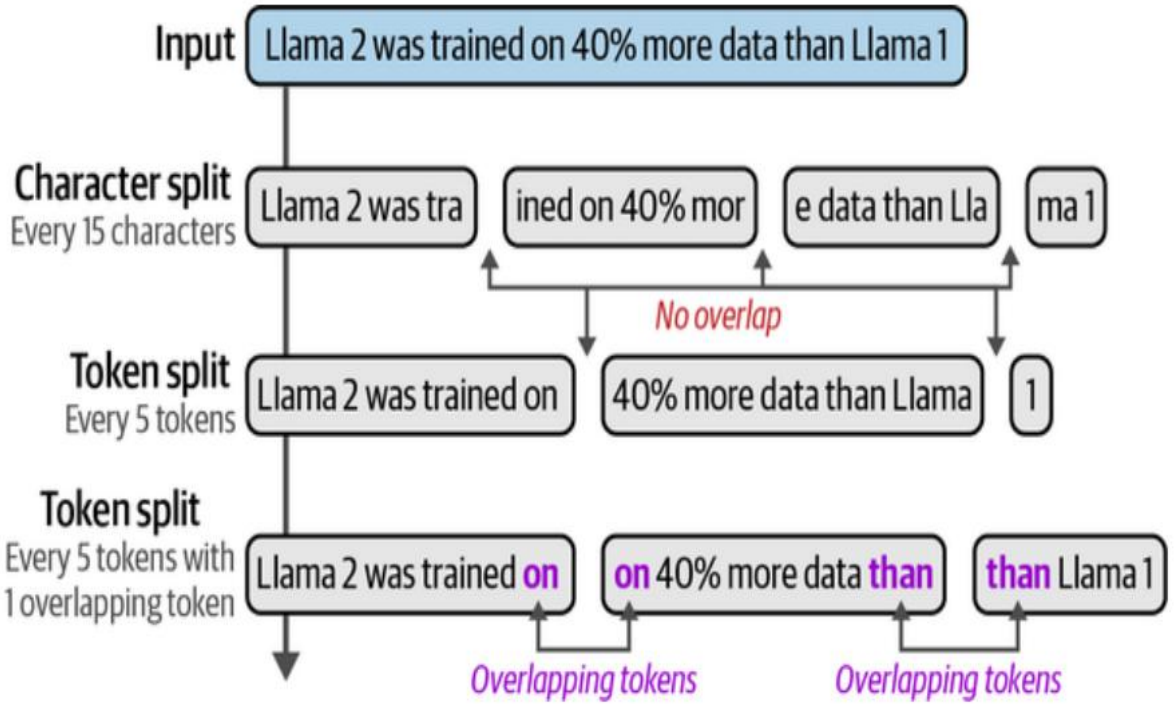
Le traitement d'un texte (document) contient :

- Chunking : Découpe en unités logiques de texte (sections, paragraphe…)
- Tokenization : Découpe de chaque chunk en tokens (~ mots)
- Embeddings : Transformation du token en une représentation sémantique



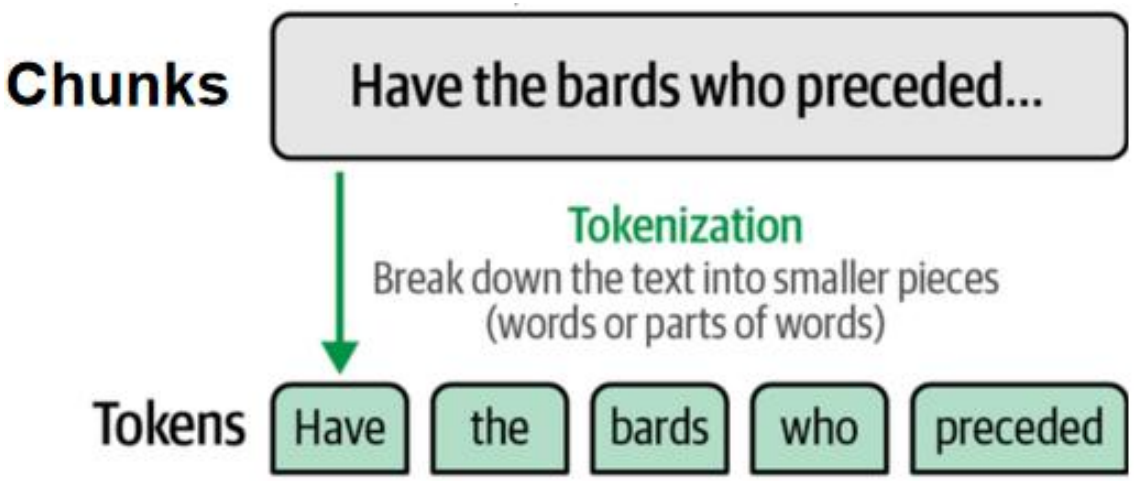
CHUNKING

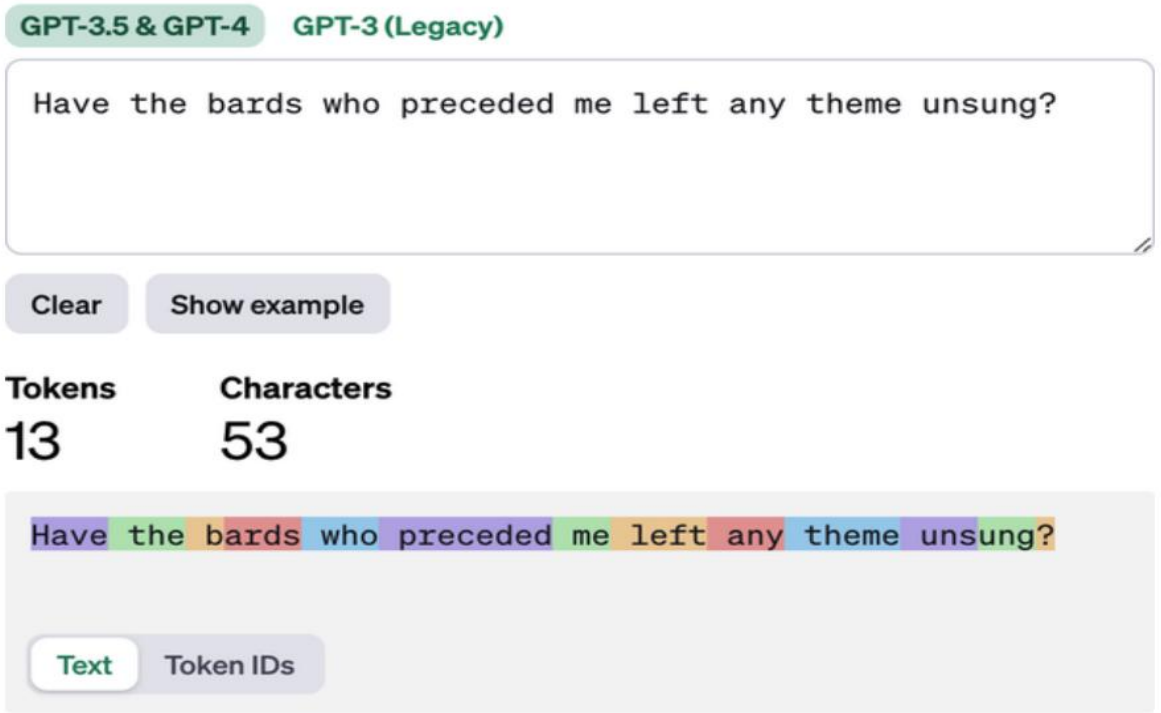
Découpe en unités logiques de texte (sections, paragraphe…)



TOKENIZATION

Transformer une séquence de caractères continue en une liste d'éléments discrets (mots, sous-mots, symboles) manipulables par la machine.

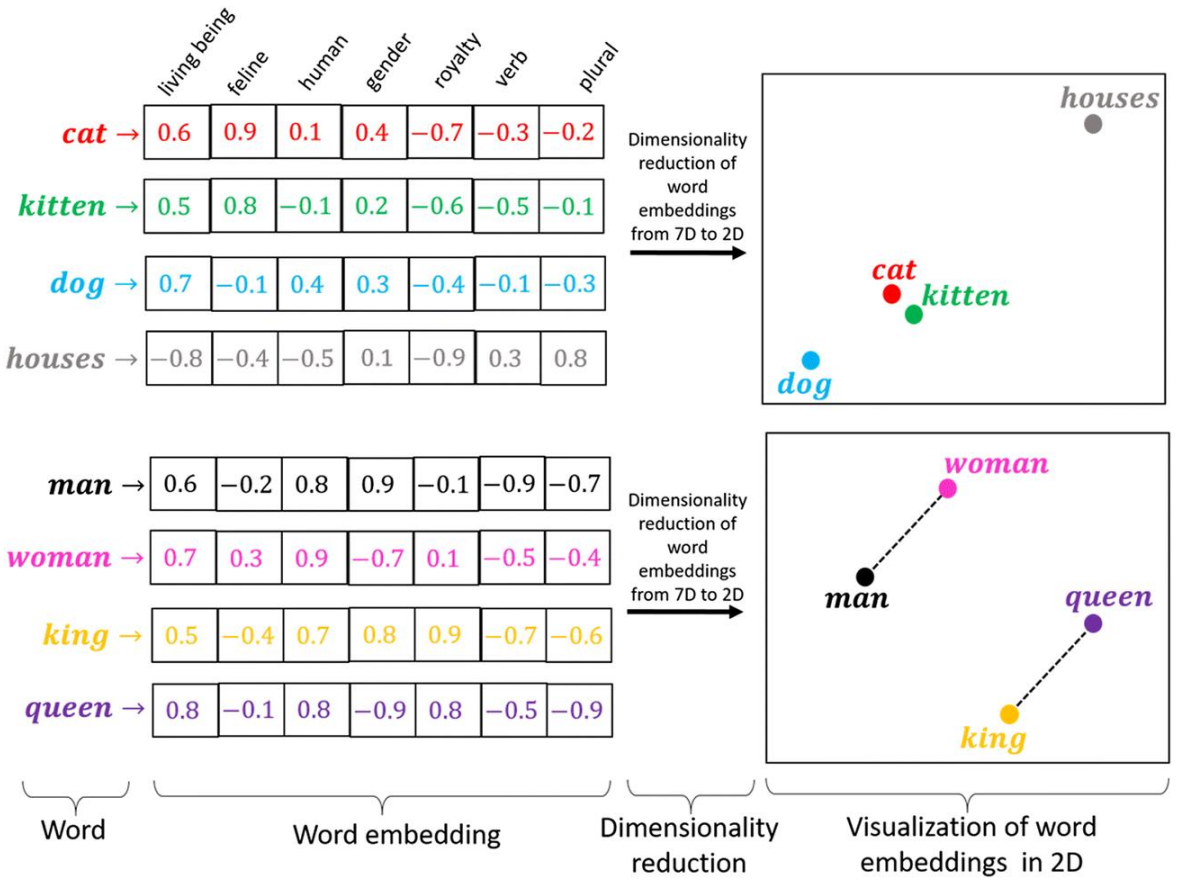




EMBEDDING

- L'embedding est un traitement qui transforme des mots en vecteurs de valeurs numériques
- Ce processus de conversion est aujourd'hui réalisé par un type de LLM dédié à cette tâche, l'*embedding model*

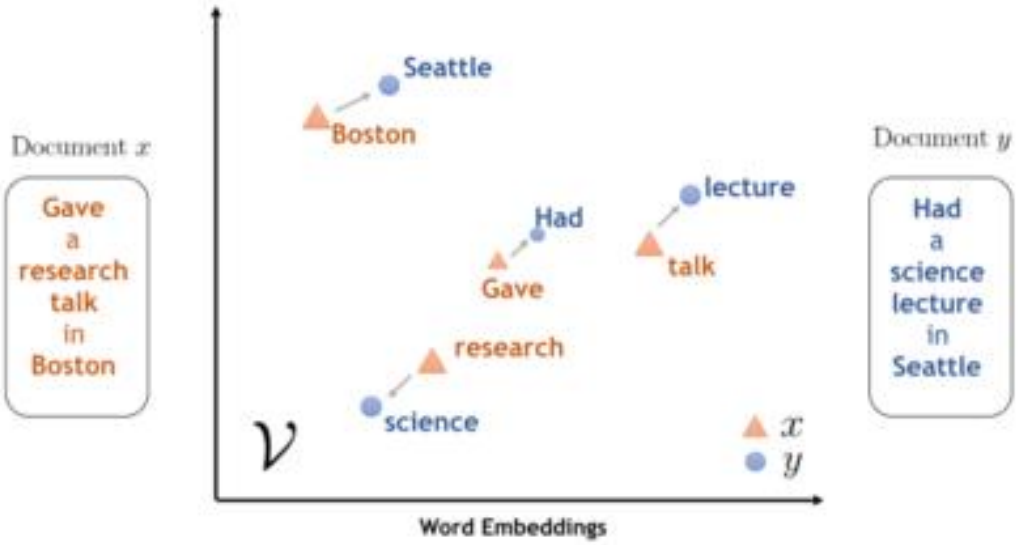
Projections dans un espace sémantique



EMBEDDING

- L'embedding est un traitement qui transforme des mots en vecteurs de valeurs numériques
- Ce processus de conversion est aujourd'hui réalisé par un type de LLM dédié à cette tâche, l'*embedding model*
- La grande force de ces représentations provient du rapprochement naturel de mots synonymes, qui permettra à la recherche de ne pas dépendre de mots-clés exacts

Projections dans un espace sémantique



VECTORSTORES

Pour stocker ces vecteurs numériques, des bases de données spécialisées se sont développées (bases de données orientées vecteurs ou *vectorstores*) au travers de solutions SaaS ou on-premise

Pinecone
(SaaS)

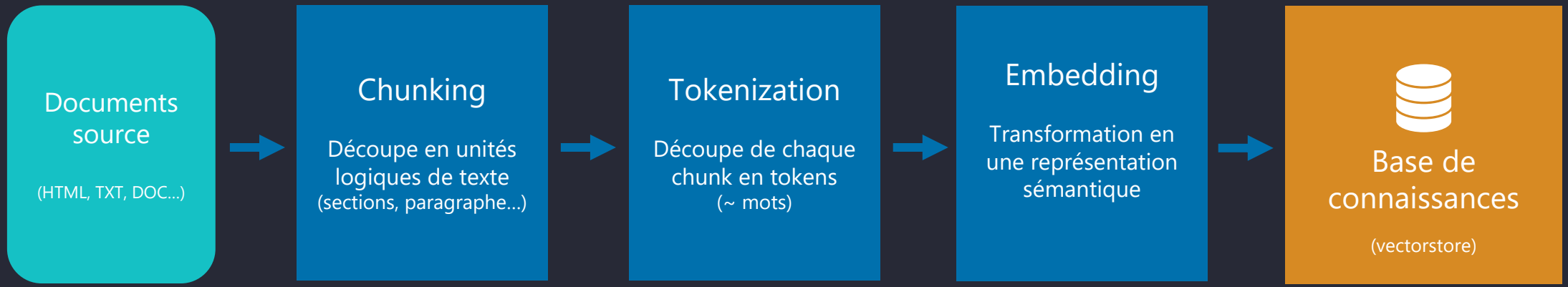
Chroma
(SaaS & On-prem)

FAISS
(On-prem)

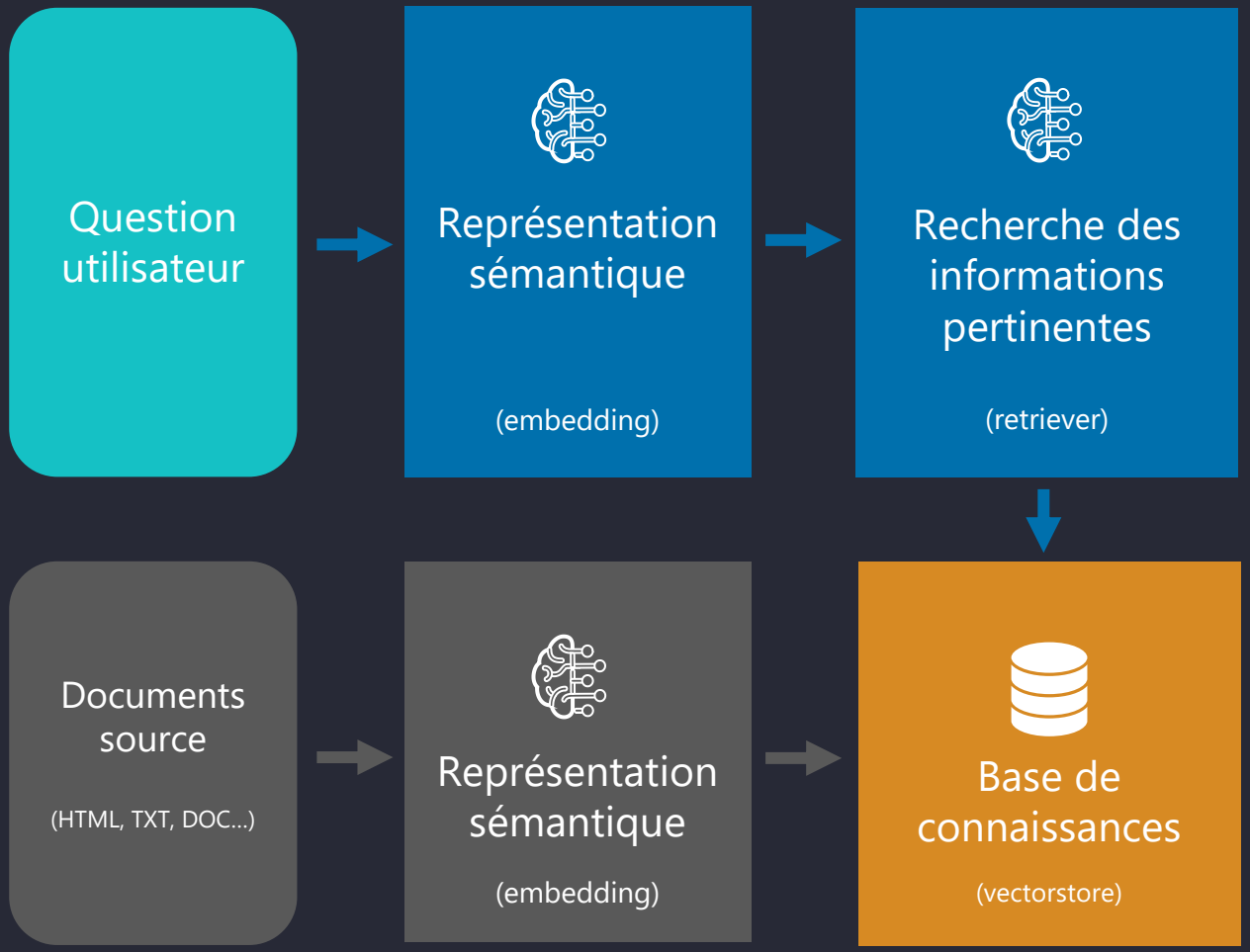
Azure Search
(SaaS)

Supabase
(SaaS & On-prem)

RETRIEVAL-AUGMENTED GENERATION (RAG)

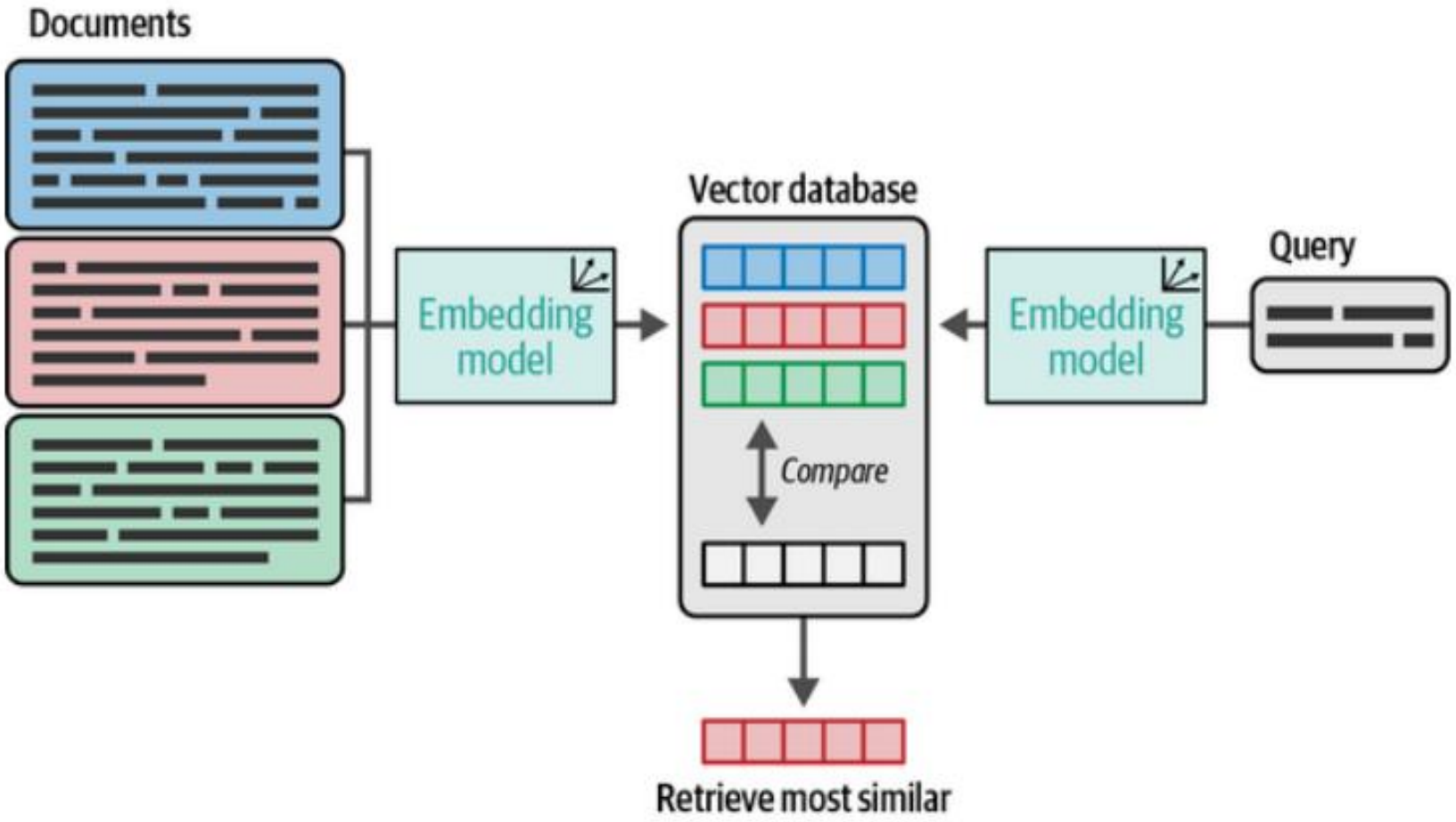


RETRIEVAL-AUGMENTED GENERATION (RAG)

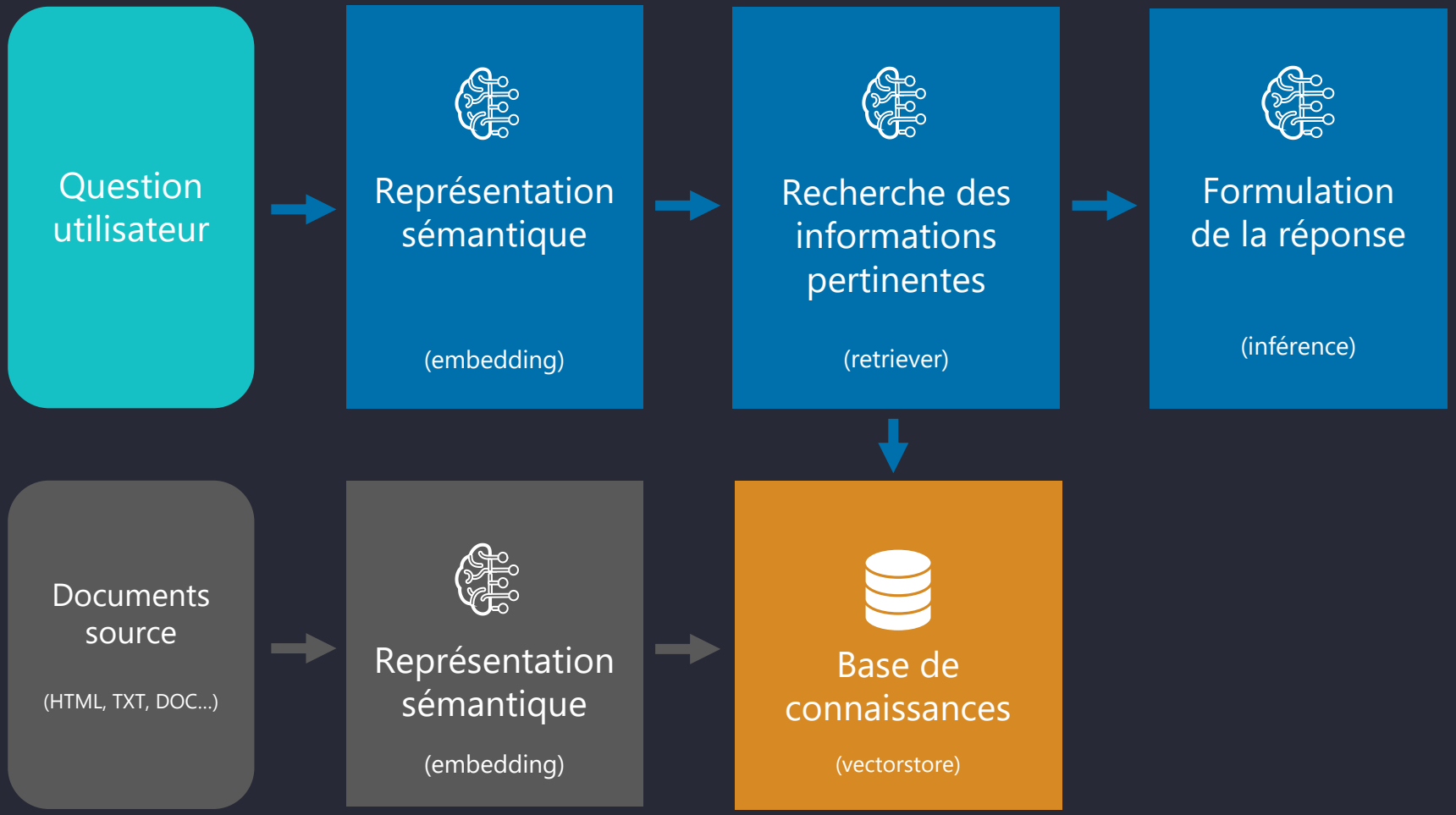


RETRIEVEMENT

- Une fois la requête vectorisée, il faut trouver les vecteurs les plus proches dans notre archive de textes



RETRIEVAL-AUGMENTED GENERATION (RAG)



RETRIEVAL-AUGMENTED GENERATION (RAG)