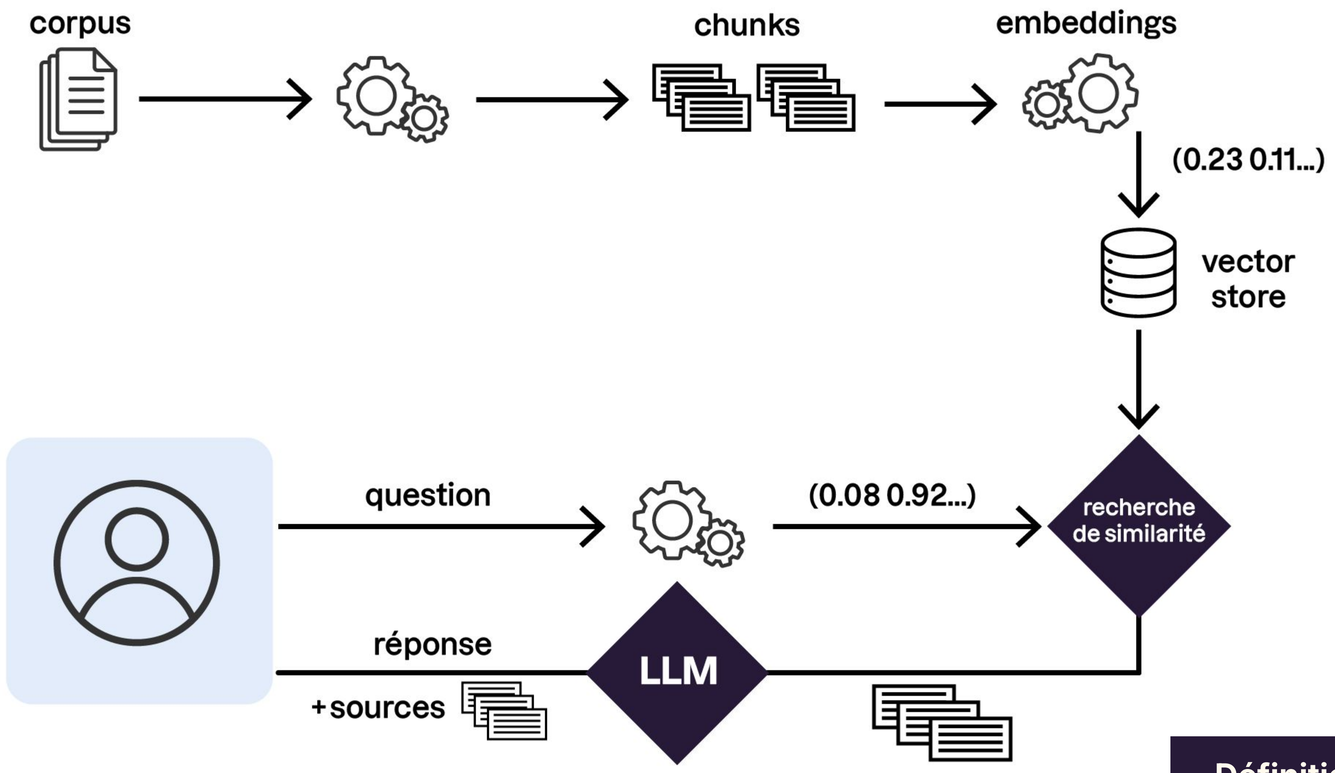
Fiche récap : Mettez en place un RAG pour un LLM

Data

1. Préparer une base de données vectorielle adaptée à vos documents.
2. Transformer les données textuelles en vecteurs pour la recherche sémantique.
3. Créer un index avec FAISS pour interroger efficacement les vecteurs.
4. Intégrer un LLM avec RAG pour générer des réponses contextualisées.
5. Développer une interface de chat avec Streamlit connectée à votre système RAG.



Bonnes pratiques 👍

- ✅ Structurer les prompts système pour cadrer le comportement du LLM.
- ✅ Utiliser des embeddings adaptés à la langue et au domaine.
- ✅ Choisir un modèle selon le besoin, performance et coût.
- ✅ Nettoyer et préparer les données avant vectorisation.
- ✅ Adopter le chunking contextuel pour découper vos documents.
- ✅ Limiter les hallucinations par des instructions claires.
- ✅ Testez avec des scénarios réels pour valider les réponses.
- ✅ Optimiser les requêtes utilisateurs pour des réponses ciblées.

Erreurs classiques 👎

- ❌ N'utiliser aucune stratégie de chunking, au risque d'une perte de contexte.
- ❌ Choisir un modèle non multilingue pour un projet en français.
- ❌ Ignorer les limites de tokens des LLMs.
- ❌ Stocker les clés API en clair dans le code.
- ❌ Faire du fine-tuning inutilement, surtout pour de petits cas.
- ❌ Négliger la gouvernance des données (RGPD, sécurité).
- ❌ Réinjecter trop de contexte, ce qui rend le prompt confus.
- ❌ Ne pas tester la qualité des embeddings, surtout sur corpus local.

Définitions 🔍

RAG (Retrieval-Augmented Generation)
Méthode combinant recherche documentaire et génération de texte par LLM.

Embedding
Vecteur numérique représentant la signification d'un texte.

Token
Unité de texte traitée par un LLM (mot, partie de mot ou caractère).

Base vectorielle
Base de données stockant des vecteurs pour recherche sémantique.

Prompt système
Instruction initiale définissant le rôle et les limites du LLM.