Comparaison entre ton RAG actuel et un RAG classique avec embeddings

# 1. Ton RAG actuel (basé sur Create\_RAG.py)

- Pas d'embedding ni de recherche vectorielle.  
- Le dataset contient des contextes déjà préparés ('correct\_doc', 'wrong\_doc', 'fake\_context').  
- À chaque question, un contexte est choisi aléatoirement (correct ou faux).  
- Le contexte choisi est injecté directement dans le prompt envoyé au LLM (phi3:mini via Ollama).  
- Pas de mécanisme de retrieval dynamique selon la question.  
- Taille du contexte limitée à la fenêtre du modèle (4k à 8k tokens environ).  
- Permet de tester la robustesse du modèle aux erreurs de contexte.

# 2. RAG classique (retrieval-augmented generation avec embeddings)

- Utilise des embeddings vectoriels pour représenter chaque morceau de document.  
- Découpe préalable des documents en petits chunks (ex : 300-1000 tokens).  
- À chaque question de l'utilisateur :  
 \* On encode la question avec le même modèle d'embedding.  
 \* On recherche les passages les plus similaires (top-k) via une base vectorielle (FAISS, Chroma, Milvus, etc).  
 \* Seuls ces passages pertinents sont injectés dans le prompt envoyé au LLM.  
- Permet de travailler avec des bases de connaissances volumineuses (plusieurs Go de documents).  
- Réduit le bruit et augmente la pertinence des réponses.  
- Optimise l'utilisation de la fenêtre contextuelle du LLM (max 4k, 32k, 100k tokens selon le modèle).

# 3. Synthèse rapide

|  |  |  |
| --- | --- | --- |
|  | Ton RAG actuel | RAG classique avec embeddings |
| Préparation des données | Manuelle (contextes déjà fournis) | Automatique (découpage + embeddings) |
| Sélection du contexte | Aléatoire (correct/faux) | Recherche vectorielle (similarité question/contenu) |
| Volume de données | Petit dataset (quelques Mo) | Possibilité de très gros corpus (plusieurs Go) |
| Fenêtre contextuelle | Utilisée entièrement pour chaque exemple | Optimisée sur les passages pertinents |
| Robustesse testée ? | Oui (test sur contextes faux) | Indirectement (retriever filtre les erreurs) |
| Complexité d'implémentation | Simple | Moyenne (embeddings, base vectorielle) |