



Découverte du RAG

Retrieval Augmented Generation

Sommaire

1. Découverte du RAG
2. Sommaire
3. Qui suis-je
4. Définition
5. Exemple
6. Concrètement
7. Embedding
8. Base de données vectorielle
9. Service tout en 1
10. Retrieval
11. Conclusions
12. Documentation

Qui suis-je

Victor Santelé

Étudiant à Unamur en informatique à finalité Data Science

- Ancien étudiant en IG à l'Henallux
- Dev autodidacte depuis une dizaine d'année
- Ancien stagiaire du MIC (2022)
- 6x Azure Certified
- Fan de serverless

Contact:

Twitter: @vsantele

LinkedIn: Victor Santelé

Définition

RAG (Retrieval Augmented Generation)

- Technique de grounding
- Design pattern d'IA Generative
- Combine LLM et sources de données externes
- Utilise (souvent) une DB vectorielle
- Permet de générer des réponses plus précises
- Plus simple et moins chère que le fine-tuning

Exemple

Copilot Chat

Utilise le contenu de la page pour améliorer son contexte et sa précision.

Pic vert

Picus viridis

Article Discussion Outils


★ Vous lisez un « [article de qualité](#) » labellisé en 2024.

[Picus viridis](#) redirige ici. Pour les autres taxons précédemment traités sous ce nom, voir [Pic de Sharpe](#) et [Pic de Levallant](#).

Le **Pic vert** (*Picus viridis*), aussi connu sous le nom vernaculaire de **Pivert**, est une [espèce](#) d'[Oiseaux](#) de l'[ordre](#) des [Piciformes](#) et de la [famille](#) des [Picidae](#). Il est commun à très commun dans son [aire de répartition](#), qui couvre l'ensemble de l'[Europe de l'Ouest](#) (excepté l'[Irlande](#), la [péninsule Ibérique](#) et une partie du [piémont pyrénéen](#) en France), de l'[océan Atlantique](#) jusqu'à la [Volga](#), en [Russie](#).

Comme son nom l'indique, il est majoritairement vert, avec un [croupion](#) jaune bien visible en vol. Il porte une calotte rouge, avec une face noire striée de blanc, ainsi qu'une moustache rouge cernée de

Picus viridis



You

D'après le site web, que mange le pic-vert?

Copilot

D'après le site web, le pic-vert se nourrit principalement de fourmis qu'il capture au sol ou dans les fourmilières [1](#) [2](#) . Il complète son alimentation avec d'autres invertébrés, des fruits, des graines et parfois des abeilles [3](#) .

References: fr.wikipedia.org

[1](#) [2](#) [3](#)

👍 🔄 📄 ⬇️ 🔊 1 of 30

🔍 Quelle est la taille d'un pic-vert?

Comment les pics-verts se reproduisent-ils?

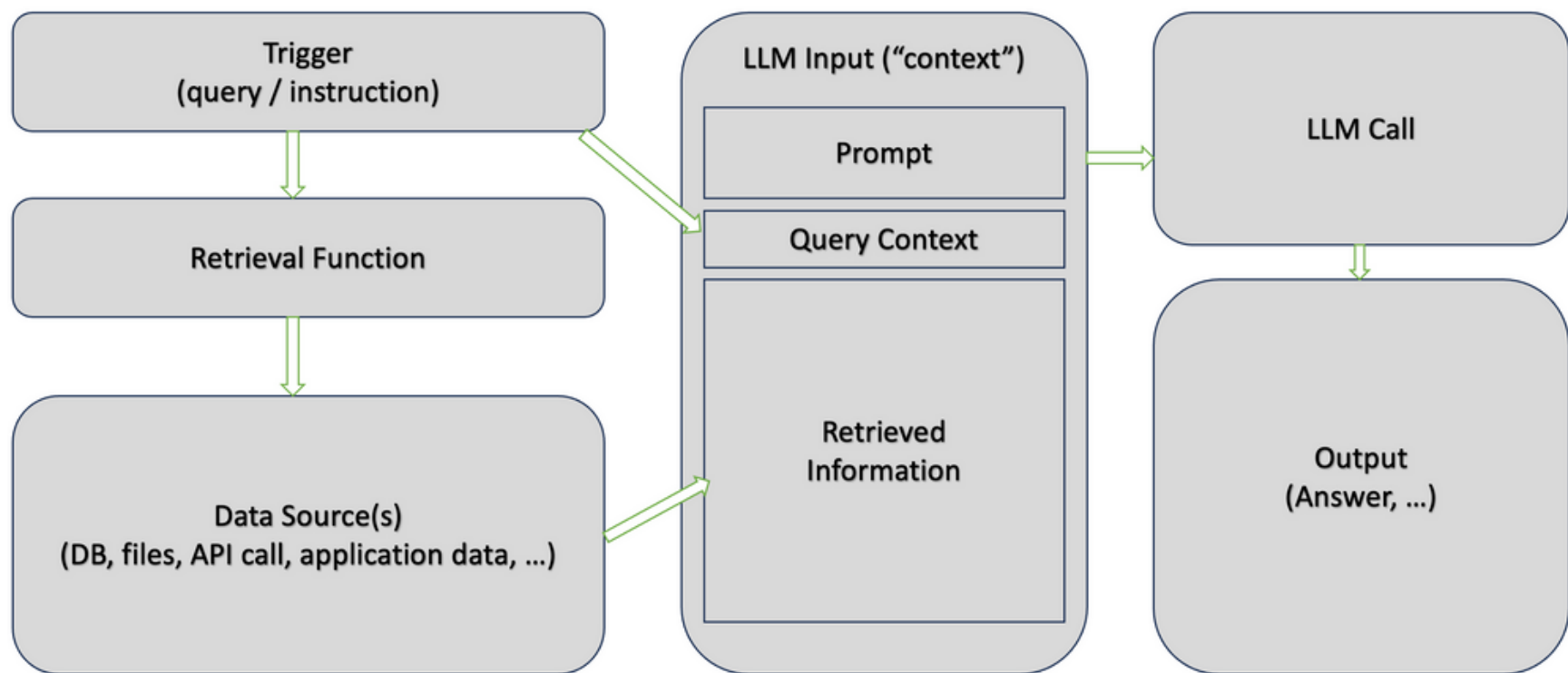
Où vivent les pics-verts ?

Ask me anything...

0/2000

Concrètement

- Besoin de données structurées ou non structurées
- Utilisation de DB vectorielle
- Embedding des documents
- Embedding des requêtes
- Recherche des plus proches voisins
- Utilisation des plus proches voisins pour améliorer la génération



Embedding

- Utilisation de modèles de langage pour transformer les données en vecteurs
- Permet de garder la sémantique du texte
- Permet de faire des recherches rapides avec un autre vecteur
- Plusieurs use-cases: recherche, clustering, classification, recommandations, détection d'anomalies, ...

Le plus connu: `text-embedding-ada-002`

Nouveau: `text-embedding-3-small` et `text-embedding-3-large`

Local: `all-MiniLM-L6-v2`

Classement: <https://huggingface.co/spaces/mteb/leaderboard>

Documentation: <https://platform.openai.com/docs/guides/embeddings/use-cases>

Base de données vectorielle

- Optimiser pour le stockage et la recherche de vecteurs
- Permet de stocker n'importe quel type de vecteurs
- Plusieurs méthodes: FAISS, Pinecone, QDrant, CosmosDB (MongoDb vCore et Postgres)
- Recherche via graph, arbre ou hash (pour info)

Documentation: <https://learn.microsoft.com/en-us/azure/cosmos-db/vector-database>

Service tout en 1

Azure Ai Search

- Full-text search
- Vector similarity search
- Facilité d'accès granulaire
- Facile d'utilisation

Liste des features: <https://learn.microsoft.com/en-us/azure/search/search-features-list>

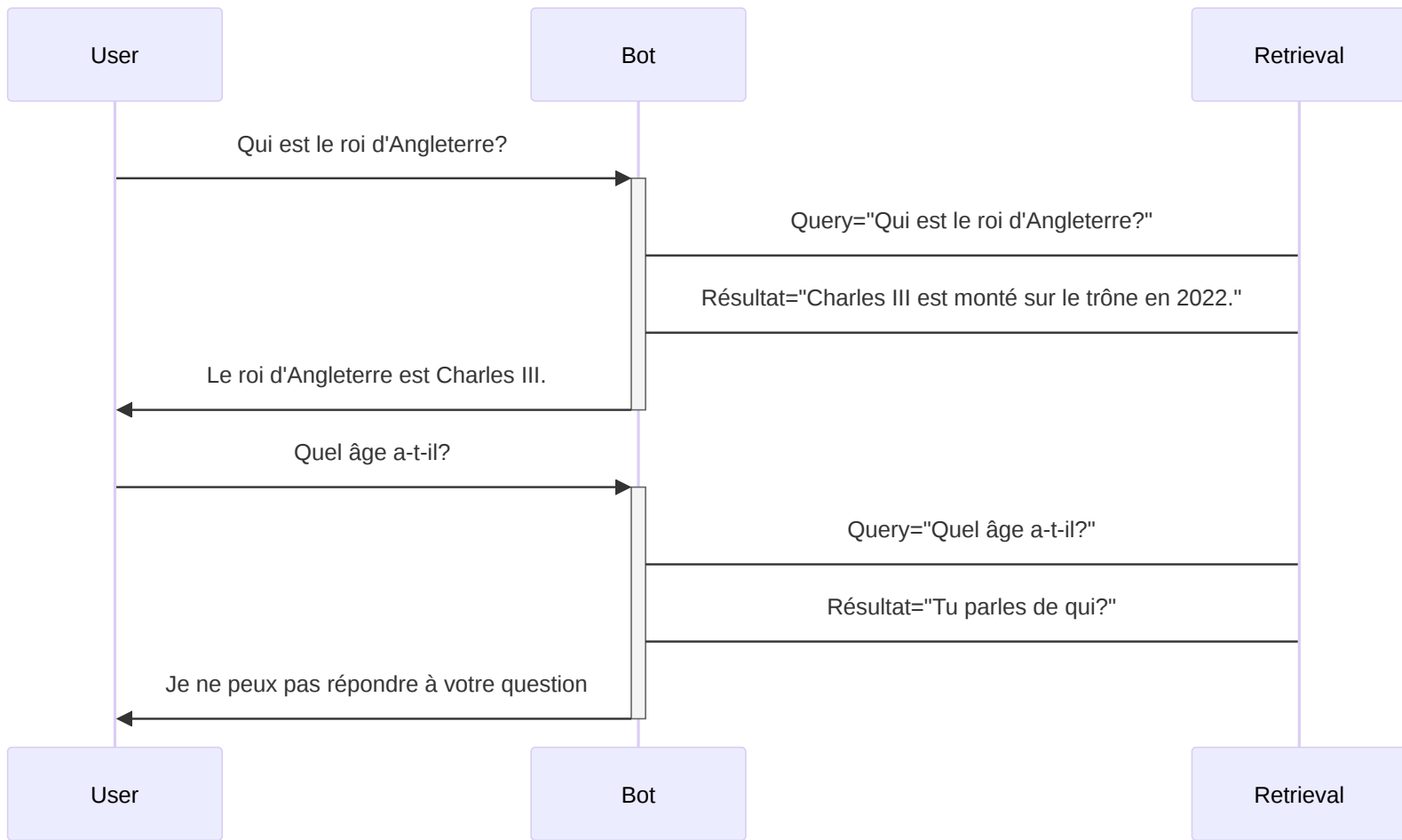
Retrieval

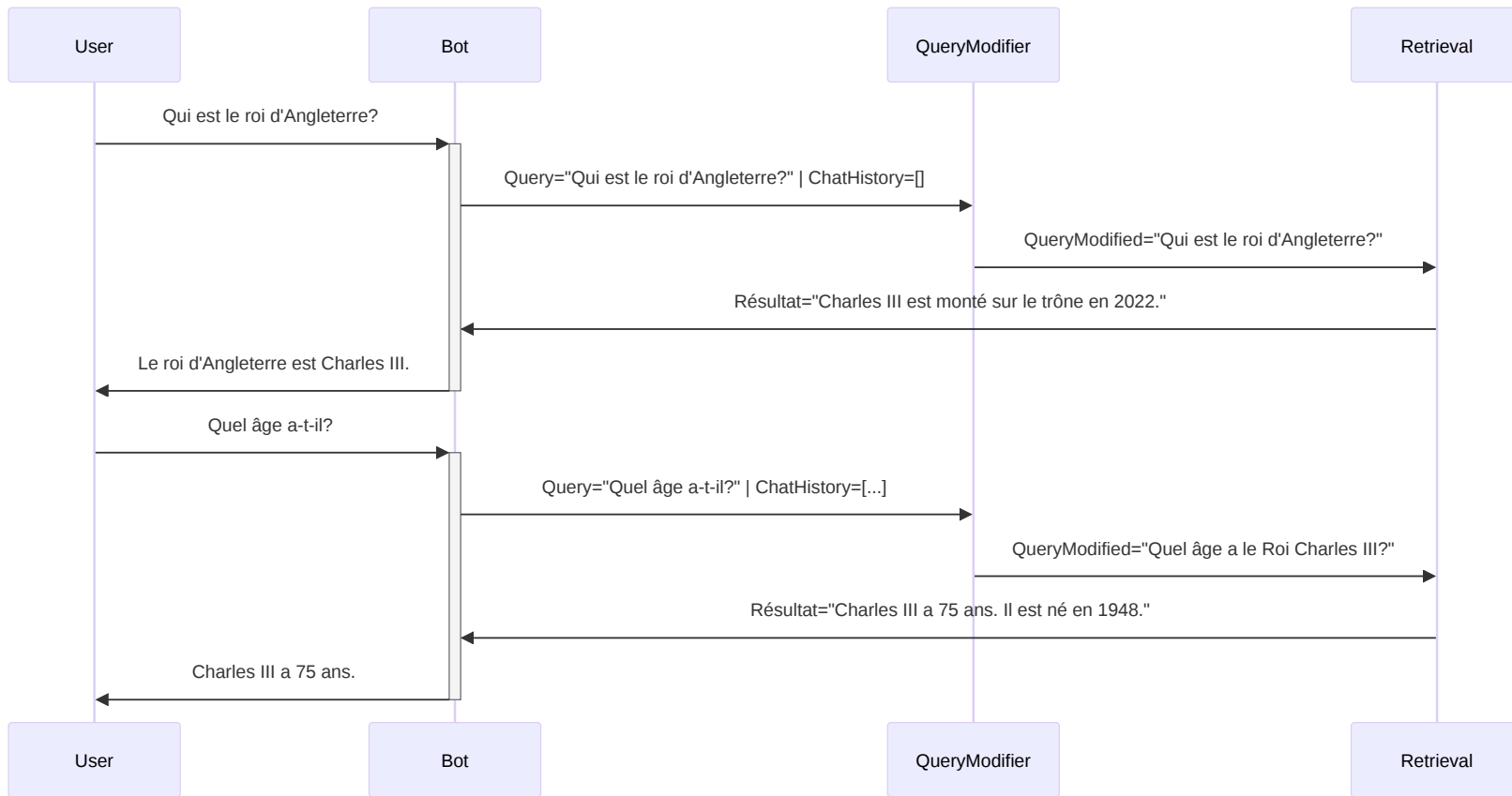
Comment retrouver les documents pertinents?

Contexte

Exemple simple de discussion à un chatbot.

- User: Qui est le roi d'Angleterre?
- Bot: Le roi d'Angleterre est Charles III.
- User: Quel âge a-t-il?
- Bot: Charles III a 75 ans.





Query Modifier?

(J'ai pas trouvé mieux)

On utilise un LLM pour modifier la requête de l'utilisateur pour qu'elle soit plus précise et compréhensible sans autre contexte.

Méthode avancée 1: Rag-Fusion

- Demander plusieurs requêtes sur base de la nouvelle question de l'utilisateur et de l'historique.
- Faire les requêtes pour toutes celles générées.
- Rerank les documents via un algo "Reciprocal Rank Fusion"
- Utiliser les k meilleurs dans le contexte.

Exemple:

"Quand faire pousser un oliver?"

-> "Quand planter un olivier?"

-> "Quel est le meilleur moment pour planter un olivier?"

-> "Quel période de l'année pour planter un oliver?"

Source: <https://github.com/Raudaschl/rag-fusion>

Reciprocal Rank Fusion

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)}$$

D: Documents

R: Rangs

k: constante = 60

r(d): rang du document d

Autrement dit:

```
1  search_results_dict = {
2      "query1": {"doc1": 0.5, "doc2": 0.3, "doc3": 0.2},
3      "query2": {"doc1": 0.4, "doc2": 0.3, "doc3": 0.3},
4      "query3": {"doc1": 0.6, "doc2": 0.2, "doc3": 0.2},
5  }
6  fused_scores = {}
7
8  for query, doc_scores in search_results_dict.items():
9      for rank, (doc, score) in enumerate(
10         sorted(doc_scores.items(), key=lambda x: x[1], reverse=True)
11     ):
12         if doc not in fused_scores:
13             fused_scores[doc] = 0
14             fused_scores[doc] += 1 / (rank + k)
15
```

Méthode avancée 2: Multi-Hop Reasoning System

j'y ai pensé avant de voir que ça existait...

- Décomposer la question en plusieurs sous-questions qui se suivent/ sont complémentaires. - Faire les requêtes pour toutes les décompositions. - Utiliser les résultats pour répondre à la question initiale.

Exemple:

"Quel âge a le roi d'angleterre?"

-> "Qui est le roi d'angleterre en 2024?"

[On récupère la réponse]

-> "Quand est née X"

[On récupère la réponse]

-> Calcul de l'âge

Autre exemple:

- Recherche dans des textes de lois
- Références à d'autres articles
- Combinaison de plusieurs sources pour une réponse

Piste de recherche

- Entity Extraction
- Question Decomposition
- Multihop QA System

Conclusions

Warning

- Ne pas se reposer uniquement sur les LLMs
- Explorer des techniques "classiques", NLP, ML, ...

Mots clés

- RAG
- LLM
- DB vectorielle
- Embedding
- Rag-Fusion
- Multi-Hop Reasoning
- NLP
- Entity Extraction

Documentation

- <https://techcommunity.microsoft.com/t5/fasttrack-for-azure/grounding-llms/ba-p/3843857>
- <https://www.linkedin.com/pulse/retrieval-augmented-generation-rag-next-frontier-covarrubias/>
- <https://medium.com/@shivansh.kaushik/this-technique-will-make-your-llm-smarter-and-more-context-aware-rag-on-steroids-b16d7cf4a42c>
- <https://medium.com/@shivansh.kaushik/talk-to-your-database-using-rag-and-llms-42eb852d2a3c>

Culture générale

- Début du RAG: <https://ai.meta.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models/>
- Explications des techniques dans FAISS: <https://www.pinecone.io/learn/series/faiss/>