RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

cnes
CENTRE NATIONAL D'ÉTUDES SPATIALES

colette,

pour des applications métier orientées GenAI

Travail technique réalisé par Jolibrain

Guillaume Alleon, Guillaume Infantes,
Antoine Jacquet, Alex Girard, Emmanuel Benazera

CNES
7 MARS 2025

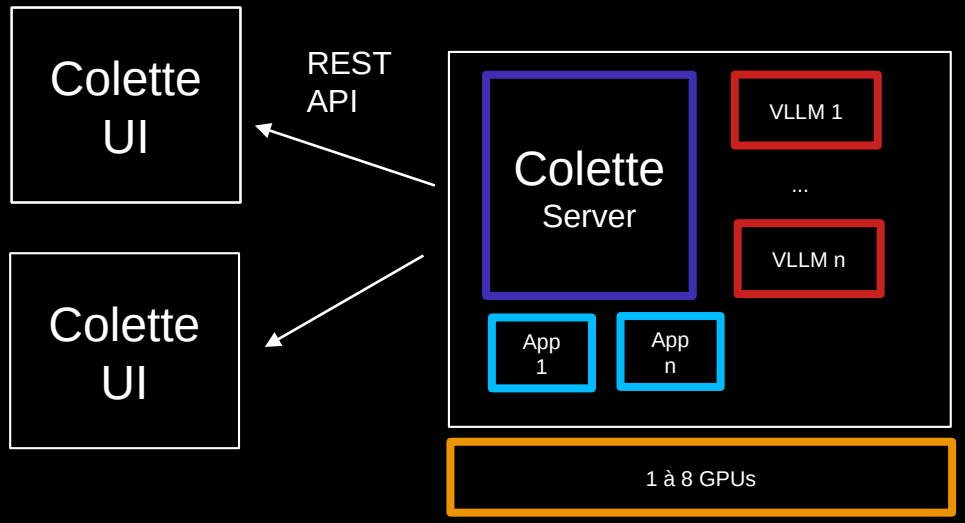
jolibrain

AIRBUS

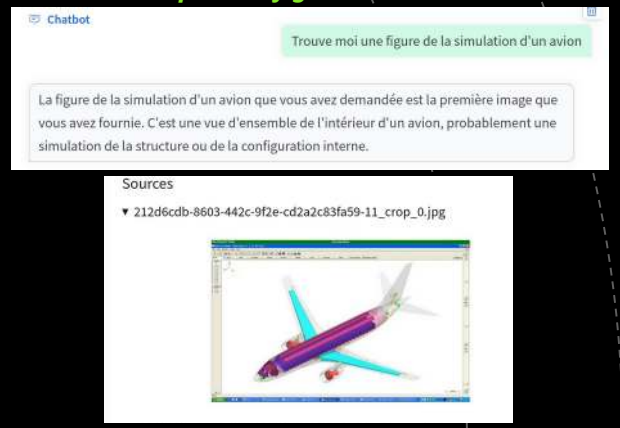
COLETTE

Description générale

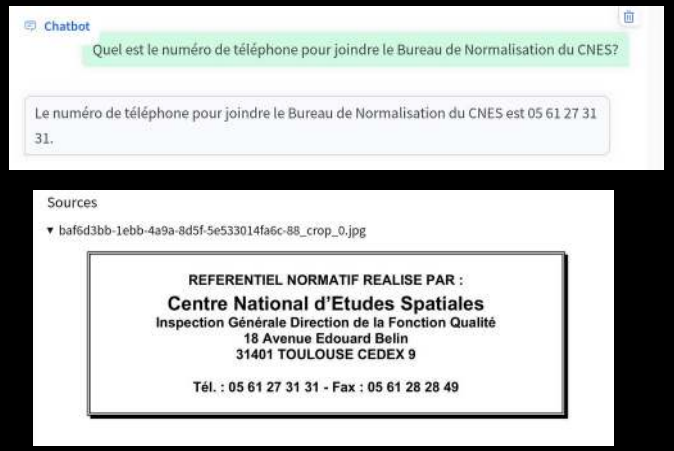
- Service de LLMs « ouverts » et auto-hébergés
- RAG basé texte
- RAG « vision » qui lit le texte, tables et figures, adapté aux documentations techniques
- Service d'aide à la programmation
- Service de génération d'images
- Serveur + UI, Open Source



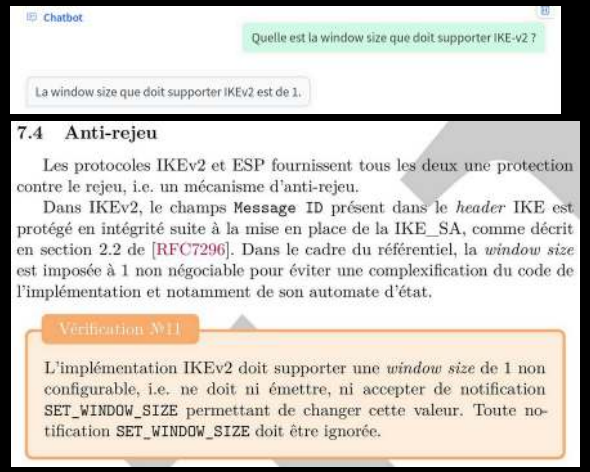
Sémantique des figures et tables



Éléments précis perdus dans les corpus



Compréhension des éléments visuels et textuels



COLETTE

RAG et papier original

- Adresse l'utilisation jointe d'une mémoire externe (dite « non-paramétrique / statique ») avec une mémoire paramétrique (un LLM)
- Ciblé sur les tâches nécessitant une documentation externe
- Publication à NeurIPS 2020
- Applique un apprentissage statistique au RAG dans sa « presque » intégralité !

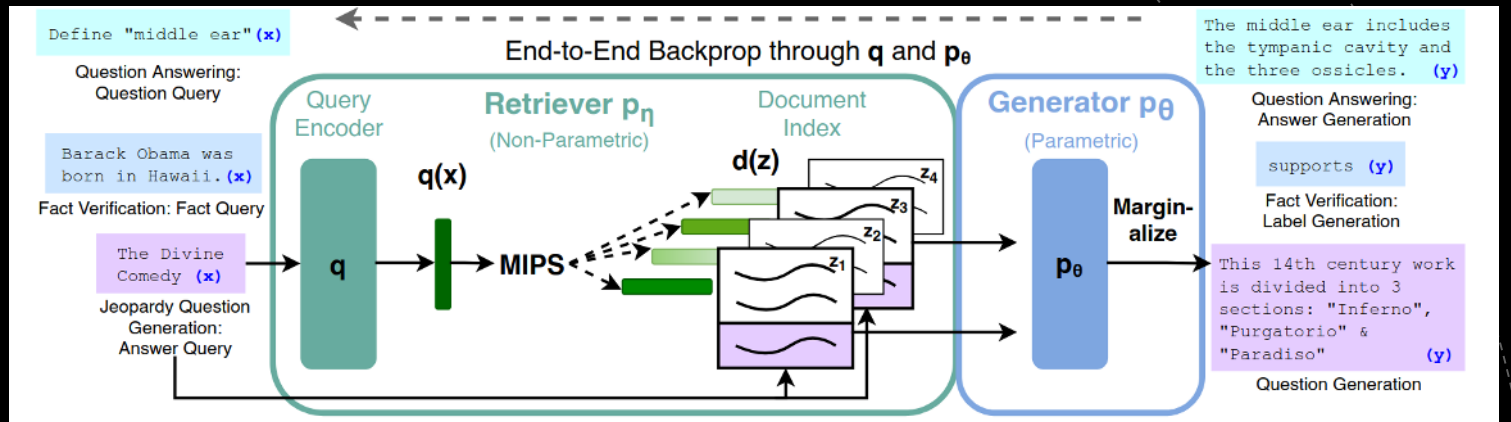


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020.

Our results highlight the benefits of combining parametric and non-parametric memory with generation for *knowledge-intensive tasks*—tasks that humans could not reasonably be expected to perform without access to an external knowledge source. Our RAG models achieve state-of-the-art results

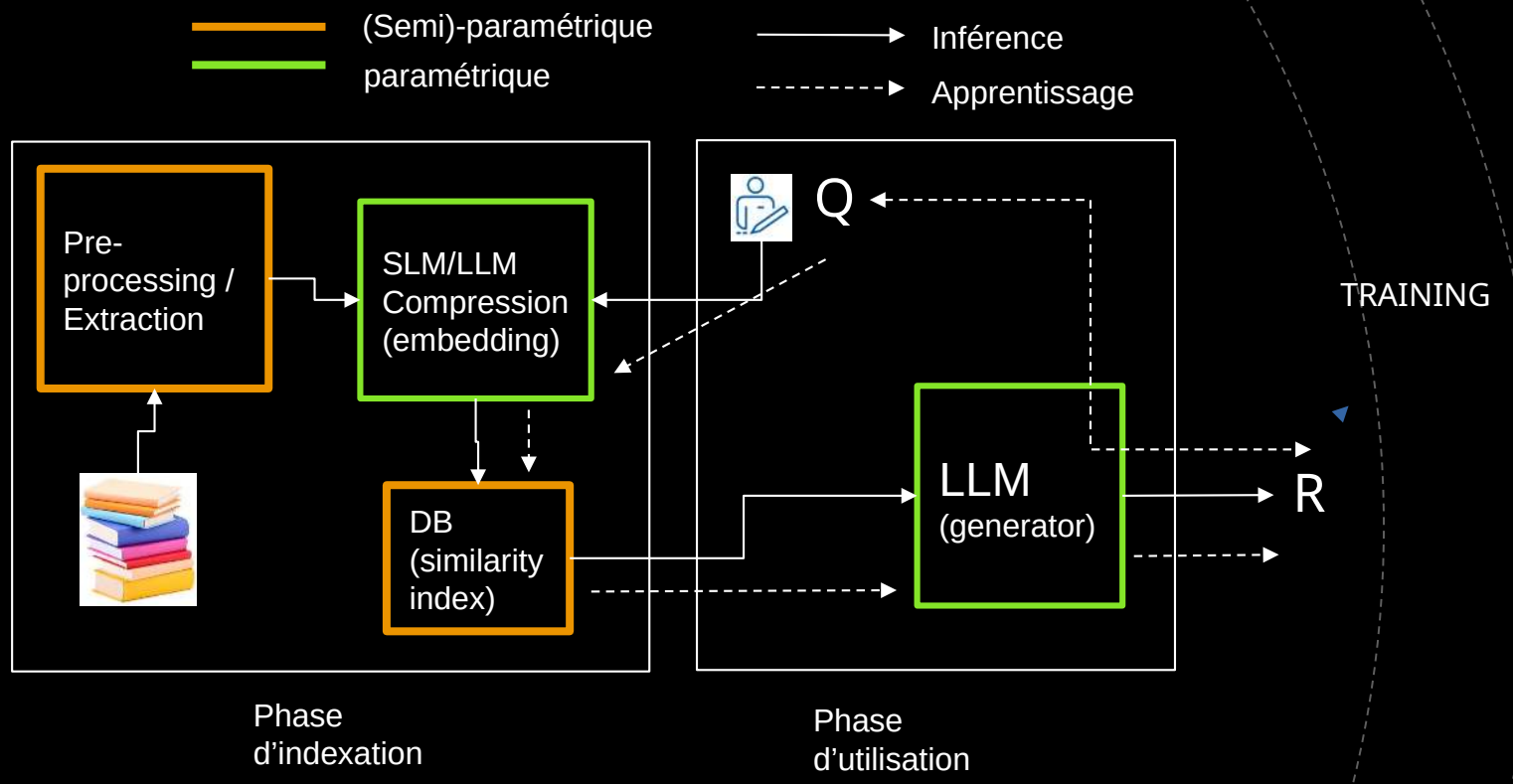
both the generator and retriever are jointly learned.

COLETTE

RAG en application (en « réalité »)

- L'extraction du texte est un art
  - Documents hétérogènes
  - Images, tables, figures, schemas, …
- L'abandon de la composante d'apprentissage (« training »)
  - Difficile d'obtenir les Q & R en masse suffisante
  - Disponibilité de modèles fondationnels (OpenAI, Anthropic, Mistral, …)

Conséquence : une accumulation d'erreurs et une mise en application réelle complexe malgré l'engouement.



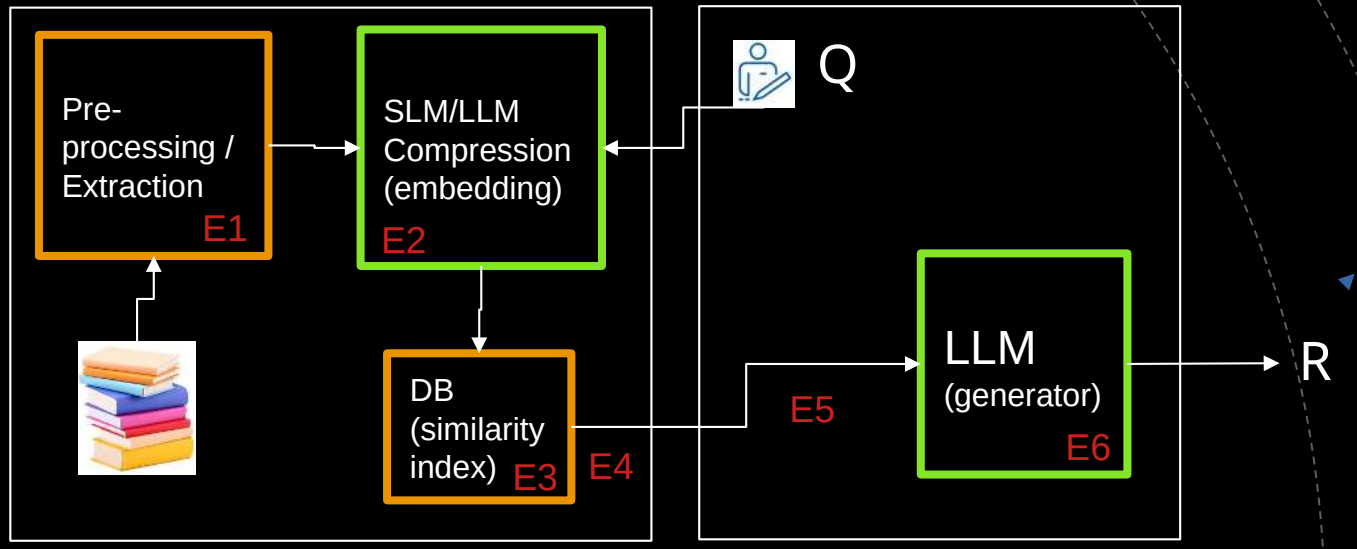
https://github.com/huggingface/transformers/tree/dcbdf7e962c4b36140cc9ee76f870016121e69e5/src/transformers/models/rag

COLETTE

Conséquence, une cascade d'erreurs et de difficultés,

Sans composante d'apprentissage automatique, celui-ci est fait « à la main » : les erreurs sont observées, et des tentatives de correction « programmatiques » sont implémentées.

Celles-ci complexifient et rendent chaque installation du RAG sur-mesure.



E1 : erreurs OCR, transformation des tables et images en texte, perte des indications visuelles.
Layout detection, Table2txt (!), figure2txt (!), correction OCR (!)

E2 : compression textuelle = approximation, biais du modèle
Découpage du texte (« chunking »), graphRAG, rephrasage des questions

E3 : index de similarité = approximation
Reranking (!)

E4 : similarité != réponse
Des questions similaires peuvent avoir des réponses complètement différentes
« answer embeddings »

E5 : Biais entre modèle de compression (embedder) et LLM (generator)
Utiliser le même modèle

E6 : « hallucinations », biais, approximations, savoir dire « je ne sais pas ».
Prompts(!), modèles fondationnels

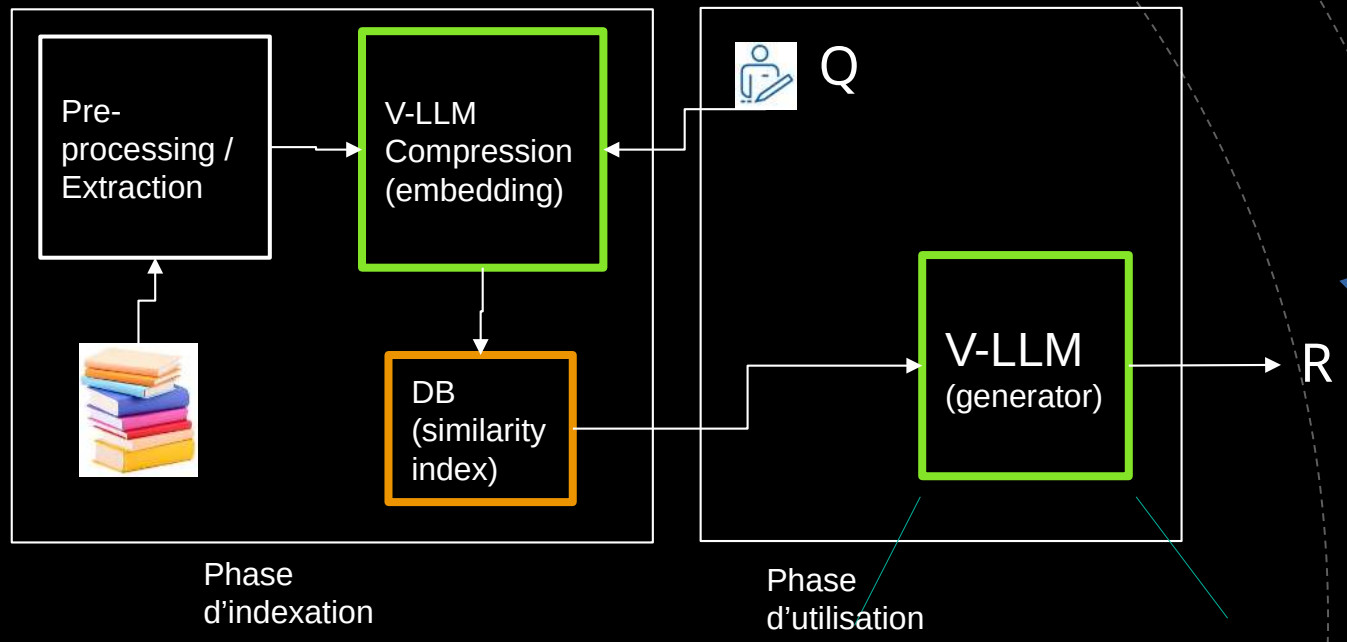
Cri de victoire des data scientists :
« Mon RAG est tombé en marche ! », souvent suivi de désillusions

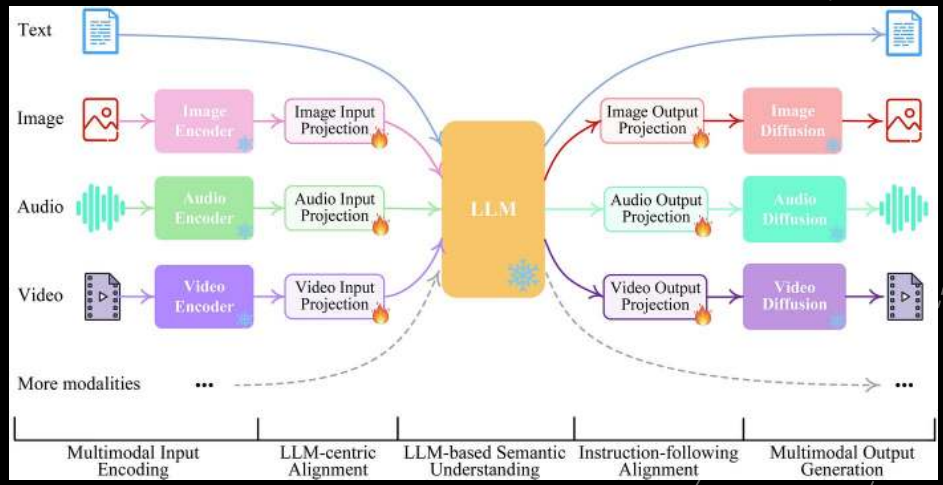
COLETTE

Objectifs pour Colette

- Amenuiser autant que possible l'ensemble des sources d'erreur
- Rendre le RAG « paramétrique » de bout en bout (cad. « apprenable »)
  - Garantie d'exactitude et de fonctionnement « adéquat »

<u>Idée clé</u> : « Vision-RAG » traite tous les documents sous forme d'images, en utilisant des LLM multimodaux (ou V-LLM pour Vision-LLM).

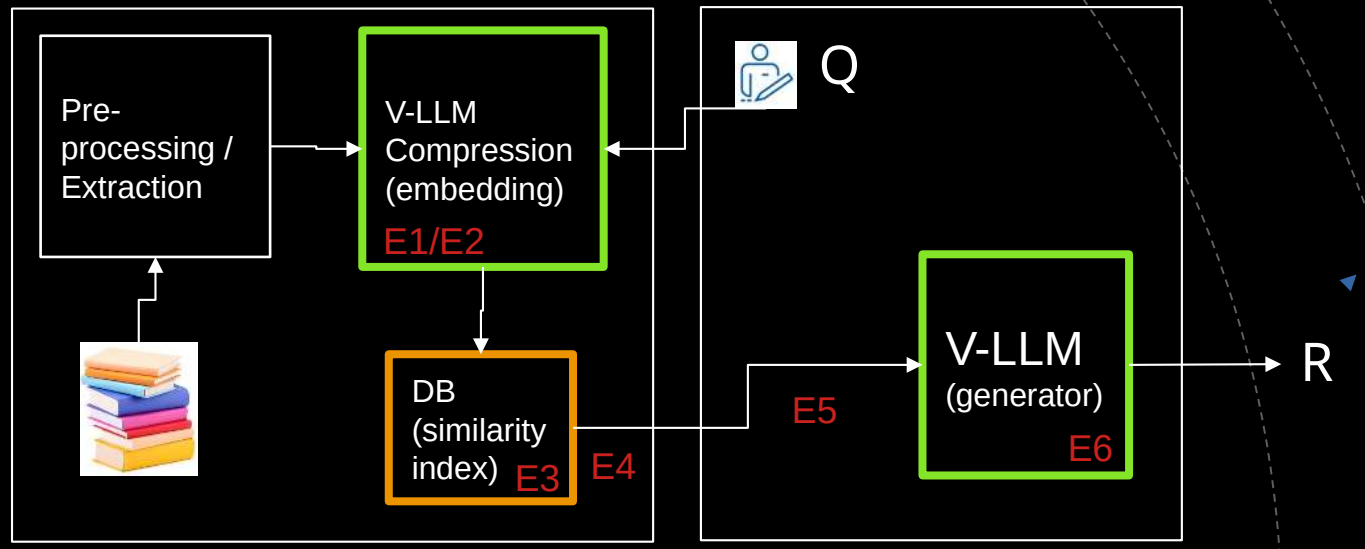




COLETTE

Qu’est-ce que le Vision-RAG de Colette ?
Comment certaines erreurs sont réduites ?

- Traitement du corpus par un « cortex visuel » = naturel
- Tous les documents sont « imprimés » sous forme d’images (pdf, xls, pptx, json, html, …)
- Le code d’extraction et d’OCR (lecture des caractères) est remplacé par un V-LLM
- Images = format dense et passage à l’échelle souple : résolution en pixels ! Taille du contexte LLM = image resolution
- Détection de la composition visuelle des documents = « Visual chunking »



E1/E2 : erreurs « OCR », compression textuelle = approximation, biais du modèle
Découpage visuel, graphRAG, rephrasage

E3 : index de similarité données denses = meilleure approximation

E4 : similarité != réponse
Des questions similaires peuvent avoir des réponses complètement différentes
« answer embeddings »

E5 : Biais entre modèle de compression (embedder) et LLM (generator)
Utiliser le même modèle

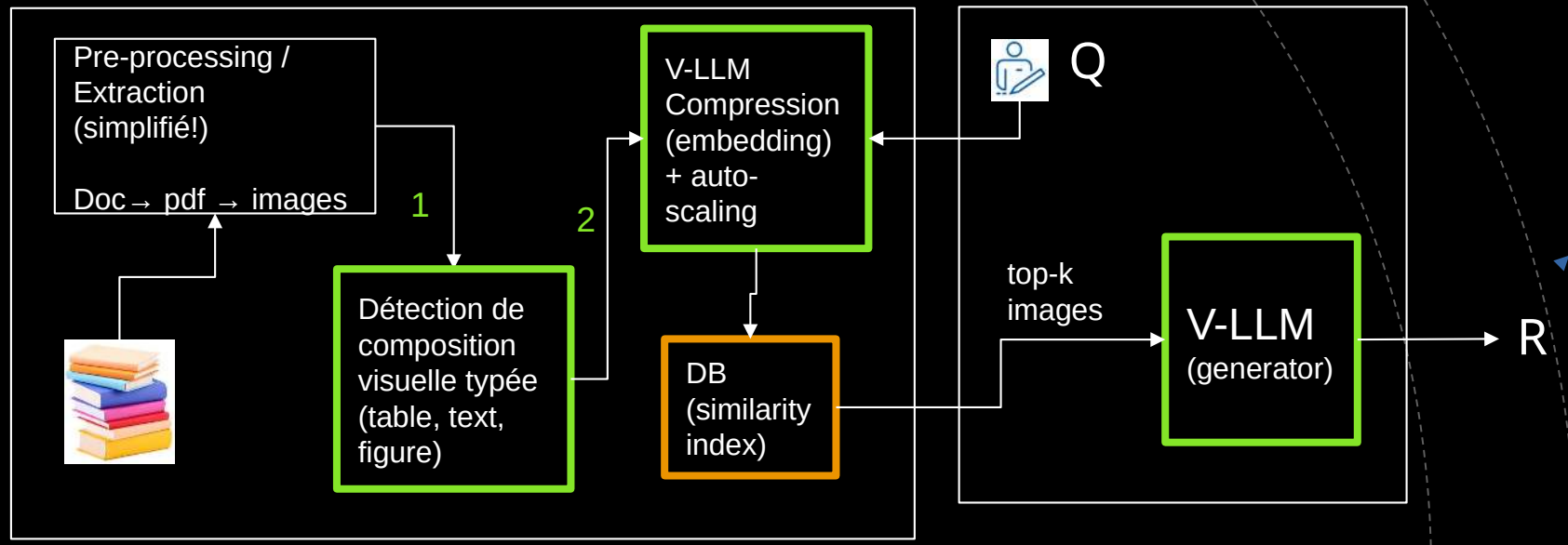
E6 : « hallucinations », biais, approximations, savoir dire « je ne sais pas ».
Prompts(!), modèles fondationnels

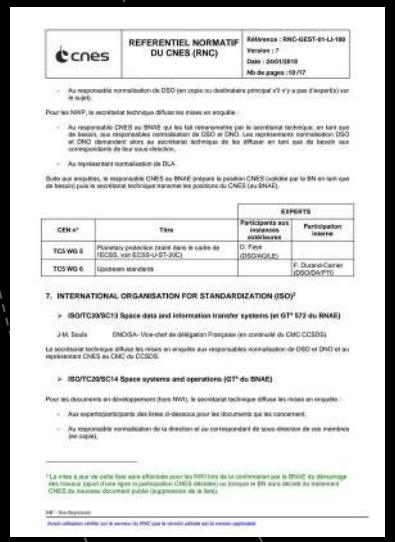
De nombreux travaux académiques sur le sujet Vision-RAG ces derniers mois, dans l’air du temps = RAG v2

COLETTE

Colette Vision-RAG → indexation

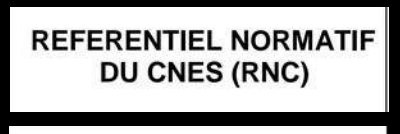
- Tout doc → PDF → images
- Détection de composition typée sur-mesure
- V-LLM pour embedding + images auto-scaling pour assurer la bonne lecture



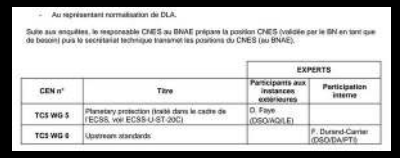


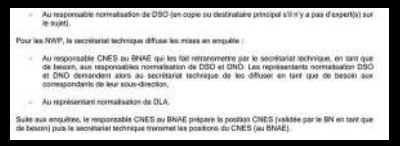
1

2



« Chunks visuels» + vue d'ensemble + type sont indexés





+

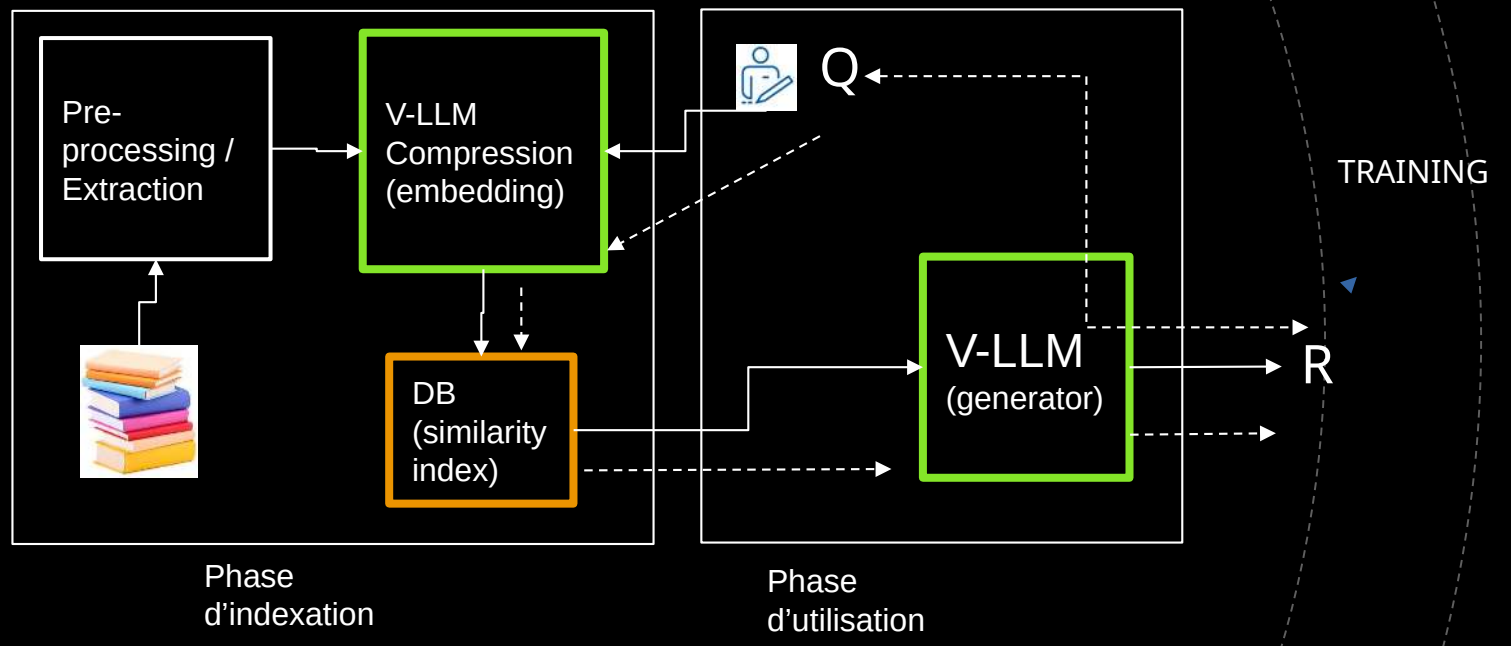
+

Auto-scaling
- Détection des mots
- Vérification de la taille de la fonte en pixels
- Redimensionnement de l'image pour meilleure lecture

COLETTE

Colette training sur 3 composantes sur 4 :

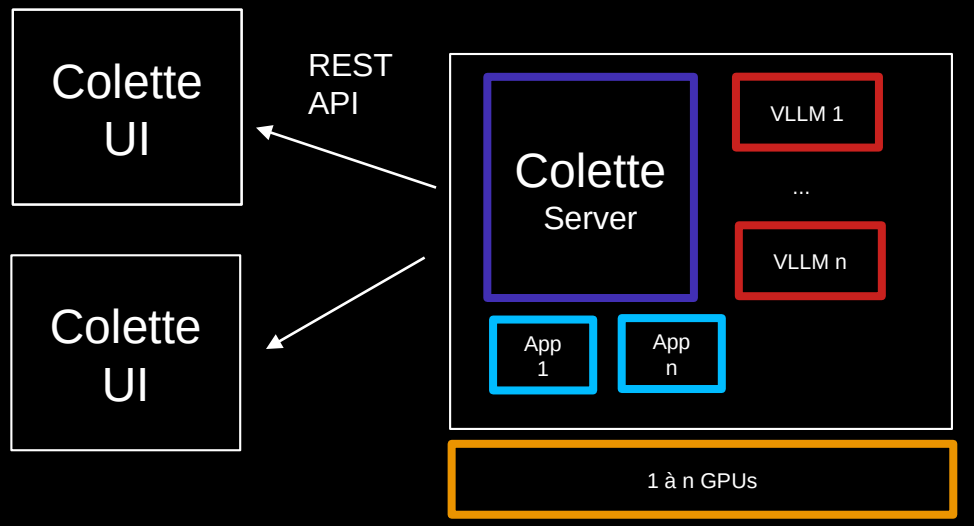
- Apprentissage du détecteur de composition → OK via outils Open Source
- « Finetuning » du V-LLM embedder/compresseur → OK dans Colette
  - Dataset : docs image / questions générées par un autre VLLM
- Apprentissage DB de similarité → OK dans Colette via DB de type ColBERT (« clustering »)
- « Finetuning » du V-LLM generateur → X non implémenté à ce stade



COLETTE

Architecture

- Serveur Multi-applications / RAGs
  - Modèles partagés (optionnel)
  - Modèles externalisés (optionnel) (ollama/vllm)
  - DB de similarité (vecteurs) internalisée, pas de dépendance
  - REST API pour applications programmatiques tierces
- UI multi-applications



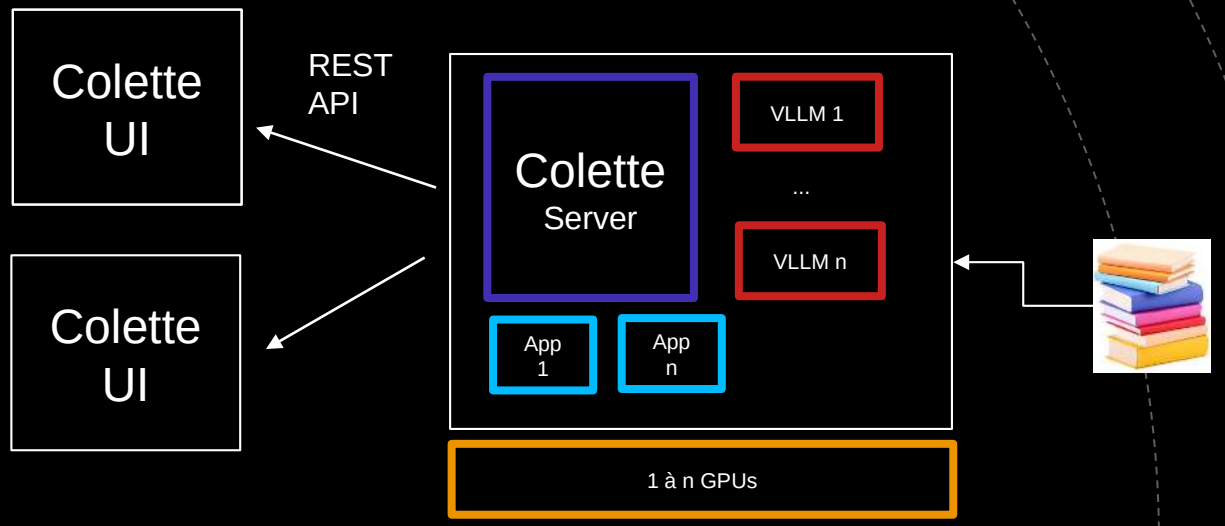
Applications supportées

- Service de modèle LLM texte Open Source
- RAG pour recherche de documents, question/réponse et conversation (Text-RAG et Vision-RAG)
- Assistant de programmation pour conversation
- Génération d'images : prompt → image

COLETTE

Indexation et cycle des applications

- Ingestion initiale des données à la création de l'application métier
  - De quelques minutes à quelques heures
- Mises à jour incrémentales de l'index
  - Par ajout de documents et appels API
  - Détecte les nouveaux fichiers automatiquement
- Application
  - index HDF5 + DB similarité + modèles
  - stockés sur disque, simples à transférer
- Actuellement : indexation est bloquante, à effectuer pendant les heures creuses
  - Indexation asynchrone à venir



COLETTE

Nomenclature HW / SW / ML

Nomenclature du Matériel

Inférence RAG-Text
- **Capacité cible est 1xGPU NVidia 8GB**

Inférence RAG-Vision
- **Capacité cible est 1xGPU NVidia 24GB**
  - Fonctionne sur GPUs avec VRAM < 24GB avec quantization
  - GPUs > 24GB permettent de meilleurs résultats
- 1 à 8 GPUs sur une seule machine
- Inférence en 32bit / 16bit / 8bit
- Pas de dépendances externes en API ou Cloud

Apprentissage
- Apprentissage détecteur de composition : 1 GPU 8GB+
- Vision LLM embedder/retriever: 1 à 8 GPUs sur une seule machine
- Apprentissage DB de similarité: CPU

Nomenclature Logicielle

Principales librairies utilisées
- Transformers (Huggingface)
- Ollama
- Vllm server

Fonctionnalités
- Vision-based RAG (défaut)
- Text RAG simple
- JSON API au niveau Python
- REST API avec serveur compatible Open API
- Serveur en container Docker (optionnel)
- Index = kvstore avec HDFS5

Interface Utilisateur
- Connecte à l'API REST
- Support pour de multiples applications
- Au choix de l'utilisateur pour définir son interface métier

Nomenclature des Modèles

Construit sans dépendances externes

Modèles par défaut (Vision-based RAG)
- Embedder/Retriever:
  - dse-qwen2-2b-mrl-v1
  - gme-Qwen2-VL-2/7B-Instruct
- Inférence: Qwen2.5-VL-7B-Instruct

Modèles supportés :
- Qwen2/Qwen2.5-VL 2B to 72B
- SmolVLM-Instruct
- Paligemma1/2 3B to 28B
- Mistral Pixtral 12B
- Phi4-multimodal
- Tous modèles text-LLM (llama3, ...)
- Deepseek-R1
- QwQ
- Deepseek-code/qwen2.5-coder, …
- Flux

Qwen2/2.5 et Llama3.2 modèles par défaut en 2024.

COLETTE

Performances

Embedder	LLM	Rouge-1	Rouge-2	Rouge-L	BertScore	Pr	Re	F1
fast-intfloat/multilingual-e5-small	llama3.1	21%	14%	19%	74%	0.05	0.19	0.08
fast-sentence-transformers/all-MiniLM-L6-v2	llama3.1	20%	13%	18%	72%	0.05	0.16	0.07
fast-hkunlp/instructor-large	llama3.1	22%	15%	19%	73%	0.05	0.16	0.07
non fast-intfloat/multilingual-e5-small	llama3.1	17%	10%	14%	72%	0.06	0.21	0.09
non fast-sentence-transformers/all-MiniLM-L6-v2	llama3.1	17%	10%	15%	72%	0.06	0.21	0.08
non fast-hkunlp/instructor-large	llama3.1	18%	11%	16%	73%	0.05	0.18	0.08
coldb	llama3.1	25%	17%	22%	75%	0.08	0.32	0.12
dse-qwen2-2b-mrl-v1	Qwen2-VL-2B-Instruct / topk-4	44%	32%	40%	82%	0.13	0.36	0.18
dse-qwen2-2b-mrl-v1	Qwen2-VL-7B-Instruct / topk-4	47%	35%	43%	83%	0.13	0.36	0.18
gme-Qwen2-VL-2B-Instruct	Qwen2-VL-2B-Instruct / topk-4	46%	33%	41%	82%	0.13	0.37	0.18
gme-Qwen2-VL-2B-Instruct	Qwen2-VL-7B-Instruct / topk-4	47%	36%	45%	83%	0.13	0.37	0.18
gme-Qwen2-VL-2B-Instruct	Qwen2.5-VL-3B-Instruct / topk-4	37%	25%	33%	80%	0.13	0.37	0.18
gme-Qwen2-VL-2B-Instruct	Qwen2.5-VL-7B-Instruct / topk-4	36%	25%	32%	79%	0.13	0.37	0.18
gme-Qwen2-VL-7B-Instruct	Qwen2.5-VL-7B-Instruct / topk-4	46%	36%	43%	83%	0.16	0.45	0.23
coldb-colqwen2-v1.0	Qwen2-VL-7B-Instruct / topk-4	41%	40%	47%	84%	0.14	0.39	0.20
jolibrain-dse2	Qwen2-VL-7B-Instruct / topk-2	47%	35%	44%	83%	0.21	0.35	0.26
jolibrain-dse2	Qwen2-VL-7B-Instruct / topk-4	48%	37%	45%	83%	0.16	0.42	0.22

COLETTE

Avantages et inconvénients → facilité et précision accrue via plus de « calcul »

Avantages	Inconvénients
Pré-processing des documents simples et sans perte	Indexation plus lente que RAG-txt car utilisation d'un VLLM
Traitement des tables, figures, schémas (par ex documentations techniques)	Empreinte mémoire plus importante au global (embedder + générateur VLLMs)
Mise en place rapide et meilleurs résultats que RAG txt sans effort particulier	VLLM générateur limité en VRAM → top-k en sortie de l'embedder limité à k=4 en pratique pour 1x GPU 24GB
Chunking visuelle naturel, facile à spécialiser	Le stockage des images est plus lourd que celui du texte
Embedder VLLM assez simple à spécialiser	Temps d'inférence peut être plus long que celui d'un RAG-txt
Retour visuel des « chunks » images de documents retournés par le RAG → simplifie l'appréciation des résultats, et des erreurs	Erreurs de lectures (« OCR » implicite) peuvent compromettre certaines réponses
Passage à l'échelle en mémoire simplifié / taille du contexte = résolution d'images	
Pas besoin de re-ranker en pratique	
Facilité de debug car documents facilement observables de bout en bout	

COLETTE

Futur & bonnes pratiques à venir

- **RAG sur base de données existantes**
  - Recommandation : ne pas indexer les documents ! Utiliser « function calling » pour appeler directement l'API de la DB : question → LLM/appel API à votre DB → docs → images → VLLM → réponse.

- **Apprentissage du RAG**
  - Implémenter la capture des retours utilisateurs au plus tôt → permet de collecter un dataset en amont → permet la correction des erreurs par apprentissage → même avec ~1k échantillons.
  - Apprentissage « embedder / retriever » → améliore les résultats sur les corpus très spécialisés !
  - « Answer embedding » → bonne pratique, dataset à construire semi-automatiquement

- **« Reasoning »**
  - Résultats récents montrent que le « raisonnement » fin sur tâche spécifique (maths, problème ciblé, …) est abordable en terme de dataset + training (S1, R1, …) → testé par Jolibrain en interne, 1k échantillons
  - Arrive pour les VLLM ! → QvQ, Insight-V, R1-V, …

COLETTE

Disponibilité:
A venir sur GitHub avril/mail 2025 → nettoyage du code, commits et documentation en cours

En attendant, possibilité de tester et de prototypage très rapide via CNES, ADS et/ou Jolibrain → nous contacter.

Par exemple :

1. Nous contacter, pointer/partager quelques documents publics d'intérêts (par ex. proches de votre activité / documentation)
2. Mise en place d'une application de test : yourapp.colette.chat
3. Portage avec vos documents sur votre infrastructure