ARCHITECTURE · SYSTÈMES À BASE DE LLM

Le pipeline RAG, de l'indexation à la réponse

La génération augmentée par récupération donne à un modèle de langage des passages pertinents au moment de répondre. Sa qualité se joue d'abord à la récupération : ce qui n'est pas retrouvé ne peut pas être cité.

Intermédiaire · 11 min · RAG · plongements · recherche vectorielle · BM25 · Exemple — non révisé par l'auteur

Sheet
by Simon-Pierre Boucher

sheet.spboucher.ai/s/pipeline-rag
17 novembre 2026

Un modèle de langage ne connaît que ce que contiennent ses paramètres : sa mémoire est figée à la fin de l'entraînement, et il ne peut pas en indiquer la provenance. La génération augmentée par récupération (*Retrieval-Augmented Generation*, RAG) lui adjoint une **mémoire externe consultable** : avant de répondre, le système va chercher des passages pertinents et les fournit au modèle [1].

Mémoire paramétrique

Les poids du modèle. Vaste, mais figée, sans provenance, et coûteuse à mettre à jour.

Mémoire non paramétrique

Un index de documents. On peut le corriger ou l'étendre sans réentraîner le modèle [1].

Le lien entre les deux

Un récupérateur qui, pour chaque question, sélectionne les quelques passages à placer sous les yeux du modèle.

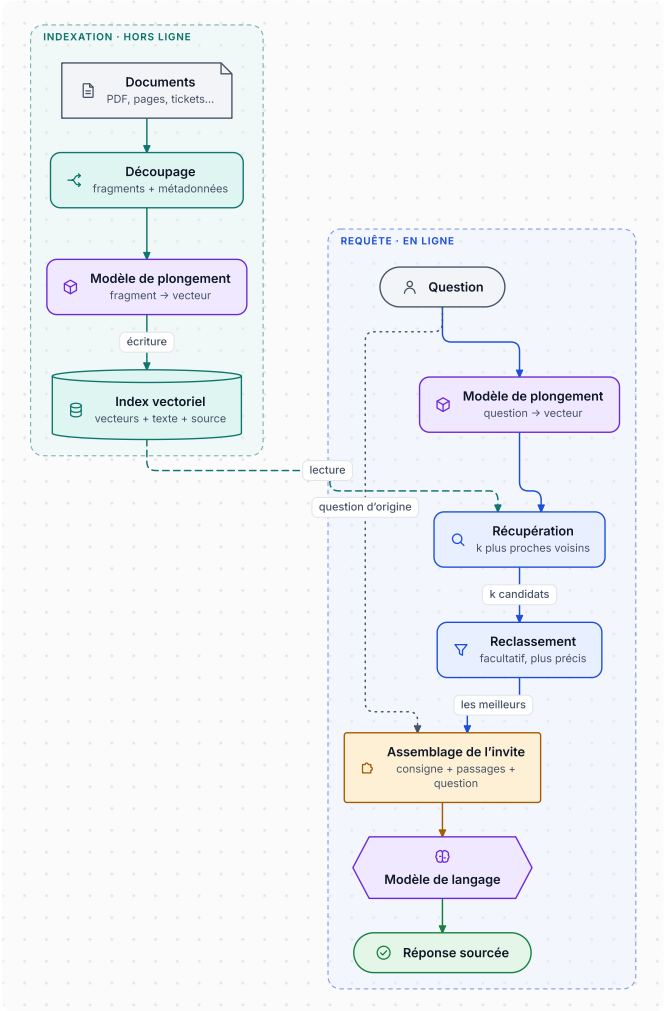
HYPOTHÈSE

RAG d'origine et RAG d'usage courant

L'article de 2020 entraîne **conjointement** un récupérateur dense et un générateur séquence-à-séquence, sur un index de Wikipédia [1]. La pratique courante, décrite ici, est plus simple : récupérateur et modèle de langage sont utilisés tels quels, et les passages sont insérés dans l'invite [5]. Le principe reste le même.

01 Deux phases, deux moments

Indexation (hors ligne) et requête (en ligne)



1. **Découper**
   — Chaque document est scindé en fragments assez courts pour porter une idée, assez longs pour rester compréhensibles hors contexte.
2. **Vectoriser et stocker**
   — Un modèle de plongement associe à chaque fragment un vecteur ; vecteur, texte et provenance sont rangés dans l'index.
3. **Vectoriser la question**
   — À l'arrivée d'une question, le même modèle la projette dans le même espace vectoriel.
4. **Récupérer**
   — L'index renvoie les k fragments dont le vecteur est le plus proche de celui de la question.
5. **Reclasser**
   — Un modèle plus coûteux relit chaque couple question-fragment et ne garde que les plus pertinents.
6. **Générer**
   — L'invite réunit la consigne, les passages retenus et la question ; le modèle rédige la réponse en s'appuyant sur ces passages.

L'indexation se fait une fois par document ; la requête se rejoue à chaque question. Les deux phases doivent utiliser le **même** modèle de plongement.

02 Retrouver les bons passages

ƒ RECHERCHE DENSE

$$\mathrm{sim}(q, p) = E_Q(q)^\top E_P(p)$$

Question et passage sont projetés dans un même espace ; leur proximité se mesure par un produit scalaire. Les vecteurs des passages sont calculés à l'avance, si bien qu'une requête se réduit à une recherche de plus proches voisins [2].

q	Question
p	Passage (fragment de document)
E_Q	Encodeur des questions, qui renvoie un vecteur de dimension d
E_P	Encodeur des passages, de même dimension

Sources : 2

ƒ RECHERCHE LEXICALE : BM25

$$\mathrm{score}(D, Q) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{f(q_i, D)\,(k_1 + 1)}{f(q_i, D) + k_1 \left(1 - b + b\,\frac{|D|}{\mathrm{avgdl}}\right)}$$

BM25 compte les mots de la question présents dans le document, en donnant plus de poids aux mots rares et en corrigeant l'effet de la longueur [3]. Il reste difficile à battre sur les termes exacts : noms propres, références, codes.

q_i	i-ième terme de la requête Q	
$f(q_i, D)$	Nombre d'occurrences de q_i dans le document D	
$\mathrm{IDF}(q_i)$	Fréquence inverse de document : élevée pour un terme rare dans le corpus	
$\lvert D\rvert$	Longueur du document	mots
avgdl	Longueur moyenne des documents du corpus	mots
k_1, b	Paramètres libres : saturation de la fréquence et normalisation par la longueur	

Sources : 3

ƒ RAPPEL À K

$$\mathrm{Recall@}k = \frac{|\mathcal{R} \cap \mathcal{T}_k|}{|\mathcal{R}|}$$

Mesure de la seule récupération : parmi les passages qu'il fallait trouver, quelle part figure dans les k premiers ? Elle se calcule sans appeler le modèle de langage.

$\mathcal{R}$	Ensemble des passages pertinents pour la question (annotés à la main)
$\mathcal{T}_k$	Ensemble des k passages renvoyés par le récupérateur

Similarité question-fragment

DONNÉES ILLUSTRATIVES

	Retours	Livraison	Garantie	Batterie	Carrières
Délai de retour ?	0,86	0,41	0,33	0,12	0,05
Frais de livraison ?	0,38	0,84	0,15	0,10	0,06
Garantie batterie ?	0,29	0,13	0,81	0,77	0,04

0,04 — 0,86

Valeurs illustratives, choisies à la main : elles montrent ce qu'on attend d'un bon modèle de plongement, pas la sortie d'un modèle réel.

	Lexicale (BM25)	Dense (plongements)	Hybride COURANT
APPARIE	les mêmes mots	le même sens	les deux
POINT FORT	termes exacts, identifiants, sigles	paraphrases, synonymes	robustesse
POINT FAIBLE	vocabulaire différent entre question et document	termes rares ou absents de l'entraînement	deux index à maintenir, fusion des scores à régler
MODÈLE REQUIS	aucun	modèle de plongement [4]	modèle de plongement

VERDICT

Les deux familles se trompent sur des cas différents ; c'est pourquoi on les combine souvent, puis on confie le tri final à un reclasseur [5].

ATTENTION

Évaluer les deux moitiés séparément

Une mauvaise réponse a deux causes possibles : le bon passage n'a pas été fourni, ou il l'a été et le modèle ne s'en est pas servi. Sans mesure distincte de la récupération (rappel à k) et de la génération (fidélité aux passages), on corrige au hasard. **Un passage non récupéré ne peut pas être rattrapé par le modèle.**

03 Générer à partir des passages

ƒ MODÈLE RAG-SEQUENCE

$$p_{\text{RAG-Sequence}}(y \mid x) \approx \sum_{z \in \text{top-}k(p_\eta(\cdot \mid x))} p_\eta(z \mid x) \prod_{i=1}^{N} p_\theta(y_i \mid x, z, y_{1:i-1})$$

Le passage récupéré z est traité comme une variable latente : la probabilité de la réponse est une moyenne sur les k meilleurs passages, pondérée par leur score de récupération [1]. Insérer les passages dans l'invite d'un modèle de langage en est une version simplifiée, où le modèle lit tous les passages à la fois.

x	Entrée (la question)
y	Séquence générée, de jetons $y_1, \ldots, y_N$
z	Passage récupéré
$p_\eta(z \mid x)$	Récupérateur, de paramètres η
p_θ	Générateur, de paramètres θ

Sources : 1

Consigne

Répondre à partir des passages fournis ; dire qu'on ne sait pas si l'information n'y figure pas.

Passages numérotés

Chaque passage porte un identifiant et sa provenance, afin que la réponse puisse y renvoyer.

Question

La question d'origine, telle que posée.

Réponse citée

Chaque affirmation renvoie au passage qui la soutient, ce qui rend la réponse vérifiable.

04 Où le pipeline casse

Modes de défaillance typiques, rangés par étape. Synthèse de pratique d'ingénierie [5].

ÉTAPE	SYMPTÔME	CAUSE PROBABLE	PISTE DE CORRECTION
Corpus	Le modèle répond « je ne sais pas » ou invente	L'information n'est tout simplement pas dans l'index	Compléter le corpus ; faire refuser plutôt qu'inventer
Découpage	Réponse partielle ou hors contexte	Le fragment coupe un tableau, une liste ou un raisonnement	Découper selon la structure ; chevauchement ; rattacher le titre de section
Récupération	Le bon passage existe mais n'est pas remonté	Écart de vocabulaire, k trop petit, question ambiguë	Recherche hybride ; augmenter k ; reformuler la question
Reclassement	Le bon passage est récupéré puis écarté	Seuil trop strict ou budget de contexte trop court	Garder plus de passages ; mesurer avant et après reclassement
Génération	La réponse contredit ou ignore les passages	Passages contradictoires, trop nombreux, ou consigne trop vague	Moins de passages, mieux triés ; exiger des citations ; vérifier la fidélité
Index	Réponse exacte hier, fausse aujourd'hui	Index périmé ou versions concurrentes d'un même document	Réindexation incrémentale ; dates et versions en métadonnées

05 Vérifier sa compréhension

QUESTION

Un assistant RAG répond mal à une question. En inspectant les journaux, on constate que le passage contenant la réponse figurait bien dans l'invite. Où chercher d'abord ?

A. Du côté de la génération : consigne, nombre et ordre des passages, passages contradictoires.

B. Du côté du modèle de plongement, qu'il faut remplacer.

C. Du côté du découpage, en réduisant la taille des fragments.

VOIR LA RÉPONSE

A — Bonne réponse. Exact : la récupération a fait son travail ; c'est l'usage du passage qui a échoué.

B — Mauvaise réponse. Le passage a été retrouvé : le plongement n'est pas en cause pour cette question.

C — Mauvaise réponse. Possible si le fragment était tronqué, mais ce n'est pas la première piste quand la réponse est entièrement dans l'invite.

Le RAG supprime-t-il les hallucinations ?

Non. Il les réduit quand les bons passages sont fournis et que le modèle s'y tient ; l'article d'origine observe des générations plus factuelles et plus précises qu'avec un modèle purement paramétrique [1]. Mais le modèle peut encore mal lire un passage, en combiner deux à tort, ou compléter de mémoire ce qui manque.

Quelle taille de fragment choisir ?

Il n'y a pas de valeur universelle : c'est un compromis à mesurer sur vos propres questions. Trop petit, le fragment perd son contexte ; trop grand, son vecteur mélange plusieurs sujets et le passage utile se dilue. Suivre la structure du document (sections, paragraphes) est un meilleur point de départ qu'un nombre fixe de caractères.

Erreur fréquente : changer de modèle de plongement sans réindexer

Les vecteurs de deux modèles différents ne vivent pas dans le même espace : comparer une question encodée par l'un à des fragments encodés par l'autre n'a aucun sens. Tout changement de modèle de plongement impose de recalculer l'index entier.

Un très grand contexte rend-il le RAG inutile ?

Pour un petit corpus stable, tout placer dans l'invite peut suffire. Au-delà, la récupération reste utile : elle limite le coût par requête, permet de tenir à jour un corpus qui change, applique des droits d'accès par document et indique d'où vient chaque information.

Références

[1] Patrick Lewis, Ethan Perez, Aleksandra 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*. arXiv:2005.11401. arxiv.org/abs/2005.11401

[2] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, Wen-tau Yih (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 6769–6781. doi:10.18653/v1/2020.emnlp-main.550

[3] Stephen Robertson, Hugo Zaragoza (2009). *The Probabilistic Relevance Framework: BM25 and Beyond*. Foundations and Trends in Information Retrieval, 4(1–2), 1–174. doi:10.1561/1500000019

[4] Nils Reimers, Iryna Gurevych (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 3980–3990. doi:10.18653/v1/D19-1410

[5] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, Haofen Wang (2023). *Retrieval-Augmented Generation for Large Language Models: A Survey*. arXiv:2312.10997. arxiv.org/abs/2312.10997