Sélection Dynamique du Contexte pour la Génération Augmentée de Recherche D'Information

Maya Iratni Mohand Boughanem Taoufiq Dkaki
IRIT, Université de Toulouse, Toulouse, France
{Malika.Iratni, Mohand.Boughanem, Taoufiq.Dkaki}@irit.fr

RÉSUMÉ

La génération augmentée par la recherche d'information (RAG) combine des modèles de langue avec des corpus externes afin de soutenir des tâches nécessitant des connaissances externes, telles que la question-réponse en domaine ouvert. Les systèmes RAG standard utilisent généralement une stratégie de recherche d'information fixe top-k, qui récupère le même nombre de passages indépendamment des besoins de la requête. Cela peut conduire soit à des preuves insuffisantes, soit à l'inclusion de contextes non pertinents, qui entraînent une dégradation des performances de génération. Dans le cadre de ce travail, nous menons une étude empirique sur la manière dont les passages non pertinents récupérés affectent la génération. Sur la base de ces observations, nous proposons un module de classification basé sur la taille du contexte, qui prédit de manière dynamique la quantité de contexte nécessaire en fonction des besoins spécifiques de la requête. Nous intégrons cette approche dans un pipeline RAG complet et démontrons une amélioration des performances par rapport à plusieurs références.

ABSTRACT

Dynamic Context Selection for Retrieval-Augmented Generation

Retrieval augmented generation (RAG) combines language models with external corpora to support knowledge-intensive tasks, such as open-domain question answering. Standard RAG systems typically employ a fixed top-k retrieval strategy, retrieving the same number of passages regardless of query needs. This can lead to either insufficient evidence, or the inclusion of irrelevant contexts that lead to a degradation of generation performance. In this work, we conduct an empirical study of how irrelevant retrieved passages affect downstream generation. Building on these insights, we propose a lightweight context-size classification module that dynamically predicts how much context is required based on query-specific needs. We integrate this approach into a full RAG pipeline, and demonstrate improved performance over several baselines.

MOTS-CLÉS : Génération augmentée par la recherche d'information (RAG), recherche d'information, reranking, question-réponse multi-hop, grands modèles de langage (LLM).

KEYWORDS: Retrieval Augmented Generation (RAG), Information Retrieval, Reranking, Multi-hop QA, Large language models (LLM).

ARTICLE ACCEPTÉ À : SIGIR 2026.
URL : https://doi.org/10.1145/3805712.3809959

1 Introduction

La génération augmentée par la recherche d'information (RAG) est une approche qui améliore les grands modèles de langue (LLM) grâce à des connaissances externes, en récupérant des passages pertinents pendant l'inférence (Gao *et al.*, 2023; Lewis *et al.*, 2020). Dans un système RAG standard, la requête de l'utilisateur est d'abord encodée, puis utilisée pour récupérer une liste classée de documents à partir d'une grande collection (Karpukhin *et al.*, 2020). L'objectif est de s'assurer que le LLM est capable de répondre à une requête donnée sur la base des preuves factuelles récupérées, afin de fournir une réponse fondée sur la vérité et de limiter les hallucinations (Ji *et al.*, 2023; Liu *et al.*, 2023; Koopman & Zuccon, 2023). Les passages top-K, où k est une valeur fixe prédéterminée, sont ensuite fournis au modèle génératif comme contexte supplémentaire pour répondre à une requête donnée (Lewis *et al.*, 2020; Guu *et al.*, 2020). Cette approche top-K est largement adoptée, car elle est simple à mettre en œuvre et fournit généralement un contexte adéquat pour de nombreuses tâches, telles que les questions-réponses en domaine ouvert (Gao *et al.*, 2023). L'un des principaux défis souvent rencontrés dans les systèmes RAG est le bruit attribué à la recherche d'information de textes non pertinents qui peut entraîner une dégradation des performances de génération (Amiraz *et al.*, 2025). Un distracteur est un texte récupéré qui semble similaire à la requête, mais qui est en réalité sémantiquement non pertinent. De tels contextes peuvent semer la confusion dans le générateur et réduire la qualité de la sortie en introduisant du bruit dans le contexte, ce qui peut induire le générateur en erreur et affaiblir l'attention portée aux passages pertinents (Amiraz *et al.*, 2025; Jin *et al.*, 2024). De ce fait, l'utilisation d'une stratégie top-k fixe présente des limites importantes et risque soit d'exclure des informations pertinentes lorsque k est trop petit, soit d'inclure des contextes non pertinents lorsque k est trop grand, ce qui en fait une approche sous-optimale pour les tâches complexes(Su *et al.*, 2024). Cela est particulièrement vrai pour les tâches de questions-réponses multi-hop, où une requête peut contenir des réponses partielles réparties sur plusieurs contextes (Bolotova-Baranova *et al.*, 2023; Gabburo *et al.*, 2024; Zamani *et al.*, 2023). Ce problème d'équilibre est directement lié au compromis entre précision et rappel dans la recherche. L'augmentation de k améliore le rappel et réduit le nombre de passages pertinents manqués, mais la précision est simultanément réduite par l'inclusion de plus de contextes non pertinents, ce qui finit par « noyer » les contextes pertinents et dégrader la qualité de la génération (Jin *et al.*, 2024). Les points ci-dessus soulignent les questions fondamentales suivantes : **(RQ1)** *Dans quelle mesure la qualité de génération est-elle affectée par la présence de distracteurs ?* **(RQ2)** *Quelle quantité de contexte une requête nécessite-t-elle pour obtenir une réponse correcte ?*

Les requêtes diffèrent en termes de complexité, de portée et d'exigences en matière d'informations. Certaines peuvent être traitées à l'aide d'un seul passage, tandis que d'autres nécessitent des preuves contextuelles plus larges. Pour relever ce défi, ce travail apporte les contributions suivantes : (1) Nous menons une expérience préliminaire afin d'évaluer l'impact des distracteurs sur la génération (2) Nous introduisons un classifieur qui prédit de manière dynamique le nombre de contextes requis (k) en fonction de la requête, qui guide ensuite un LLM pour sélectionner le nombre de contextes «prédit par le classifieur» k parmi un ensemble récupéré pour la génération. (3) Nous intégrons le classifieur dans un système RAG complet et comparons ses performances à celles d'une base de référence à k fixe, démontrant ainsi des améliorations constantes en matière de génération.

2 État de l'Art

Les systèmes RAG sont connus pour être très sensibles à la qualité du contexte récupéré, les passages distracteurs constituant un défi majeur. Plusieurs études montrent que la recherche d'information

optimisée pour le rappel introduit souvent des contextes thématiquement liés, mais non pertinents, qui dégradent la qualité de la génération. Le bruit joue un rôle essentiel dans les performances du RAG (Cuconasu *et al.*, 2024), et il a été démontré que l'apprentissage du filtrage des contextes récupérés améliore les performances (Wang *et al.*, 2023). Le problème des distracteurs est particulièrement prononcé dans les questions-réponses multi-hop, qui nécessitent d'agréger des preuves provenant de plusieurs contextes pour formuler une réponse (Min *et al.*, 2019; Karpukhin *et al.*, 2020).

Des travaux antérieurs sur le RAG ont exploré plusieurs stratégies pour faire face à cette complexité. Les approches basées sur la chaîne de pensée guident la recherche et le raisonnement à travers des étapes intermédiaires, en entrelaçant la recherche d'information avec des étapes de raisonnement générées (Trivedi *et al.*, 2023; Press *et al.*, 2023). Les méthodes de recherche itératives récupèrent des preuves en plusieurs étapes, affinant les requêtes en fonction des contextes ou des réponses partielles précédemment récupérés (Yadav *et al.*, 2020; Trivedi *et al.*, 2023; Xiong *et al.*, 2020). D'autres approches décomposent les requêtes complexes en sous-requêtes plus simples, qu'elles récupèrent et auxquelles elles répondent de manière séquentielle (Perez *et al.*, 2020; Min *et al.*, 2019). Plus récemment, les LLM se sont révélés efficaces pour le reclassement de passages, effectuant un reclassement sans apprentissage ou non supervisé avec d'excellents résultats (Ma *et al.*, 2023; Khramtsova *et al.*, 2024), ce qui motive leur utilisation comme juges de pertinence sémantique dans les pipelines RAG. RankRAG (Yu *et al.*, 2024) intègre davantage le reclassement dans la génération en ajustant un LLM pour classer conjointement les passages et produire des réponses. DynamicRAG (Sun *et al.*, 2025) ajuste un LLM à l'aide d'un apprentissage par renforcement qui effectue à la fois la sélection et le reclassement des passages récupérés. La recherche explore désormais des pipelines RAG plus adaptatifs, où les décisions de recherche d'information sont dynamiques. Ces systèmes décident quand récupérer (Jeong *et al.*, 2024; Asai *et al.*, 2024), quoi récupérer via un reclassement dynamique ou une sélection basée sur le retour d'information (Jiang *et al.*, 2023; Su *et al.*, 2024), et quelle quantité de contexte récupérer (Sun *et al.*, 2025; Taguchi *et al.*, 2025). Contrairement aux méthodes précédentes, notre approche utilise un classifieur qui prédit directement le nombre précis de contextes requis pour chaque requête, fournissant au module de sélection LLM une ligne directrice sur la quantité de contexte à récupérer afin de limiter les distracteurs et de fournir un contexte suffisant.

3 Analyse préliminaire

3.1 Impact des distracteurs sur la generation

Dans cette expérience, nous évaluons l'impact des distracteurs sur la génération à l'aide de deux jeux de données et de trois générateurs. Les détails des jeux de données sont décrits dans la Section 5.1. Dans cette analyse, le générateur a reçu une requête, l'ensemble de ses passages « gold » et un nombre croissant de distracteurs à chaque itération (de zéro, « 0 », à trois, « 3 »). Cela a été réalisé afin d'évaluer dans quelle mesure les distracteurs dégradent la qualité de la génération, même lorsque le générateur dispose du contexte « gold » complet. Les résultats présentés dans le Tableau 1 montrent une tendance clairement à la baisse sur toutes les lignes de base, un seul distracteur réduisant les performances jusqu'à 22% (sur Llama2), et les distracteurs supplémentaires amplifiant cet effet. Cette dégradation systématique suggère que les distracteurs interfèrent activement avec la génération, affaiblissant l'attention portée aux contextes pertinents. La cohérence de cette tendance à travers différentes architectures de générateurs met en évidence un défi fondamental dans le maintien de la concentration sur le contenu pertinent à mesure que la longueur du contexte augmente, indépendamment du modèle de génération.

TABLE 1 – Distractor analysis on MuSiQue & 2Wiki using EM

	MuSiQue				2Wiki			
Model	0	1	2	3	0	1	2	3
Flan-t5-XXL	64.7	61.2	56.5	52.8	67.8	63.7	61.4	58.6
Llama2-7b	22.0	17.1	14.4	12.3	47.9	40.8	36.6	34.6
Llama3-8b	45.7	40.7	39.5	37.5	64.9	58.0	54.6	51.5

4 Approche Proposée

Sur la base de notre analyse préliminaire, nous proposons un système RAG conçu pour améliorer la qualité de génération pour les questions-réponses multi-hop, en réduisant l'impact des distracteurs envoyés au générateur. Cette section présente l'architecture et les composants de notre système. Nous étendons le pipeline Vanilla RAG, qui se compose d'un module de recherche d'information, d'un reranker optionnel et d'un générateur, avec deux modules supplémentaires conçus pour réduire le bruit provenant des passages récupérés non pertinents. La figure 1 illustre le pipeline VanillaRAG par rapport à l'architecture que nous proposons, qui comprend un module de classification des requêtes et un sélecteur de contexte basé sur un LLM.

La principale extension consiste en l'intégration d'un prédicteur de k spécifique à la requête, où k correspond au nombre de contextes fournis au générateur pour la tâche de question-réponse. Nous introduisons un classifieur chargé d'estimer le nombre de contextes (k) nécessaires pour répondre complètement à une requête. Le classifieur prend en entrée une requête donnée et produit une valeur entière (k) indiquant le nombre de contextes requis pour y répondre.

La seconde extension du pipeline repose sur l'ajout d'un module de sélection de contexte basé sur un LLM. Dans le pipeline complet, nous récupérons d'abord un nombre fixe k de contextes candidats au moyen d'une étape classique de recherche d'information suivie d'un reranking. La valeur de k prédite par le classifieur, la requête originale ainsi que les contextes les mieux classés sont ensuite fournis en entrée à un LLM agissant comme sélecteur de contexte. Le LLM est alors incité à sélectionner les k passages les plus pertinents parmi l'ensemble candidat, en fonction de leur pertinence vis-à-vis de la requête. Les passages sélectionnés sont ensuite utilisés comme contexte lors de l'étape finale de génération. Cette valeur $predicted_k$ guide le LLM afin de sélectionner un contexte suffisant tout en excluant autant que possible les passages distracteurs. Le pipeline est donc modifié comme suit :

1. Le classifieur traite la requête q pour prédire le nombre de résultats :

$$k_{\text{pred}} = \text{Classifieur}(q)$$

2. Le module de recherche d'information récupère k_{fixed} passages.
3. Le sélecteur LLM filtre les passages :

$$P_{\text{filtré}} = \text{LLM\_Sélecteur}(q, k_{\text{pred}}, P_{\text{fixe}})$$

$$|P_{\text{filtré}}| = k_{\text{pred}}$$

4. Le générateur produit une sortie : $y = \text{Generateur}(q, P_{\text{filtré}})$

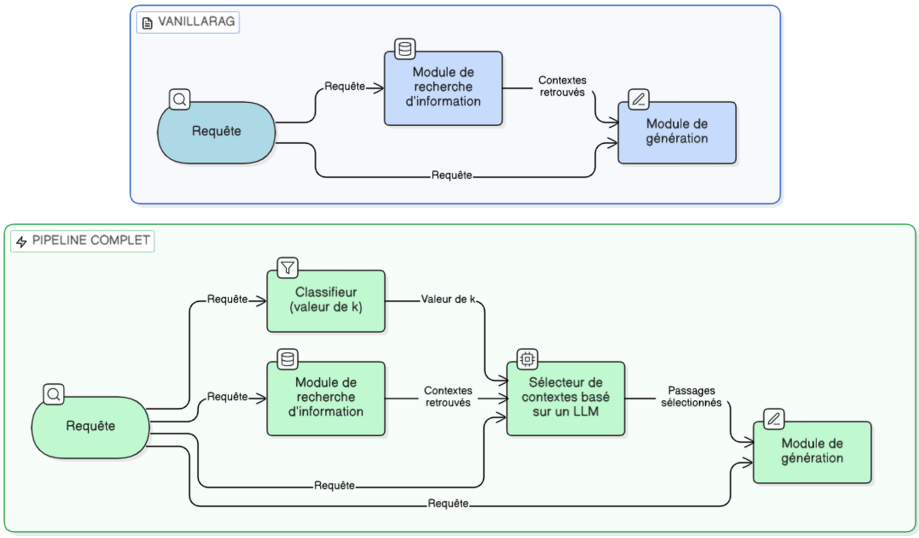

FIGURE 1 – Schéma du pipeline proposé

5 Protocole expérimental

5.1 Jeux de données et stratégie de recherche d'information

Nous avons sélectionné trois jeux de données multi-hop afin d'évaluer les performances du pipeline proposé. **MuSiQue** (Trivedi *et al.*, 2022), **2WikiMultihopQA** (Ho *et al.*, 2020) et **Multi-hop RAG** (Tang & Yang, 2024) sont tous composés de requêtes qui nécessitent un nombre variable de passages pour y répondre. Les requêtes vont de relativement simples, nécessitant deux passages, à plus complexes, nécessitant trois ou quatre passages.

MuSiQue Nous utilisons l'ensemble de validation MuSiQue-Ans, composé d'environ 2 500 requêtes, ainsi que le corpus prétraité proposé par Trivedi et al. (Trivedi *et al.*, 2023), constitué d'environ 130 000 passages. Un ensemble de 1000 passages est récupéré à l'aide d'une étape unique de recherche d'information dense (single-round dense retrieval). Les passages sont encodés à l'aide du modèle SentenceTransformer `all-MiniLM-L6-v2` , indexés dans ChromaDB, puis récupérés via la similarité cosinus. Les passages récupérés sont ensuite rerankés à l'aide du modèle `BAAI/bge-reranker-large`, dont les 25 premiers sont conservés comme contexte fourni au LLM.

2WikiMultiHopQA (2Wiki) Ce jeu de données contient environ 12 600 requêtes. Nous utilisons le jeu de données pré-récupéré basé sur BM25 fourni par *Rankify* (Abdallah *et al.*, 2025), comprenant les 1000 passages les mieux classés par requête, récupérés à partir d'un corpus de 21 millions de passages issus de Wikipédia. Nous appliquons ensuite la même configuration de reranking que ci-dessus.

MultihopRAG (MHR) Ce jeu de données comprend environ 2 600 requêtes ainsi qu'un corpus de 603 textes complets. Un découpage (chunking) simple est appliqué pour la phase de recherche d'information. Une division 80/20 est utilisée, avec 80 % des requêtes dédiées à l'entraînement du classifieur et 20 % réservées à l'évaluation, soit 451 requêtes pour l'évaluation. Nous adoptons la

même configuration de recherche d'information que pour MuSiQue, mais en récupérant un ensemble réduit de 100 passages pour le reranking, dont les 25 premiers sont conservés.

5.2 Classifieur

Afin de prédire dynamiquement le nombre de passages (k) à récupérer pour chaque requête, nous effectuons un fine-tuning d'un modèle RoBERTa sur les ensembles d'entraînement combinés des trois jeux de données (MuSiQue-Ans train, 2Wiki-train et les 80% de la portion d'entraînement de MHR). Le classifieur est entraîné dans un cadre de classification multi-classes, où l'entrée correspond à une requête et la sortie à une étiquette catégorielle indiquant le nombre de hop de raisonnement requis, soit 2-hop, 3-hop ou 4-hop ($k \in 2, 3, 4$). Ces étiquettes sont dérivées du nombre de reasoning hops de référence (ground truth) fourni par chacun des trois jeux de données. Nous évaluons le classifieur sur un jeu de données combiné composé de l'ensemble de validation MuSiQue-Ans, de l'ensemble de validation 2Wiki ainsi que des 20% restants de MHR dédiés au test, obtenant un score F1 macro-moyenné de 0,87 ainsi qu'une exactitude équilibrée (balanced accuracy) de 0,86.

Configuration de l'entraînement : le modèle est affiné pendant 5 époques à l'aide de l'optimiseur AdamW (taux d'apprentissage 2×10^{-5}) avec un linear scheduling. Afin de pallier le déséquilibre de la classe 2-hop, nous utilisons un *WeightedRandomSampler* pondéré par l'inverse des fréquences de classe. Une division 80/20 est appliquée en utilisant un échantillonnage stratifié afin de préserver les proportions de classes. L'entraînement est réalisé avec une taille de batch de 16, et le meilleur modèle est sélectionné sur la base de la précision obtenue sur l'ensemble de validation.

5.3 Sélecteur LLM

Pour la sélection du contexte, nous utilisons un `Mistral Nemo Instruct` (12,2 milliards de paramètres) avec un prompting en zéro-shot. La stratégie de prompting a demandé au modèle : (1) d'analyser et de raisonner sur la requête donnée et les passages récupérés, (2) de sélectionner $predicted_k$ passages en fonction de leur pertinence par rapport à la requête, et (3) de renvoyer les passages pertinents.

5.4 Génération et mesures d'évaluation

La phase finale de génération a utilisé trois modèles pour les comparer entre eux et par rapport à des références établies : (1) Flan-T5-XXL (Chung *et al.*, 2024), (2) Llama2-7b (Touvron *et al.*, 2023) et (3) Llama3-8b (Dubey *et al.*, 2024). Le générateur reçoit la requête et les passages prédits sélectionnés par le LLM afin de produire la réponse finale.

Évaluation : La génération a été évaluée à l'aide des mesures Exact Match (EM) et F1-score (F1), et la recherche d'information à l'aide des mesures standard Precision (P) et recall (R).

6 Résultats et discussion

6.1 Analyse comparative

Cette section fournit une évaluation des performances de bout en bout de notre approche, par rapport à une base de référence VanillaRAG, avec une configuration identique de recherche d'information et de reranking, et un k fixe standard de 5 passages pour la génération, ainsi que quatre modèles de

TABLE 2 – Résultats d'évaluation de bout en bout

	MuSiQue		2Wiki		MHR	
Model	EM	F1	EM	F1	EM	F1
VanillaRAG (Flan-T5-XXL)	15.6	24.5	32.0	38.2	60.1	64.5
AdaptiveRAG (Flan-T5-XXL)	20.6	28.5	-	-	-	-
Notre approche (Flan-T5-XXL)	22.6	32.6	36.9	43.3	64.1	68.4
VanillaRAG (Llama2-7b)	3.7	12.5	23.4	30.6	63.3	63.4
DRAGIN (Llama2-7b)	-	-	22.0	29.3	-	-
DynRAG (Llama2-7b)	6.6	7.0	26.8	32.8	59.3	60.0
Notre approche (Llama2-7b)	10.3	19.6	28.8	35.2	69.2	69.6
VanillaRAG (Llama3-8b)	13.3	23.3	27.4	34.4	63.6	64.3
DynRAG (Llama3-8b)	14.7	26.9	32.8	38.6	72.0	73.1
RankRAG (Llama3-8b)	-	-	31.4	36.9	-	-
Notre approche (Llama3-8b)	17.6	28.0	31.3	38.2	64.9	65.8

référence : (1) AdaptiveRAG (Jeong *et al.*, 2024), (2) DRAGIN (Su *et al.*, 2024), (3) RankRAG (Yu *et al.*, 2024) et (4) DynamicRAG (Sun *et al.*, 2025). Nous réimplémentons DynamicRAG en utilisant notre configuration de recherche d'information et de reranking pour une comparaison directe dans des conditions identiques. AdaptiveRAG et DRAGIN utilisent une recherche d'information itérative incompatible avec notre pipeline, et RankRAG n'est pas accessible au public. Nous rapportons donc leurs résultats publiés. Nous divisons la comparaison en fonction du modèle générateur utilisé (ou fine-tuned) par chaque référence afin d'assurer une comparaison équitable.

Les résultats du tableau 2 démontrent que notre pipeline surpasse considérablement le VanillaRAG à k fixe, obtenant des gains substantiels pour toutes les références. Notre approche améliore statistiquement les performances de VanillaRAG dans presque toutes les configurations (+2,1-8,0 F1, p corrigé par Bonferroni $< 0{,}001$), à l'exception des paramètres MHR Llama2 et Llama3 ($p = 0{,}09$ & 0,12), probablement en raison de la petite taille de l'échantillon de l'ensemble de données. Par rapport aux travaux précédents, notre approche atteint des performances compétitives, surpassant plusieurs références. Nous obtenons de solides performances sur MuSiQue, surpassant considérablement DynamicRAG sur chaque référence (+2,9-3,7 EM) et AdaptiveRAG sur Flan-T5-XXL (+2 EM & +4,1 F1). Ces résultats indiquent que MuSiQue est particulièrement sensible aux distracteurs et bénéficie d'une sélection contextuelle adaptative. Les performances sur 2Wiki sont moins constantes, avec des gains significatifs sur Llama2-7b par rapport à DRAGIN et DynamicRAG (+6,8 et +2 EM respectivement), et des gains plus faibles sur Llama3-8b (-0,1 EM et +1,3 F1) sur RankRAG, et (-1,5 EM et -0,4 F1) sur DynamicRAG. DynamicRAG surpasse largement MHR avec Llama3-8b, mais pas avec Llama2-7b. Il est intéressant de noter que notre approche enregistre les gains de performance les plus faibles sur la base de référence Llama3, ce qui suggère que ce modèle est plus robuste face aux distracteurs (ceci est confirmé par l'étude sur les distracteurs de la section 3, où il a enregistré la plus faible diminution en % entre les distracteurs « 0 » et « 3 » par rapport aux autres générateurs). L'évaluation de notre approche est particulièrement notable dans la mesure où aucun fine-tuning n'est appliqué à aucune étape du pipeline, à l'exception du module de classification, contrairement aux modèles DynamicRAG et RankRAG qui effectuent un fine-tuning direct de leurs modules de reranking et de génération. Par ailleurs, notre méthode repose sur une unique étape de recherche d'information, contrairement aux stratégies de recherche d'information multi-itérations employées par DRAGIN et AdaptiveRAG, lesquelles interrogent le module de recherche d'information de manière répétée sur la base de générations intermédiaires ou d'états de raisonnement successifs.

6.2 Évaluation du module de recherche d'information

Le tableau 3 présente une évaluation des performances en matière de recherche, de reranking et de sélection de contexte basée sur le LLM. L'évaluation a été réalisée uniquement sur MuSiQue en raison de l'absence de référence en matière de recherche pour Multihop-RAG et des scores de recherche non communiqués par Rankify pour 2WikiMultihopQA. Nous rapportons les mesures pour les seuils top-5 et top-25. Le top-5 correspond à la base de référence VanillaRAG qui utilise les 5 passages reclassés les mieux classés pour la génération, tandis que le top-25 représente le contexte reclassé fourni à notre sélecteur LLM. Notre sélecteur LLM sélectionne un nombre variable de passages ($k \in 2, 3, 4$) pour la génération. Nous calculons donc la précision et le rappel comme des moyennes sur la variable *k* plutôt que comme une limite fixe pour notre pipeline. Afin d'évaluer le sélecteur LLM indépendamment des erreurs de la recherche d'information en amont, nous mesurons le rappel du LLM par rapport à l'ensemble des passages pertinents présents dans le top 25 reclassé, en prenant le score du reclassement de 61,8 % à R@25 comme rappel maximal réalisable. En d'autres termes, si un seul contexte pertinent est présent parmi les 25 premiers et que le LLM l'a sélectionné, cela constitue un rappel complet. Selon cette mesure, le LLM atteint un rappel de 69 % tout en ne sélectionnant que ($k \in 2, 3, 4$) passages par requête. Pour l'évaluation de bout en bout, nous effectuons des mesures par rapport à tous les passages de référence du jeu de données. Le pipeline complet atteint une précision de 51 % et un rappel de 44,1 %. Bien que notre rappel soit légèrement inférieur au R@5 du reclassement (50,2 %), la précision s'améliore considérablement, passant de 25,5 % à 51 %. Cela démontre que malgré une couverture réduite, le LLM filtre efficacement le bruit, tout en conservant les preuves les plus pertinentes, car notre pipeline a surpassé le VanillaRAG à k fixe dans toutes les bases de référence.

TABLE 3 – Évaluation de la recherche d'information, du réordonnancement et de la sélection de contexte par le LLM sur MuSiQue

	P@5	**R@5**	**P@25**	**R@25**
Recherche d'information	14.6	39.3	4.2	54.3
Reranking	25.5	50.2	6.3	61.8
	Precision		**Recall**	
LLM (Relatif au reranker)	–		69.0	
Bout en bout	51.0		44.1	

TABLE 4 – Étude d'ablation

		MuSiQue		2Wiki		MHR	
Generateur	***Config.***	EM	F1	EM	F1	EM	F1
Flan-T5-XXL	(1)	20.3	30.5	34.9	41.4	60.0	64.9
	(2)	15.8	24.7	34.8	40.0	59.4	61.4
	Notre approche	22.6	32.6	36.9	43.3	64.1	68.4
Llama2-7b	(1)	7.9	17.0	27.1	33.6	66.4	66.5
	(2)	3.8	12.5	25.5	32.3	65.6	66.4
	Notre approche	10.3	19.6	28.8	35.2	69.2	69.6
Llama3-8b	(1)	16.1	26.5	30.1	37.2	63.0	63.7
	(2)	13.5	23.5	27.7	34.7	63.7	64.5
	Notre approche	17.6	28.0	31.3	38.2	64.9	65.8

6.3 Étude d'ablation

Dans cette section, nous quantifions l'« utilité » du classifieur dans le pipeline. Cette évaluation a été réalisée à l'aide de deux configurations : le sélecteur LLM reçoit les 25 textes les mieux classés et est invité soit (1) à sélectionner autant de passages pertinents qu'il en identifie, sans l'aide du classifieur, soit (2) à récupérer les 5 passages les plus pertinents selon les critères standard. Les résultats présentés dans le tableau 4 démontrent l'importance du classifieur et de la sélection adaptative du contexte. Le pipeline complet surpasse systématiquement les deux configurations d'ablation pour chaque ensemble de données et chaque générateur. De plus, le fait de permettre au LLM de déterminer dynamiquement le nombre de contextes à récupérer (config. 1) surpasse la sélection k fixe (config. 2) pour toutes les bases de référence. Ces résultats suggèrent que, bien que le LLM bénéficie d'une aide supplémentaire pour déterminer la quantité de contexte à sélectionner, une approche dynamique de la sélection du contexte est supérieure à une base de référence k fixe, quel que soit l'ensemble de données ou le modèle utilisé, ce qui démontre la généralisation des stratégies de recherche d'information adaptative. L'idée clé du classifieur réside dans la séparation explicite de l'évaluation de la complexité de la requête (par le classifieur) et de la sélection du contexte avec le LLM. En déterminant d'abord le nombre de passages requis, nous fournissons au sélecteur LLM un objectif clair et limité, le guidant pour sélectionner un nombre approprié de passages.

7 Conclusion

Dans ce travail, nous avons proposé un pipeline RAG dans le but d'atténuer l'impact des distracteurs sur la génération. Nous avons mené une analyse préliminaire montrant que les distracteurs dégradent les performances de génération, et avons introduit un cadre de sélection de contexte qui combine un classifieur spécifique à la requête avec un sélecteur basé sur un LLM afin de déterminer la quantité de contexte requise pour une requête donnée. Cette approche a été évaluée en comparant la recherche d'information et la génération de bout en bout à des références établies, ce qui a montré une amélioration des performances. Ces résultats soulignent la nécessité de méthodes de sélection de contexte plus dynamiques.

Les travaux futurs pourraient améliorer davantage le cadre proposé en rendant la sélection de contexte plus adaptative. Par exemple, au lieu de s'appuyer uniquement sur la requête pour estimer la quantité de contexte nécessaire, de futures approches pourraient analyser conjointement la requête et les passages récupérés afin de mieux déterminer la quantité de contexte réellement utile. De plus, le classifieur actuel est entraîné à partir des annotations de vérité terrain fournies par les jeux de données existants, mais des stratégies de supervision plus robustes pourraient être explorées, telles que l'apprentissage par renforcement ou l'apprentissage auto-supervisé, afin de mieux capturer la pertinence contextuelle. Une autre direction prometteuse consisterait à étudier des mécanismes de recherche itératifs ou guidés par rétroaction, dans lesquels le modèle de génération demanderait dynamiquement du contexte supplémentaire uniquement lorsque cela est nécessaire au cours du raisonnement.

8 Remerciements

Ce travail a bénéficié d'un accès aux ressources de calcul haute performance (HPC) de l'IDRIS dans le cadre de l'allocation AD011015885 accordée par GENCI.

Références

ABDALLAH A., PIRYANI B., MOZAFARI J., ALI M. & JATOWT A. (2025). Rankify : A comprehensive python toolkit for retrieval, re-ranking, and retrieval-augmented generation. *arXiv preprint arXiv :2502.02464*.

AMIRAZ C., CUCONASU F., FILICE S. & KARNIN Z. (2025). The distracting effect : Understanding irrelevant passages in rag. *arXiv preprint arXiv :2505.06914*.

ASAI A., WU Z., WANG Y., SIL A. & HAJISHIRZI H. (2024). Self-rag : Learning to retrieve, generate, and critique through self-reflection.

BOLOTOVA-BARANOVA V., BLINOV V., FILIPPOVA S., SCHOLER F. & SANDERSON M. (2023). Wikihowqa : A comprehensive benchmark for multi-document non-factoid question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 5291–5314.

CHUNG H. W., HOU L., LONGPRE S., ZOPH B., TAY Y., FEDUS W., LI Y., WANG X., DEHGHANI M., BRAHMA S. *et al.* (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, **25**(70), 1–53.

CUCONASU F., TRAPPOLINI G., SICILIANO F., FILICE S., CAMPAGNANO C., MAAREK Y., TONELLOTTO N. & SILVESTRI F. (2024). The power of noise : Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 719–729.

DUBEY A., JAUHRI A., PANDEY A., KADIAN A., AL-DAHLE A., LETMAN A., MATHUR A., SCHELTEN A., YANG A., FAN A. *et al.* (2024). The llama 3 herd of models. *arXiv e-prints*, p. arXiv–2407.

GABBURO M., JEDEMA N. P., GARG S., RIBEIRO L. F. & MOSCHITTI A. (2024). Measuring retrieval complexity in question answering systems. *arXiv preprint arXiv :2406.03592*.

GAO Y., XIONG Y., GAO X., JIA K., PAN J., BI Y., DAI Y., SUN J., WANG H. & WANG H. (2023). Retrieval-augmented generation for large language models : A survey. *arXiv preprint arXiv :2312.10997*, **2**(1).

GUU K., LEE K., TUNG Z., PASUPAT P. & CHANG M. (2020). Retrieval augmented language model pre-training. In *International conference on machine learning*, p. 3929–3938 : PMLR.

HO X., NGUYEN A.-K. D., SUGAWARA S. & AIZAWA A. (2020). Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv :2011.01060*.

JEONG S., BAEK J., CHO S., HWANG S. J. & PARK J. C. (2024). Adaptive-rag : Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv :2403.14403*.

JI Z., LEE N., FRIESKE R., YU T., SU D., XU Y., ISHII E., BANG Y. J., MADOTTO A. & FUNG P. (2023). Survey of hallucination in natural language generation. *ACM computing surveys*, **55**(12), 1–38.

JIANG Z., XU F. F., GAO L., SUN Z., LIU Q., DWIVEDI-YU J., YANG Y., CALLAN J. & NEUBIG G. (2023). Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 7969–7992.

JIN B., YOON J., HAN J. & ARIK S. O. (2024). Long-context llms meet rag : Overcoming challenges for long inputs in rag. *arXiv preprint arXiv :2410.05983*.

KARPUKHIN V., OGUZ B., MIN S., LEWIS P. S., WU L., EDUNOV S., CHEN D. & YIH W.-T. (2020). Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, p. 6769–6781.

KHRAMTSOVA E., ZHUANG S., BAKTASHMOTLAGH M. & ZUCCON G. (2024). Leveraging llms for unsupervised dense retriever ranking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 1307–1317.

KOOPMAN B. & ZUCCON G. (2023). Dr chatgpt tell me what i want to hear : How different prompts impact health answer correctness. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, p. 15012–15022.

LEWIS P., PEREZ E., PIKTUS A., PETRONI F., KARPUKHIN V., GOYAL N., KÜTTLER H., LEWIS M., YIH W.-T., ROCKTÄSCHEL T. *et al.* (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, **33**, 9459–9474.

LIU N. F., ZHANG T. & LIANG P. (2023). Evaluating verifiability in generative search engines. *arXiv preprint arXiv :2304.09848*.

MA X., ZHANG X., PRADEEP R. & LIN J. (2023). Zero-shot listwise document reranking with a large language model. *arXiv preprint arXiv :2305.02156*.

MIN S., ZHONG V., ZETTLEMOYER L. & HAJISHIRZI H. (2019). Multi-hop reading comprehension through question decomposition and rescoring. *arXiv preprint arXiv :1906.02916*.

PEREZ E., LEWIS P., YIH W.-T., CHO K. & KIELA D. (2020). Unsupervised question decomposition for question answering. *arXiv preprint arXiv :2002.09758*.

PRESS O., ZHANG M., MIN S., SCHMIDT L., SMITH N. A. & LEWIS M. (2023). Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics : EMNLP 2023*, p. 5687–5711.

SU W., TANG Y., AI Q., WU Z. & LIU Y. (2024). Dragin : Dynamic retrieval augmented generation based on the information needs of large language models. *arXiv preprint arXiv :2403.10081*.

SUN J., ZHONG X., ZHOU S. & HAN J. (2025). Dynamicrag : Leveraging outputs of large language model as feedback for dynamic reranking in retrieval-augmented generation. *arXiv preprint arXiv :2505.07233*.

TAGUCHI C., MAEKAWA S. & BHUTANI N. (2025). Efficient context selection for long-context qa : No tuning, no iteration, just adaptive-k. *arXiv preprint arXiv :2506.08479*.

TANG Y. & YANG Y. (2024). Multihop-rag : Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv :2401.15391*.

TOUVRON H., MARTIN L., STONE K., ALBERT P., ALMAHAIRI A., BABAEI Y., BASHLYKOV N., BATRA S., BHARGAVA P., BHOSALE S. *et al.* (2023). Llama 2 : Open foundation and fine-tuned chat models. *arXiv preprint arXiv :2307.09288*.

TRIVEDI H., BALASUBRAMANIAN N., KHOT T. & SABHARWAL A. (2022). Musique : Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, **10**, 539–554.

TRIVEDI H., BALASUBRAMANIAN N., KHOT T. & SABHARWAL A. (2023). Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1 : long papers)*, p. 10014–10037.

WANG Z., ARAKI J., JIANG Z., PARVEZ M. R. & NEUBIG G. (2023). Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv :2311.08377*.

XIONG W., LI X. L., IYER S., DU J., LEWIS P., WANG W. Y., MEHDAD Y., YIH W.-T., RIEDEL S., KIELA D. *et al.* (2020). Answering complex open-domain questions with multi-hop dense retrieval. *arXiv preprint arXiv :2009.12756*.

YADAV V., BETHARD S. & SURDEANU M. (2020). Unsupervised alignment-based iterative evidence retrieval for multi-hop question answering. *arXiv preprint arXiv :2005.01218*.

YU Y., PING W., LIU Z., WANG B., YOU J., ZHANG C., SHOEYBI M. & CATANZARO B. (2024). Rankrag : Unifying context ranking with retrieval-augmented generation in llms. *Advances in Neural Information Processing Systems*, **37**, 121156–121184.

ZAMANI H., TRIPPAS J. R., DALTON J., RADLINSKI F. *et al.* (2023). Conversational information seeking. *Foundations and Trends® in Information Retrieval*, **17**(3-4), 244–456.