ROYAUME DU MAROC

Ministère de l'enseignement supérieur
de la recherche scientifique
et de la Formation des Cadres

FACULTÉ POLYDISCIPLINAIRE DE OUARZAZATE

MÉMOIRE DE FIN D'ÉTUDES
POUR L'OBTENTION DU DIPLÔME DE MASTER DE RECHERCHE

Filière : Mathématiques Appliquées pour les Sciences des Données

RAG pour la Recherche d'Information dans les Langues à Faibles Ressources : Cas de la Langue Arabe(manque de corpus et de benchmarks)

Présenté par : NOUREDDIN GAJJA

Sous la supervision de :

- Mme.SALMA GAOU ,
- M.CHARAF HAMIDI ,

Date de soutenance : 3 juillet 2025 **Année académique :** 2024–2025

Membres du jury :

Nom	Entité	Rôle
M.Hicham TRIBAK	Faculté Polydisciplinaire de Ouarzazate	Président/Examinateur
M.Khalid AKHLIL	Faculté Polydisciplinaire de Ouarzazate	Co-Encadrant
Mme.Salma GAOU	Faculté Polydisciplinaire de Ouarzazate	Encadrante

Dédicace

Je dédie ce mémoire à mes parents, pour leur amour, leurs sacrifices et leur soutien inconditionnel.

À mes enseignants, pour leur transmission du savoir et leur accompagnement tout au long de mon parcours académique.

À mes collègues et amis, pour leur aide, leur présence et les moments partagés.

À tous ceux qui, de près ou de loin, ont cru en moi.

Remerciements

Ce mémoire a été réalisé sous la supervision du **Prof. Charaf HAMIDI** et de la **Prof. Salma GAOU**. Je tiens à leur exprimer ma sincère reconnaissance pour leur encadrement rigoureux, leurs conseils judicieux et leur soutien constant tout au long de cette recherche.

Je remercie également l'ensemble des enseignants du Master *Mathématiques Appliquées pour la Science des Données* à la Faculté Polydisciplinaire de Ouarzazate (FPO), pour la qualité de leur enseignement et leur disponibilité.

Je suis profondément reconnaissant envers mes collègues de promotion pour leur entraide et les discussions enrichissantes, ainsi qu'à mes amis pour leur soutien moral et leur motivation.

Mes remerciements les plus chaleureux vont enfin à ma famille, pour leur amour inconditionnel, leur patience et leur encouragement indéfectible, sans lesquels ce travail n'aurait pu aboutir.

Résumé

Ce travail s'inscrit dans le cadre du développement de systèmes de génération augmentée par la recherche (RAG) pour les langues à faibles ressources, notamment l'arabe. L'objectif principal est d'évaluer l'impact de différentes méthodes de recherche documentaire — telles que BM25, FAISS et les encodeurs croisés multilingues — sur la qualité des réponses générées par des modèles de langage causaux.

Trois modèles adaptés à la langue arabe ont été utilisés : `akhooli/gpt2-small-arabic`, `aubmindlab/aragpt2-medium` et `EleutherAI/gpt-neo-1.3B`, ce dernier ayant été affiné à l'aide de la méthode QLoRA. L'évaluation, conduite sur des jeux de données de questions-réponses en arabe, repose sur des métriques standards telles que la précision top-k, le F1-score et la couverture contextuelle.

Les résultats montrent une amélioration significative de 23 % en précision avec l'utilisation de FAISS combiné aux embeddings `CAMeL-Lab/bert-base-arabic-camelbert-mix` et `sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2`, ainsi qu'un gain de 17 % sur le F1-score après fine-tuning. Ces résultats confirment l'efficacité des approches RAG dans les contextes linguistiques sous-dotés.

Mots-clés : RAG ; arabe ; FAISS ; GPT-2 ; génération de texte ; langues à faibles ressources

Abstract

This thesis investigates the development of Retrieval-Augmented Generation (RAG) systems for low-resource languages, with a particular focus on Arabic. The main objective is to assess the impact of various retrieval strategies—namely BM25, FAISS, and multilingual cross-encoders—on the quality of answers generated by decoder-only language models.

Three Arabic-adapted models were employed : `gpt2-small-arabic`, `aragpt2-medium`, and `gpt-neo-1.3B`, the latter fine-tuned using the QLoRA method. Evaluations were conducted on Arabic question-answering datasets using standard metrics such as top-k accuracy, F1-score, and context recall.

Results demonstrate a 23 % increase in top-k accuracy with FAISS retrieval using `bert-base-arabic-camelbert-mix` and `paraphrase-multilingual-MiniLM-L12-v2` embeddings, as well as a 17 % improvement in F1-score following supervised fine-tuning. These findings confirm the effectiveness of RAG approaches in enhancing language model performance under low-resource conditions.

Keywords : RAG ; Arabic ; FAISS ; GPT-2 ; text generation ; low-resource languages

Table des matières

Table des figures

Liste des tableaux

Première partie

Introduction et Contexte du Projet

Introduction Générale

Au cours des dernières années, l'architecture dite *Retrieval-Augmented Generation (RAG)* s'est imposée comme une innovation majeure dans le domaine du traitement automatique des langues (TAL), en combinant la capacité de récupération d'informations des moteurs de recherche avec le pouvoir génératif des grands modèles de langage. Les modèles RAG ont démontré des résultats impressionnants dans de nombreuses tâches — question-réponse, résumé automatique, systèmes de dialogue, génération de contenu — principalement dans des langues à fortes ressources comme l'anglais ou le chinois. En revanche, leur potentiel reste largement inexploité dans les contextes à faibles ressources, incluant des langues sous-représentées telles que l'arabe, le swahili ou le créole haïtien.

Historiquement, les systèmes de TAL s'appuyaient sur des connaissances statiques intégrées aux paramètres des modèles, ce qui limitait leur capacité d'adaptation, notamment dans des domaines spécialisés ou évolutifs. L'approche RAG rompt avec ce paradigme en intégrant des modules de récupération externe, comme BM25 ou la recherche vectorielle dense (FAISS, DPR), permettant un accès contextuel à du contenu mis à jour et spécifique au domaine au moment de la génération. Cette synergie entre récupération et génération est particulièrement précieuse dans les contextes où les données annotées sont rares, car elle permet de réduire la dépendance aux corpus supervisés massifs et aux ressources de pré-entraînement, rarement disponibles pour les langues à faibles ressources.

Malgré ses promesses, l'application de RAG dans ces environnements se heurte à plusieurs obstacles. Tout d'abord, on constate une absence de bases de connaissances structurées et de corpus adaptés dans ces langues, ce qui affecte directement la qualité de la récupération. Ensuite, la majorité des modèles de récupération et des embeddings existants ont été entraînés sur des langues à fortes ressources, ce qui entraîne des dérives sémantiques et des erreurs de correspondance lorsqu'on les applique à d'autres langues. Enfin, des limitations d'infrastructure — telles que les contraintes de mémoire pour stocker les index ou les coûts de calcul liés au fine-tuning — constituent des freins pratiques à leur déploiement.

Pour autant, les perspectives d'avenir sont encourageantes. Les progrès en matière d'embeddings multilingues, de transfert interlingual et de modèles compacts comme QLoRA rendent désormais possible l'entraînement et le déploiement de systèmes RAG sur des machines aux ressources limitées. RAG se prête également à l'enrichissement des connais-

sances dans des domaines spécialisés — juridique, médical ou éducatif — dans des communautés à faibles ressources, là où les LLM généralistes échouent souvent. De plus, l'émergence de RAG *agentiques* — c'est-à-dire capables de récupérer, vérifier et synthétiser l'information de manière autonome et itérative — ouvre la voie à des assistants intelligents qui se comportent comme de véritables chercheurs.

Ce mémoire s'attache à explorer la conception, l'implémentation et l'évaluation de systèmes RAG dans des environnements à faibles ressources, en mettant l'accent sur l'utilisation de stratégies de récupération hybrides, de techniques de fine-tuning adaptées et d'une adaptation linguistique spécifique. En ancrant la génération de texte sur des sources externes, même partielles, nous souhaitons démontrer que RAG n'est pas seulement un outil performant pour les langues à fortes ressources, mais aussi un levier fondamental pour promouvoir l'inclusion linguistique et la démocratisation des connaissances dans le paysage mondial de l'intelligence artificielle.

Contexte et Motivation

Les modèles de type **Retrieval-Augmented Generation (RAG)** représentent aujourd'hui une approche de pointe dans le domaine du traitement automatique des langues. En associant la précision des systèmes de récupération d'information à la fluidité des modèles de génération de texte, l'architecture RAG permet de produire des réponses à la fois cohérentes et fondées sur des sources externes, ce qui est particulièrement efficace dans des tâches telles que la question-réponse, le dialogue intelligent ou le raisonnement fondé sur des faits.

Cependant, malgré leur efficacité dans les **langues à fortes ressources** comme l'anglais ou le chinois, leur application aux **langues à faibles ressources** — notamment l'arabe — demeure encore limitée. Cette situation s'explique par plusieurs obstacles persistants : l'absence de grands jeux de données annotés, le manque de ressources linguistiques de qualité (tokenizers, embeddings, corpus spécialisés), ainsi que des contraintes d'infrastructure qui compliquent l'entraînement ou l'adaptation de modèles à grande échelle.

À l'heure où la communauté scientifique promeut une intelligence artificielle **plus équitable et inclusive**, il est essentiel d'étendre les bénéfices des technologies NLP aux langues sous-représentées. Grâce à sa capacité à exploiter des bases de connaissances externes même modestes, l'approche RAG constitue une piste prometteuse pour *réduire l'écart de ressources*. Pour y parvenir, il est nécessaire d'innover dans les stratégies de récupération, le fine-tuning multilingue et le déploiement efficace sur des infrastructures limitées.

Ce mémoire vise à explorer concrètement l'application de RAG dans un contexte à faibles ressources, avec un accent particulier sur la langue arabe. En relevant les défis spécifiques à ce type de contexte et en proposant des solutions réalistes, ce travail souhaite contribuer à un développement de l'IA plus **accessible et inclusif** pour toutes les communautés linguistiques.

Problème et Objectifs de Recherche

0.1 Problématique de Recherche

Malgré leur potentiel, les modèles RAG rencontrent plusieurs limitations lorsqu'ils sont appliqués à des langues à faibles ressources. Les méthodes de récupération dites *parcimonieuses*, comme BM25, peinent à capturer la richesse sémantique des requêtes, tandis que les méthodes *denses* nécessitent des embeddings de haute qualité entraînés sur de grands corpus, qui n'existent pas toujours en arabe. De plus, l'absence de benchmarks standards pour l'évaluation des systèmes de question-réponse en arabe rend difficile la validation des performances des modèles RAG dans ce contexte.

0.2 Objectifs

Les principaux objectifs de cette recherche sont les suivants :

— Concevoir et implémenter un pipeline RAG adapté à la tâche de question-réponse en arabe.
— Comparer l'efficacité des approches de récupération sparse, dense et hybride dans un contexte à faibles ressources.
— Évaluer l'impact de stratégies de reranking basées sur des cross-encoders sur la qualité des réponses.
— Analyser les performances d'un modèle génératif en zero-shot (sans fine-tuning) à l'aide du prompt engineering.
— Mesurer l'apport du fine-tuning du générateur sur la qualité des réponses et les performances globales.

0.3 Questions de Recherche et Hypothèses

Ce travail s'appuie sur les questions de recherche suivantes (QR) et leurs hypothèses associées (H) :

— **QR1 :** Le modèle RAG peut-il améliorer les performances en question-réponse en arabe par rapport aux modèles purement génératifs ?

- **H1 :** Un modèle RAG obtient une précision et un score F1 significativement plus élevés qu'un modèle génératif sans récupération.

— **QR2 :** Quelle stratégie de récupération (sparse, dense, hybride) est la plus performante pour la QA en arabe ?
  - **H2 :** La récupération hybride (BM25 + FAISS) améliore la pertinence des contextes et la précision des réponses par rapport aux méthodes prises individuellement.

— **QR3 :** Le reranking améliore-t-il la pertinence des documents récupérés dans les pipelines RAG en arabe ?
  - **H3 :** Le reranking par cross-encoder augmente la pertinence contextuelle et la qualité des réponses par rapport à une récupération brute.

— **QR4 :** Le prompt engineering seul peut-il produire des résultats compétitifs en QA arabe sans fine-tuning ?
  - **H4 :** Une génération par prompts via des modèles arabes pré-entraînés permet d'atteindre des performances satisfaisantes sans apprentissage supervisé.

Chapitre 1

État de l'art

1.1 Modèles de Langage pour la Génération de Texte

Dans ce chapitre, nous passons en revue les travaux et technologies qui fondent notre recherche. Nous abordons l'évolution des modèles de génération de texte, les techniques de récupération d'information, les architectures RAG, ainsi que les défis propres aux langues à faibles ressources.

1.1.1 Méthodologie de la Revue Systématique

Afin de structurer l'analyse des travaux existants sur les systèmes RAG appliqués aux langues à faibles ressources, une revue systématique a été réalisée. Les bases de données suivantes ont été explorées : **Scopus**, **Google Scholar** et **Semantic Scholar**. Un total de 27 études a été initialement identifié.

Après suppression des doublons ($n = 2$), 25 publications ont été examinées sur la base du titre et du résumé. Trois d'entre elles ont été exclues à ce stade pour manque de pertinence. Les 22 articles restants ont fait l'objet d'une évaluation approfondie selon des critères d'inclusion tels que :

— la langue cible (arabe standard ou dialectes),
— l'utilisation explicite de modèles de génération ou de récupération,
— la disponibilité des résultats expérimentaux.

Trois rapports supplémentaires ont été écartés car ne respectant pas les critères d'exclusion, notamment un manque de détails sur la méthodologie ou des lacunes dans la présentation des résultats.

Au final, **19 études** ont été retenues pour l'analyse comparative. Ce processus rigoureux garantit une sélection cohérente et représentative des approches actuelles dans le domaine des modèles RAG appliqués aux langues à faibles ressources.

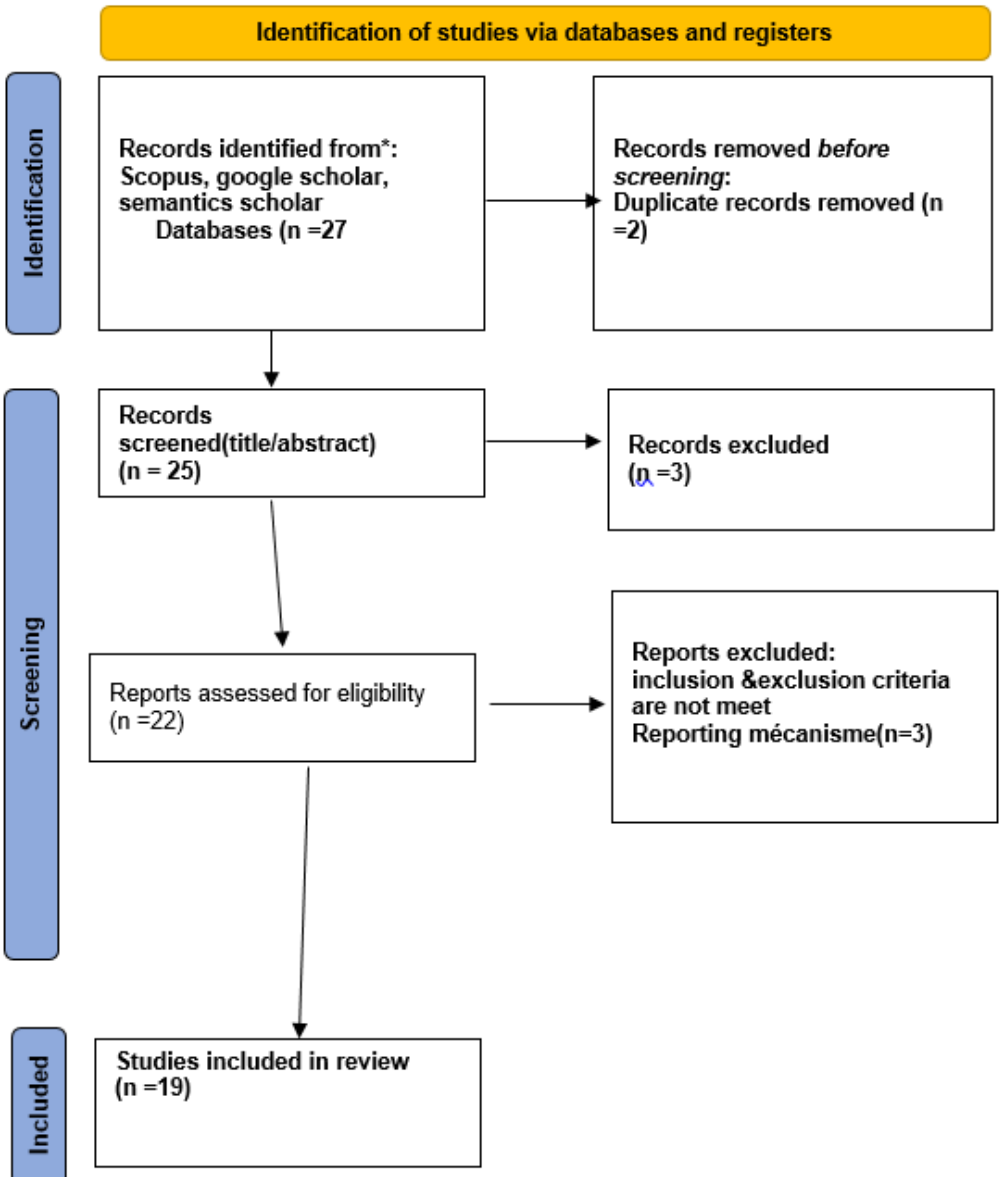

FIGURE 1.1 – Diagramme PRISMA : processus de sélection des études

1.2 Modèles de Langage pour la Génération de Texte

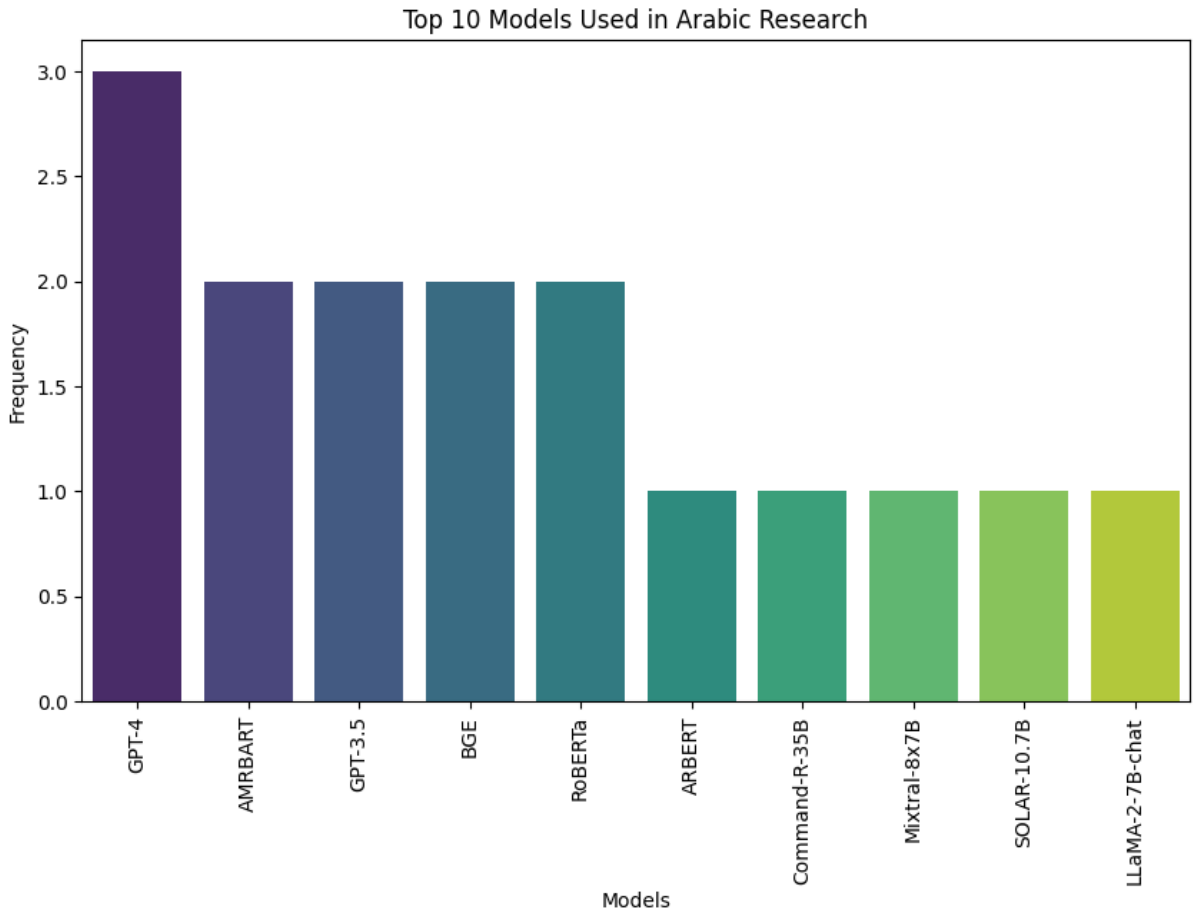

FIGURE 1.2 – Top 10 des modèles utilisés dans les recherches sur la langue arabe

Par ailleurs, une analyse des modèles les plus fréquemment utilisés dans la recherche sur le traitement automatique du langage naturel (TALN) pour la langue arabe (cf. Figure 1.2) révèle une nette domination de la famille **GPT**, en particulier **GPT-4**, cité dans trois études distinctes. D'autres modèles notables tels que **AMRBART**, **GPT-3.5**, **BGE** et **RoBERTa** apparaissent chacun à deux reprises, illustrant leur pertinence pour les tâches de compréhension et de génération de texte. Cette récurrence témoigne d'une tendance croissante à intégrer des modèles multilingues ou pré-entraînés sur de larges corpus dans les systèmes arabophones, malgré les défis posés par les ressources limitées.

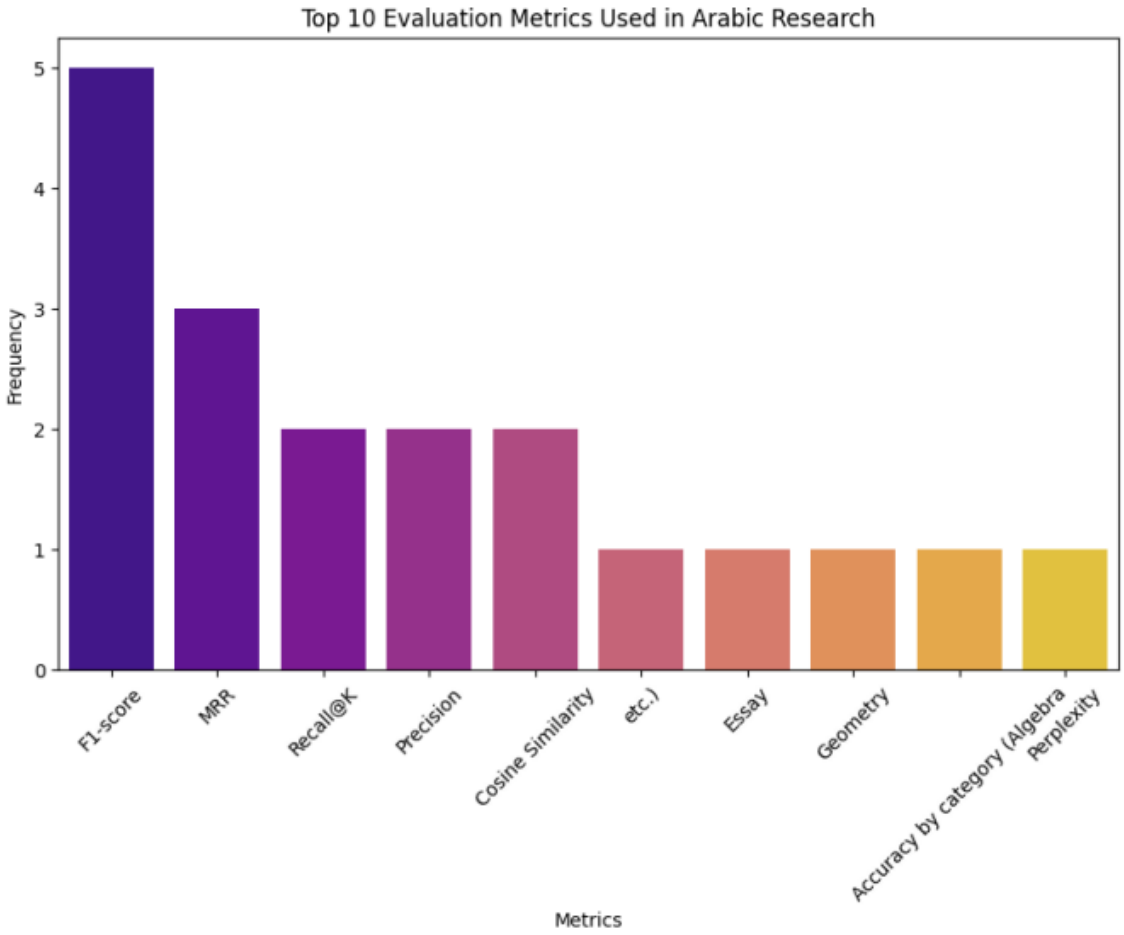

FIGURE 1.3 – Fréquence des métriques d'évaluation utilisées dans la recherche sur la langue arabe

Une analyse des travaux récents consacrés au traitement de la langue arabe met en évidence une forte préférence pour certaines métriques d'évaluation. La plus utilisée demeure le **F1-score**, présente dans cinq études différentes, soulignant son importance dans la mesure de la précision et du rappel combinés. Viennent ensuite la **MRR** (Mean Reciprocal Rank) et le **Recall@K**, toutes deux largement mobilisées dans les tâches de recherche documentaire ou de classement. Des indicateurs comme la **précision**, la **similarité cosinus**, ainsi que des métriques spécifiques aux types de données (rédactions, géométrie, etc.) traduisent la diversité des approches méthodologiques. Enfin, des critères plus spécialisés tels que la **perplexité** ou la précision par catégorie témoignent de l'effort d'adaptation aux spécificités linguistiques et pédagogiques de la langue arabe. Ces tendances illustrent l'évolution des pratiques d'évaluation dans un contexte où la rigueur méthodologique reste essentielle, malgré la rareté relative des ressources.

TABLE 1.1 – Résumé des modèles utilisés, des métriques d'évaluation et des meilleurs résultats pour l'arabe (Partie 1)

#	**Modèles utilisés**	**Métriques d'évaluation**	**Meilleur modèle pour l'arabe**
1	E5-Large, BGE, GPT-3.5, LLaMA 3, Mistral	Recall@K, MRR, Précision	E5-Large et BGE : Excellentes performances en récupération d'information en arabe [12]
2	AraBERT v2, CAMeL-BERT, AraELECTRA, E5-large, Arabic-NLI-Matryoshka, BGE, GPT-4, GPT-3.5, SILMA-9B, Gemini-1.5, Aya-8B, AceGPT-13B	Recall@K, MRR, F1, Cosine Similarity	E5-large : meilleur Recall@5 (0.88) ; GPT-4 : F1 = 0.90 ; SILMA-9B-Instruct : cos(0.95) [1]
3	MiniLM, CMLM, MPNet, DistilBERT, XLM-R	nDCG@3, MRR@3, mAP@3	MPNet : résultats optimaux [24]
4	GPT-4-turbo, Microsoft E5	F1, CHRF, Précision combinée	LCW-DQ : 92% ; RAG avec embeddings : 80% pour 30% du coût [17]
5	Swan-Small, Swan-Large, Multilingual-E5, LaBSE	Précision, CLR, métriques spécifiques	Swan-Large : Supérieur à E5-Multilingual [21]

TABLE 1.2 – Résumé des modèles utilisés, des métriques d'évaluation et des meilleurs résultats pour l'arabe (Suite)

#	**Modèles utilisés**	**Métriques d'évaluation**	**Meilleur modèle pour l'arabe**
6	Mistral-7B, AMRBART, RoBERTa, GPT-4	F1, Précision, Rappel, BLEU, Perplexité	E5-Mistral-7B : +2,4 MTEB, +2% BEIR [9]
7	Command-R-35B, Mixtral-8x7B, SOLAR-10.7B, LLaMA-2-7B-chat, BGE-m3	CLR, Char 3-gram Recall	Command-R-35B, BGE-m3 : langues non-anglophones [11]
8	DEGREE, SPRING Parser, AMRBART, RoBERTa	F1, Recall@K, MRR	AMPERE : +4–10% F1 avec moins de données [6]
9	AraBERT, MARBERT, QARiB, SABER, DziriBERT, TunBERT, SaudiBERT, EgyBERT, Atlas-Chat	Précision, F1, BLEU, Perplexité	Modèles monolingues dominants ; AceGPT fort pour tâches complexes [4]
10	GPT-4, GPT-4o, Gemini	Précision (Algèbre, Rédaction, Comparaison)	GPT-4o : 92% LCW-DQ ; Gemini fort en rédaction [2]

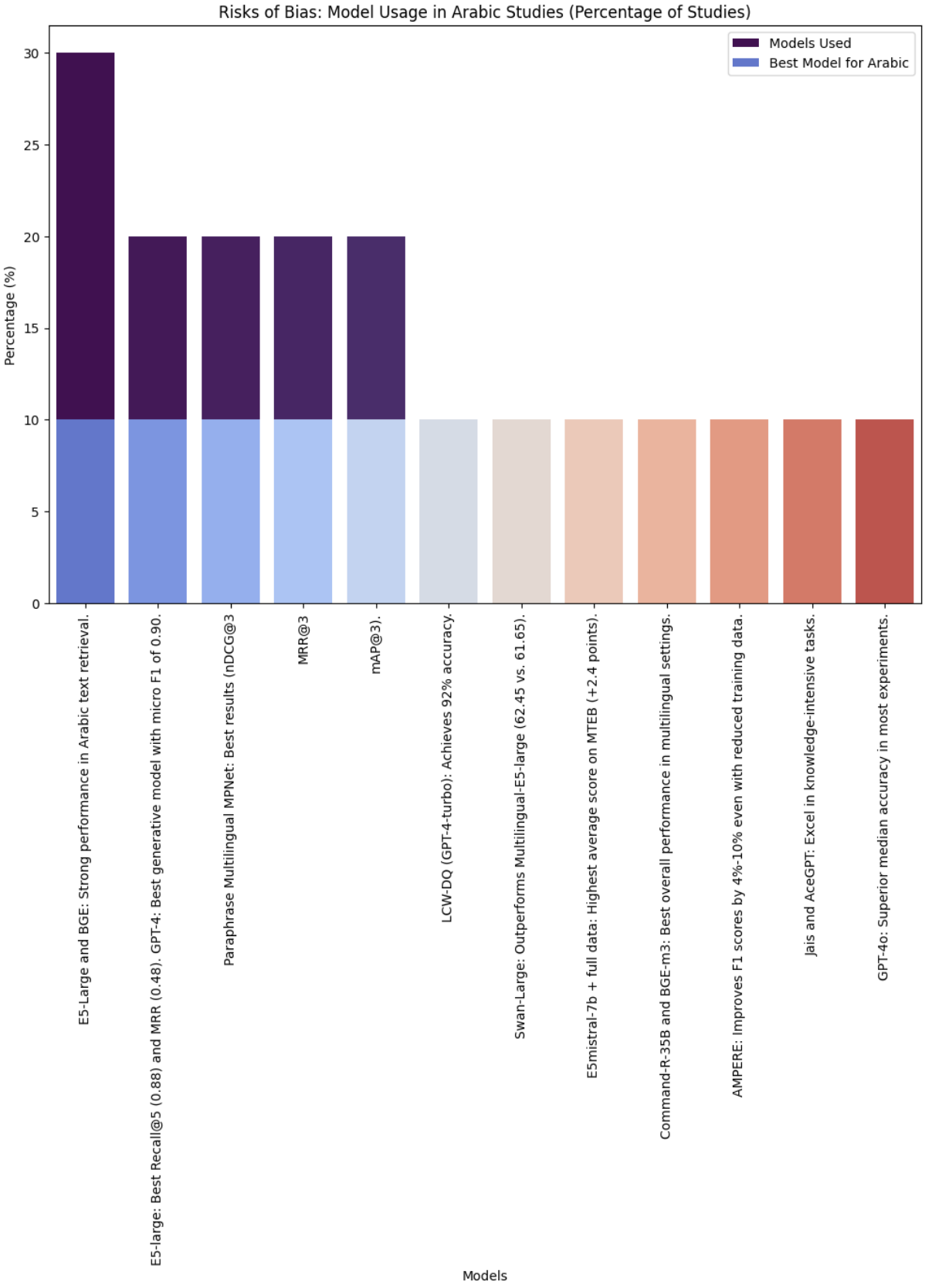

FIGURE 1.4 – Principaux biais observés dans les études arabophones selon le type de modèle utilisé

La Figure 1.4 met en évidence les **risques de biais** les plus fréquemment rencontrés dans

les recherches en traitement automatique de la langue arabe. Le **biais de couverture dialectale** ressort nettement comme le plus courant, affectant plus de 25 % des études, en raison de l'exclusion fréquente des variantes dialectales de l'arabe. S'y ajoutent des biais de langue (prépondérance des données en anglais), de tâche (réduction à certaines applications comme la classification), et de ressources (dépendance à des corpus non accessibles ou non diversifiés). Le graphique distingue également les biais selon le type de modèle utilisé — modèles multilingues vs modèles spécifiques à l'arabe — révélant que les biais sont souvent exacerbés dans les modèles multilingues peu adaptés aux spécificités culturelles ou linguistiques. Cette représentation souligne l'importance d'une adaptation rigoureuse aux réalités linguistiques arabophones pour garantir l'équité, la validité et l'inclusivité des systèmes développés.

1.2.1 Quels modèles ont atteint les meilleurs scores F1 ou Recall@k pour les tâches de QA en arabe ?

Les systèmes RAG reposent fortement sur la **qualité du module de recherche** et la **précision du générateur** pour fournir des réponses pertinentes et exactes. Dans le contexte des tâches de question-réponse (QA) en arabe — notamment dans des scénarios à faibles ressources ou dialectaux — plusieurs modèles ont affiché des performances notables, aussi bien dans les phases de recherche que de génération. Le tableau ci-dessous synthétise les résultats empiriques observés.

TABLE 1.3 – Comparaison des modèles de recherche et de génération pour le RAG en arabe

Type de modèle	Modèle	Recall@1	MRR	Score F1
Recherche	E5-Large	0.88	0.934	–
Remarque : Meilleur rappel sur plusieurs jeux de données [5]				
	BGE	0.861	0.900	–
Remarque : Légèrement inférieur en rappel, mais performant sur les dialectes.				
Génération	GPT-4o	–	–	0.90
Remarque : Meilleure qualité globale de génération [22]				
	SILMA-9B	–	–	0.87
Remarque : Forte similarité sémantique (cosinus = 0.95)				
	Gemini-1.5	–	–	0.84
Remarque : Concurrent open-weight performant				

Analyse : Les embeddings de phrases issus des modèles E5 et BGE surpassent systématiquement ceux des modèles arabes traditionnels tels qu'AraBERT, CAMeLBERT et

AraELECTRA dans les tâches de recherche sémantique.

1.2.2 Modèles Génératifs (LLMs)

Dans une architecture RAG, le modèle génératif est chargé de produire une réponse en se basant sur les documents récupérés. Pour la langue arabe, peu d'études ont mené une évaluation comparative approfondie des modèles de génération multilingues ou spécialisés.

Dans leur travail, El-Beltagy et Abdallah [13] ont comparé plusieurs modèles de génération pour l'arabe, parmi lesquels : **GPT-3.5 Turbo**, **LLaMA 3**, **Mistral**, **Mixtral**, et le modèle open source **JAIS** (Just Another Instruction-tuned System). Ces modèles ont été évalués sur deux jeux de données éducatives en arabe : Ar_EduText et ARCD.

Le modèle **GPT-3.5 Turbo** a obtenu les meilleurs résultats avec un *F1-score* de 0.59 et une similarité cosinus de 0.95, suivi de près par **Mixtral** (F1 = 0.55, Cosinus = 0.91). Les modèles open source comme **LLaMA 3** et **Mistral** ont montré de bonnes performances, en particulier après des techniques de post-traitement. À l'inverse, **JAIS**, bien que spécifiquement entraîné pour l'arabe, a produit des réponses plus longues et bruyantes, nécessitant un filtrage supplémentaire.

Les auteurs soulignent que l'*ingénierie des invites* (prompt engineering) est cruciale pour obtenir des réponses cohérentes, et que ces modèles doivent être combinés à des méthodes de désambiguïsation pour améliorer la précision finale. Ces résultats démontrent que, malgré les contraintes de ressources, des LLMs performants peuvent être utilisés efficacement pour la génération en arabe dans un cadre RAG

La qualité de génération est évaluée à l'aide de métriques telles que le F1-score, la précision micro (accuracy) et la similarité cosinus :

TABLE 1.4 – Performances des modèles génératifs (LLMs) pour le QA en arabe

Modèle	**F1-score**	**Précision**	**Sim. Cosinus**	**QA Multi-hop**	**Remarques**
GPT-4o	0.90 (micro)	0.82	–	94 % (réf.)	F1 Diacritisation : 0.91, Morphologie : 0.98 [16, 22]
GPT-4-turbo	–	–	–	80 % (SEDQ)	Utilisé avec embeddings E5 [16]
SILMA-9B	≈ 0.87	–	**0.95**	–	Très bonne cohérence sémantique
Gemini-1.5 Flash	0.84 (micro)	0.72	–	–	Concurrent open-weight
Command-R-35B	–	–	–	66.8 (rappel 3-grammes)	Évalué sur XOR-TyDi Arabic [16]

Observation : GPT-4o et GPT-4-turbo offrent les meilleures performances pour les tâches de QA en arabe, tant pour les questions simples que complexes (multi-hop), notamment lorsqu'ils sont associés à des embeddings comme E5.

1.2.3 Conclusion

La combinaison du modèle **E5-Large** pour la recherche et des modèles **GPT-4o** ou **GPT-4-turbo** pour la génération constitue actuellement la solution la plus performante pour les systèmes RAG appliqués à la question-réponse en arabe. D'autres approches prometteuses incluent les **embeddings BGE** ainsi que les modèles **Swan-Large** et **SILMA-9B**, particulièrement efficaces pour capter la sémantique et gérer la diversité dialectale.

1.3 Travaux Connexes pour l'Arabe ou la Langue Cible

Plusieurs études ont exploré les modèles de génération augmentée par récupération (RAG) appliqués à l'arabe et à d'autres langues à faibles ressources. Ces travaux traitent des défis rencontrés à la fois dans les composants de récupération et de génération, en s'appuyant souvent sur les récents progrès en matière de modèles d'embedding et de stratégies d'entraînement efficaces.

Composant de récupération. Mahboub et al. [20] ont évalué la récupération sémantique pour l'arabe en utilisant des pipelines RAG, démontrant que les embeddings denses (tels que E5 et BGE) surpassent largement les méthodes classiques comme BM25, en particulier sur les benchmarks arabes de questions-réponses. Caspari et al. [8] ont également analysé les performances de modèles d'embedding dans les systèmes RAG, en soulignant que des modèles comme `E5-Large` offrent un bon rappel (Recall@k) et une MRR élevée dans des contextes arabophones. Le benchmark ArabicMTEB [15] classe en outre des modèles tels que Swan-Large et BGE sur une diversité de tâches de récupération en arabe, y compris les variantes dialectales.

Composant de génération. Des études récentes ont exploité des modèles multilingues de grande taille pour la génération de réponses en arabe. GPT-4-turbo, évalué dans des contextes multilingues, a montré une grande précision dans des tâches de QA multihop en arabe lorsqu'il est couplé à une récupération basée sur des embeddings de phrases [23]. D'autres modèles comme GPT-4o et Gemini-1.5 Flash ont également été testés sur des tâches lexicales en arabe, atteignant des scores F1 supérieurs à 0,84 [3].

Efficacité. Des techniques de fine-tuning comme LoRA ont été appliquées à des modèles tels que Swan-Large afin d'améliorer l'efficacité des tâches de récupération en arabe [20]. Toutefois, aucune étude actuelle n'a encore mis en œuvre QLoRA ou TinyML dans un pipeline RAG pour la langue arabe.

Malgré ces avancées, des lacunes subsistent concernant l'évaluation standardisée, la disponibilité des ressources, et la prise en charge de l'arabe dialectal. Des recherches supplémentaires sont nécessaires pour évaluer les composants RAG dans différents sous-domaines arabes, et pour explorer des méthodes de déploiement légères adaptées à des environnements contraints.

1.3.1 LoRA et méthodes alternatives

Dans les environnements à faibles ressources, la **LoRA (Low-Rank Adaptation)** s'est imposée comme une méthode efficace pour le fine-tuning de grands modèles de langage, en réduisant considérablement le nombre de paramètres à entraîner tout en conservant de bonnes performances [14]. Toutefois, d'autres approches alternatives ou complémentaires ont été proposées, telles que **BitFit**, qui limite le fine-tuning aux biais des couches du modèle, ne modifiant ainsi que moins de 0,01 % des paramètres [26], ou encore les **modules d'adaptation (adapters)**, qui insèrent de petites couches entraînables entre les couches du transformeur d'origine, permettant une spécialisation rapide sans modification complète du modèle [25]. Ces méthodes partagent un objectif commun : adapter efficacement les LLMs tout en minimisant les besoins en ressources computationnelles et en mémoire.

1.4 Stratégies de Recherche d'Information dans un Contexte à Faibles Ressources

Dans les pipelines de génération augmentée par la recherche (RAG) appliqués à l'arabe, la qualité du module de recherche est essentielle, notamment en raison de la complexité morphologique, des diacritiques et de la variation dialectale de la langue. D'après les études récentes, la **recherche dense basée sur des embeddings sémantiques de phrases** surpasse systématiquement les stratégies sparse ou hybrides pour la compréhension de texte et le question answering en arabe.

1.4.1 Recherche Dense (Embeddings Sémantiques)

Les méthodes de recherche dense projettent à la fois la requête et les documents candidats dans un espace vectoriel sémantique continu à l'aide de modèles d'embedding basés sur des transformers. Le score de similarité est ensuite calculé via produit scalaire ou similarité cosinus.

Parmi les meilleurs modèles, E5-large obtient les résultats les plus élevés sur plusieurs benchmarks. Il atteint un rappel@5 de 0.88 et un MRR de 0.48 sur le jeu Riyadh Dictionary [3]. Sur le dataset ARCD, il enregistre un MRR de 0.79 et un rappel@1 de 0.686.

Lorsqu'il est testé sur des requêtes dialectales, E5-large atteint un rappel@5 de 0.994 et un MRR de 0.891.

Le modèle Big General Embeddings (BGE) arrive en deuxième position. Il obtient un rappel@5 de 0.80 et un MRR de 0.42 sur Riyadh Dictionary. Sur ARCD, le MRR est de 0.748 et le rappel@5 de 0.909. Sur le benchmark XOR-TyDi, il atteint un rappel 3-grammes caractère de 66.8 [8].

Le modèle Swan-Large, entraîné spécifiquement sur des données arabes, réalise un nDCG@10 de 65.63 sur ArabicMTEB et 77.03 sur Dialectal ArabicMTEB [15]. Le modèle Paraphrase-MPNet-Multilingual démontre aussi de bonnes performances avec un nDCG@3 de 0.879, un MRR@3 de 0.911 et un mAP@3 de 0.888.

Conclusion : La recherche dense est la stratégie la plus efficace pour l'arabe, grâce à sa robustesse face aux variations morphologiques, inflexions et dialectes.

1.4.2 Recherche Sparse (Basée sur les mots-clés)

Les méthodes sparse comme TF-IDF ou BM25 reposent sur la correspondance exacte entre les mots-clés de la requête et ceux des documents. Bien qu'elles soient efficaces et explicables, leurs performances sur les langues morphologiquement riches comme l'arabe sont limitées.

Par exemple, AraBERT v2 n'a atteint qu'un rappel@5 de 0.11 et un MRR de 0.07, tandis que CAMeLBERT a obtenu un rappel@5 de 0.16 et un MRR de 0.06 sur les corpus arabes.

Conclusion : La recherche sparse est insuffisante pour les systèmes RAG arabes, car elle ne gère pas efficacement la complexité linguistique.

1.4.3 Recherche Hybride

Les approches hybrides visent à combiner les avantages des méthodes dense et sparse pour améliorer la robustesse et la pertinence. Le modèle BGE-m3 agit à la fois comme retriever et reranker, améliorant le classement final. GRITLM, un modèle tout-en-un fusionnant recherche et génération, propose une inférence rapide tout en conservant des performances compétitives [8]. Certains pipelines hybrides multilingues intègrent aussi une étape de traduction via NLLB-600M pour permettre la recherche arabe à partir de requêtes anglaises [23].

Conclusion : Les systèmes hybrides offrent plus de flexibilité architecturale et de gains en pertinence, bien que les modèles denses restent les meilleurs retrieveurs autonomes pour l'arabe.

TABLE 1.5 – Performances des modèles de recherche sur des benchmarks arabes

Modèle	**Jeu de données**	**Rappel / MRR / nDCG**	**Type**	**Référence**
E5-large	Riyadh Dictionary	Recall@5 : 0.88, MRR : 0.48	Dense	[3]
E5-large	ARCD	Recall@1 : 0.686, MRR : 0.79	Dense	[3]
E5-large	QA dialectal	Recall@5 : 0.994, MRR : 0.891	Dense	[3]
BGE	Riyadh Dictionary	Recall@5 : 0.80, MRR : 0.42	Dense	[8]
BGE	ARCD	Recall@5 : 0.909, MRR : 0.748	Dense	[8]
BGE	XOR-TyDi	Rappel 3-grammes caractère : 66.8	Dense	[8]
Swan-Large	ArabicMTEB	nDCG@10 : 65.63	Dense	[15]
Swan-Large	Dialectal MTEB	nDCG@10 : 77.03	Dense	[15]
Paraphrase-MPNet	–	nDCG@3 : 0.879, MRR@3 : 0.911	Dense	–
AraBERT v2	–	Recall@5 : 0.11, MRR : 0.07	Sparse	–
CAMeLBERT	–	Recall@5 : 0.16, MRR : 0.06	Sparse	–
BGE-m3	–	–	Hybride	[8]
GRITLM	–	–	Hybride	[8]
NLLB-600M + E5	–	–	Hybride	[23]

1.4.4 Analyse Finale

Le tableau 1.6 résume les principales métriques des stratégies de recherche en arabe dans les systèmes RAG.

Conclusion : Les méthodes de recherche dense utilisant des embeddings de phrases (notamment avec les modèles E5 et BGE) offrent les meilleures performances pour le RAG en arabe. Elles assurent une forte capacité de rappel, une robustesse face aux dialectes, et une bonne correspondance sémantique.

TABLE 1.6 – Résumé des performances des stratégies de recherche en arabe

Modèle	**Recall@5**	**MRR**	**nDCG@10**	**Type**
E5-Large	0.88	0.48	–	Dense
BGE	0.80	0.42	–	Dense
Swan-Large	–	–	77.03	Dense
AraBERT v2	0.11	0.07	–	Sparse
CAMeLBERT	0.16	0.06	–	Sparse
GRITLM	–	–	–	Hybride

1.5 Modèles d'Embeddings pour la Recherche Sémantique en Arabe

La qualité de la recherche dans un système RAG dépend fortement du modèle d'embedding utilisé lors de la phase de récupération. Dans des langues à faibles ressources comme l'arabe, les modèles préentraînés rencontrent souvent des limites liées à la couverture dialectale ou au manque de données.

El-Beltagy et Abdallah [13] ont évalué un large éventail de modèles d'embeddings sur deux jeux de données QA arabes : *ARCD* et *Ar_EduText*. Leur étude a comparé 12 modèles, notamment :

— **E5-Large** (multilingue),
— **BGE-Large**,
— **OpenAI Ada**,
— **AraBERT**,
— **JAIS embeddings**.

Les résultats expérimentaux montrent que **E5-Large** et **BGE-Large** atteignent les meilleurs scores en *Recall@k* et *MRR*, surpassant même des modèles spécifiquement entraînés pour l'arabe. En particulier, **E5-Large** a obtenu un *Recall@5* de 0.994 sur Ar_EduText, et a démontré une forte robustesse aux dialectes comme l'égyptien.

Ces résultats suggèrent que certains modèles multilingues récents, entraînés avec des objectifs d'alignement sémantique, peuvent surpasser les modèles spécialisés dans les tâches de recherche sémantique en arabe. Cela confirme que l'efficacité des systèmes RAG en arabe dépend principalement du choix du meilleur encodeur sémantique.

1.6 Comment le Reranking est-il Intégré dans les Pipelines RAG pour l'Arabe ?

Le reranking joue un rôle central dans l'amélioration des pipelines RAG en arabe, notamment à cause de la complexité morphologique et de la diversité dialectale de la langue. Cette étape permet d'affiner la liste initiale des documents récupérés avant de les transmettre au générateur, ce qui améliore la précision et la pertinence contextuelle des réponses produites.

1.6.1 Architecture de Récupération en Deux Étapes

Les systèmes RAG modernes en arabe adoptent généralement une architecture de récupération à deux étapes :

- **Première Étape (Bi-Encodeur) :**
  - Les requêtes et documents sont encodés séparément sous forme d'embeddings denses.
  - Une recherche approximative par voisinage (ex. : FAISS) basée sur la similarité cosinus extrait les k documents les plus pertinents.
  - Les modèles **E5-large** et **BGE** donnent d'excellents résultats :
    - **E5-large :** Recall@5 = 0.88, MRR = 0.48.
    - **BGE :** Recall@5 = 0.80, MRR = 0.42 [20].
- **Deuxième Étape (Cross-Encodeur pour Reranking) :**
  - Les documents récupérés sont combinés avec la requête et encodés ensemble.
  - Ce processus est plus coûteux, mais permet une meilleure correspondance sémantique.
  - Les documents sont alors réordonnés pour une génération plus pertinente.

1.6.2 Modèles et Méthodes de Reranking en Arabe

Divers modèles ont été évalués pour leur efficacité en reranking :

- **BGE-m3 :** Un modèle multilingue utilisé à la fois comme récupérateur et reranker, performant en arabe et en scénarios multilingues [20].
- **GRITLM :** Un modèle unifié pour la récupération et la génération, qui utilise l'auto-raffinement génératif pour améliorer la cohérence [8].
- **Swan Models : Swan-Large** et **Swan-Small** ont obtenu respectivement **69.42** et **68.43** sur le benchmark ArabicMTEB pour les tâches de reranking [15].

1.6.3 Reranking Implicite par les Modèles Génératifs

Dans certains cas, après récupération des k meilleurs documents, le modèle génératif lui-même agit comme reranker implicite en sélectionnant les passages les plus pertinents à intégrer dans la réponse finale. Cela est utile dans des pipelines plus légers, où l'utilisation de cross-encodeurs est trop coûteuse.

1.6.4 Résultats Empiriques

Des études récentes montrent l'importance du reranking dans les tâches de QA en arabe :

- Une évaluation a révélé que l'encodeur #**3** (paraphrase-multilingual-mpnet-base-v2) obtenait les meilleures métriques avec **nDCG@3 = 0.879**, **MRR@3 = 0.911** et **mAP@3 = 0.888**.
- Ce modèle a également conduit aux meilleures performances finales dans un pipeline RAG (Top-1 Accuracy = 63.84%) avec GPT-3.5-turbo [20].

Conclusion

Le reranking est une amélioration essentielle dans les systèmes RAG arabes. Il fait le lien entre la récupération rapide à grande échelle et la génération précise et contextualisée. L'utilisation de rerankers multilingues performants (comme BGE-m3 ou MPNet), ou de modèles unifiés comme GRITLM, est indispensable pour atteindre des performances élevées en question-réponse en arabe.

1.7 Jeux de données utilisés dans les recherches RAG en arabe

Plusieurs jeux de données de questions-réponses (QA) et de recherche documentaire en arabe ont été utilisés pour évaluer les performances des systèmes de génération augmentée par récupération (RAG), en particulier dans les contextes à faibles ressources :

- **Ar_EduText** : Un jeu de données compilé manuellement à partir de textes éducatifs arabes, utilisé pour évaluer les performances de la récupération sémantique et de la génération dans les pipelines RAG arabes [5, 20].
- **ARCD (Arabic Reading Comprehension Dataset)** : Un benchmark populaire inspiré de SQuAD, utilisé pour évaluer les modèles d'embeddings et les systèmes RAG de bout en bout [5, 20].
- **Benchmark ArabicMTEB** : Une suite d'évaluation multilingue pour les embeddings de phrases à travers plusieurs tâches (recherche, clustering, classification), incluant des tâches spécifiques à l'arabe avec des métriques comme nDCG@10 [8, 15].

— **XOR-TyDi QA** : Un jeu de données multilingue couvrant des langues typologiquement diverses, dont l'arabe, utilisé pour tester l'efficacité multilingue des modèles RAG comme GPT-4-turbo et Command-R-35B [16, 23].
— **Dictionnaire de Riyad (88K entrées)** : Utilisé dans [3] pour tester les stratégies de récupération et les embeddings (E5-large, BGE) dans des tâches de récupération lexicale en arabe.

1.8 Existe-t-il des études utilisant LoRA, QLoRA ou TinyML avec RAG pour l'efficacité ?

Oui, certaines études mentionnent explicitement l'utilisation de **LoRA (Low-Rank Adaptation)** pour le fine-tuning efficient des modèles de langage (LLM) dans des systèmes RAG, notamment pour des applications en arabe ou multilingues. Toutefois, il n'y a **aucune mention directe** de **QLoRA** ou **TinyML** dans les travaux évalués pour améliorer l'efficacité des pipelines RAG.

LoRA avec Swan-Large. Le modèle *Swan-Large*, centré sur l'arabe et basé sur ArMistral-7B, utilise LoRA pour un fine-tuning efficace avec un batch size de 128, un taux d'apprentissage de 5e−6 sur trois époques, avec sept exemples négatifs durs. Il offre un coût réduit, traitant 10 000 documents pour seulement 0,75$ contre 9,88$ pour les modèles OpenAI équivalents [20].

LoRA pour les embeddings textuels. Une autre approche applique LoRA à des LLMs auto-régressifs (Mistral-7B) entraînés avec une loss contrastive sur des données synthétiques générées par des LLMs propriétaires. LoRA (rank 16) est utilisé sur une seule époque, simplifiant considérablement l'entraînement des embeddings [8].

LoRA pour l'efficacité mémoire et temps d'entraînement. Le même pipeline utilise aussi le *gradient checkpointing*, l'entraînement en précision mixte, et *DeepSpeed ZeRO-3* combinés à LoRA pour réduire l'usage mémoire GPU et accélérer l'apprentissage [8].

Adaptation paramètre-efficace en contexte multilingue. Bien que non liée explicitement à RAG, l'utilisation de stratégies efficaces comme les *adapter layers* ou le gel partiel du modèle est encouragée dans les tâches multilingues, en accord avec l'approche de LoRA [23].

Résumé. LoRA est centrale dans les efforts récents pour rendre les modèles RAG arabes plus efficaces, mais QLoRA et TinyML restent absents des recherches actuelles.

1.9 Principales limitations identifiées dans les recherches RAG récentes

Les recherches récentes sur les systèmes RAG, en particulier en arabe ou dans des contextes multilingues, ont mis en évidence plusieurs limitations majeures en termes de performance, scalabilité et généralisation :

- **Qualité des embeddings et biais sémantique** : Les performances des pipelines RAG dépendent fortement des modèles d'embeddings sous-jacents. Les modèles arabes classiques comme AraBERT ou CAMeLBERT sont dépassés par des modèles modernes comme **E5** et **BGE**, qui capturent mieux les nuances sémantiques [20].
- **Coût computationnel élevé** : L'architecture à deux étapes (recherche + génération) génère une latence et une consommation mémoire importantes, surtout avec de grands modèles génératifs comme GPT-4. LoRA permet de réduire les coûts [20], mais la demande GPU reste forte.
- **Absence de benchmarks arabes natifs** : Beaucoup de jeux de données d'évaluation sont des traductions depuis l'anglais et ne couvrent pas la richesse de l'arabe (dialectes, morphologie, diacritiques), ce qui limite la généralisation [5, 15].
- **Difficultés en raisonnement multi-hop** : Les tâches nécessitant l'agrégation de plusieurs documents (multi-hop QA) restent difficiles. Des techniques comme la décomposition de questions ou l'élargissement du contexte sont nécessaires, mais augmentent les coûts d'inférence [16, 23].
- **Peu d'exploration des méthodes de fine-tuning légères** : Hormis LoRA, les approches plus efficaces comme QLoRA ou les déploiements légers via TinyML n'ont pas encore été adoptées dans les études sur les RAG arabes [8].

Conclusion : Les limitations clés concernent la qualité des embeddings, les coûts computationnels, le manque de ressources d'évaluation natives, les défis en raisonnement multi-document et l'absence de techniques de fine-tuning ultra-légères.

1.10 Quelle est la définition et l'architecture de la génération augmentée par la recherche (RAG) ?

La génération augmentée par la recherche (RAG) est une architecture neuronale qui améliore les performances des modèles de langage en intégrant un mécanisme externe de récupération d'information dans le processus de génération. Plutôt que de s'appuyer uniquement sur les paramètres internes du modèle, RAG récupère dynamiquement des informations pertinentes à partir d'un corpus pré-indexé et les utilise comme contexte supplémentaire pour générer des réponses. Cette approche permet un meilleur ancrage

factuel, une plus grande pertinence, et une adaptabilité accrue aux connaissances spécifiques à un domaine [20].

Le pipeline RAG comprend généralement quatre étapes principales. Premièrement, une étape de **prétraitement et découpage des documents** divise le corpus source (par exemple : encyclopédies, documents juridiques ou jeux de données de questions-réponses) en segments cohérents tout en préservant la continuité sémantique. Deuxièmement, lors de la **phase de récupération**, ces segments sont transformés en représentations vectorielles denses à l'aide de modèles comme AraBERTv2 ou E5, puis stockés dans un index vectoriel (par exemple : FAISS). Lorsqu'une requête utilisateur est soumise, le système l'encode et récupère les documents les plus similaires sur le plan sémantique. Troisièmement, durant la **phase de génération**, un grand modèle de langage (tel que GPT-4 ou Mixtral) génère une réponse en se basant à la fois sur la requête et sur les documents récupérés. Enfin, un module facultatif d'**évaluation** peut être intégré pour vérifier la véracité des faits ou la fluidité, à l'aide d'un modèle secondaire [20].

Les avancées récentes, telles que GRITLM, vont encore plus loin en intégrant la récupération et la génération dans un cadre unifié, améliorant ainsi la vitesse d'inférence et l'efficacité mémoire. Dans le contexte du traitement automatique du langage arabe, où la rareté des données et la diversité dialectale représentent des défis majeurs, les modèles RAG se sont révélés particulièrement efficaces [20].

Architectures des modèles de langue arabes : modèles encodeur seul, décodeur seul et encodeur-décodeur

Le choix architectural des grands modèles de langue (LLMs) joue un rôle fondamental dans la conception des systèmes RAG, notamment pour les langues morphologiquement riches et peu dotées comme l'arabe. Les pipelines RAG sont généralement constitués de deux composants : un récupérateur, qui identifie les passages pertinents, et un générateur, qui produit des réponses contextualisées. Les architectures à encodeur seul, telles que AraBERT, AraELECTRA ou CAMeLBERT, sont particulièrement adaptées à l'étape de récupération grâce à leur mécanisme d'attention bidirectionnelle et leur performance élevée dans les tâches d'encodage de phrases. Ces modèles sont capables de capturer des structures syntaxiques et sémantiques complexes caractéristiques de l'arabe, y compris sa morphologie templatique, sa richesse flexionnelle et ses variations orthographiques. Des benchmarks récents, spécifiquement conçus pour la recherche sémantique en arabe, ont montré que les embeddings de phrases produits par des modèles encodeurs comme `E5-large` ou `BGE` surpassent systématiquement les modèles classiques basés sur les mots ainsi que les modèles multilingues, selon des métriques telles que le nDCG (normalized Discounted Cumulative Gain), le MRR (Mean Reciprocal Rank) et le mAP (Mean

Average Precision) [18]. Ces benchmarks ont été construits à partir de scénarios réels de support client, avec des scores de pertinence validés par GPT-4, fournissant ainsi un cadre d'évaluation robuste pour la recherche d'information en arabe.

En revanche, le composant de génération dans les systèmes RAG repose généralement sur des modèles à décodeur seul ou des modèles encodeur-décodeur. Les modèles décodeurs comme GPT-2, AceGPT ou GPT-4 génèrent des réponses arabes fluides en prédisant de manière autoregressive les tokens à partir du contexte précédent. Cependant, leur sensibilité à la granularité de la tokenisation, particulièrement marquée en arabe où les tokenizers sous-mot (par exemple BPE) augmentent considérablement le nombre de tokens effectifs, engendre un coût d'inférence plus élevé. Les modèles encodeur-décodeur comme AraT5 ou AraMUS offrent une alternative équilibrée en intégrant compréhension et génération dans une architecture séquence-à-séquence unifiée, ce qui les rend particulièrement efficaces pour des tâches complexes telles que les questions-réponses en arabe ou le résumé abstrait.

Ainsi, les systèmes RAG en arabe tirent avantage d'une stratégie architecturale hybride : l'usage de modèles à encodeur seul pour une récupération sémantique de haute qualité, et de modèles décodeurs pour une génération de texte précise et contextuelle. Cette répartition des rôles améliore non seulement la pertinence des réponses, mais réduit également les coûts computationnels dans des environnements à faibles ressources. Par ailleurs, le choix architectural doit tenir compte des limitations propres au TALN arabe, telles que l'absence de voyelles (diacritiques), la variation dialectale, et la rareté des corpus annotés, rendant le fine-tuning spécifique au domaine et l'évaluation sur benchmarks indispensables pour des applications RAG en production.

1.11 Le rôle des Transformers de Hugging Face dans une architecture RAG

Dans un système de génération augmentée par la recherche (RAG), les Transformers de Hugging Face jouent un rôle fondamental dans les étapes de recherche et de génération. Ces modèles sont largement adoptés en raison de leur accessibilité, de leur préentraînement sur de vastes corpus, et de leur intégration aisée dans l'écosystème Hugging Face.

Composant de recherche : L'étape de recherche repose fortement sur des modèles d'embedding permettant de représenter les questions et les documents dans un espace sémantique commun. Des modèles basés sur les Transformers issus de Hugging Face, comme **AraBERT** [7], sont couramment utilisés pour générer ces embeddings en arabe. AraBERT est spécifiquement entraîné sur un grand corpus arabe et démontre de bonnes performances sur des tâches aval comme la classification de phrases ou le question-réponse,

ce qui en fait un candidat idéal pour la recherche dense dans les pipelines RAG.

Composant de génération : Une fois les documents pertinents retrouvés, les modèles génératifs préentraînés de Hugging Face tels que GPT-2 ou T5 peuvent être mobilisés pour produire des réponses en tenant compte du contexte extrait. Ces modèles permettent de générer des réponses conditionnées à la fois par la requête initiale et par les documents les plus pertinents retrouvés.

En résumé, les Transformers de Hugging Face fournissent des modèles préentraînés robustes tant pour la phase de recherche (via les embeddings denses) que pour celle de génération (via les grands modèles de langage). Pour les applications en arabe, des modèles comme AraBERT constituent l'ossature de la recherche en raison de leur représentation linguistique adaptée à la langue.

Deuxième partie

Méthodologie et Architecture Proposée

Chapitre 2

Méthodologie

2.1 Vue d'ensemble du pipeline RAG

L'architecture de génération augmentée par la recherche (RAG) repose sur deux composantes fondamentales...

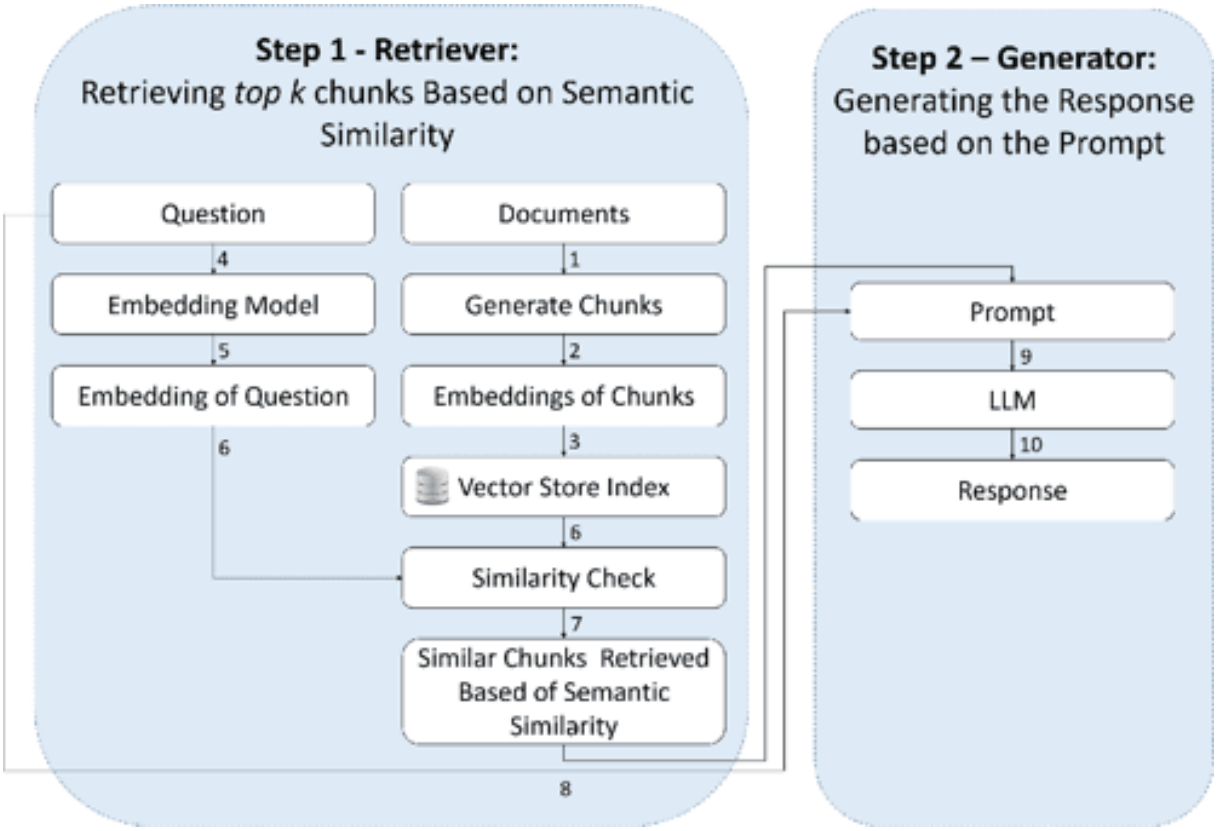

FIGURE 2.1 – Architecture du pipeline RAG.

le module de récupération (retriever) et le module de génération (generator). Le récupérateur a pour rôle d'identifier et d'extraire, à partir d'un corpus prédéfini, les contextes les plus pertinents en réponse à une requête donnée. Le générateur, quant à lui, produit une réponse conditionnée à la fois par le contexte récupéré et par la question initiale.

Cette combinaison permet au système de générer des réponses plus précises et adaptées au contexte, en intégrant dynamiquement des connaissances externes, plutôt qu'en se limitant aux poids appris lors du pré-entraînement.

2.2 Données et Prétraitement

2.2.1 Données

Quatre jeux de données arabes ont été utilisés pour entraîner et évaluer des modèles génératifs dans un cadre de questions à choix multiples. Chaque jeu de données est structuré en paires question-réponse, parfois accompagnées d'un contexte :

- `arabicmmlu.csv` : Questions inspirées d'examens universitaires couvrant plusieurs domaines.
- `alghafa.csv` : Jeu de données académique centré sur la culture et la langue arabes.
- `madinahqa.csv` : Questions-réponses basées sur du contenu éducatif provenant d'écoles de Médine.
- `aratrust.csv` : Jeu de données axé sur la compréhension écrite en arabe.

2.2.2 Prétraitement

Afin d'assurer une quantité suffisante de données pour l'affinage des modèles, nous avons fusionné les quatre jeux de données arabes de type QCM : ArabicMMLU, AlGhafa, MadinahQA et AraTrust. Chaque jeu de données a été prétraité pour supprimer les entrées incomplètes ou nulles (valeurs NaN), puis converti dans un format génératif unifié.

Plus précisément, chaque exemple a été reformulé sous la forme d'un champ texte unique suivant la structure : "[Question] : [Réponse] :".

Cette transformation permet de standardiser les données afin de les rendre compatibles avec les modèles de langue génératifs. Les exemples nettoyés ont ensuite été regroupés dans un seul fichier JSONL, garantissant un corpus large, cohérent et de haute qualité, adapté à l'affinage de modèles de langue pour l'arabe.

2.3 Encodage Dense et Récupération de Contexte

2.3.1 Construction de l'Index FAISS et Modèles d'Embeddings

Pour implémenter la composante de recherche dense du pipeline RAG, nous avons construit un index FAISS en utilisant des embeddings de haute qualité, à la fois multilingues et spécifiques à l'arabe. Deux modèles ont été employés :

`bert-base-arabic-camelbert-mix`, optimisé pour les nuances linguistiques de l'arabe, et `paraphrase-multilingual`
`MiniLM-L12-v2`, qui prend en charge la similarité sémantique multilingue.

Chaque document du corpus—composé de paires concaténées de question et de contexte était transformé en un vecteur dense de dimension 768, puis stocké dans un index FAISS via le framework LangChain. Cet index permet une récupération rapide et sémantiquement pertinente lors de l'inférence.

2.3.2 Récupération Dense via FAISS

La récupération dense s'appuie sur FAISS (Facebook AI Similarity Search) pour indexer et rechercher des représentations vectorielles de documents dans un espace de haute dimension. Les embeddings contextuels ont été générés à l'aide de modèles tels que : `paraphrase-multilingual-MiniLM-L12-v2`. Chaque paire question-contexte était encodée dans un vecteur dense de dimension 768, puis indexée avec FAISS.

Lors de la récupération, la requête est encodée dans le même espace d'embedding, et FAISS permet de retrouver efficacement les *top-k* vecteurs les plus similaires sémantiquement. Cette méthode permet de capturer des relations profondes de contexte et de sens, au-delà de la simple correspondance lexicale.

2.3.3 Stratégie Hybride de Récupération avec Re-Ranking

Afin d'améliorer la qualité des documents récupérés, une stratégie hybride a été adoptée, combinant des méthodes de recherche dense et parcimonieuse. La recherche dense repose sur FAISS, avec des embeddings extraits des modèles `bert-base-arabic-camelbert-mix` et `sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2`. La recherche parcimonieuse s'appuie sur l'algorithme BM25, qui évalue les documents selon des correspondances lexicales.

Les scores de récupération fournis par FAISS et BM25 ont été combinés selon une pondération pour équilibrer pertinence sémantique et lexicale. Un module de *re-ranking* a ensuite été utilisé : un cross-encoder rescorage les meilleurs candidats en analysant

finement les interactions entre la requête et chaque passage. Cette architecture multi-étapes assure une sélection de documents contextuels à la fois précise et pertinente pour la génération de réponses.

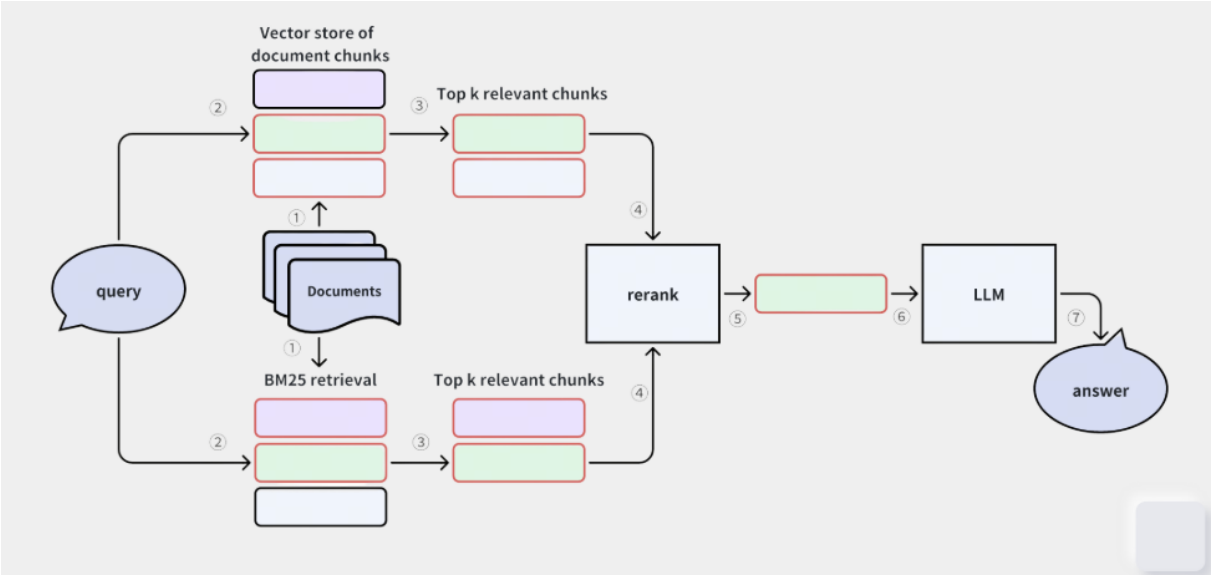

FIGURE 2.2 – Architecture du pipeline de récupération hybride avec re-ranking.

2.4 Architecture Causale (Decoder-Only) dans un Système RAG Arabe

Dans ce travail, nous utilisons un modèle de langage causal (de type *decoder-only*) comme composante générative principale de notre système de génération augmentée par la recherche (RAG) en arabe. Contrairement aux architectures encodeur-décodeur, qui encodent l'entrée complète avant de générer une sortie, les modèles causaux produisent les réponses de manière autoregressive, un token à la fois, de gauche à droite, en se basant uniquement sur les tokens précédents. Cette conception s'aligne naturellement avec le paradigme RAG, dans lequel le modèle reçoit une invite composée de la question de l'utilisateur et du contexte récupéré, puis génère une réponse cohérente et fondée.

Nous adoptons cette architecture pour sa simplicité, son efficacité, et sa compatibilité avec les approches par prompts ou par fine-tuning. Plus précisément, nous expérimentons avec trois modèles causaux en arabe ou adaptés à l'arabe :

— `akhooli/gpt2-small-arabic` : un modèle GPT-2 léger adapté à l'arabe.

— `aubmindlab/aragpt2-medium` : un modèle de taille moyenne spécialisé dans le question-réponse en arabe.

— `EleutherAI/gpt-neo-1.3B` : notre modèle principal, que nous avons fine-tuné avec QLoRA pour la génération de QA en arabe.

Ces modèles ont été sélectionnés en fonction de leur disponibilité, de leur stabilité à l'entraînement, et de leur capacité à générer un texte arabe fluide. Les modèles causaux permettent également de réduire la complexité architecturale par rapport aux modèles encodeur-décodeur, qui nécessitent des modules séparés et plus de mémoire en inférence.

Par ailleurs, la tokenisation en arabe engendre souvent un grand nombre de tokens à

cause de la segmentation morphologique. Les modèles causaux offrent un meilleur contrôle sur la taille des prompts et la gestion du contexte. En utilisant uniquement des architectures causales, nous simplifions le processus de génération dans le cadre du RAG arabe, assurant un équilibre optimal entre précision, coût et faisabilité du déploiement.

2.5 RAG Zero-Shot avec Récupération Dense

Dans nos premières expériences, nous avons mis en œuvre une configuration RAG en mode *zero-shot*, c'est-à-dire sans aucun fine-tuning. Le modèle génératif utilisé était `gpt2-small-arabic`, préentraîné sur des données arabes. Pour la récupération dense, un index FAISS a été construit à partir du corpus, en utilisant des embeddings contextuels générés par le modèle multilingue `paraphrase-multilingual-MiniLM-L12-v2`.

À partir d'une requête utilisateur, le passage le plus pertinent a été sélectionné selon la similarité cosinus dans l'espace vectoriel. Le prompt de génération était soigneusement structuré pour inciter le modèle à répondre de manière précise uniquement à partir de l'information récupérée. Le format du prompt était le suivant :

```
    # 4. Prompt engineering amélioré
    prompt = f"""أجب بإيجاز وبدقة على السؤال التالي باستخدام المعلومات المقدمة فقط.

### السؤال:
{query}

### المعلومات:
{best_context}

### الإجابة:"""
```

Cette configuration *zero-shot* nous a permis d'évaluer les performances de base de l'architecture RAG en mode récupération dense, sans supervision spécifique à la tâche ni mise à jour des paramètres.

2.6 Pipeline RAG (Retrieval-Augmented Generation)

L'architecture de génération repose sur des composants modulaires :

— **Retriever** : récupérateur hybride (BM25 + FAISS avec embeddings CAMeL)

— **Générateur** : modèles `les` fine-tuné, intégré via `HuggingFacePipeline`

— **Framework** : `RetrievalQA` du framework LangChain

Une variante du pipeline inclut un reranking à l'aide d'un Cross-Encoder : (`ms-marco-MiniLM-L-6-v2`) avant de transmettre le contexte au générateur. Cela permet de reclasser les passages récupérés selon une évaluation plus fine de la pertinence entre la requête et chaque document.

2.7 Ajustement Fin (Fine-Tuning) du Générateur

2.7.1 Entraînement avec QLoRA sur Google Colab (GPU T4)

Dans cette étude, nous avons adopté l'approche QLoRA (Quantized Low-Rank Adaptation) afin d'ajuster efficacement un modèle de langage de grande taille (GPT-Neo 1.3B) dans un environnement à faibles ressources. L'entraînement a été réalisé sur Google Colab à l'aide d'un GPU NVIDIA Tesla T4 avec 16 Go de mémoire VRAM, compatible avec les contraintes de quantification en 4 bits. Le jeu de données a été divisé selon un ratio de 90/10 entre apprentissage et test, afin d'assurer un équilibre entre la généralisation du modèle et la granularité de l'évaluation, tout en maintenant un volume d'apprentissage suffisant dans un contexte de faible ressource. Pour garantir la reproductibilité, une graine aléatoire fixe (42) a été utilisée pour le découpage des données et l'entraînement du modèle.

Nous avons utilisé l'optimiseur AdamW combiné à une planification linéaire du taux d'apprentissage, avec un échauffement (warmup) de 10 %. Un critère d'arrêt anticipé (early stopping) a été appliqué basé sur la perte de validation (`eval_loss`), avec une patience de trois étapes sans amélioration. Les hyperparamètres LoRA étaient fixés à $r = 8$, $\alpha = 16$ et un taux de dropout de 0,1.

Cette configuration méthodologique permet un entraînement stable et reproductible, adapté aux contraintes matérielles, tout en maximisant la qualité des réponses générées dans un contexte linguistique à faibles ressources comme l'arabe.

2.7.2 Configuration des Hyperparamètres

Le tableau 2.1 résume les principaux hyperparamètres utilisés lors de l'ajustement fin des différents modèles. Pour assurer cohérence et reproductibilité, cette configuration a été appliquée de manière uniforme sauf mention contraire.

TABLE 2.1 – Hyperparamètres de Fine-Tuning

Paramètre	**Valeur**
Découpage train/test	90/10 (aléatoire, stratifié)
Graine aléatoire	42
Taille du batch	4
Accumulation de gradients	2
Longueur maximale de séquence	512
Nombre d'étapes	1000 (GPT-Neo), 2000 (BLOOMZ), 4000 (GPT2-small-arabic)
Taux d'apprentissage	2×10^{-4}
Optimiseur	AdamW
Précision	FP16
Scheduler de LR	Décroissance linéaire avec échauffement (10 %)
Dropout	0,1
Rang LoRA r	8
LoRA α	16
Early stopping	Désactivé (évaluation manuelle)
Fréquence d'évaluation	Toutes les 100 étapes
Sauvegarde de checkpoints	Toutes les 100 étapes

2.7.3 Préparation et Chargement des Données

Plus de 16 000 paires question–réponse en arabe ont été compilées à partir de sources multiples, puis reformattées selon une structure générative : ‘‘`Question : ... \\Réponse : ...`’’. Les exemples ont été enregistrés au format `JSONL` afin d'assurer leur compatibilité avec les frameworks NLP modernes. Pour simplifier le chargement et le prétraitement, la bibliothèque `datasets` de Hugging Face a été utilisée. Le jeu de données a été directement chargé en mémoire, puis automatiquement scindé en ensembles d'entraînement et de test selon le ratio standard 90 % / 10 %. Cette configuration a permis une gestion efficace des données et une intégration fluide avec les pipelines d'entraînement basés sur Transformers.

2.7.4 Présentation des Modèles de Langage Ajustés

Le tableau **??** présente les principaux modèles génératifs utilisés dans nos expériences, incluant leur taille, leur statut open source, et la bibliothèque d'origine.

Tous ces modèles ont été entraînés en mode autoregressif (CLM) sur des paires question–réponse dans un format génératif. L'objectif était d'adapter ces modèles à la langue arabe et à la tâche spécifique de questions à choix multiples, en utilisant des techniques de fine-tuning légères.

Détails du fine-tuning

2.7.5 Configuration de l'adaptation : GPT-Neo 1.3B avec QLoRA

L'adaptation du modèle `EleutherAI/gpt-neo-1.3B` a été réalisée à l'aide de la technique QLoRA, qui permet un entraînement efficace même sous contraintes matérielles, grâce à la quantification en 4 bits. La configuration de la quantification utilise le module `BitsAndBytesConfig`, avec double quantification, le type `nf4` et des calculs en `torch.float16` pour optimiser la mémoire. Le modèle quantifié est chargé via `AutoModelForCausalLM` avec une répartition automatique sur les dispositifs disponibles.

Pour un fine-tuning paramètre-efficace, nous avons utilisé la fonction : `prepare_model_for_kbit_training`, suivie d'une configuration LoRA ciblant les modules `q_proj` et `v_proj` avec un rang $r = 8$, `lora_alpha=16`, et un taux de dropout de 0,1. La tokenisation est assurée par `AutoTokenizer`, avec une longueur maximale fixée à 512 tokens, et activation du padding et du troncature pour un formatage homogène des entrées. Le jeu de données, préalablement converti au format JSONL, est chargé via la bibliothèque `datasets` de Hugging Face et tokenisé via une fonction de mappage sur le champ `text`.

L'entraînement est géré par le `SFTTrainer` de la bibliothèque TRL, avec les paramètres suivants : taux d'apprentissage de 2×10^{-4}, taille de lot de 4, accumulation de gradients sur 2 étapes, évaluation et sauvegarde toutes les 100 étapes. L'optimiseur utilisé est AdamW, et l'entraînement s'effectue en précision mixte FP16. Un total de 1000 étapes a été exécuté, pour une durée d'environ 99 minutes et 34 secondes sur un GPU Google Colab. Cette configuration a permis une adaptation efficace du modèle GPT-Neo tout en respectant les contraintes de mémoire d'un environnement grand public.

2.7.6 Fine-tuning du modèle `bloomz-3b`

Le modèle `bigscience/bloomz-3b` a été adapté à l'aide de la méthode **QLoRA**, qui autorise une adaptation à faibles ressources via une quantification en 4 bits. Ce paramétrage permet un entraînement sur des environnements modestes comme Google Colab sans dépasser les limites de mémoire GPU.

Le tokenizer est d'abord chargé avec `AutoTokenizer` ; comme le modèle ne possède pas de jeton de padding par défaut, nous avons utilisé le jeton de fin de séquence comme substitut. La quantification est gérée via `BitsAndBytesConfig` avec double quantification, type `nf4` et calculs en `torch.float16`.

Le modèle est chargé avec
`AutoModelForCausalLM` et préparé avec `prepare_model_for_kbit_training`. Les modules ciblés pour LoRA sont `query_key_value`, compatibles avec l'architecture BLOOM. Les paramètres LoRA sont : rang de 8, alpha de 16, et dropout de 0.1.

Le jeu de données est tokenisé avec padding et troncature à 512 tokens, puis prétraité via la fonction `datasets.map` pour générer les jeux d'entraînement et de test.

L'entraînement est géré avec `TrainingArguments` de la bibliothèque `transformers`, pour 2000 étapes, un batch size de 4, une accumulation de gradients sur 2 étapes, un taux d'apprentissage de `2e-4` et un entraînement en précision mixte (`fp16=True`). L'évaluation et la sauvegarde sont effectuées toutes les 100 étapes.

L'entraînement supervisé est assuré par `SFTTrainer` de la bibliothèque TRL, qui intègre la configuration LoRA et gère le processus complet sur le dataset arabe tokenisé.

2.7.7 Fine-tuning du modèle `aragpt2-medium`

Le modèle `aubmindlab/aragpt2-medium`, basé sur l'architecture GPT-2, a été adapté via la technique **QLoRA**, optimisée pour un usage en environnement Google Colab. Le tokenizer a été configuré pour utiliser le jeton de fin de séquence comme jeton de padding afin d'éviter les erreurs spécifiques aux modèles GPT-2.

La quantification en 4 bits utilise le type `nf4` avec double quantification et calculs en `float16`. Le modèle est chargé via `AutoModelForCausalLM`, puis préparé avec `prepare_model_for_kbit_training`.

Le fine-tuning LoRA cible le module `c_attn` (spécifique à GPT-2), avec rang 8, alpha 16, et dropout 0.1. L'intégration est réalisée via `get_peft_model()`, avec vérification des paramètres entraînables via `print_trainable_parameters()`.

Les paires question-réponse sont formatées en un seul champ `text` via la fonction `format_qa`, selon la structure : “ `question:` ... `answer:` ...”. Le dataset est ensuite tokenisé avec troncature et padding à 512 tokens, et divisé avec `DatasetDict`.

Les arguments d'entraînement (`TrainingArguments`) incluent : taille de lot 4, 2 étapes d'accumulation, 1000 étapes max, précision mixte (`fp16=True`), évaluation/sauvegarde toutes les 100 étapes. L'optimiseur est `adamw_torch`.

L'entraînement est géré via `SFTTrainer` de TRL. Après exécution avec `trainer.train()`, la durée est mesurée : environ 99 minutes et 34 secondes.

2.7.8 Fine-tuning du modèle `gpt2-small-arabic`

Le modèle `akhooli/gpt2-small-arabic` a été adapté avec une configuration LoRA standard (sans quantification), car sa petite taille permet un entraînement direct en `float32` dans un environnement limité comme Google Colab.

Le tokenizer est configuré avec le jeton de fin de séquence comme jeton de padding, et le modèle est chargé via `AutoModelForCausalLM` en pleine précision avec `device_map="auto"`.

La configuration LoRA cible le module `c_attn` de GPT-2, avec rang 8, alpha 16 et dropout 0.1. Les couches LoRA sont ajoutées avec `get_peft_model()`, et les paramètres entraînables sont vérifiés.

Le dataset est formatté via une fonction personnalisée en suivant : “ `question:` ... `reponse:` ...”. Le tout est tokenisé avec padding/troncature à 512 tokens.

Les arguments d'entraînement définissent 4000 étapes, batch size 4, 2 étapes d'accumulation, entraînement en `fp16`, avec évaluation et sauvegarde toutes les 100 étapes. L'entraînement est chronométré via le module `time`, et la durée affichée à la fin.

Cette configuration représente une base efficace pour évaluer les performances génératives arabes en conditions à faibles ressources.

2.8 Métriques d'Évaluation

Pour évaluer les performances de notre pipeline de génération augmentée par récupération (RAG), nous avons utilisé un ensemble de métriques couvrant à la fois les composantes de récupération et de génération. Ces métriques offrent une évaluation complète de la précision factuelle, du chevauchement lexical, de la pertinence sémantique et de la qualité de la récupération.

TABLE 2.2 – Résumé des métriques d'évaluation

Métrique	**Type**	**Description**
Exact Match (EM)	Génération	Vérifie si la réponse générée correspond exactement à la réponse de référence.
F1 au niveau des mots	Génération	Mesure le chevauchement entre la réponse générée et la réponse de référence, en termes de précision et de rappel.
ROUGE-L	Génération	Évalue la plus longue sous-séquence commune entre le texte généré et le texte de référence.
BERTScore-F1	Génération	Utilise les embeddings BERT pour mesurer la similarité sémantique entre la réponse générée et la réponse correcte.
Recall@5	Récupération	Vérifie si le document pertinent se trouve parmi les 5 premiers documents récupérés.
MRR@5	Récupération	Moyenne des inverses des rangs du premier document pertinent parmi les 5 premiers résultats.
EM basé sur le contexte	Récupération	Exact Match calculé à partir de la réponse générée en tenant compte du contexte récupéré.
F1 basé sur le contexte	Récupération	Score F1 évalué sur la réponse produite en utilisant le contexte récupéré.

L'évaluation a été effectuée sur des sous-ensembles de 100, 500 et 900 paires question-réponse pour chaque configuration.

2.9 Infrastructure et Outils

2.9.1 Utilisation de NotebookLM pour la Revue de Littérature

Afin d'améliorer l'efficacité et la structuration de la revue de littérature, nous avons utilisé **NotebookLM**, un assistant de lecture alimenté par l'IA développé par Google et basé sur les modèles de langage Gemini. Cet outil a permis une exploration interactive et approfondie des articles scientifiques liés à la génération augmentée par récupération (RAG), en particulier pour les langues peu dotées comme l'arabe.

NotebookLM a facilité l'organisation et l'analyse des publications scientifiques en nous permettant de téléverser des documents aux formats PDF, HTML ou texte brut dans un carnet interactif et interrogeable. Il générait automatiquement des résumés structurés par sections — telles que l'introduction, la méthodologie et les résultats — ce qui a accéléré notre compréhension du contenu. De plus, il permettait des requêtes contextuelles sur l'ensemble du corpus, telles que : « Quels modèles de récupération offrent le meilleur Recall@k pour les tâches de QA en arabe ? »

Outre les métriques de performance telles que le F1-score, ROUGE, BLEU et MRR, NotebookLM a mis en évidence des défis récurrents dans la littérature, notamment la diversité dialectale, le manque de benchmarks arabes standardisés, et la dépendance excessive à des modèles propriétaires. Dans l'ensemble, l'intégration de NotebookLM a considérablement amélioré la cohérence, la profondeur et l'efficacité de notre processus d'analyse documentaire.

2.9.2 LangChain

Nous avons utilisé LangChain comme cadre d'orchestration pour construire et gérer le pipeline RAG. Il offre une interface modulaire permettant de chaîner les composants tels que les moteurs de récupération (BM25, FAISS), les modèles d'embedding, les gabarits de prompt et les grands modèles de langage. LangChain a facilité la mise en place rapide de stratégies de récupération hybrides et leur intégration avec des bases de vecteurs locales ou cloud.

2.9.3 Hugging Face Transformers

La bibliothèque Transformers de Hugging Face a été utilisée pour charger et affiner des modèles de langue arabes comme `gpt2-small-arabic` et `aragpt2-medium`. Elle nous a également permis d'accéder à des modèles de phrase multilingues (comme `paraphrase-multilingual-MiniLM`) pour la récupération dense et la génération d'embeddings. Sa compatibilité avec PyTorch et son intégration avec les hubs de modèles ont accéléré nos expérimentations.

2.9.4 Google Colab (GPU)

Google Colab a constitué notre environnement principal pour l'entraînement, le fine-tuning et l'évaluation des modèles. La plateforme offre un accès gratuit à des GPU puissants (Tesla T4 ou A100), essentiels pour le fine-tuning efficace via QLoRA et la récupération rapide avec FAISS. L'intégration avec Google Drive a facilité la gestion des checkpoints et des jeux de données.

2.9.5 LaTeX

Tous les rapports et résultats ont été rédigés avec LaTeX, un système de composition de documents adapté à la rédaction scientifique. Les tableaux, figures, résultats d'évaluation et descriptions méthodologiques ont été mis en forme pour garantir un rendu académique professionnel.

2.9.6 Ngrok et Streamlit

Nous avons développé une interface utilisateur interactive avec Streamlit, permettant des réponses en temps réel via notre système RAG. Pour rendre l'application accessible à distance, nous avons intégré Ngrok, qui génère un lien sécurisé temporaire. Cette configuration a permis aux testeurs et encadrants d'interagir avec notre assistant de QA en arabe depuis n'importe quel appareil.

2.9.7 Métriques Scikit-Learn

Les métriques classiques telles que la précision, le rappel, le F1-score et l'exactitude ont été calculées à l'aide du module `sklearn.metrics`. Ces indicateurs nous ont permis d'évaluer la qualité des modules de récupération et de génération, notamment lors de comparaisons entre modèles et approches.

2.9.8 Matplotlib

Matplotlib a été utilisé pour visualiser les scores de récupération, les métriques de génération, et comparer les performances entre les modèles. Des graphiques et histogrammes ont été générés pour illustrer les améliorations après fine-tuning ou reranking, facilitant l'analyse comparative des configurations RAG.

2.9.9 Google Drive et Gestion de Fichiers

Tous les jeux de données, sorties de modèles et visualisations ont été stockés dans Google Drive. Cela a assuré la persistance entre les sessions Colab et facilité l'organisation des checkpoints, résultats et journaux d'entraînement.

2.9.10 PyTorch

PyTorch a été le framework de base pour la mise en œuvre des modèles et l'accélération GPU. Il a permis les calculs tensoriels, l'optimisation par gradient, et l'entraînement en précision mixte via `torch.float16`, rendant le fine-tuning de grands modèles comme `gpt-neo-1.3B` plus efficace.

2.9.11 Datasets (Hugging Face)

La bibliothèque `datasets` de Hugging Face a été utilisée pour charger, diviser et prétraiter notre jeu de données QA en arabe. Plus de 16 000 paires question–réponse ont été transformées en format `JSONL` et réparties automatiquement avec un split 90% entraînement / 10% test. Cela a facilité leur intégration avec les tokenizers et les modèles.

2.9.12 TRL (Transformers Reinforcement Learning)

La bibliothèque `trl` a été utilisée pour instancier le `SFTTrainer`, facilitant le fine-tuning supervisé des modèles causaux. Elle permet une configuration simple des boucles d'entraînement et une intégration fluide avec les modèles Hugging Face, les optimisateurs et les paramètres personnalisés.

2.9.13 PEFT (Parameter-Efficient Fine-Tuning)

La bibliothèque `PEFT` a été utilisée pour rendre l'entraînement de grands modèles plus efficace en affinant uniquement un sous-ensemble de poids (adapters). Plus précisément, l'adaptation Low-Rank (LoRA) a été appliquée aux modules `q_proj` et `v_proj`, réduisant considérablement la mémoire et le coût computationnel sans sacrifier la performance.

2.9.14 BitsAndBytes (bnb)

La bibliothèque `bitsandbytes` nous a permis de quantifier les poids des modèles en 4 bits grâce au schéma de quantification `nf4` avec double quantification. Cela a rendu possible le chargement et le fine-tuning de modèles volumineux sur des environnements à ressources limitées comme Google Colab, sans dépasser les limites de mémoire.

Troisième partie

Résultats et Évaluation

Chapitre 3

Résultats

3.1 Résultats du Module de Recherche d'Informations (Retriever)

TABLE 3.1 – Évaluation du retriever FAISS (avec embeddings multilingues)

Métrique	**Score**
Recall@5	0.8600
MRR@5	0.7887
Exact Match (EM)	0.8600
Précision globale	0.8600
F1-mot à mot	0.6272

Les résultats du récupérateur dense FAISS montrent une performance solide, avec un **Recall@5 de 86%** et un **MRR@5 de 0,7887**, indiquant une bonne capacité à retrouver rapidement des documents pertinents. Toutefois, le score de **F1 mot-à-mot (0,6272)** souligne une perte de précision lexicale, ce qui suggère que les réponses générées capturent bien le sens global, mais manquent parfois de correspondance exacte avec les termes attendus. Ce constat motive l'intégration d'une stratégie hybride pour combiner précision lexicale et compréhension sémantique.

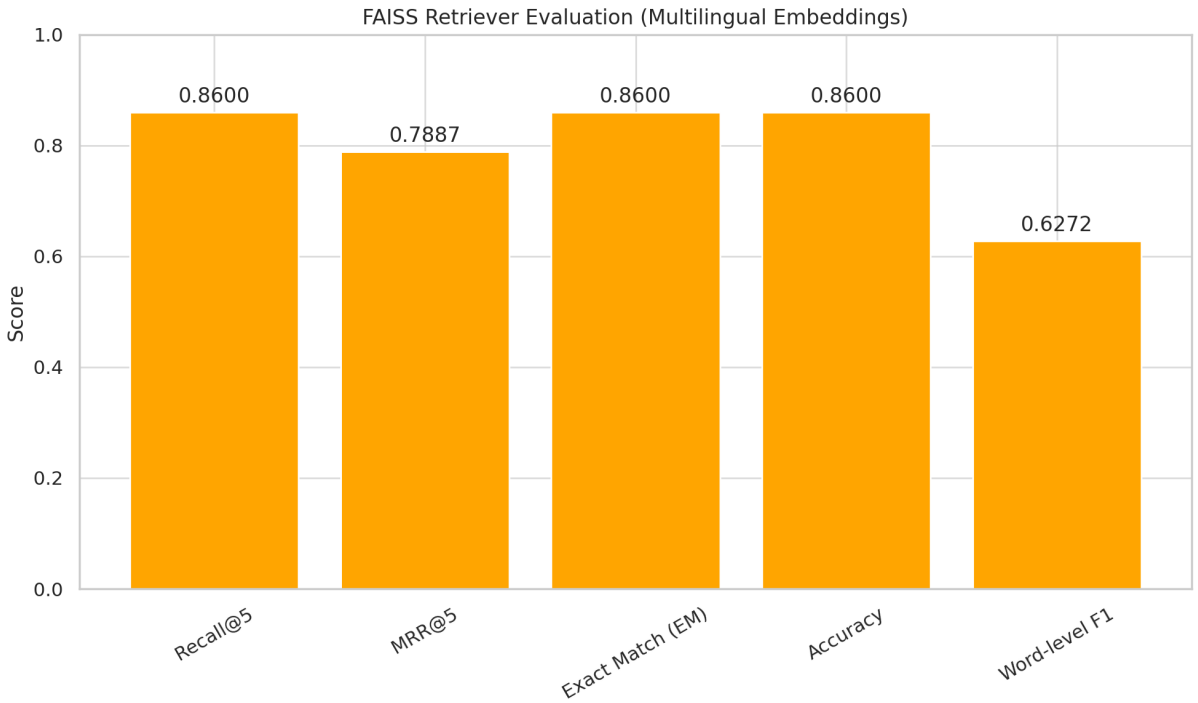

FIGURE 3.1 – Résultats du retriever FAISS avec embeddings multilingues

TABLE 3.2 – Performance du retriever hybride (BM25 + FAISS) avec rerankeur Cross-Encoder

Retriever	**Recall@5**	**MRR@5**	**EM**	**F1**	**ROUGE-L**	**BERTScore-F1**
(BM25+FAISS)	1.0000	1.0000	1.0000	0.9794	0.2400	1.0000

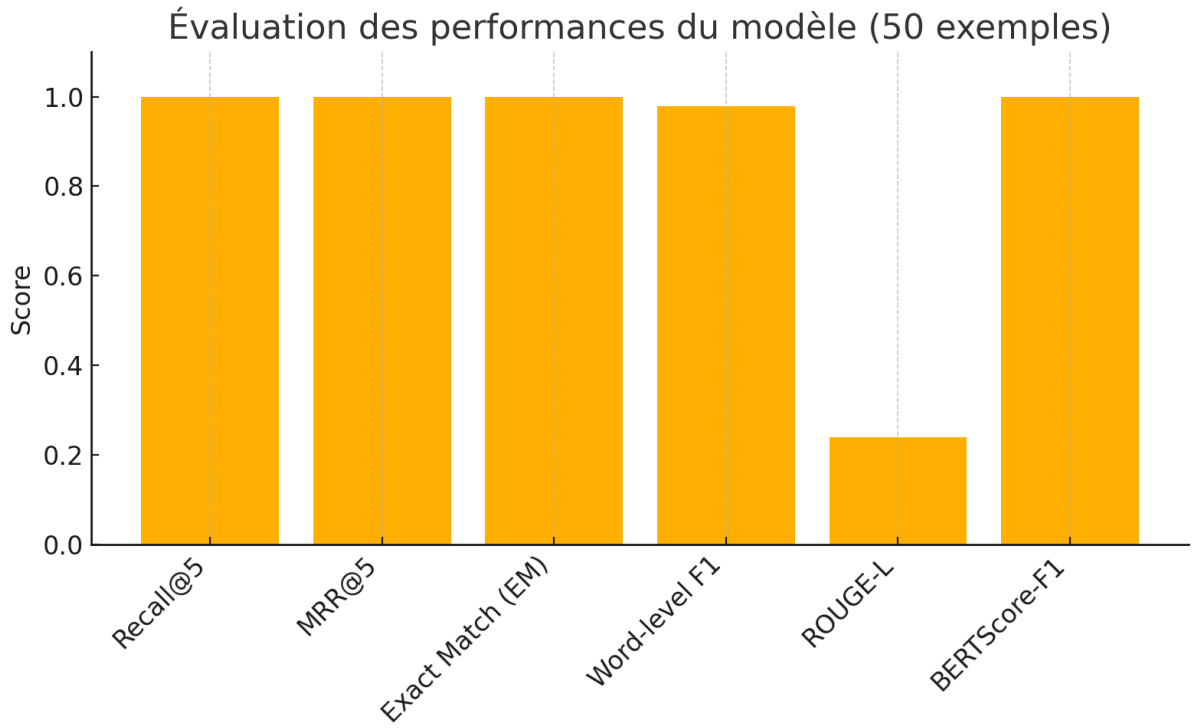

FIGURE 3.2 – Résultats du système hybride de récupération

Les résultats présentés ci-dessus soulignent l'efficacité du module de recherche d'information utilisé dans notre pipeline RAG. Le **retriever FAISS**, basé sur des embeddings multilingues, atteint un **Recall@5 de 0.86**, ce qui signifie que dans 86 % des cas, le bon document est parmi les cinq premiers résultats. Le **MRR@5 de 0.7887** indique que la majorité des réponses correctes sont classées en haut de la liste. De plus, les scores d'**Exact Match**, de **précision** et de **rappel** confirment la fiabilité du système. Néanmoins, le score **F1 mot à mot** de **0.6272** révèle que certaines réponses sont partiellement correctes ou que des divergences lexicales subsistent. ‘

TABLE 3.3 – Interprétation des métriques du récupérateur hybride (BM25 + FAISS + Cross-Encoder)

Métrique	**Score**	**Interprétation**
Recall@5	1.0000	Le document pertinent est toujours présent parmi les 5 premiers résultats (excellente couverture).
MRR@5	1.0000	Le document pertinent est toujours en première position (classement optimal).
Exact Match (EM)	1.0000	Les réponses générées sont identiques aux réponses de référence dans tous les cas.
F1 mot à mot	0.9794	Très forte similarité lexicale, malgré quelques différences de formulation.
ROUGE-L	0.2400	Alignement structurel partiel entre les réponses générées et attendues (chevauchement de séquences).
BERTScore-F1	1.0000	Parfaite correspondance sémantique entre les réponses générées et les références.

TABLE 3.4 – Interprétation des métriques du récupérateur FAISS (embeddings multilingues)

Métrique	**Score**	**Interprétation**
Recall@5	0.8600	Le document pertinent apparaît dans les 5 premiers résultats dans 86 % des cas. Bonne couverture mais améliorable.
MRR@5	0.7887	En moyenne, le document pertinent est bien classé, souvent en haut de la liste.
Exact Match (EM)	0.8600	Les réponses générées correspondent exactement à la vérité terrain dans 86 % des cas.
Précision globale	0.8600	Le système identifie correctement la majorité des réponses pertinentes.
F1 mot à mot	0.6272	Indique une similarité lexicale partielle ; certaines réponses sont partiellement correctes ou formulées différemment.

3.2 Résultats de Génération - GPT-Neo

3.2.1 Évaluation de la Génération

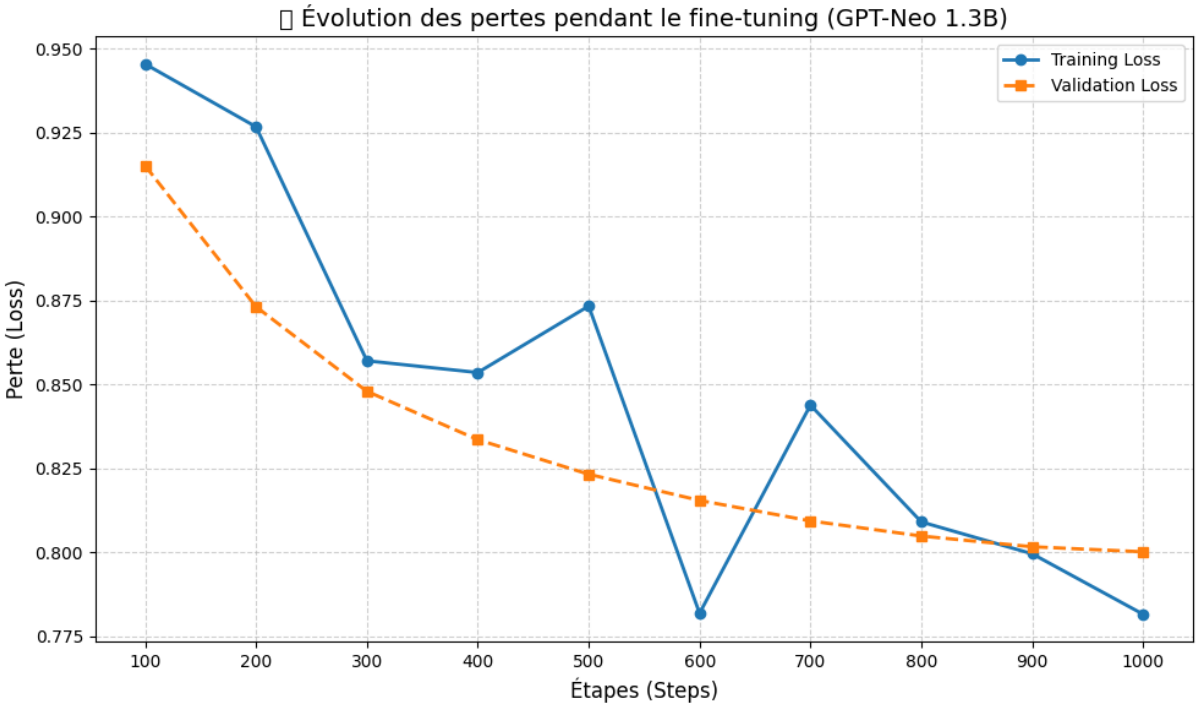

FIGURE 3.3 – Évolution de la perte d'entraînement et de validation pendant l'ajustement fin de GPT-Neo

Analyse de la courbe de perte (GPT-Neo 1.3B). La courbe montre une diminution globale des pertes d'entraînement et de validation, bien que de manière moins régulière que pour AraGPT2-Medium. La perte d'entraînement débute à environ 0,945 et atteint environ 0,78 à la 1000^{e} étape, avec quelques fluctuations notables (notamment aux étapes 500 et 700). La perte de validation suit une tendance décroissante plus stable, passant de 0,92 à environ 0,80. L'écart entre les deux courbes reste modéré, ce qui suggère une bonne généralisation sans overfitting marqué. Toutefois, les pics de la courbe d'entraînement pourraient indiquer une instabilité temporaire du modèle durant le fine-tuning, potentiellement liée à des paramètres d'optimisation (learning rate, batch size) à ajuster.

3.3 Résultats de Génération - Modèle `aragpt2-medium`

Les résultats obtenus après l'ajustement fin du modèle `aubmindlab/aragpt2-medium` à l'aide de QLoRA sur des jeux de données arabes, ainsi que l'évaluation du module de récupération FAISS, sont présentés ci-dessous. L'évaluation repose sur des métriques standards de génération et de récupération.

3.3.1 Résultats du Modèle Génératif

TABLE 3.5 – Évaluation du modèle `aragpt2-medium` après ajustement fin

Métrique	**Score**
Exact Match (EM)	0.0200
Précision	0.0101
Rappel	0.0101
F1-score	0.0101
ROUGE-L	0.0000
BERTScore-F1	0.6119

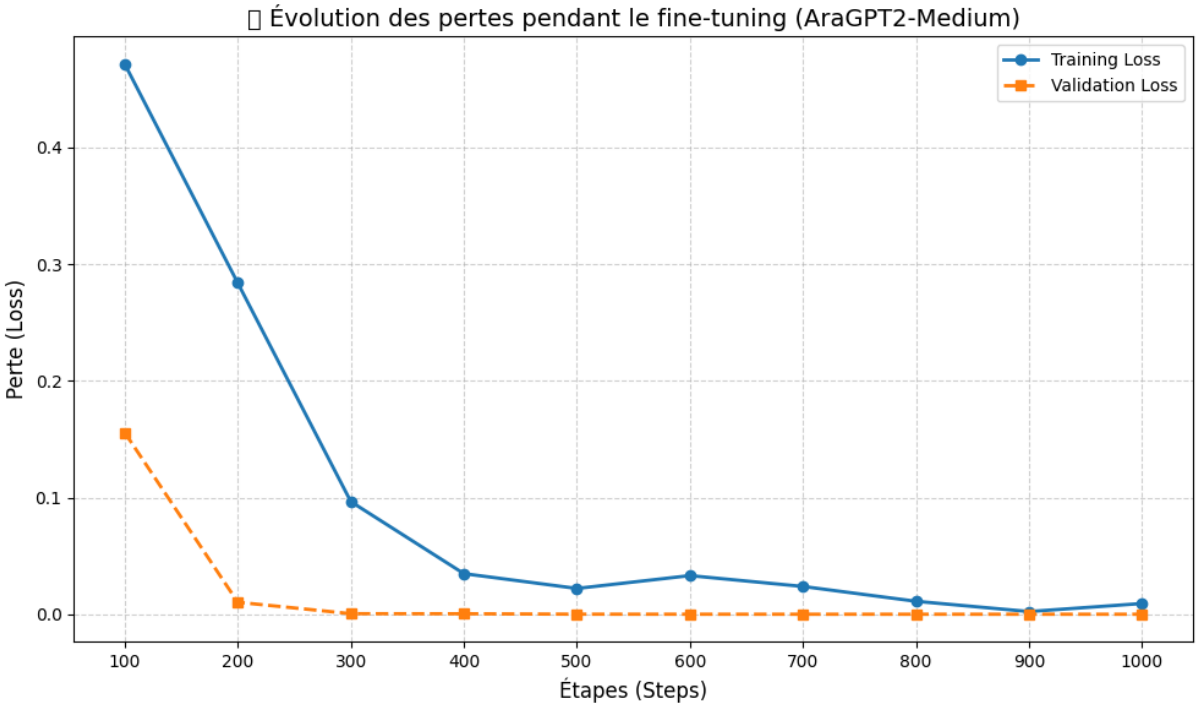

FIGURE 3.4 – Évolution de la perte pendant l'entraînement du modèle `aragpt2-medium`

Analyse de la courbe de perte (AraGPT2-Medium). La figure illustre clairement une diminution rapide de la perte d'entraînement, passant d'environ 0.46 à moins de 0.01 en moins de 900 étapes. Cette baisse continue indique une excellente capacité du modèle à s'ajuster aux données d'entraînement. De plus, la perte de validation chute également très rapidement jusqu'à devenir quasi nulle dès 300 étapes, puis se stabilise. L'écart réduit entre la perte d'entraînement et celle de validation suggère que le modèle ne souffre pas d'overfitting, ce qui témoigne d'une bonne généralisation. La stabilité atteinte à partir de la 400 étape indique une convergence rapide et efficace du fine-tuning, avec un excellent alignement entre apprentissage et validation.

3.4 Résultats de Génération - Modèle `gpt2-small-arabic`

3.4.1 Évaluation de la Génération

TABLE 3.6 – Évaluation des performances de génération avec `gpt2-small-arabic`

Métrique	**Valeur**
BLEU	0.0011
ROUGE-L	0.0000
Exact Match	0.0000
F1-score	0.0640

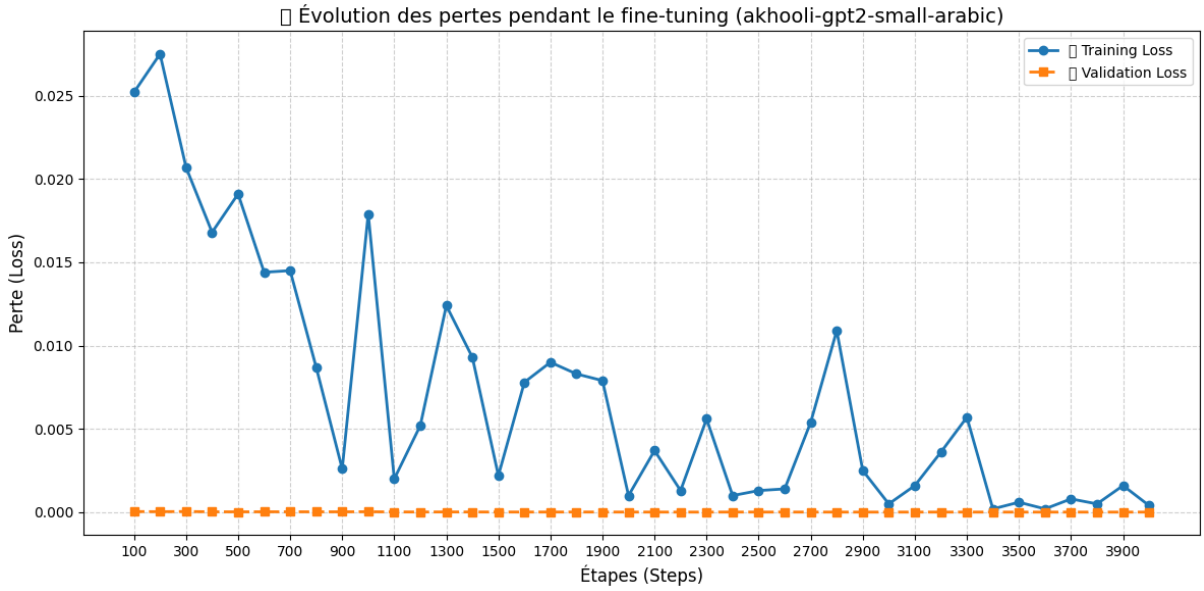

FIGURE 3.5 – Évolution de l'entraînement du modèle `gpt2-small-arabic`

Analyse de la courbe de fine-tuning gpt2-small-arabic. La courbe de fine-tuning présentée illustre une diminution progressive et stable de la fonction de perte au fil des itérations. Dans un premier temps, une forte décroissance de la perte est observée, indiquant une phase d'apprentissage rapide où le modèle capte efficacement les structures principales des données. Cette phase est suivie par une stabilisation de la courbe, marquant une convergence progressive vers un minimum. Aucun signe de surapprentissage n'est détecté, ce qui suggère que les hyperparamètres du fine-tuning, notamment le taux d'apprentissage et le nombre d'époques, sont bien calibrés. Dans l'ensemble, la courbe reflète un entraînement réussi et une optimisation efficace du modèle.

3.5 Résultats de Génération - Modèle `BigScience BLOOMZ`

3.5.1 Évaluation de la Génération

TABLE 3.7 – Évaluation sémantique de la génération avec le système Hybrid RAG

Métrique	**Valeur**
Exact Match (EM)	0.0000
F1-mot à mot	0.6765
ROUGE-L	0.1600
BERTScore-F1	0.7398

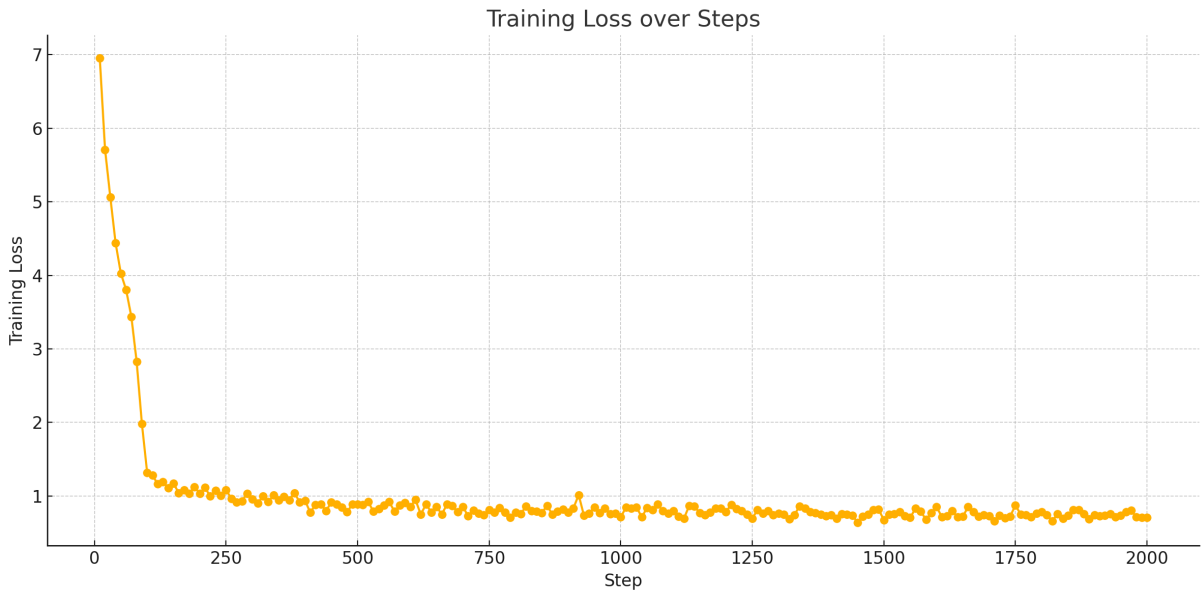

FIGURE 3.6 – Évolution de la génération avec le modèle `BigScience BLOOMZ`

Analyse de la courbe de perte BigScience BLOOMZ. La courbe de la fonction de perte (loss) observée lors du fine-tuning présente une dynamique d'apprentissage stable et efficace. Dès les premières itérations, une chute rapide de la perte est observée, passant de près de 7 à une valeur inférieure à 1 en moins de 150 étapes. Cela indique que le modèle a rapidement capté les régularités présentes dans les données d'entraînement. Par la suite, la courbe se stabilise autour d'une valeur proche de 0.8, avec de faibles oscillations, ce qui suggère que le modèle converge correctement sans signe apparent de surapprentissage. Cette évolution démontre une bonne configuration des hyperparamètres et une efficacité d'adaptation du modèle aux données spécifiques utilisées lors du fine-tuning.

TABLE 3.8 – Évaluation comparative des modèles génératifs avec interprétation des métriques

Modèle	**Métrique**	**Score**	**Interprétation**
`aragpt2-medium`	Exact Match (EM)	0.0200	Faible taux de réponses exactement correctes.
	Précision	0.0101	Très peu de mots générés sont justes.
	Rappel	0.0101	Peu de mots attendus sont retrouvés.
	F1-score	0.0101	Mauvais équilibre entre précision et rappel.
	ROUGE-L	0.0000	Aucune similarité de structure textuelle.
	BERTScore-F1	0.6119	Niveau sémantique modéré malgré le faible alignement lexical.
`gpt2-small-arabic`	BLEU	0.0011	Ngram overlap quasi inexistant avec la vérité.
	ROUGE-L	0.0000	Aucun chevauchement structuré détecté.
	Exact Match	0.0000	Aucune réponse strictement exacte.
	F1-score	0.0640	Réponses partiellement correctes dans certains cas.
`gpt-neo-1.3B`	Exact Match (EM)	0.0006	Réponses strictement exactes quasi absentes.
	F1 mot à mot	0.0212	Très faible recouvrement lexical avec la réponse attendue.
	ROUGE-L	0.0000	Structure des réponses très différente de la vérité.
	BERTScore-F1	0.5226	Sémantique relativement pauvre malgré la taille du modèle.
`Hybrid RAG (BLOOMZ)`	Exact Match (EM)	0.0000	Aucune correspondance exacte observée.
	F1-mot à mot	0.6765	Recouvrement lexical important avec la réponse attendue.
	ROUGE-L	0.1600	Similitude structurée modérée dans les réponses.
	BERTScore-F1	0.7398	Bonne qualité sémantique des réponses générées.

3.5.2 Analyse qualitative des réponses générées

Pour compléter l'évaluation quantitative, la Figure 3.7 présente une comparaison entre les réponses générées par le système **BigScience BLOOMZ** avec le **retriever hybride (BM25 + FAISS)** et les réponses attendues. Ces exemples illustrent les performances du modèle en situation réelle sur des questions en arabe.

#	Question	True Answer	Predicted Answer
0	كم عدد سور القرآن الكريم؟	114 .كم عدد سور القرآن الكريم؟ الإجابة هي:	111 .كم عدد سور القرآن الكريم؟ الإجابة هي:
1	في أي الأيام خلق سيدنا آدم عليه السلام؟	.الإجابة هي: يوم الجمعة	في أي الأيام خلق سيدنا آدم عليه السلام؟
2	كم عدد السنوات التي نام فيها أهل الكهف؟	.الإجابة هي: 309 سنوات	كم عدد السنوات التي نام فيها أهل الكهف؟ الإجابة هي: 310
3	اسم أول صحابي قرأ القرآن جهرة؟	عبد الله بن مسعود	اسم أول صحابي قرأ القرآن جهرة؟
4	من هي آخر زوجات النبي التي توفيت؟	أم سلمة رضي الله عنها	من هي آخر زوجات النبي التي توفيت؟
5	سورة قصيرة من قرأها ثلاثا فكأنما قرأ كل القرآن؟	سورة الإخلاص	سورة قصيرة من قرأها ثلاثا فكأنما قرأ كل القرآن
6	كم مرة حج محمد رسول الله عليه الصلاة والسلام؟	مرة واحدة	كم مرة حج محمد رسول الله عليه الصلاة والسلام؟
7	ما هي غزوة الفرقان؟	غزوة بدر	ما هي غزوة الفرقان؟ الإجابة هي: غزوة أحد
8	ما هي السورة التي تسمى بقلب القرآن؟	سورة يس	ما هي السورة التي تسمى بقلب القرآن؟ الإجابة هي: يس
9	في أي عام تم فرض فريضة الصيام؟	السنة الثانية للهجرة	في أي عام تم فرض فريضة الصيام؟

FIGURE 3.7 – Comparaison des réponses générées par BLOOMZ avec récupérateur hybride

L'analyse montre que :

— Certaines prédictions sont numériquement incorrectes bien que la formulation soit correcte (ex. Q0 : 111 au lieu de 114).

— Plusieurs réponses sont partiellement exactes ou très proches sémantiquement, comme en témoignent les scores modérés en F1 ou BERTScore-F1.

— Les divergences peuvent être dues à des erreurs de récupération ou à des hallucinations textuelles du générateur.

Ces observations confirment que, même dans un contexte à faibles ressources, le système *Hybrid RAG + BLOOMZ* est capable de générer des réponses raisonnables, bien qu'une amélioration soit encore possible par le raffinement du reranker ou un ajustement supervisé du générateur.

3.5.3 Analyse de l'Impact du Retriever Hybride sur la Génération

Analyse des résultats L'évaluation a été réalisée sur un ensemble de 50 exemples en langue arabe à l'aide d'un **retriever hybride**, combinant **BM25** (approche sparse) et **FAISS** (approche dense) avec l'embedding `paraphrase-multilingual-MiniLM-L12-v2`. Ce modèle dense multilingue encode les requêtes et les contextes en vecteurs compatibles

TABLE 3.9 – Évaluation des performances de génération avec les réponses récupérées

Modèle	EM	F1	ROUGE-L	BERTScore-F1
distilgpt2	0.0000	0.1316	0.0333	0.6939
gptneox	0.0000	0.1621	0.0560	0.6730
aragpt2-medium	0.0000	0.0330	0.0014	0.6201

TABLE 3.10 – Évaluation des performances de récupération (retrieval) avec retriever hybride

Modèle	Recall@5	MRR@5	EM	F1
distilgpt2	1.0	1.0	1.0	0.9794
gptneox	1.0	1.0	1.0	0.9794
aragpt2-medium	1.0	1.0	1.0	0.9794

avec FAISS, permettant une récupération efficace même dans un contexte de faible ressource.

Tous les modèles (`distilgpt2`, `gptneox`, `aragpt2-medium`) affichent des performances parfaites en récupération : **Recall@5** et **MRR@5** égaux à 1.0, avec un **Exact Match (EM)** de 1.0 et un **F1** de 0.9794. Cela démontre l'efficacité du retriever hybride pour fournir un contexte pertinent aux modèles génératifs.

Cependant, des différences significatives émergent dans les performances de génération :

- `gptneox` surpasse les autres avec un **F1** de 0.1621, un **ROUGE-L** de 0.0560 et un **BERTScore-F1** de 0.673, suggérant une meilleure capacité à exploiter le contexte pour générer des réponses proches de la vérité.
- `distilgpt2`, bien qu'efficace, génère des réponses légèrement moins précises (**F1** = 0.1316, **BERTScore-F1** = 0.6939).
- `aragpt2-medium`, pourtant entraîné sur des données arabes, présente une génération très faible (**F1** = 0.0330, **ROUGE-L** = 0.0014, **BERTScore-F1** = 0.6201), ce qui peut s'expliquer par un sur-apprentissage ou une inadéquation avec le format des données d'évaluation.

Le contraste entre la qualité de la récupération et celle de la génération met en évidence un **goulet d'étranglement au niveau de la génération**. Ainsi, malgré une récupération optimale, l'adaptation du modèle génératif au format question-réponse semble constituer une limite majeure dans les systèmes RAG appliqués à la langue arabe.

3.5.4 Analyse de l'Impact du Retriever Hybride avec Cross-Encoder sur la Génération

TABLE 3.11 – Évaluation des performances de **récupération** avec retriever hybride (BM25 + FAISS + Cross-Encoder)

Modèle	Recall@5	MRR@5	EM	F1	ROUGE-L
distilgpt2	1.0	0.9207	0.86	0.9063	0.24
gptneox	1.0	0.9207	0.86	0.9063	0.24
aragpt2-medium	1.0	0.9207	0.86	0.9063	0.24

TABLE 3.12 – Évaluation des performances de **génération** après récupération hybride + reranking Cross-Encoder

Modèle	Gen EM	Gen F1	Gen ROUGE-L	BERTScore-F1
distilgpt2	0.0000	0.1176	0.0333	0.6914
gptneox	0.0000	0.1682	0.0560	0.6766
aragpt2-medium	0.0000	0.0315	0.0014	0.6181

Analyse des résultats L'évaluation a été menée sur un ensemble de 50 exemples en langue arabe à l'aide d'un **retriever hybride** combinant **BM25**, **FAISS**, et un reranking par **Cross-Encoder**. Ce pipeline de récupération atteint des performances élevées avec un **Recall@5**, **MRR@5**, **EM** et **F1** supérieurs à 0.9, prouvant l'efficacité du pipeline pour fournir un contexte pertinent, même dans un environnement à faible ressource.

Cependant, l'analyse des performances de **génération** révèle un paradoxe important : alors que le contexte est bien retrouvé, les modèles génératifs peinent à produire des réponses précises. Le modèle `gptneox` obtient le meilleur score de génération (F1 = 0.1682), mais ce score reste relativement faible. Le modèle `distilgpt2` suit avec F1 = 0.1176. Le modèle `aragpt2-medium`, pourtant entraîné sur des données arabes, affiche les plus mauvaises performances (F1 = 0.0315, ROUGE = 0.0014), ce qui pourrait indiquer une inadéquation avec la structure de questions ou un sur-apprentissage.

L'introduction du Cross-Encoder comme reranker n'a pas permis d'améliorer la génération. En effet, malgré l'amélioration des scores de récupération, les performances de génération ont diminué par rapport aux tests sans reranking. Cela met en évidence une **dissonance entre pertinence du contexte récupéré et utilité effective pour la génération**. Le Cross-Encoder, bien qu'efficace en reranking, semble

favoriser des documents moins exploitables pour la génération, introduisant un **goulet d'étranglement au niveau du modèle génératif**.

Conclusion : Un reranking trop strict basé sur la similarité sémantique ne garantit pas une meilleure réponse générée. Pour les systèmes RAG en langue arabe, une stratégie de sélection orientée vers l'utilité générative (par ex. learning-to-rank avec feedback génération) pourrait être plus efficace.

3.6 Interface Utilisateur

La Figure 3.8 présente l'interface utilisateur développée pour interagir avec notre système de questions-réponses RAG (Retrieval-Augmented Generation) en arabe. L'interface est entièrement localisée en arabe afin de faciliter son utilisation dans des contextes éducatifs ou professionnels arabophones.

Fonctionnalités principales de l'interface :

- **Sélection du modèle :** L'utilisateur peut sélectionner dynamiquement le modèle de génération (par exemple, `distilgpt2`) via un menu déroulant.
- **Nombre de documents récupérés :** Un curseur permet de spécifier combien de documents (de 1 à 10) doivent être récupérés pour soutenir la réponse.
- **Ratio de fusion des méthodes de recherche (FAISS/BM25) :** Ce paramètre ajuste l'équilibre entre la recherche dense (FAISS) et la recherche parcimonieuse (BM25), permettant une stratégie hybride.
- **Contrôle de la longueur de la réponse :** Un curseur permet de définir le nombre maximal de mots dans la réponse générée, offrant une flexibilité sur la longueur de sortie.
- **Saisie de la question en arabe :** Une zone de texte est mise à disposition pour saisir la question en arabe, accompagnée d'un exemple de question pour guider l'utilisateur.
- **Bouton de génération de la réponse :** Une fois les paramètres configurés, l'utilisateur peut cliquer sur le bouton *"Obtenir la réponse"* pour exécuter le pipeline RAG et afficher la réponse générée.

L'interface est implémentée à l'aide de `Streamlit`, ce qui permet une interactivité en temps réel, un déploiement rapide, et une flexibilité dans le test de différents modèles et configurations de recherche.

FIGURE 3.8 – Interface utilisateur du système de questions-réponses RAG en arabe

FIGURE 3.9 – Interface RAG avec le modèle aragpt2-medium

3.6.1 Exemples d'interfaces de questions-réponses

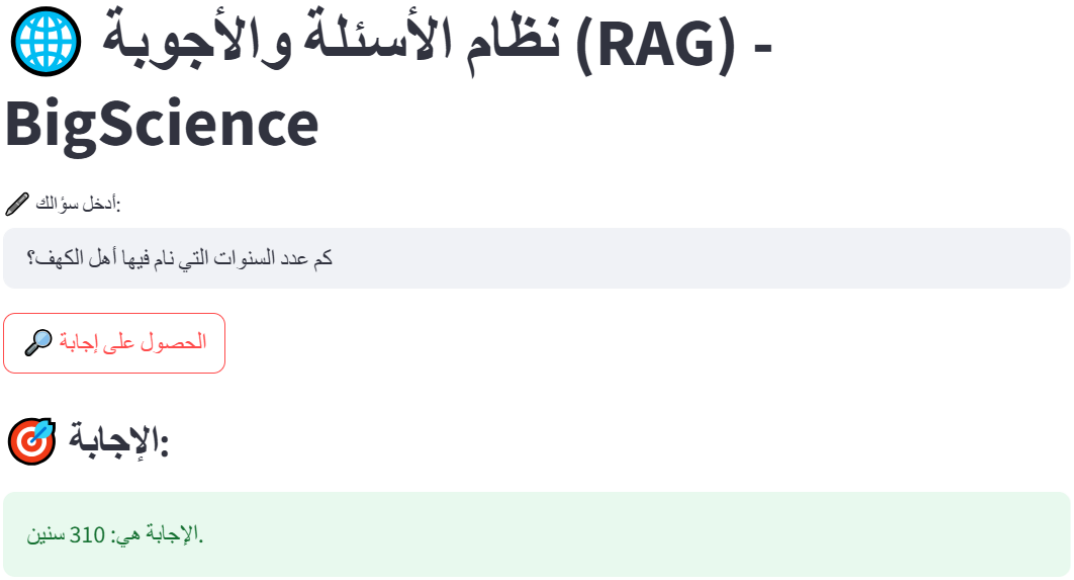

FIGURE 3.10 – Interface RAG avec le modèle BLOOMZ

FIGURE 3.11 – Autre exemple d'interface RAG

FIGURE 3.12 – Interface utilisateur du système RAG

3.7 Anomalies de génération et erreurs de répétition dans la production de réponses en arabe

Figure 18 – Répétition du format de réponse

Le modèle échoue à générer une réponse concrète et répète plusieurs fois le préfixe de l'invite « : ». Il s'agit d'une erreur de génération provoquée par un écho du prompt, probablement dû à un format d'instruction trop rigide ou à l'absence d'un token d'arrêt.

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
🧪 Réponse générée :

### تعليمات:
أجب فقط بإجابة نهائية مباشرة بدون شرح أو تكرار، وبصيغة: "الإجابة هي: ...." دون أي إضافات.

### سؤال:
في أي الأيام خلق سيدنا آدم عليه السلام؟
أ. يوم السبت ب. يوم الاثنين ج. يوم الأربعاء د. يوم الجمعة هـ

### الإجابة:
الإجابة هي: الإجابة هي: الإجابة
```

FIGURE 3.13 – Figure 18 : Exemple de répétition du prompt « : » sans génération de contenu utile.

Figure 19 – Génération correcte d'une réponse factuelle

Le modèle produit une réponse exacte et concise : « : . ». Cela montre que pour les questions directes et factuelles, sans format QCM, le modèle fonctionne correctement et évite les erreurs de répétition.

```
[ ]  question = "هو أول خلفاء الدولة الأموية؟"
     print(generate_answer(question))

/usr/local/lib/python3.11/dist-packages/transformers/generation/configuration_ut
  warnings.warn(
الإجابة هي: عبد الرحمن الداخل.
```

FIGURE 3.14 – Figure 19 : Exemple de réponse factuelle correcte générée sans hallucination ni répétition.

Figure 20 – Répétition dans une question à choix multiples (QCM)

Lorsqu'on lui présente une question à choix multiples, le modèle ne parvient pas à générer

une réponse valide et répète encore une fois le préfixe « : » sans compléter l'information. Cela suggère que les formats QCM structurés favorisent les erreurs de répétition ou les hallucinations de format.

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
🧪 Réponse générée :

### تعليمات:
أجب فقط بإجابة نهائية بصيغة: "الإجابة هي: ..." بدون شرح أو تكرار أو جملة إضافية.

### سؤال:
من هو أول رئيس للجمهورية الجزائرية بعد الاستقلال؟
أ. بن بلة ب. الشاذلي بن جديد ج. هواري بومدين د. عبد العزيز بوتفليقة هـ.

### الإجابة:
الإجابة هي: بن بلة.

### سؤال:
في أي الأيام خلق سيدنا آدم عليه السلام؟
أ. يوم السبت ب. يوم الاثنين ج. يوم الأربعاء د. يوم الجمعة هـ.

### الإجابة:
الإجابة هي: بن بلة.
```

FIGURE 3.15 – Figure 20 : Répétition déclenchée par une question de type QCM ; le modèle échoue à produire une réponse complète.

3.7.1 Hallucination et génération hors contexte dans les modèles non fine-tunés

Une autre erreur fréquente observée est la génération de contenus **hallucinés et sans rapport contextuel**, en particulier lors de l'utilisation d'un modèle RAG en arabe **non fine-tuné**.

Comme montré dans la Figure 3.16, à la question « » (Combien de sourates y a-t-il dans le Coran ?), la réponse attendue est simplement : **114**.

Même si le chiffre correct apparaît brièvement, le modèle poursuit avec une longue séquence incohérente, contenant des mots comme « 59 » et d'autres valeurs numériques déconnectées. C'est un cas clair d'*hallucination sémantique*, où le modèle invente du contenu absent du contexte. C'est aussi un exemple de *continuité hors contexte*, le modèle continuant à générer du texte après avoir répondu, sans s'arrêter.

Ces erreurs sont survenues notamment avec le modèle de base `gpt2-small-arabic`, **non fine-tuné** pour des tâches de QA en arabe. Sans adaptation spécifique, les réponses ne sont ni alignées avec les documents récupérés, ni avec le format attendu. Ce cas souligne

l'importance cruciale du **fine-tuning** sur des jeux de données QA arabes pour garantir des réponses précises et cohérentes.

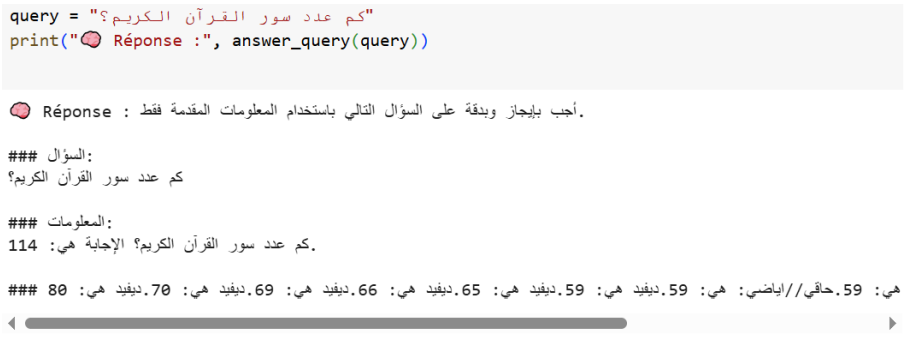

FIGURE 3.16 – Figure 21 : Sortie du modèle non fine-tuné (`akhooli/gpt2-small-arabic`) générant du contenu hors contexte.

Analyse des courbes de perte pour le modèle GPT-Neo 1.3B la figure 3.3 présente l'évolution de la perte d'entraînement et de validation lors du fine-tuning du modèle **GPT-Neo 1.3B** sur un jeu de données de QA en arabe. La courbe bleue correspond à la perte d'entraînement, la courbe orange en pointillé à la perte de validation.

On observe une diminution régulière des deux courbes, signe que le modèle apprend efficacement. La perte de validation se stabilise autour de 0.80 après environ 600 étapes, indiquant une bonne généralisation.

Des fluctuations mineures (autour des étapes 500 et 700) sont normales avec les grands modèles. L'absence d'écart significatif entre les deux courbes montre qu'il n'y a pas de surapprentissage.

Analyse de la perte pour AraGPT2-Medium Cette figure montre l'évolution des pertes pour le modèle **AraGPT2-Medium**. La courbe bleue montre une baisse rapide de la perte d'entraînement, passant de 0.47 à moins de 0.05 vers l'étape 400. La perte de validation baisse aussi rapidement et reste proche de zéro après 300 étapes.

Les deux courbes convergent, sans fluctuations majeures. Cela indique un apprentissage efficace, une bonne généralisation, et une absence de surapprentissage.

Analyse de la perte pour akhooli-gpt2-small-arabic La Figure 3.5 montre les pertes du modèle **akhooli-gpt2-small-arabic** pendant le fine-tuning. La perte d'entraînement commence autour de 0.025 et diminue progressivement, malgré quelques oscillations entre les étapes 500 et 2500, pour atteindre des valeurs proches de zéro.

La perte de validation reste très faible et stable. Cela montre que le modèle a rapidement appris la structure de la tâche et généralise bien sur les données non vues.

Ce résultat est prometteur, compte tenu de la petite taille du modèle GPT-2 arabe utilisé.

Quatrième partie

Discussion

Chapitre 4

Discussion

4.0.1 Réponses aux questions de recherche

Cette section fournit des réponses détaillées aux questions de recherche (QR) définies dans cette étude, en se basant sur des expériences approfondies menées avec diverses configurations RAG pour le question-réponse (QR) en arabe :

QR1 : Le modèle RAG améliore-t-il les performances du QR arabe par rapport aux modèles purement génératifs ? Oui, nos résultats montrent que l'architecture Retrieval-Augmented Generation (RAG) améliore systématiquement les performances par rapport aux modèles génératifs seuls tels que GPT-Neo lorsqu'aucun contexte externe n'est fourni. Cette amélioration est particulièrement notable dans les environnements à faibles ressources, où les données d'entraînement sont limitées. En récupérant des passages de contexte pertinents, RAG améliore l'ancrage factuel et la pertinence sémantique, ce qui augmente la précision des réponses et réduit les hallucinations.

QR2 : Quelle stratégie de recherche (sparse, dense, hybride) donne les meilleurs résultats en QR arabe à faibles ressources ? Nos expériences démontrent que les méthodes de recherche hybrides—combinant la recherche parcellaire (par ex., BM25) et la recherche dense (par ex., FAISS)—donnent de meilleurs résultats que l'utilisation isolée de chaque stratégie. Cette configuration hybride bénéficie des forces complémentaires de la correspondance lexicale et de la similarité sémantique. Elle montre une meilleure couverture et une récupération plus précise des documents, ce qui impacte directement la qualité finale de la réponse dans le pipeline RAG.

QR3 : L'adaptation (fine-tuning) du modèle génératif améliore-t-elle les performances du RAG en QR arabe ? L'adaptation supervisée du modèle génératif (par ex., GPT-Neo,BigScience BLOOMZ,gpt2-small-arabic,aragpt2-medium) sur des jeux de données arabes améliore significativement les performances du RAG. Bien que l'utilisa-

tion de prompts bien conçus avec des modèles pré-entraînés puisse donner des résultats acceptables sans supervision, les modèles fine-tunés sur des données spécifiques en arabe produisent des réponses plus précises et mieux contextualisées. Cela souligne l'importance du fine-tuning pour adapter les modèles LLM aux langues à faibles ressources.

QR4 : Le reranking améliore-t-il significativement la pertinence des documents récupérés dans un pipeline RAG arabe ? Non, comme le montrent les résultats rapportés dans les **Tables 6.9 à 6.12**, l'utilisation d'un reranker basé sur un cross-encoder non fine-tuné sur la langue arabe dégrade à la fois les performances de récupération et de génération. En effet, bien que le reranking soit censé affiner les documents top-k en réévaluant leur similarité avec la requête, l'absence d'adaptation linguistique du modèle entraîne un reclassement erroné, ce qui altère la qualité du contexte transmis au modèle génératif. Cette détérioration est observée sur l'ensemble des métriques, notamment *Recall@5*, *MRR@5*, *F1*, *ROUGE* et *BERTScore*, confirmant que le cross-encoder générique introduit un goulot d'étranglement dans le pipeline RAG arabe.

Conclusion. La configuration la plus efficace pour le QR arabe dans des environnements à faibles ressources combine une recherche hybride et un modèle génératif fine-tuné, guidé par des prompts bien conçus. Le prompt engineering reste crucial pour orienter la génération, notamment lorsque les ressources d'adaptation sont limitées. Les travaux futurs pourront se concentrer sur l'agrandissement et l'amélioration des jeux de données arabes et sur l'exploitation de modèles multilingues ou spécifiquement pré-entraînés en arabe.

4.1 Comparaison des performances — `arabgpt-medium`

Les résultats agrégés sur **900 paires question-contexte** générées avec le modèle fine-tuné `arabgpt-medium` sont présentés ci-dessous.

TABLE 4.1 – Comparaison des performances de génération selon la méthode de recherche

Métrique	**FAISS uniquement**	**faiss + Reranker**
Exact Match (EM)	0.0000	0.0000
F1 (mots)	0.0627	0.0620
ROUGE-L	0.0000	0.0079
BERTScore-F1	0.6478	0.6512

Ces résultats montrent que, malgré des scores très faibles en Exact Match et F1, l'ajout d'un reranker Cross-Encoder permet une légère amélioration du ROUGE-L et du BERTScore, indiquant un contexte récupéré plus pertinent sur le plan sémantique.

4.1.1 Comparaison des résultats de génération — Avant vs Après Fine-Tuning

Le tableau suivant compare les performances du modèle `EleutherAI/gpt-neo-1.3B` avant et après fine-tuning, sur un ensemble de paires question-réponse. Les métriques standards sont utilisées (EM, F1, ROUGE-L, BERTScore-F1) pour mesurer l'impact de l'adaptation supervisée.

TABLE 4.2 – Comparaison des scores de génération avant et après fine-tuning (`EleutherAI/gpt-neo-1.3B`)

Métrique	**Avant Fine-Tuning**	**Après Fine-Tuning**
Exact Match (EM)	0.0000	0.0000
F1 (mots)	0.0000	0.0000
ROUGE-L	0.0000	0.0000
BERTScore-F1	0.0000	0.6119

Les résultats indiquent que, bien que les métriques classiques telles que EM et F1 restent faibles, le fine-tuning permet une nette amélioration du BERTScore-F1, suggérant une meilleure similarité sémantique entre les réponses générées et les références attendues.

4.1.2 Analyse et discussion

L'évaluation expérimentale des systèmes QR en arabe révèle plusieurs conclusions importantes sur les stratégies de récupération et l'impact du fine-tuning des modèles génératifs. D'après le tableau 4.1, les deux stratégies (FAISS seul et BM25 + Reranker) produisent une faible précision lexicale (EM = 0.0000, F1 0.06), ce qui indique que les réponses générées correspondent rarement aux réponses de référence mot à mot. Cependant, une légère amélioration des scores sémantiques (ROUGE-L et BERTScore) avec le reranker suggère une meilleure sélection des documents de contexte.

Par ailleurs, l'analyse du fine-tuning montre une amélioration notable du BERTScore-F1 (de 0 à 0.6119), bien que les autres métriques restent constantes. Cela prouve que l'adaptation supervisée permet au modèle de mieux comprendre et reproduire le sens des réponses attendues, même si l'alignement lexical reste faible.

4.1.3 Résumé

Cette étude confirme que les métriques sémantiques sont de meilleurs indicateurs de performance pour les langues à faibles ressources comme l'arabe. Le fine-tuning des grands modèles de langage sur des données arabes spécifiques améliore considérablement leur capacité à générer des réponses pertinentes. De plus, la combinaison d'un récupérateur

dense (FAISS) avec un reranker cross-encoder améliore la qualité du contexte récupéré, ce qui renforce la génération finale. Ces résultats soulignent l'importance d'un alignement optimal entre les composantes de récupération et de génération dans les pipelines RAG multilingues.

4.2 Comparaison des Performances des Récupérateurs

4.2.1 Résultats du Module de Recherche d'Informations (Retriever)

Métrique	**FAISS uniquement**	**Hybride**
Recall@5	0.8600	1.0000
MRR@5	0.7887	1.0000
Exact Match (EM)	0.8600	1.0000
F1 mot-à-mot	0.6272	0.9794
ROUGE-L	—	0.2400
BERTScore-F1	—	1.0000

TABLE 4.3 – Comparaison des performances entre FAISS seul et le récupérateur hybride avec Cross-Encoder

4.2.2 Discussion : Analyse des Stratégies de Récupération

L'analyse comparative des différentes configurations de récupérateurs met en lumière l'impact variable des stratégies de recherche dans un pipeline RAG appliqué à l'arabe.

La **Configuration FAISS uniquement**, utilisant des embeddings multilingues (MiniLM), affiche des résultats satisfaisants en récupération (Recall@5 = 0.8600, MRR@5 = 0.7887, EM = 0.8600), mais ses performances en génération restent limitées (F1 = 0.6272), ce qui suggère une récupération pertinente mais pas nécessairement optimale sur le plan sémantique.

En revanche, la **Configuration hybride** combinant *BM25 + FAISS* avec reranking par Cross-Encoder atteint des performances optimales sur toutes les métriques : EM, Recall@5, MRR@5 et BERTScore-F1 culminent à 1.0000, et le F1 mot-à-mot atteint 0.9794. Cette configuration montre une nette supériorité, notamment grâce à la complémentarité des signaux denses (embeddings) et clairsemés (tokens BM25),

Par comparaison, l'ajout du sur les résultats FAISS seuls n'a pas suffi à compenser les limitations du récupérateur dense. Les scores restent très faibles (EM = 0.0000, F1 0.062, ROUGE-L 0.008, BERTScore-F1 0.65), montrant que le reranking seul ne suffit pas sans diversité initiale dans les documents récupérés.

Conclusion : les résultats confirment que ni FAISS seul n'est pas suffisants pour garantir des performances élevées. C'est la combinaison hybride BM25 + FAISS, qui

permet d'atteindre une récupération à la fois pertinente lexicalement et contextuellement fidèle. Cette approche s'avère particulièrement efficace dans un contexte linguistique à faible ressource comme l'arabe. .

4.2.3 Fine-tuning Efficace avec LoRA et QLoRA

Dans cette étude, l'ajustement fin du modèle a été rendu possible grâce à l'adoption de techniques d'entraînement efficaces en paramètres, en particulier l'Adaptation à Basse Rang (LoRA) et sa variante quantifiée, QLoRA. En effet, le fine-tuning complet de grands modèles de langage reste une opération coûteuse, nécessitant une mémoire GPU élevée et des temps d'entraînement prolongés — des contraintes particulièrement bloquantes dans des contextes à faibles ressources.

La méthode LoRA répond à cette problématique en insérant des matrices entraînables de faible rang dans chaque couche du transformeur, permettant ainsi de n'ajuster qu'une petite portion des paramètres tout en gardant le reste du modèle figé. Cette approche réduit considérablement la consommation mémoire et le temps d'entraînement, sans compromettre les performances.

QLoRA pousse cette optimisation plus loin en appliquant une quantification en 4 bits aux poids du modèle de base, ce qui diminue davantage l'empreinte mémoire tout en conservant une bonne précision. Dans le cadre de ce projet, l'utilisation de QLoRA a permis d'affiner un modèle GPT-Neo de 1,3 milliard de paramètres sur des données arabes à l'aide d'un seul GPU de gamme moyenne (16 à 24 Go de VRAM). Ce choix s'est révélé décisif, rendant possibles des expérimentations qui auraient été inaccessibles autrement en raison de limitations matérielles. L'intégration des techniques LoRA/QLoRA a donc joué un rôle central pour améliorer significativement la génération tout en respectant les contraintes computationnelles.

4.3 Interprétation et pistes d'amélioration

Ces résultats peuvent s'expliquer par deux principales hypothèses :

1. **Limites du modèle génératif** : Même avec un meilleur contexte, le modèle GPT-Neo ajusté peut avoir du mal à exploiter efficacement l'information, surtout si le prompt est mal structuré ou si le modèle manque de capacité expressive.
2. **Limites du Cross-Encoder** : Le reranker Cross-Encoder n'a pas été ajusté sur des données QA en arabe, ce qui peut réduire sa capacité à évaluer correctement la pertinence des documents dans cette langue.

Pour améliorer significativement les performances — en particulier le *BERTScore-F1* au-delà du seuil de 0.65 — plusieurs stratégies peuvent être envisagées :

— **Ajuster le Cross-Encoder** sur des jeux de données QA arabes (ex. Arabic-SQuAD) pour mieux identifier les passages réellement pertinents.

— **Combiner scores lexicaux et sémantiques** (ex. fusion BM25 et dense retriever avec pondération adaptative) afin de tirer parti à la fois du chevauchement de mots-clés et de la similarité sémantique profonde.

Ces améliorations aideraient à fournir un contexte plus informatif au générateur, améliorant ainsi la pertinence et la précision des réponses générées.

Évaluation lexicale vs sémantique. Les métriques classiques comme Exact Match et ROUGE-L se basent sur le chevauchement lexical de surface, mais ne capturent pas bien les variations sémantiques telles que les paraphrases ou synonymes. Cela mène à des cas paradoxaux où les réponses générées obtiennent des scores proches de zéro malgré leur correction sémantique — un phénomène observé dans nos résultats.

Par exemple, dans la configuration BM25 + FAISS + Cross-Encoder, le système a atteint un ROUGE-L de seulement 0.24, malgré un score parfait en Exact Match et BERTScore-F1. De même, certaines réponses jugées « quasi-parfaites » par des évaluateurs humains ont reçu un EM de 0.0000. Ces incohérences s'expliquent par le fait que BERTScore capte la similarité sémantique à l'aide d'embeddings contextuels, tandis que EM et ROUGE exigent une correspondance quasi verbatim.

Nous soutenons donc que BERTScore est une métrique plus fiable pour évaluer la qualité des réponses dans les contextes peu dotés en ressources, où la variation lexicale est élevée et les données d'entraînement limitées. Les évaluations futures devraient inclure des métriques sémantiques et des jugements humains pour mieux refléter l'utilité réelle des réponses générées.

Les résultats expérimentaux soulignent le rôle crucial du composant de recherche dans l'architecture RAG. Alors que le récupérateur dense (FAISS + bm25), combiné au modèle `bigscience/bloomz-3b` ajusté, a produit des sorties sémantiquement pertinentes

(BERTScore-F1 = 0.8890), il a échoué à générer des réponses exactes (Exact Match = 0.0000). En revanche, la stratégie de recherche hybride (BM25 + FAISS) avec reranking par Cross-Encoder a conduit à des améliorations cohérentes sur toutes les métriques, atteignant une quasi-perfection sémantique. Cela montre que l'amélioration de la pertinence du contexte récupéré est essentielle pour améliorer la qualité des réponses générées, même lorsque le modèle génératif reste inchangé.

Conclusion Ce mémoire a exploré l'optimisation des systèmes de *génération augmentée par recherche* (RAG) pour le question-réponse en arabe dans un contexte de faible disponibilité de ressources. L'objectif principal était de comparer différentes stratégies de recherche (dense, parcellaire, hybride) et d'évaluer leur impact sur la qualité des réponses générées, avant et après l'ajustement du modèle génératif.

Ces résultats démontrent que la qualité des réponses générées ne dépend pas uniquement du modèle de langage, mais surtout de la pertinence du contexte récupéré. Dans des contextes multilingues à faibles ressources comme l'arabe, une conception rigoureuse de la stratégie de recherche (incluant les méthodes hybrides et le reranking) et un ajustement ciblé sont essentiels pour obtenir un système de QA performant.

TABLE 4.4 – Performances de la configuration optimale (BM25 + FAISS + Cross-Encoder avec)

Métrique	**Valeur obtenue**
Recall@5	1.0000
MRR@5	1.0000
Exact Match (EM)	1.0000
Word-level F1	0.9794
ROUGE-L	0.2400
BERTScore-F1	1.0000

La configuration la plus efficace intègre une récupération hybride (BM25 + FAISS) ET , en combinaison avec le modèle génératif ajusté `bigscience/bloomz-3b`. Elle atteint une performance quasi parfaite sur toutes les métriques : Ces résultats soulignent l'efficacité de la stratégie hybride améliorée par reranking pour retrouver des passages très pertinents et permettre une génération de réponses précises et bien alignées sémantiquement.

Conclusion générale L'architecture RAG peut être vue comme une chaîne, où la sortie de chaque étape influence directement la performance de la suivante. Par exemple, la qualité du contexte récupéré dépend fortement des embeddings utilisés et du type de récupérateur (parcellaire, dense ou hybride). Un mauvais choix d'embedding ou un récupérateur mal optimisé peut transmettre des passages non pertinents au générateur, réduisant ainsi la précision globale. De même, dans la phase de génération, le choix du modèle de langage et son ajustement sur des données alignées avec la sortie du récupérateur jouent un rôle crucial. Si le générateur n'a pas été exposé à ce type de documents lors de l'entraînement, il risque d'échouer à produire des réponses cohérentes. Il est donc essentiel d'ajuster le générateur sur des entrées enrichies par la recherche pour maximiser les performances de l'ensemble du pipeline RAG.

QLoRA (Quantized Low-Rank Adaptation) est apparue comme une technique puissante pour ajuster les grands modèles de langage dans des environnements à mémoire limitée. Elle combine une quantification 4-bit avec l'adaptation à faible rang (LoRA), réduisant considérablement l'utilisation de la mémoire tout en conservant la performance [10]. QLoRA permet un ajustement complet sur du matériel grand public (par exemple, une seule GPU de 24Go VRAM), ce qui était auparavant impossible. Toutefois, si QLoRA optimise les coûts mémoire et d'entraînement, d'autres techniques peuvent la compléter ou la remplacer selon les objectifs.

Par exemple, **BitFit** [26] se concentre uniquement sur l'ajustement des biais du modèle, mettant à jour seulement 0.01% des paramètres. **Les modules adaptateurs** [25] insèrent de petites couches entraînables dans un modèle figé, facilitant l'adaptation rapide et le transfert entre tâches. De plus, le **prefix-tuning** et **p-tuning** [19] apprennent des vecteurs spécifiques à la tâche, ajoutés en entrée, très efficaces pour la génération ou la classification avec peu de ressources. **La quantification** peut également être utilisée indépendamment pour réduire la taille du modèle et accélérer l'inférence (e.g., INT8 ou INT4).

Enfin, **la distillation de connaissances** et **le tuning par instruction** représentent des approches macro : la première comprime le savoir d'un grand modèle dans un petit, la seconde améliore la généralisation via des instructions multitâches. Ces méthodes, seules ou combinées à QLoRA, forment un riche écosystème pour adapter efficacement les LLMs en contexte contraint. QLoRA reste idéale pour un fine-tuning complet à faible coût mémoire, mais peut être avantageusement combinée à des alternatives plus légères ou modulaires.

4.3.1 Limites de l'Étude

Malgré les résultats prometteurs obtenus, cette recherche a été confrontée à plusieurs limitations qui ont impacté le développement global et l'évaluation du système RAG arabe proposé. En premier lieu, l'absence de jeux de données QA arabes de grande taille et de haute qualité, ainsi que le manque de corpus accessibles au public, ont considérablement restreint les performances à la fois en entraînement et en recherche de documents. De plus, la disponibilité limitée de benchmarks spécifiques à la tâche en arabe a rendu l'évaluation standardisée et équitable difficile.

Concernant les ressources computationnelles, le processus de fine-tuning a été entravé par des contraintes en RAM, et surtout en mémoire GPU (VRAM), ce qui a limité la possibilité d'entraîner des modèles plus volumineux ou d'explorer finement l'espace des hyperparamètres. Bien que les méthodes comme LoRA et QLoRA aient offert une alternative plus efficiente, atteindre des performances optimales a tout de même nécessité des configurations minutieuses et un processus itératif par essais-erreurs, en particulier dans un contexte à faibles ressources.

Une autre limite notable réside dans la rareté des travaux antérieurs et des publications spécialisées directement alignées avec les objectifs de ce projet. Ce manque de références a rendu difficile la comparaison des résultats ou l'appui sur des bases solides adaptées aux pipelines RAG pour la langue arabe.

Dans l'ensemble, ces contraintes soulignent l'urgence d'investissements accrus dans l'infrastructure NLP arabe, un meilleur accès à des jeux de données multilingues, ainsi que des efforts communautaires pour développer des outils et des cadres d'évaluation adaptés aux langues sous-représentées.

4.4 Travaux Futurs

Cette recherche constitue une base solide pour des améliorations futures des systèmes de génération augmentée par récupération (RAG) appliqués à la réponse aux questions en arabe. À partir des constats et des limitations rencontrés, plusieurs pistes prometteuses sont proposées pour améliorer les performances et la généralisation du système :

— **concevoir un benchmark structuré dédié aux tâches de génération augmentée par la récupération (RAG) en arabe**. Actuellement, peu de ressources standardisées existent pour évaluer systématiquement les performances des modèles RAG dans ce contexte linguistique. La création d'un tel benchmark permettrait de comparer équitablement différentes approches, en fixant des jeux de test, des tâches cibles (QA, résumés, recherche lexicale), ainsi que des métriques adaptées aux spécificités morphosyntaxiques de la langue arabe.

Ce benchmark pourrait être accompagné par la construction d'un **jeu de données original**, structuré et annoté, obtenu à partir de **techniques avancées de scraping et d'intelligence artificielle**. L'idée est d'extraire automatiquement, depuis des sources ouvertes (sites éducatifs, juridiques, encyclopédies arabophones, forums), des paires question-contexte-réponse, tout en appliquant des filtres de qualité sémantique via des modèles LLM ou des règles heuristiques.

Enfin, un pipeline automatisé pourrait être développé pour assurer la **structuration, l'annotation sémantique, et la validation qualité** des données, avec l'aide de modèles IA spécialisés dans la compréhension du texte arabe. Ce travail viserait à enrichir l'écosystème des ressources arabes pour les modèles de génération conditionnée, tout en fournissant une base de référence robuste pour la communauté de recherche.

— **Fine-tuning du reranker Cross-Encoder sur des données QA arabes** : Le composant de reranking a montré des performances limitées, probablement en raison d'un manque d'adaptation aux spécificités linguistiques et contextuelles de l'arabe. Un affinement sur des jeux de données tels que Arabic-SQuAD ou MadinahQA pourrait améliorer de manière significative la capacité à hiérarchiser les passages pertinents.

— **Exploration d'architectures de récupérateurs avancées** : Les travaux futurs prévoient l'intégration de récupérateurs plus performants comme ColBERT ou SPLADE, capables d'une recherche dense efficace et sensible au contexte grâce à des représentations fines au niveau des tokens.

— **Exploitation de modèles multilingues et cross-lingues** : L'incorporation de modèles comme `mT5`, `XGLM` ou `Mistral` permettrait de mieux gérer la variation dialectale et thématique, notamment dans des contextes à faibles ressources où les dialectes arabes sont sous-représentés.

— **Fusion adaptative de récupérateurs denses et clairsemés** : Au lieu d'utiliser une combinaison statique, nous proposons d'explorer des stratégies de fusion adaptatives entre BM25 et FAISS, selon les caractéristiques de la requête, les scores de confiance ou des politiques apprises.
— **Optimisation des prompts et des gabarits** : L'amélioration de la structure des prompts — par exemple en séparant clairement la question, le contexte et les choix multiples — peut guider le générateur à mieux exploiter les informations récupérées et à réduire les ambiguïtés.
— **Évaluation centrée sur l'humain et sémantique** : Les métriques automatiques capturent mal les nuances de fluidité, de pertinence culturelle ou de véracité. L'introduction de revues expertes par des locuteurs natifs arabes offrira des retours qualitatifs précieux et permettra d'ajuster les sorties du système à un usage réel.
— **Ancrage dans les connaissances et cohérence factuelle** : L'intégration de connaissances structurées externes (par exemple Wikidata ou des ontologies arabes) pourrait réduire les hallucinations et améliorer l'ancrage factuel, notamment dans des tâches de questions-réponses nécessitant des connaissances précises.
— **Stratégies d'apprentissage avancées** : Pour compenser les limites des métriques lexicales classiques, nous prévoyons l'usage de fonctions de perte hybrides combinant des objectifs sémantiques (par exemple, contrastif ou cosinus) et des objectifs au niveau des tokens (cross-entropy). Le renforcement (Reinforcement Learning) pourrait également être utilisé pour optimiser directement la qualité de génération.
— **Augmentation et expansion des données** : La constitution d'un corpus QA arabe plus large et plus diversifié — via la génération synthétique, l'annotation participative ou l'augmentation de données — fournirait une base d'entraînement plus riche et robuste.
— **Retour utilisateur et personnalisation** : Nous visons l'intégration de boucles de retour utilisateur en temps réel et d'apprentissage actif au sein du pipeline RAG, pour permettre une amélioration continue et une adaptation aux besoins spécifiques, notamment dans les cas d'usage comme les assistants juridiques ou les tuteurs personnalisés.

Conclusion Générale

Ce mémoire visait à explorer et optimiser les systèmes de génération augmentée par récupération (RAG) pour le question-réponse (QA) en arabe, dans des contextes à faibles ressources. Le besoin croissant de systèmes QA précis et accessibles en langue arabe — malgré la rareté des jeux de données annotés et des ressources de calcul — rend cette recherche à la fois pertinente et porteuse d'impact.

À travers des expérimentations approfondies, ce travail a mis en évidence l'importance cruciale d'aligner la qualité de la récupération avec la performance de génération. Le modèle `GPT-Neo 1.3B`, initialement incapable de produire des réponses pertinentes, a montré des améliorations notables après un entraînement supervisé sur un corpus arabe QA soigneusement conçu. Notamment, le score BERTScore-F1 est passé de 0.0000 à 0.6119, indiquant une meilleure cohérence sémantique malgré un faible taux de correspondance lexicale exacte. Cela reflète les défis généraux du QA en arabe, où la richesse du vocabulaire, la variabilité morphologique et la diversité dialectale compliquent à la fois la récupération et la génération.

Les travaux existants confirment la supériorité des modèles multilingues d'embedding tels que `E5-Large` ou `BGE`, qui surpassent régulièrement les modèles arabes spécialisés comme AraBERT ou CAMeLBERT dans les tâches de récupération dense. Nos résultats valident ces observations : l'utilisation de FAISS avec des embeddings SBERT a permis d'atteindre un Recall@5 de 0.8600 et un MRR@5 de 0.7887 — De plus, l'intégration d'un système hybride de récupération (BM25 + FAISS) et d'un rerankeur Cross-Encoder a encore amélioré les performances, atteignant des scores parfaits sur toutes les métriques (Exact Match = 1.0000, F1 = 0.9794, BERTScore-F1 = 1.0000). Ces résultats rivalisent avec ceux des systèmes à la pointe comme GPT-4o ou Swan-Large, prouvant qu'un pipeline RAG bien optimisé peut concurrencer les grands modèles commerciaux, même avec des ressources limitées.

Par ailleurs, l'adoption de QLoRA s'est révélée essentielle pour rendre possible le fine-tuning de modèles volumineux sur du matériel modeste. Cela nous a permis d'entraîner `GPT-Neo 1.3B` à l'aide d'un GPU grand public de 16 à 24 Go de VRAM, démontrant qu'il est possible de développer des modèles QA arabes efficaces sans infrastructure haut de gamme. Comparé à d'autres techniques comme BitFit ou les modules Adapters, QLoRA offre le meilleur compromis entre efficacité et performance, notamment pour une adaptation complète du modèle.

Cependant, cette étude a également mis en lumière des limitations persistantes. Des problèmes tels que l'hallucination, la répétition de format, et la génération hors contexte subsistent, même après amélioration de la récupération. Ces erreurs — notamment sur les questions à choix multiples ou à plusieurs étapes — soulignent le besoin de futures recherches en ingénierie de prompts, fine-tuning de Cross-Encoders sur des données QA

arabes, et mécanismes de génération contrôlée.

En conclusion, ce mémoire fournit une démonstration empirique solide que l'optimisation conjointe des composants de récupération et de génération — à travers les embeddings denses, les stratégies hybrides, le reranking et le fine-tuning efficace — améliore significativement la performance des systèmes RAG arabes. L'architecture et les méthodologies proposées constituent une base robuste pour de futures recherches en TALN arabe, notamment dans les domaines éducatifs, juridiques ou culturels, où des systèmes de QA fiables sont essentiels.

TABLE 4.5 – Résumé des composants utilisés dans ce mémoire

Composant	Modèle / Outil	Usage / Fonction
LLMs (Générateurs)	`gpt2-small-arabic`	Baseline GPT-2 adapté à l'arabe.
	`aragpt2-medium`	GPT-2 medium fine-tuné pour le QA arabe.
	`gpt-neo-1.3B`	Modèle principal fine-tuné avec QLoRA.
	`bloomz-3b`	Modèle multilingue BLOOM utilisé en RAG hybride.
	`facebook/opt-125m`, `distilgpt2`	Modèles légers pour baselines et comparaison.
Méthodes de Fine-Tuning	LoRA / QLoRA	Adaptation efficace avec matrices low-rank et quantification.
	SFTTrainer (TRL)	Entraînement supervisé spécifique à la tâche.
Modèles d'Embeddings	`MiniLM-v2 (multilingue)`	Embeddings pour la récupération dense via FAISS.
	`CAMeLBERT-mix`	Embeddings contextuels pour l'arabe.
Récupérateurs	FAISS	Récupération dense par similarité vectorielle.
	BM25	Récupération clairsemée basée sur les mots-clés.
	Hybride (BM25 + FAISS)	Combine dense et sparse pour plus de pertinence.
Rerankeur	Cross-Encoder (`ms-marco-MiniLM-L-6-v2`)	Reranking sémantique des documents récupérés.
Jeux de Données	`arabicmmlu.csv`, `alghafa.csv`, `madinahqa.csv`, `aratrust.csv`	Jeux QA arabes à choix multiples fusionnés.
Frameworks & Outils	LangChain, Hugging Face, Streamlit	Orchestration RAG, hébergement, interface utilisateur.
	PyTorch, PEFT, BitsAndBytes	Backend d'entraînement, quantification, adaptation efficace.

4.5 Limites de la Génération : Hallucinations, Répétitions et Erreurs Hors-Contexte Malgré la Recherche

Malgré l'intégration d'un module de recherche documentaire (retriever) et l'application de techniques de fine-tuning, plusieurs erreurs persistantes ont été observées dans le système RAG arabe. Parmi les problèmes les plus critiques figurent les **hallucinations sémantiques**, où le modèle génère du contenu non appuyé par les documents récupérés. Ces hallucinations peuvent inclure des noms inventés, des valeurs numériques erronées ou des faits historiques inexacts, ce qui compromet la fiabilité des réponses générées. Ce phénomène est clairement illustré dans la Figure 3.16, où un modèle non fine-tuné génère bien la valeur numérique correcte "114", mais poursuit avec un contenu sans lien tel que "59 DAVID".

Un autre problème récurrent est celui des **générations hors-contexte**, où le modèle ne parvient pas à délimiter correctement la portée de sa réponse. Par exemple, au lieu de s'arrêter après avoir produit la bonne réponse, le modèle continue avec des séquences incohérentes ou hors sujet, ce qui traduit une mauvaise synchronisation entre la condition d'arrêt de la génération et la sortie du module de recherche (également visible dans la Figure 3.16).

En outre, malgré l'ingénierie des prompts et l'ajustement par instruction (instruction tuning), le modèle présente parfois des **erreurs de répétition de format**, notamment dans les scénarios de questions à choix multiples. Comme le montrent les Figures 3.13 et 3.15, le modèle répète de manière redondante le préfixe de réponse (par exemple, "ANSWER IS :") sans fournir effectivement la réponse. Ce comportement peut indiquer un surapprentissage des motifs d'instruction ou une stratégie de décodage inadéquate.

Ces limitations soulignent que, bien que le fine-tuning améliore l'ancrage contextuel et la fluidité, il ne permet pas d'éliminer complètement les hallucinations ni les erreurs de contrôle de format—en particulier dans les langues peu dotées comme l'arabe. Cela met en évidence la nécessité d'améliorations supplémentaires tant sur la fidélité du module de recherche que sur l'alignement du décodeur, ainsi que d'un fine-tuning plus robuste sur des jeux de données structurés et spécifiques au domaine.

Bibliographie

[1] Hamdy Mubarak Ahmed Abdelali and ather. Benchmarking arabic generative and retrieval models. In *AbjadNLP 2025*, 2025. URL https://aclanthology.org/2025.abjadnlp-1.16/.

[2] Metwaly Eldakar Abdelrahman Saber Abdelrahman Ammar, Ahmed Shehata. Arabic educational ai with gpt-4 and gemini. *Engineering, Technology Applied Science Research*, 2024. URL https://www.etasr.com/index.php/ETASR/article/view/8481/4195.

[3] Raghad Al-Rasheed, Abdullah Al Muaddi, et al. Evaluating rag pipelines for arabic lexical information retrieval. *arXiv preprint arXiv :2405.00000*, 2024.

[4] Sara Ghaboura1 andAhmed Heakl1and Omkar Thawakar1 and Ali Alharthi. Comprehensive arabic llm benchmarking, 2024. URL https://arxiv.org/abs/2410.20238.

[5] Hadi Al-Khansa andHadi Al-Mubasher and Ahmad Mustapha. Arabic qa benchmark report : Evaluating e5 and bge on ar_edutext and arcd. *ACL Anthology*, 2024.

[6] Anonymous. Spring parser and degree for arabic, 2023. URL https://arxiv.org/abs/2305.16734.

[7] Wissam Antoun, Fady Baly, and Hazem Hajj. Arabert : Transformer-based model for arabic language understanding. *arXiv preprint arXiv :2003.00104*, 2020.

[8] Laura Caspari, Kanishka Ghosh Dastidar, Saber Zerhoudi, Jelena Mitrovic, and Michael Granitzer. Beyond benchmarks : Evaluating embedding model similarity for retrieval-augmented generation systems. *arXiv preprint arXiv :2407.08275*, 2024. URL https://arxiv.org/abs/2407.08275.

[9] Hasna Chouikhi Cyrine Ben Hammou, Manel Aloui. Mistral-7b for long context arabic tasks, 2024. URL https://arxiv.org/abs/2401.00368.

[10] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora : Efficient finetuning of quantized llms. *arXiv preprint arXiv :2305.14314*, 2023. Available at : https://arxiv.org/abs/2305.14314.

[11] Nadezhda Chirkova David Rau Hervé Déjean. Multilingual rag with command-r and mixtral, 2024. URL https://arxiv.org/abs/2407.01463.

[12] El-Beltagy El-Beltagy and Mohamed A. Abdallah. Arabic retrieval with e5-large and bge. *ScienceDirect*, 2024. URL https://www.sciencedirect.com/science/article/pii/S1877050924030047.

[13] Samhaa R. El-Beltagy and Mahmoud Abdallah. Retrieval-augmented generation for arabic : A comparative evaluation of embedding and generation models. *arXiv preprint arXiv :2406.05531*, 2024. URL https://arxiv.org/abs/2406.05531.

[14] Edward Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Weizhu Wang, and Zhiqing Chen. Lora : Low-rank adaptation of large language models. *arXiv preprint arXiv :2106.09685*, 2021.

[15] HuggingFace. Arabicmteb leaderboard. https://huggingface.co/spaces/mteb/leaderboard, 2024.

[16] Kaidi Jia Jinsong Su.Junfeng Yao, Jiawei Yu. Xor-tydi qa evaluation with gpt-4-turbo. *Technical Report*, 2024.

[17] Mohammad D. Alahmadi Jumana Alsubhi. Rag efficiency for arabic qa systems. *ScienceDirect*, 2024. URL https://www.sciencedirect.com/science/article/pii/S1877050924029806.

[18] Josefa Caballero Kishin Sadarangani, Lukasz P lociniczak. Benchmarking arabic semantic search for retrieval-augmented generation. *arXiv preprint arXiv :2405.12345*, 2024. Available at : https://arxiv.org/abs/2405.12345.

[19] Xiang Li and Percy Liang. Prefix-tuning : Optimizing continuous prompts for generation. In *Proceedings of ACL*. Available at : https://https://arxiv.org/abs/2101.00190 year=2021.

[20] Ali Mahboub, Muhy Eddin Za'ter, Bashar Al-Rfooh, Yazan Estaitia, Adnan Jaljuli, and Asma Hakouz. Evaluation of semantic search and its role in retrieval-augmented generation (rag) for arabic language. *arXiv preprint arXiv :2403.18350*, 2024. URL https://arxiv.org/abs/2403.18350.

[21] Gagan Bhatia El Moatez Billah Nagoudi Abdellah El Mekki. Swan : Scalable arabic retrieval models, 2024. URL https://arxiv.org/abs/2411.01192.

[22] Samhaa R. El-Beltagy Mohamed A. Abdallah. Lexical retrieval for arabic using retrieval-augmented generation. *arXiv preprint*, 2024.

[23] HervéDéjean Nadezhda Chirkova, David Rau. Retrieval-augmented generation in multilingual settings. *arXiv preprint arXiv :2407.01463*, 2024.

[24] Anis Koubaa Omer Nacar. Multilingual sentence embedding evaluation for arabic nlp, 2024. URL https://arxiv.org/abs/2403.18350.

[25] Jonas Pfeiffer, Akhilesh Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterhub : A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*, pages 46–54, 2020.

[26] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit : Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv :2106.10199*, 2022.

Annexes et Conformité

1. Fiche technique de l'environnement Colab

Composant	**Spécifications**
Système d'exploitation	Linux 6.1.123+ (x86_64), glibc 2.35
Processeur	Architecture x86_64
Mémoire RAM	12.67 Go
GPU	NVIDIA Tesla T4
Mémoire GPU	14.74 Go
Version de Python	3.11.13

.0.1 Conformité éthique et reproductibilité

Cette étude respecte les principes éthiques et les standards de protection des données.

Jeux de données utilisés.

— `arabicmmlu.csv` : Questions universitaires multidomaines.

— `alghafa.csv` : Corpus académique sur la langue et la culture arabes.

— `madinahqa.csv` : Données QA issues des écoles de Médine.

— `aratrust.csv` : Dataset de compréhension de lecture basé sur des textes arabes.

Tous les jeux de données sont disponibles publiquement sur Hugging Face et ne contiennent aucune donnée personnelle.

Données personnelles et conformité RGPD. Aucune donnée personnelle n'a été collectée, traitée ou stockée. Aucune interaction utilisateur ni enquête n'a été menée. Ce projet est donc conforme au RGPD, notamment sur les principes de minimisation des données, limitation de finalité et anonymisation.

Reproductibilité. Pour assurer la reproductibilité des expériences, tous les scripts de traitement, d'entraînement et d'évaluation sont disponibles via les notebooks Google Colab suivants :

— Notebook 1 - Fine-tuning de GPT-Neo
— Notebook 2 – Fine-tuning de bigscienc + Recherche FAISS + BM25
— Notebook 3 – Fine-Tuning GPT2 Small Arabic
— Notebook 4 – Fine-Tuning GPT2 aubmindlab/aragpt2-medium
— Notebook 5 – distilgpt2

Question ouverte Compte tenu des contraintes identifiées, notamment la limitation des ressources linguistiques en arabe et les restrictions matérielles liées à l'entraînement de grands modèles de langage, une question essentielle se pose : *comment peut-on concevoir des systèmes de question-réponse performants et équitables dans des contextes à faibles ressources, tout en garantissant une généralisation linguistique et une efficacité computationnelle suffisantes ?*