Vers une optimisation de RAG en français : conception d'un reranker open source, fine-tuning et évaluation

Ying Zhang[1], Matthieu Petit Guillaume[1], Aurélien Krauth[1]

[1] Leviatan, 725 Bd Robert Barrier, 73100 Aix-les-Bains

{y.zhang, matthieu, aurelien}@leviatan.fr

Résumé

Dans les systèmes RAG, la qualité des documents extraits conditionne fortement la pertinence des réponses générées. Nous proposons un reranker français open source, basé sur un modèle de type Cross-Encoder, afin de réordonner les documents candidats et ainsi améliorer la cohérence du texte généré. Nous détaillons la construction d'un jeu de données combinant un corpus de similarité sémantique et des données de questions-réponses en français, puis comparons plusieurs stratégies de fine-tuning (ajustement fin). Les résultats montrent qu'un modèle open source optimisé peut rivaliser avec des solutions commerciales, à condition de bien choisir le modèle pré-entraîné et d'apporter une attention particulière à la génération d'exemples négatifs. Nous proposons enfin des pistes d'amélioration pour renforcer davantage la performance globale des systèmes RAG.

Mots-clés

RAG (Retrieval-Augmented Generation), Reranker, Retriever (module de recherche), Cross-Encoder, Fine-tuning, Questions-Réponses, Open Source, Traitement automatique du français

Abstract

In Retrieval-Augmented Generation (RAG) systems, the quality of retrieved documents is crucial for producing accurate and coherent answers. We introduce an open-source French reranker based on a Cross-Encoder architecture, designed to reorder candidate documents and enhance the coherence of the generated text. We describe the creation of a dataset that integrates semantic similarity corpora with French question-answering data and compare several fine-tuning strategies. Our results demonstrate that a carefully optimized open-source model can rival commercial rerankers, provided there is thoughtful selection of the pretrained model and careful design of negative examples. We conclude by suggesting improvements to further enhance overall RAG performance.

Keywords

RAG (Retrieval-Augmented Generation), Reranker, Retriever, Cross-Encoder, Fine-tuning (ajustement fin), Question Answering, Open Source, French NLP

1 Introduction

Dans les systèmes de génération augmentée par la recherche (Retrieval-Augmented Generation, RAG), la qualité des documents extraits par le module de recherche (Retriever) est un facteur déterminant pour la fiabilité et la pertinence des réponses produites. Les techniques de recherche traditionnelles, bien qu'efficaces pour un large éventail d'applications, peuvent toutefois introduire du bruit informationnel en sélectionnant des documents moins pertinents [3, 13]. Cette limitation se répercute sur la qualité du texte généré, qui s'appuie sur ces documents comme source d'information.

Pour pallier ce problème, l'intégration d'un reranker se révèle cruciale. Ce composant permet de réordonner et de prioriser les documents déjà extraits afin de mettre en avant les plus pertinents et les plus fiables. De cette façon, le générateur de texte dispose d'un contexte documenté mieux ciblé et plus pertinent, ce qui améliore la cohérence et la qualité globale de la réponse finale.

Dans cet article, nous mettons en évidence le rôle central joué par le reranker dans un système RAG. Nous montrons comment ce module peut être affiné (micro-ajusté) à l'aide de données privées pour renforcer davantage la pertinence du réordonnancement. Nous décrivons la préparation du jeu de données ainsi que l'expérimentation de micro-ajustement menée dans le cadre de plusieurs configurations expérimentales. Nous comparons ensuite nos résultats à ceux de solutions commerciales et open source dans un benchmark, mettant en évidence des performances contrastées : certaines configurations s'avérant très efficaces, d'autres moins. Nous analysons ces résultats et proposons des pistes d'optimisation.

Enfin, il convient de souligner que les modèles, les données d'entraînement, et les procédures de fine-tuning décrits dans cette étude sont entièrement accessibles en open source[1], afin de promouvoir la transparence et de faciliter l'adoption de ces travaux par la communauté.

1. Le projet est disponible à l'adresse suivante : https://github.com/LeviatanAI/reranker-cross-encoder, et les modèles ainsi que les jeux de données sont accessibles à l'adresse : https://huggingface.co/LeviatanAIResearch.

2 Contexte et problématique

La technique de RAG combine un module de recherche (Retriever) et un générateur de texte (par exemple GPT [18] ou Llama [16]) afin d'améliorer la pertinence et l'exactitude des réponses produites. Contrairement aux modèles purement pré-entraînés qui se reposent exclusivement sur leurs connaissances internes, un système RAG s'appuie également sur une base de connaissances ou un ensemble de documents externes (base documentaire, corpus textuel, etc.), permettant ainsi d'offrir des réponses plus complètes et régulièrement mises à jour.

Dans sa forme la plus élémentaire, un système RAG comporte deux composantes principales :

1. Le Retriever (module de recherche) : il interroge une base de connaissances ou une base de données pour identifier les documents ou passages les plus proches de la requête de l'utilisateur. Parmi les méthodes de recherche les plus courantes, on retrouve la recherche sémantique par vecteurs (particulièrement adaptée aux textes longs ou non structurés) [11, 22], la recherche par mots-clés (souvent efficace pour des textes courts ou structurés) [23, 24] ou encore la recherche hybride (qui combine mots-clés et représentations vectorielles) [28].

2. Le Générateur : il s'agit généralement d'un grand modèle de langage (par exemple GPT [18] ou Llama [16]) qui, pour construire sa réponse, tient compte à la fois des documents extraits et de ses propres connaissances linguistiques.

L'un des avantages majeurs de RAG réside dans la réduction du phénomène d'« hallucination » — lorsque le modèle invente ou déforme des informations. En consultant des documents externes, le générateur est en mesure de vérifier le contenu avant de l'incorporer dans sa réponse, ce qui augmente la fiabilité du texte produit.

Par ailleurs, RAG offre d'autres avantages importants :

- Flexibilité et mise à jour des connaissances : en séparant les données du modèle lui-même, RAG permet d'adapter facilement les informations accessibles sans nécessiter un nouvel entraînement. Cela facilite l'intégration de nouvelles connaissances et permet l'exploitation de données privées dans un environnement sécurisé, sans les exposer ni les inclure dans l'apprentissage du modèle.
- Meilleure traçabilité : le lien explicite entre un document et la réponse générée renforce la confiance des utilisateurs, qui peuvent vérifier l'origine des informations fournies.

Dans la majorité des scénarios industriels de RAG, les données manipulées sont principalement non structurées (textes libres, documents PDF, pages web, etc.). Une méthode répandue pour en extraire des passages pertinents repose sur la recherche sémantique vectorielle, où la requête de l'utilisateur est convertie en une représentation vectorielle pour identifier les passages les plus proches, transmis ensuite au Générateur pour produire la réponse.

L'idée sous-jacente est de projeter à la fois la requête et les documents (ou passages) dans un espace vectoriel de dimension fixe, puis de comparer leurs représentations pour en mesurer la proximité sémantique. La méthode du bi-encoder [22] est couramment utilisée dans ce cadre :

- Un encodeur (souvent basé sur BERT [6], RoBERTa [14] ou des modèles similaires) transforme chaque document ou passage en un vecteur d'embedding. Ce calcul peut être effectué hors ligne et le résultat stocké dans une base de données vectorielle (comme FAISS [7], Milvus [30] ou Elasticsearch [8]).
- Lors de la requête, le même encodeur (ou un encodeur symétrique) génère un vecteur d'embedding représentant la question de l'utilisateur.
- Enfin, une mesure de similarité (produit scalaire ou distance cosinus) est calculée entre le vecteur de requête et les vecteurs stockés. Les documents présentant le score de similarité le plus élevé sont considérés comme les plus pertinents.

Si cette approche offre l'avantage d'un traitement rapide grâce à l'indexation optimisée et d'une bonne couverture sémantique, elle présente toutefois certaines limites :

- Sensibilité aux biais : le modèle, entraîné pour produire des embeddings généraux, peut négliger certaines nuances contextuelles ou linguistiques propres à la requête.
- Scores de pertinence approximatifs : pour des requêtes complexes (plusieurs sous-questions, contraintes de style, etc.), l'utilisation exclusive d'une similarité vectorielle peut demeurer insuffisante pour hiérarchiser correctement les documents.

Pour contourner ces limites et maximiser les chances de fournir au Générateur les documents les plus adéquats, une pratique courante consiste à extraire un grand nombre de candidats au stade du Retriever (bi-encoder). Cependant, l'envoi d'un volume important de documents au Générateur pose deux problèmes majeurs : d'une part, la consommation de ressources de calcul explose ; d'autre part, la qualité du texte généré peut se dégrader.

C'est pour répondre à ces défis qu'intervient le Reranker [17, 22] : en réordonnant les documents sélectionnés par le bi-encoder, il permet d'augmenter la précision globale du système RAG tout en réduisant la charge computationnelle du Générateur, comme illustré à la figure 1. Le module Reranker se place ainsi comme un maillon essentiel dans la chaîne de traitement, garantissant une sélection plus ciblée et plus pertinente des informations destinées à la phase de génération.

3 État de l'art

3.1 Principales approches de reranking

Plusieurs approches de reranking coexistent, chacune présentant des avantages et des inconvénients. Parmi les méthodes les plus courantes, on distingue notamment :

1. Le Cross-Encoder [17] basé sur des Transformers [29]

Le Cross-Encoder constitue aujourd'hui l'une des approches de reranking les plus populaires. Contrairement aux bi-encoders, qui génèrent un embedding séparé pour la requête et pour chaque document, son principe repose sur

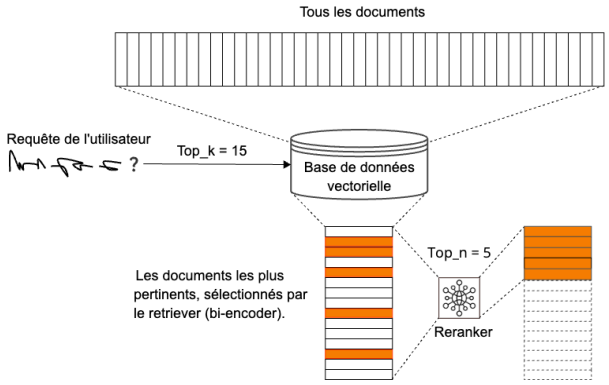

FIGURE 1 – Système de recherche en deux étapes [1]

l'encodage simultané de la requête et du document dans une même séquence en entrée du modèle (par exemple, [CLS] QUESTION [SEP] DOCUMENT [SEP]) afin de produire un score de pertinence. Cette méthode permet de capturer de manière fine les interactions entre le texte de la requête et celui du document, ce qui se traduit souvent par des performances remarquables en termes de classement. Cependant, cette granularité accrue implique un coût computationnel élevé : en effet, chaque paire (REQUÊTE, DOCUMENT) doit être traitée conjointement par le réseau, ce qui peut devenir onéreux lorsque le nombre de candidats à réordonner est important.

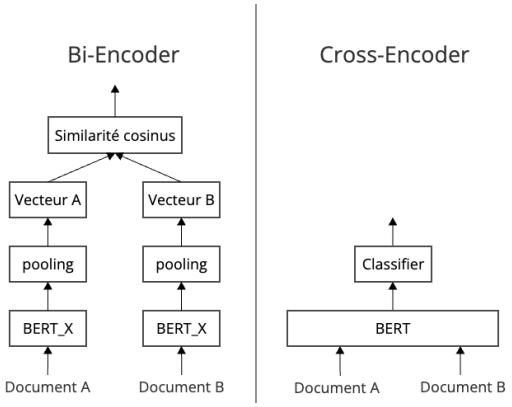

FIGURE 2 – Structure de bi-encoder et de cross-encoder [22]

2. Le Reranker génératif

Dans cette catégorie, on utilise un modèle génératif (par exemple T5 [20], GPT [18] ou d'autres grands modèles de langage) pour évaluer la pertinence ou produire directement un score de classement. Les approches génératives peuvent se subdiviser en deux sous-catégories :

— Reranker basé sur T5 [9]

T5, entraîné à réaliser de multiples tâches de traitement automatique du langage naturel, peut être adapté pour « générer » un score de pertinence ou un label de pertinence. Les formulations dites « prompt-based » permettent d'exploiter la flexibilité du modèle pour réaliser l'inférence des scores de classement. L'avantage de cette approche réside dans sa polyvalence et sa capacité à généraliser sur divers jeux de données. Toutefois, son efficacité dépend de la qualité du prompt et du fine-tuning réalisé, et son déploiement peut s'avérer coûteux en ressources si le modèle est de grande taille.

— Reranker basé sur les LLM [21]

À l'instar de GPT ou d'autres grands modèles de langage, cette variante suit une logique similaire, mais tire parti de la puissance (et parfois des connaissances internalisées) des modèles très volumineux. D'un côté, cela peut conduire à de meilleures performances, notamment pour des questions ambiguës ou nécessitant des connaissances larges. De l'autre, l'important coût de calcul et les contraintes d'accès (modèles propriétaires ou services payants) peuvent limiter l'adoption à grande échelle.

3.2 Comparaison et enjeux

Les Cross-Encoders se distinguent par leur capacité à capturer les interactions requête-document avec un haut niveau de précision, au prix toutefois d'un coût de calcul plus conséquent. Les méthodes génératives, quant à elles, offrent une plus grande flexibilité et peuvent intégrer des signaux sémantiques complexes, mais restent encore très dépendantes de la taille du modèle et du prompt utilisé.

Dans cette étude, nous nous concentrons sur l'approche la plus largement adoptée en pratique : le Cross-Encoder. Notre objectif est de mettre au point un reranker multilingue, facile à déployer, peu coûteux en ressources et disponible en open source. Dans un premier temps, les expériences présentées porteront sur un ensemble de données en langue française, afin d'illustrer la faisabilité et l'efficacité de l'approche proposée dans un contexte concret.

4 Expérimentation

4.1 Composition du jeu de données

4.1.1 Source des jeux de données

Dans le cadre de nos travaux, nous nous sommes appuyés sur deux grandes catégories de données pour entraîner et évaluer notre reranker :

1. Données dédiées à la similarité sémantique

Nous avons utilisé le STS Benchmark multilangue (stsb) [2], un corpus de référence pour l'évaluation de la similarité sémantique entre paires de phrases. Sa structure, composée de paires de textes et de scores de similarité, se prête particulièrement bien à l'apprentissage de modules de réordonnancement (ou reranking).

2. Données de questions-réponses en langue française

Afin de disposer d'un jeu de données riche et varié, nous avons rassemblé quatre sources ouvertes de questions-réponses en français :

— PIAF [12]
— FQuAD [15]
— SQuAD-French [10]
— pandora-s/neural-bridge-rag-dataset-12000-google-translated [19], filtrée pour la langue française (pandora-rag-fr)

L'utilisation conjointe de ces corpus vise à couvrir un large éventail de thématiques et de formulations linguistiques, permettant ainsi de tester la robustesse du reranker à différents styles de questions et à divers types de documents.

4.1.2 Processus général de construction des jeux de données

Pour le jeu de données STS Benchmark multilingue, nous avons d'abord extrait la portion en français, puis normalisé les scores de similarité afin de les contraindre à l'intervalle [0, 1].
Dans le but de garantir une cohérence optimale dans l'entraînement du reranker, un protocole d'agrégation et de transformation a été établi pour le deuxième type de jeu de données (données de questions-réponses), comprenant notamment les étapes suivantes :

1. Chargement et filtrage linguistique

Chaque corpus (PIAF, FQuAD, SQuAD-French, pandora-rag-fr) est d'abord chargé individuellement, afin de repérer et d'extraire les échantillons pertinents (en l'occurrence, uniquement ceux rédigés en français lorsque le corpus présente une composante multilingue).

2. Génération des paires positives

Les entrées constituées d'un contexte et d'une ou plusieurs questions associées forment les exemples « positifs » (LABEL = 1). Chaque couple (CONTEXTE, QUESTION) ainsi validé illustre une correspondance avérée entre la question et l'extrait textuel sélectionné.

3. Création d'exemples négatifs

Pour renforcer la capacité du modèle à discriminer le pertinent de l'impertinent, des paires artificiellement « négatives » (LABEL = 0) sont générées. Concrètement, celles-ci associent une question aléatoire issue d'un autre document ou d'une autre entrée, de sorte qu'elle ne corresponde pas au contexte considéré.

4. Fusion et structuration finale

Les jeux de données issus de chaque corpus sont ensuite fusionnés en un ensemble unique, organisé sous forme de tableaux structurés (champs contexte, question, label et source). Cette étape permet de regrouper de manière homogène les différents types d'exemples (positifs et négatifs).

5. Préparation pour l'entraînement

Afin de faciliter l'intégration dans les pipelines de traitement du langage, nous avons intégré et mis en ligne le jeu de données sur Hugging Face. La compilation des exemples finaux couvre un large spectre de cas où la question est (ou non) liée au contexte fourni. Concernant la répartition des données, nous respectons la division train/test d'origine. Autrement dit, le jeu de données initialement dédié à l'entraînement demeure utilisé pour le train set, et celui destiné aux tests reste inchangé. Seul le jeu de données « PIAF » n'ayant pas de découpage préexistant, nous avons procédé à une répartition séquentielle en affectant les 70 % des données initiales au train set et les 30 % restants au test set.

Grâce à ce processus, un volume conséquent de données étiquetées, composé de paires cohérentes et incohérentes, est mis à la disposition du modèle de reranking. L'objectif est d'entraîner celui-ci à attribuer un score de pertinence élevé aux paires contextes-questions justifiées, et un score faible dans les cas où la question ne correspond pas au contenu textuel.

4.2 Fine-tuning (ajustement fin)

4.2.1 Combinaison des jeux de données et des modèles

Pour l'entraînement de notre reranker, nous nous sommes appuyés sur la méthodologie proposée par l'équipe Sentence Transformers. Celle-ci fournit une classe dédiée CROSSENCODER qui s'appuie en interne sur la classe AUTOMODELFORSEQUENCECLASSIFICATION de Hugging Face, tout en offrant des fonctionnalités simplifiées pour l'entraînement et la prédiction des scores de similarité ou de pertinence [26].
Afin de procéder au fine-tuning, nous avons dû sélectionner un modèle pré-entraîné issu de la famille Transformer compatible avec AUTOMODEL. Pour évaluer l'impact du volume et de la diversité des données sur les performances du reranker, dans la première phase de nos tests, nous avons expérimenté 4 combinaisons de jeux de données et de modèles de base :

- **EXPÉRIENCE 1.** BERT de Google [6] + STS Benchmark en français (stsb-fr)
- **EXPÉRIENCE 2.** DistilRoBERTa-base [25] + (PIAF, FQuAD, SQuAD-french)
- **EXPÉRIENCE 3.** DistilRoBERTa-base + (PIAF, FQuAD, SQuAD-french, pandora-rag-fr)
- **EXPÉRIENCE 4.** DistilRoBERTa-base + (PIAF, FQuAD, SQuAD-french, pandora-rag-fr, stsb-fr)

Cette comparaison vise à mettre en évidence l'influence conjointe de la nature des données (questions-contextes versus paires de phrases corrélées sémantiquement) et du modèle de base (BERT vs. DistilRoBERTa) sur la qualité du réordonnancement.

4.2.2 Configuration du fine-tuning

Tous nos processus de fine-tuning ont été réalisés sur une carte GPU T4 et reposent exclusivement sur l'ensemble d'entraînement. Il est toutefois important de souligner que, contrairement à ces processus, les étapes d'inférence et de benchmark, détaillées par la suite, ont été effectuées sur CPU.
Afin de garantir une comparaison équitable entre les différentes configurations de fine-tuning, nous avons, après une phase de tests préliminaires, adopté une configuration unique pour l'ensemble des entraînements. Nous avons fixé les paramètres d'entraînement suivants, tandis que les autres valeurs ont été maintenues par défaut :

- EPOCH : 4
- TRAIN_BATCH_SIZE : 16
- SAVE_BEST_MODEL : True
- WARMUP_STEPS : 10 % des données d'entraînement

Concernant l'évaluation durant le processus d'entraînement, une distinction importante a dû être faite en fonction de la nature de nos jeux de données. En effet, notre jeu de données

comprend deux types de données :

1. Des scores continus (dans le cas du jeu de données STS Benchmark, normalisé pour produire des valeurs dans l'intervalle [0, 1])
2. Des scores binaires (issus des jeux de données de questions-réponses)

Par conséquent, le choix de l'évaluateur a varié selon les différentes expériences. Pour les expériences 2 et 3, où les jeux de données de fine-tuning ne contenaient pas le STS Benchmark et comportaient exclusivement des étiquettes binaires, l'évaluation a été réalisée en utilisant la précision moyenne et le meilleur score F1 possible. Dans ce contexte, nous avons employé le CEBINARYCLASSIFICATIONEVALUATOR [27].
En revanche, pour les expériences 1 et 4, qui intégraient le jeu de données STS Benchmark, nous avons opté pour le CECORRELATIONEVALUATOR [27], plus adapté à l'évaluation de la corrélation entre les scores prédits et les scores de référence continus.

4.3 Résultats préliminaires des fine-tunings

4.3.1 Comparaison des résultats entre les expériences

Dans cette section, nous proposons une analyse comparative des quatre expériences de fine-tuning réalisées. Pour chaque expérience, les modèles ont été évalués sur plusieurs ensembles de données, à savoir :

1. Les ensembles de développement et de test du STS Benchmark en français (stsb-fr Dev et stsb-fr Test). Voir la Table 1.

Expérience	stsb-fr Dev Set		stsb-fr Test Set	
	Pearson	**Spearman**	**Pearson**	**Spearman**
Exp. 1	0,8722	0,8692	**0,8362**	**0,8245**
Exp. 2	0,4710	0,6119	0,3492	0,4673
Exp. 3	0,5258	0,6990	0,4225	0,5908
Exp. 4	**0,9219**	**0,9187**	0,7565	0,7460

TABLE 1 – Résultats des différentes configurations sur le jeu de données STS Benchmark

2. Trois jeux de données internes élaborés à partir de données prétraitées provenant de PIAF, FQuAD, SQuAD-French, pandora-rag-fr et stsb-fr :
   (a) Un premier jeu intégrant l'ensemble des sources, évalué via la corrélation (Table 2).
   (b) Un second jeu constitué de PIAF, FQuAD et SQuAD-French, évalué à l'aide d'indicateurs de classification binaire (Table 3).
   (c) Un troisième jeu rassemblant PIAF, FQuAD, SQuAD-French et pandora-rag-fr, également évalué à l'aide d'indicateurs de classification binaire (Table 3).

Expérience	Pearson	Spearman
Exp. 1	0,7741	0,7608
Exp. 2	0,8892	0,8372
Exp. 3	0,8985	0,8594
Exp. 4	**0,9522**	**0,8966**

TABLE 2 – Résultats sur l'ensemble combiné des données PIAF, FQuAD, SQuAD-French, pandora-rag-fr et stsb-fr

Exp.	PIAF, FQuAD et SQuAD-Fr				
	Acc.	**F1**	**Prec.**	**Rec.**	**Avg. Prec.**
Exp. 1	0,9527	0,9529	0,9455	0,9603	0,9889
Exp. 2	**0,9765**	**0,9765**	**0,9785**	0,9745	0,9931
Exp. 3	0,9763	0,9762	0,9769	0,9756	**0,9955**
Exp. 4	0,9753	0,9754	0,9720	**0,9788**	0,9954
Exp.	**PIAF, FQuAD, SQuAD-Fr et pandora-rag-fr**				
	Acc.	**F1**	**Prec.**	**Rec.**	**Avg. Prec.**
Exp. 1	0,9468	0,9472	0,9410	0,9534	0,9858
Exp. 2	0,9747	0,9747	0,9778	0,9716	0,9935
Exp. 3	**0,9771**	**0,9770**	**0,9803**	0,9736	**0,9951**
Exp. 4	0,9767	0,9767	0,9791	**0,9743**	0,9952

TABLE 3 – Résultats sur les jeux de données de questions-réponses (Exp. : Expérience, Acc. : Accuracy (exactitude), F1 : F1 score, Prec : Precision, Rec. : Recall, Avg. Prec. : Average Precision)

4.4 Analyse comparative

Les résultats obtenus révèlent plusieurs points intéressants. D'une part, l'impact du modèle de base et des données utilisées est significatif. L'expérience 1, qui utilise le modèle BERT de Google avec uniquement le STS Benchmark, montre de bonnes performances en termes de corrélation sur stsb-fr. Toutefois, ses résultats sur les jeux de données internes restent inférieurs à ceux obtenus avec les modèles basés sur DistilRoBERTa-base. En effet, les expériences 2 à 4, qui se fondent sur DistilRoBERTa-base, affichent une amélioration notable sur les jeux de données internes, notamment en ce qui concerne les métriques de classification (accuracy, F1, etc.). Cependant, l'inclusion ou non de certains jeux de données (notamment pandora-rag-fr et stsb-fr) influence significativement les performances sur stsb-fr.
D'autre part, des disparités apparaissent entre les évaluations sur stsb-fr et les jeux de données internes. Pour l'expérience 2, par exemple, les scores de corrélation sur le stsb-fr (Pearson de 0,4710 en dev et de 0,3492 en test) sont nettement inférieurs à ceux obtenus sur le jeu de données combiné interne (Pearson 0,8892). Cette disparité suggère que le fine-tuning sur des données spécifiques à la tâche (questions-contextes) peut favoriser les performances sur ces derniers, au détriment d'un benchmark généraliste comme stsb-fr. En outre, l'expérience 4 présente les meilleures performances sur le stsb-fr Dev Set (Pearson 0,9219, Spearman 0,9187) et sur le jeu de données combiné interne (Pearson 0,9522, Spearman 0,8966), bien que la performance sur le stsb-fr Test Set soit

moins élevée (Pearson 0,7565, Spearman 0,7460). Ce résultat pourrait être lié à une suradaptation aux jeux de données internes ou à la nature hétérogène du benchmark stsb-fr.
Enfin, l'effet de l'ajout progressif des corpus se révèle pertinent. L'intégration de pandora-rag-fr, comme observé dans l'expérience 3, améliore globalement les performances sur les jeux de données internes par rapport à l'expérience 2, tant en termes de corrélation qu'en classification. Cela indique que la diversité des exemples contribue à la robustesse du reranker. De plus, l'intégration complète – c'est-à-dire l'ajout de stsb-fr en plus de pandora-rag-fr dans l'expérience 4 – permet d'atteindre des scores de corrélation maximaux sur le stsb-fr Dev Set et de maintenir des performances stables en classification, démontrant ainsi l'intérêt de combiner des jeux de données hétérogènes pour couvrir une grande variété de scénarios.

5 Benchmarks de nos modèles, du reranker commercial et du reranker open source

Dans cette section, nous comparons les performances de nos quatre modèles issus des expériences précédentes avec deux rerankers de référence : le reranker commercial COHERE RERANK-MULTILINGUAL-V2.0 [4] et le reranker open source DANGVANTUAN/CROSSENCODER-CAMEMBERT-LARGE [5].
L'évaluation a été réalisée en utilisant deux ensembles de test distincts, issus respectivement des jeux de données FQuAD et PIAF. Dans chaque cas, le système RAG utilise l'ensemble de test comme base documentaire.

5.1 Configuration de l'évaluation

Pour ces deux benchmarks, nous avons employé le modèle d'embedding INTFLOAT/MULTILINGUAL-E5-LARGE de Microsoft [31] en tant que bi-encoder. Le premier filtrage, réalisé par ce bi-encoder, repose sur une mesure de similarité cosinus qui permet de sélectionner les 30 candidats les plus pertinents.
Le benchmark basé sur le jeu de données FQuAD Test comprend un total de 3188 questions, tandis que celui reposant sur le jeu de données PIAF Test en inclut 1151. Dans le cas de PIAF, en l'absence d'une partition initiale, 70 % des données ont été utilisées pour l'entraînement et les 30 % restants pour le test, conformément à la méthode décrite en section 4.1.2. Il est à noter qu'aucune donnée de l'ensemble de test n'a été intégrée durant l'entraînement.

5.2 Résultats obtenus

Les performances ont été évaluées en mesurant le taux de rappel selon la métrique Recall@k, considérée pour trois niveaux de sélection : top 5, top 7 et top 10. Les résultats obtenus pour chacun des deux ensembles de test sont présentés dans la Table 4 et la Table 5.

5.3 Analyse des résultats

Les résultats obtenus permettent de dégager plusieurs observations pertinentes :

1. Performances comparées

Modèle	Top 5	Top 7	Top 10
Cohere Reranker	**92,50%**	**93,48%**	**94,26%**
D.V. CrossEncoder	52,23%	62,33%	71,46%
Expérience 1	**84,54%**	**87,92%**	**90,90%**
Expérience 2	30,14%	38,33%	49,53%
Expérience 3	39,12%	47,33%	56,81%
Expérience 4	**72,49%**	**78,67%**	**84,07%**

TABLE 4 – Benchmark réalisé sur l'ensemble de test de FQuAD (Cohere Reranker : COHERE RERANK-MULTILINGUAL-V2.0, D.V. CrossEncoder : DANGVANTUAN/CROSSENCODER-CAMEMBERT-LARGE)

Modèle	Top 5	Top 7	Top 10
Cohere Reranker	**95,57%**	**96,35%**	**97,22%**
D.V. CrossEncoder	61,77%	69,24%	78,63%
Expérience 1	**94,87%**	**96,00%**	**96,96%**
Expérience 2	38,66%	48,05%	58,99%
Expérience 3	56,73%	64,90%	73,59%
Expérience 4	**90,70%**	**93,83%**	**95,83%**

TABLE 5 – Benchmark réalisé sur l'ensemble de test de PIAF (Cohere Reranker : COHERE RERANK-MULTILINGUAL-V2.0, D.V. CrossEncoder : DANGVANTUAN/CROSSENCODER-CAMEMBERT-LARGE)

Le reranker commercial COHERE RERANK-MULTILINGUAL-V2.0 affiche des scores très élevés sur les deux ensembles de test, atteignant près de 97 % en top 10 sur le jeu de données PIAF. En comparaison, le reranker open source DANGVANTUAN/CROSSENCODER-CAMEMBERT-LARGE obtient des performances nettement inférieures, avec un écart particulièrement prononcé sur le FQuAD test set (71,46 % en top 10 contre 94,26 % pour cohere).

2. Performances des expériences

Parmi nos expériences, l'expérience 1 présente des résultats compétitifs, se rapprochant des performances du modèle commercial sur le jeu de données PIAF et obtenant de bonnes performances sur le jeu de données FQuAD. En revanche, les expériences 2 et 3 montrent des scores faibles, suggérant que la configuration et les données utilisées dans ces tests n'ont pas permis d'exploiter pleinement le potentiel du reranker. L'expérience 4, quant à elle, s'appuie sur la configuration la plus riche en données parmi nos modèles développés, lui permettant d'atteindre des performances proches de celles du modèle commercial, notamment sur le jeu de données PIAF.

5.4 Analyse des erreurs

Les performances relativement faibles observées dans nos expériences s'expliquent principalement par la méthode de construction des exemples négatifs. Dans les données d'origine issues des questions-réponses, un même contexte est associé à une ou plusieurs questions ; chaque couple (QUESTION, CONTEXTE) constitue un exemple positif puisque le contexte contient la réponse. Or, pour créer des exemples négatifs, nous avons associé aléatoirement d'autres questions

au contexte, ce qui ne reflète pas la réalité de la tâche, car ce choix aléatoire produit des couples (QUESTION, CONTEXTE) qui abordent deux sujets complètement différents. En effet, au lieu de déterminer si la réponse se trouve dans le contexte, le modèle apprend à détecter si la question et le contexte traitent du même sujet.

Prenons l'exemple de la question « Quelle dynastie accélère la croissance de Babylone ? » issue du jeu de données FQuAD. Dans l'expérience 3, les top 10 documents ont tous des scores supérieurs à 0,98, car ils proviennent du même article sur Babylone, ne faisant que varier de passage en passage. Cette homogénéité entraîne une similarité thématique généralisée, rendant difficile la distinction du document contenant réellement la réponse.

Ces observations suggèrent qu'il serait pertinent de repenser la stratégie de génération des exemples négatifs, en incluant des cas présentant des similarités thématiques plus fines et des écarts de score plus discriminants, afin de mieux simuler la véritable complexité de la tâche.

6 Conclusion et perspectives

Dans cette étude, nous avons examiné le rôle central du reranker dans les systèmes de génération augmentée par la recherche (RAG). Nos expériences ont permis de comparer différentes configurations de fine-tuning, mettant en exergue l'impact du choix des données et du modèle de base sur la qualité du réordonnancement. Ce benchmark met en évidence la compétitivité des solutions commerciales par rapport aux approches open source et nos propres expériences. Tandis que le reranker commercial cohere offre des performances de très haut niveau, nos expériences, notamment l'expérience 1 et l'expérience 4, démontrent que des configurations optimisées peuvent se rapprocher de ces performances. Ces résultats ouvrent la voie à des travaux futurs visant à affiner davantage nos modèles, notamment en explorant des stratégies de fine-tuning plus robustes et en intégrant des jeux de données complémentaires.

Sur le plan des perspectives, nous envisageons d'ajuster la construction de la partie négative du jeu de données. Pour une question donnée, nous ne nous contenterons pas de sélectionner aléatoirement des documents sur des thématiques différentes, mais nous intégrerons également les autres segments de l'article ne contenant pas la réponse afin de constituer des exemples négatifs plus représentatifs. Une fois ce nouveau jeu de données finalisé, nous réitérerons l'ensemble des analyses et élargirons l'évaluation à d'autres modèles de type BERT base.

À long terme, nous souhaitons également construire un reranker multilingue open source, capable de fonctionner sur CPU, et dont les performances seraient comparables à celles des solutions commerciales.

Références

[1] James Briggs. Rerankers and two-stage retrieval. https://www.pinecone.io/learn/series/rag/rerankers/. In : *Retrieval Augmented Generation*, Accessed : 2025-03-05.

[2] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1 : Semantic textual similarity multilingual and crosslingual focused evaluation. In Steven Bethard, Marine Carpuat, Marianna Apidianaki, Saif M. Mohammad, Daniel Cer, and David Jurgens, editors, *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. https://aclanthology.org/S17-2001/.

[3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1870–1879, Vancouver, Canada, July 2017. Association for Computational Linguistics. https://aclanthology.org/P17-1171/.

[4] Cohere. Improve search performance with a single line of code. https://cohere.com/rerank, 2025. Accessed : 2025-03-05.

[5] Van Tuan DANG. dangvantuan/crossencoder-camembert-large. https://huggingface.co/dangvantuan/CrossEncoder-camembert-large, 2022. Accessed : 2025-03-05.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT : pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. http://arxiv.org/abs/1810.04805.

[7] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024. https://arxiv.org/abs/2401.08281.

[8] Elastic. Elasticsearch : The official distributed search & analytics engine for all types of data. https://www.elastic.co/elasticsearch/, 2021. Accessed : 2025-03-05.

[9] Kai Hui, Tao Chen, Zhen Qin, Honglei Zhuang, Fernando Diaz, Mike Bendersky, and Don Metzler. Retrieval augmentation for t5 re-ranker using external sources. https://arxiv.org/abs/2210.05145, 2022.

[10] Ali Kabbadj. French-squad : French machine reading for question answering. https://github.com/Alikabbadj/French-SQuAD, 2019. Accessed : 2025-03-05.

[11] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906, 2020. https://arxiv.org/abs/2004.04906.

[12] Rachel Keraron, Guillaume Lancrenon, Mathilde Bras, Frédéric Allary, Gilles Moyse, Thomas Scialom, Edmundo-Pavel Soriano-Morales, and Jacopo

Staiano. Project piaf : Building a native french question-answering dataset. https://arxiv.org/abs/2007.00968, 2020.

[13] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. *CoRR*, abs/2005.11401, 2020. https://arxiv.org/abs/2005.11401.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta : A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. http://arxiv.org/abs/1907.11692.

[15] d'Hoffschmidt Martin, Vidal Maxime, Belblidia Wacim, and Brendlé Tom. FQuAD : French Question Answering Dataset. *arXiv e-prints*, page arXiv :2002.06071, Feb 2020. https://arxiv.org/abs/2002.06071.

[16] Meta. Llama 3.2 : Revolutionizing edge ai and vision with open, customizable models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/, September 2024. Accessed : 2025-02-21.

[17] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019. http://arxiv.org/abs/1901.04085.

[18] OpenAI. Openai platform models. https://platform.openai.com/docs/models. Accessed : 2025-02-21.

[19] pandora s. pandora-s / neural-bridge-rag-dataset-12000-google-translated. https://huggingface.co/datasets/pandora-s/neural-bridge-rag-dataset-12000-google-translated, 2024. Accessed : 2025-03-05.

[20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140) :1–67, 2020. http://jmlr.org/papers/v21/20-074.html.

[21] Mandeep Rathee, Sean MacAvaney, and Avishek Anand. Guiding retrieval using llm-based listwise rankers. https://arxiv.org/abs/2501.09186, 2025.

[22] Nils Reimers and Iryna Gurevych. Sentence-bert : Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. https://arxiv.org/abs/1908.10084.

[23] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3) :129–146, 1976. https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.4630270302.

[24] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing Management*, 24(5) :513–523, 1988. https://www.sciencedirect.com/science/article/pii/0306457388900210.

[25] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert : smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019. https://arxiv.org/abs/1910.01108.

[26] SBERT.net. Cross encoder - training overview. https://sbert.net/docs/cross_encoder/training_overview.html, 2025. Accessed : 2025-03-05.

[27] SBERT.net. Evaluation. https://sbert.net/docs/package_reference/cross_encoder/evaluation.html, 2025. Accessed : 2025-03-05.

[28] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR : A heterogenous benchmark for zero-shot evaluation of information retrieval models. *CoRR*, abs/2104.08663, 2021. https://arxiv.org/abs/2104.08663.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. http://arxiv.org/abs/1706.03762.

[30] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. Milvus : A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627, 2021. https://dl.acm.org/doi/pdf/10.1145/3448016.3457550.

[31] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual e5 text embeddings : A technical report. https://arxiv.org/abs/2402.05672, 2024.