Encodeurs croisés légers pour ré-ordonner les résultats de recherche d'information

Mathias Vast[1, 3] Victor Morand[1] Basile Van Cooten[3] Laure Soulier[1]
Josiane Mothe[2] Benjamin Piwowarski[1]
(1) Sorbonne Université, CNRS, ISIR, Paris, France
(2) Univ. Toulouse, IRIT, CLLE, CNRS, Toulouse, France
(3) Sinequa by ChapsVision, Paris, France
vast[at]isir.upmc.fr
victor.morand[at]isir.upmc.fr
bvancooten[at]chapsvision.com
laure.soulier[at]isir.upmc.fr
josiane.mothe[at]irit.fr
benjamin.piwowarski[at]cnrs.fr

RÉSUMÉ

En supprimant les interactions superflues dans les encodeurs croisés, notre architecture divise par quatre leur temps d'inférence. Ainsi, les coûts de calcul deviennent comparables à celles de modèles à interaction tardive comme ColBERT, tout en conservant l'essentiel de la performance des encodeurs croisés en termes de résultat de recherche d'information. Notre architecture obtient en outre une meilleure généralisation sur des collections hors distribution.

ABSTRACT

Cross-Encoders with Minimal Interaction for efficient Re-ranking

By removing superfluous interactions, our novel architecture reduces inference latency by a factor of four compared to standard cross-encoders. It achieves performance comparable to late-interaction models such as ColBERT, while retaining most of the identification capacity of cross-encoders and exhibiting improved generalization in out-of-distribution settings.

MOTS-CLÉS : Système d'information, Recherche d'information, Modèles d'ordonnancement neuronaux, Encodeur croisé, Efficacité, Interaction tardive, Compromis efficacité–performance.

KEYWORDS: Information systems, Information retrieval, Neural ranking models, Cross-encoder, Efficiency, Late interaction, Effectiveness–efficiency trade-off.

1 Introduction

Depuis l'avènement des modèles basés sur les transformers (Vaswani *et al.*, 2017), une large gamme d'architectures de Recherche d'Information (RI) neuronale a été proposée, allant de simples modèles de représentation (bi-encodeurs) aux modèles basés sur l'interaction (encodeurs croisés), utilisant des représentations denses ou parcimonieuses, à vecteur unique ou multi-vecteurs.

Bien que les encodeurs croisés offrent les meilleures performances d'ordonnancement, leur application exhaustive à de grands corpus demeure trop coûteuse (Yates *et al.*, 2021a; Nogueira & Cho, 2019).

Ce goulot d'étranglement a maintenu le paradigme dominant en deux étapes : *recherche puis ré-ordonnancement*, où une première recherche rapide sélectionne dans le corpus un ensemble de documents candidats (Karpukhin *et al.*, 2020; Amati & Van Rijsbergen, 2002) qui est ensuite ré-ordonné par un encodeur croisé (Nogueira & Cho, 2019). Cette approche a substantiellement amélioré les résultats par rapport aux modèles pré-BERT (Yates *et al.*, 2021b).

Cependant, le paradigme de *recherche et ré-ordonnancement* introduit des limitations fondamentales. Premièrement, l'ordre final est borné par la première étape de recherche, c'est-à-dire que si les documents ou passages pertinents n'ont pas été retrouvés, le ré-ordonnanceur n'a plus la possibilité de les inclure. Deuxièmement, l'étape de ré-ordonnancement reste coûteuse dûe à l'utilisation d'encodeur(s) croisé(s).

Pour répondre à la première limitation, des architectures améliorent l'efficacité tout en restant suffisamment performantes pour la recherche sur corpus complet. Parmi elles, les *modèles à interaction tardive*, tels que ColBERT (Khattab & Zaharia, 2020), constituent une approche représentative. Ils conservent des représentations au niveau des tokens pour les requêtes et les documents, comme les encodeurs croisés, mais retardent (uniquement les représentations contextualisées) et simplifient (maximum de similarité) leurs interactions. Cependant, ces modèles n'atteignent pas le niveau d'efficacité des encodeurs croisés.

Pour répondre à la deuxième limitation, le coût d'inférence des encodeurs croisés peut être réduit soit en optimisant ou en élaguant l'attention propre dans le transformer (Schlatt *et al.*, 2024), soit en réduisant le nombre de candidats vus par le ré-ordonnanceur par des stratégies d'élagage/cascade (Meng *et al.*, 2024; Campagnano *et al.*, 2025).

Dans notre travail nous proposons un ré-ordonnanceur avec une efficacité comparable aux modèles à interaction tardive, directement à partir d'un encodeur croisé conventionnel. Plus précisément, nous poursuivons deux objectifs principaux.

Premièrement, nous développons une approche de masquage où nous supprimons certains mécanismes d'interaction utilisés dans le transformer pour abaisser le coût d'inférence des encodeurs croisés. Plus précisément, nous nous concentrons sur les interactions considérées comme potentiellement superflues par des études d'interprétabilité (Lu *et al.*, 2025; Zhan *et al.*, 2020) afin de minimiser l'impact sur la performance. Nous présentons ici les résultats principaux de l'analyse du masquage de certaines interactions. Nous utilisons ensuite ces résultats pour adapter l'architecture de l'encodeur croisé en ne conservant que les interactions nécessaires pour obtenir une nouvelle architecture épurée spécialement conçue pour la tâche de RI.

Cette nouvelle architecture (Encodeur Croisé à Interaction Minimale) est décrite dans la section 3. Nous montrons que pour un modèle basé sur BERT (Devlin *et al.*, 2019), cette architecture conserve les performances d'ordonnancement de l'encodeur croisé initial, tout en étant $4\times$ plus rapide, égalant les coûts d'inférence de ColBERT.

Dans ce travail, nous abordons la question de recherche suivante : *Peut-on concevoir une architecture de type interaction tardive, tout en maintenant la performance des encodeurs croisés ?*

Les sections suivantes sont centrées sur la nouvelle architecture ; les résultats principaux sur l'analyse de la suppression de certaines interactions sont simplement résumés ici et détaillés dans l'article arXiv (Vast *et al.*, 2026).

2 Travaux connexes

Modèles de base pour une RI neuronale efficiente

Les encodeurs croisés (Nogueira & Cho, 2019) sont très performants, mais leur coût a conduit la communauté de la RI à explorer des alternatives plus légères. Une approche consiste à encoder séparément les requêtes et les documents (Karpukhin *et al.*, 2020), évitant un encodage croisé coûteux à l'inférence. Les bi-encodeurs indexent les représentations des documents et calculent les scores de pertinence sous forme de produits scalaires, ce qui en fait des moteurs de sélection de documents candidats très efficaces. Mais cette capacité s'accompagne d'une performance réduite, en particulier dans les scénarios hors domaine (Rosa *et al.*, 2022; Thakur *et al.*, 2021), car le modèle ne peut pas modéliser explicitement les interactions requête-document.

Les modèles à interaction tardive, tels que ColBERT (Khattab & Zaharia, 2020) ont été développés pour surpasser les moteurs de recherche denses en performance d'ordonnancement. Ils encodent les requêtes et les passages en plusieurs vecteurs (typiquement un par token) et agrègent le score de chaque token de la requête en calculant la similarité maximale (opérateur *MaxSim*) avec un token du document. Cela se traduit par une plus grande efficacité par rapport aux bi-encodeurs simples, mais nécessite de stocker un bien plus grand nombre de vecteurs, ce que des travaux ultérieurs ont cherché à optimiser (Santhanam *et al.*, 2022b,a). Parallèlement, les modèles de représentations creuses apprises, tels que SPLADE (Formal *et al.*, 2021), améliorent également la performance des bi-encodeurs, en particulier en termes de généralisation hors domaine, mais aussi leur efficacité lors de la recherche grâce à leur compatibilité avec des index inversés.

Enfin, les poly-encoders (Humeau *et al.*, 2020) sont des architectures hybrides, plus efficaces que les encodeurs croisés et plus performantes que les bi-encodeurs. Ils résument les documents en M vecteurs, réduisant la quantité de calculs effectués dans l'attention propre du modèle. Cependant, les poly-encoders sont très sensibles au choix de M et n'atteignent l'efficacité d'un encodeur croisé que pour des valeurs de M élevées (360 dans l'article original, plus que le nombre de mots moyen d'un document dans MS MARCO (Bajaj *et al.*, 2016)). Une autre direction consiste à préserver leurs capacités tout en les rendant plus efficaces. Comme pour tous les modèles basés sur les transformers (Vaswani *et al.*, 2017), cela signifie se concentrer sur leur principal goulot d'étranglement computationnel : le mécanisme d'attention propre (Lin *et al.*, 2017).

Cibler le mécanisme d'attention propre pour améliorer l'efficacité

L'attention propre sous-tend les représentations des transformers mais ralentit leur inférence. De nombreux travaux, pas seulement en RI (Tay *et al.*, 2022), ont tenté d'en réduire la complexité (Wang *et al.*, 2020; Wu *et al.*, 2021). Malgré des résultats prometteurs dans des tâches de modélisation du langage, ces approches se transfèrent généralement mal aux tâches spécifiques, en particulier à la RI, où l'attention propre est fortement impliquées dans la détection des correspondances sémantiques et lexicales entre les tokens de la requête et du document (Lu *et al.*, 2025). Alternativement, les approches d'attention parcimonieuses, telles que Longformer (Beltagy *et al.*, 2020) ou BigBird (Zaheer *et al.*, 2020), améliorent l'efficacité de l'attention propre en restreignant les interactions entre tokens à des fenêtres locales. Ces méthodes postulent que toutes les interactions entre tokens ne sont pas requises dans l'attention propre pour atteindre le même niveau d'efficacité. Elles ont été appliquées en RI (Sekulić *et al.*, 2020), y compris sur des encodeurs croisés (Schlatt *et al.*, 2024), améliorant substantiellement l'efficience sans nuire à l'efficacité.

Améliorer l'efficience en réduisant la taille des modèles

Les travaux concernant la réduction de la taille des modèles incluent la distillation de connaissances (Hinton *et al.*, 2015), l'élagage (Frankle & Carbin, 2019; Campos *et al.*, 2023), qui consiste à supprimer les parties inutiles du modèle pour réduire les temps de calculs, ou la compression des représentations de documents pour réduire la surcharge (Déjean & Clinchant, 2025; Santhanam *et al.*, 2022a). Ces approches ont été appliquées en RI pour améliorer l'efficience des LLMs (Lei *et al.*, 2025; Schlatt *et al.*, 2025). Enfin, les transformers à fusion intermédiaire (Cao *et al.*, 2020; Tan & Bansal, 2019) ciblent également l'efficience des modèles. Cette architecture encode séparément la requête et le document dans les couches inférieures d'un modèle transformer avant d'utiliser les couches supérieures pour modéliser l'interaction entre les représentations contextualisées. En RI, MacAvaney *et al.* (2020) ont introduit PreTTR, un encodeur croisé à fusion intermédiaire, tout en abordant la question du stockage des vecteurs de documents en les compressant efficacement avec un auto-encodeur. Bien que PreTTR parvienne à améliorer l'efficience de l'encodeur croisé de base, sa stratégie d'encodage de documents hors ligne suppose que la longueur de la requête soit connue à l'avance. Notre travail atténue notamment ce problème en encodant de manière indépendante la requête et le document avec le modèle de base, BERT (Devlin *et al.*, 2019), avant une fusion intermédiaire.

3 Encodeur Croisé à Interaction Minimale

3.1 Contexte

Les encodeurs croisés classifient un couple requête-document (Q, D). Leur entrée est typiquement composée de la séquence de tokens : `[CLS]` $q_1 \dots q_n$ `[SEP1]` $d_1 \dots d_m$ `[SEP2]` (n tokens de la requête et m tokens du document). Le transformer met à jour la représentation des tokens après chaque couche $\ell \in [1 \dots L]$ en effectuant deux étapes de traitement principales : la mise à jour résiduelle par attention propre et celle d'un perceptron bi-couches. Enfin, l'encodeur croisé applique sa tête de classification à la représentation du token `[CLS]` de la dernière couche L, qui produit un score de pertinence pour le document vis-à-vis de la requête.

Des études passées (Zhan *et al.*, 2020; Lu *et al.*, 2025) suggèrent que la performance de ces modèles est dûe principalement au mécanisme d'attention propre, leur permettant de capturer les signaux de correspondance entre la requête et le document afin de modéliser la pertinence. Plus généralement, ces études ont montré que la décision sur la pertinence est décomposée en plusieurs étapes au sein des encodeurs croisés. Premièrement, le modèle contextualise les tokens de la requête et du document. À ce stade, les interactions requête-document ne jouent qu'un rôle mineur. Une fois leur sémantique correctement encodée, le modèle peut utiliser les signaux d'interaction requête-document et les stocker dans les tokens de la requête (Lu *et al.*, 2025). Finalement, dans les dernières couches, des têtes d'attention spécifiques parcourent les tokens de la requête pour en extraire l'information de pertinence et l'encoder dans la représentation du token `[CLS]` qui sera lue pour la prédiction finale.

Les résultats de ces recherches suggèrent que l'information ne circule pas librement entre les parties de l'entrée des encodeurs croisés, mais circule plutôt des tokens du document vers les tokens de la requête (après avoir été respectivement contextualisés), puis des tokens de la requête vers le `[CLS]`.

Dans cet article, nous désignons les parties d'entrée $X, Y \in \{\texttt{[CLS]}, Q, \texttt{[SEP1]}, D, \texttt{[SEP2]}\}$ et $Y \leftarrow X$ (resp. $Y \not\leftarrow X$) un transfert d'information (resp. le blocage du transfert) d'un token dans X *vers* un token dans Y via le mécanisme d'attention propre (ou, de manière équivalente, que Y *porte attention à* X).

Étude Préliminaire Afin d'identifier les interactions superflues dans un encodeur croisé standard, nous avons dans un premier temps mené une étude préliminaire, détaillée dans l'article original, dans laquelle nous masquons progressivement certaines interactions tout en contrôlant l'impact sur la performance en ré-ordonnancement. Nous en tirons les conclusions principales suivantes :

Premièrement, le `[CLS]` n'a pas besoin de porter attention au document pour recevoir les signaux de pertinence appropriés. Au contraire, les résultats OOD (*Out-Of-Domain*) montrent que les encodeurs croisés semblent capturer des *corrélations parasites* lorsque l'on autorise ce transfert d'information. Deuxièmement, bloquer l'attention du document vers la requête ($D \not\leftarrow Q$) sur toutes les couches ne nuit pas à la performance des encodeurs croisés, alors que cette ablation peut potentiellement conduire à des gains substantiels d'efficacité.

Enfin, il est possible de contextualiser les tokens de la requête et du document indépendamment, sans nuire à l'efficacité ID (*In-Domain*), jusqu'à la couche $\ell = 4$ (sur 12 couches au total pour MiniLM). Deux explications potentielles existent pour la baisse de performance ID après $\ell \geq 5$: (1) les *couches d'interaction* restantes ne sont pas suffisantes pour détecter correctement tous les signaux de pertinence ; (2) après la couche $\ell = 4$, contextualiser indépendamment les tokens de la requête et du document introduit des signaux dans leur représentation qui sont préjudiciables à la tâche de RI. Nous abordons partiellement cette question en Section 3.

Plus généralement, cette étude montre que le masquage d'interactions superflues dans un encodeur croisé peut améliorer ses performances de ré-ordonnancement. Ce gain est particulièrement prononcé dans les scénarios OOD, suggérant que cela agit comme une forme de régularisation, empêchant le modèle de sur-apprendre le corpus d'entraînement (Vast *et al.* (2026) présente davantage de détails).

3.2 Architecture

A partir de ces conclusions, nous répondons ici à **??** en proposant une nouvelle architecture de ré-ordonnanceur. En écartant les interactions superflues, nous montrons que notre architecture – proche d'une architecture à interaction tardive – devient significativement plus légère que les encodeurs croisés standards tout en maintenant des performances de ré-ordonnancement compétitives.

Comparé à un encodeur croisé standard, notre architecture diffère par trois choix architecturaux principaux : (1) Fusion intermédiaire : la requête Q et le document D sont d'abord encodés séparément avant d'être concaténés et de poursuivre l'inférence dans les couches supérieures ; (2) Attention croisée : le transfert d'information est limité, l'attention croisée n'est calculée que depuis les représentations du document, *gelées*, vers la requête. Nous implémentons également un transfert uni-directionnel $\texttt{[CLS]} \leftarrow Q$ en masquant dans l'attention croisée les transferts du document vers le `[CLS]` ($\texttt{[CLS]} \not\leftarrow D$) ; (3) Élagage de couches : nous élaguons les couches finales.

3.3 Paramètres expérimentaux

Modèle de base. Pour notre étude, nous considérons l'architecture BERT (Devlin *et al.*, 2019), car son comportement a déjà été largement étudié (Rogers *et al.*, 2020; Ferrando *et al.*, 2024) et a été appliqué avec succès à la tâche de RI. En particulier, nous étudions un encodeur croisé basé sur MiniLM-v2 (Wang *et al.*, 2021), un modèle compact et performant, basé sur BERT mais pré-entraîné via une distillation d'attention propre profonde. Nous détaillons sa configuration dans la Table 1.

TABLE 1 – Détails de la configuration de MiniLM. * signifie qu'il existe un point de contr ôle HuggingFace

Base *	`microsoft/MiniLM-L12-H384-uncased`
Encodeur croisé *	`cross-encoder/ms-marco-MiniLM-L12-v2`
Architecture	**# Couches**=12, **Params**=33M, **Taille**=384, **# Têtes**=12

Détails des paramètres d'entraînement Pour l'affinage sur la tâche RI, nous nous appuyons sur la distillation avec la fonction coût MarginMSE (Hofstätter *et al.*, 2020). Cette dernière est plus efficace qu'une supervision avec l'entropie croisée binaire (BCE) standard (Xu *et al.*, 2025), et préserve l'amplitude de la différence de pertinence (marge) entre passages pertinents et non-pertinents. Comme modèle enseignant, nous utilisons l'ensemble des scores de ré-ordonnanceur sur MS MARCO (Bajaj *et al.*, 2016) de Hofstätter *et al.*, déjà utilisé pour entraîner des architectures de ré-ordonnancement efficientes telles que ColBERT (Khattab & Zaharia, 2020).

Pour assurer la cohérence et la comparabilité stricte tout au long de notre étude, nous utilisons les mêmes hyper-paramètres d'entraînement dans toutes les expériences. Tous les modèles sont entraînés pendant 125 000 étapes avec une taille de lot de 32, un taux d'apprentissage de 7×10^{-6} et 5 000 étapes de chauffe. Nous validons toutes les 10 000 étapes sur la base des performances RR@10 sur l'ensemble de développement MS MARCO et conservons le meilleur point de contrôle.

Jeux de données Pour les évaluation OOD, nous utilisons les ensembles de données de BEIR [1], un ensemble de tests de performance hétérogène comprenant diverses tâches de recherche d'informations. BEIR offre un cadre commun et simple pour évaluer les modèles. Nous utilisons les 13 jeux de tests accessibles publiquement : ArguAna (Ar), Climate-FEVER (CF), DBPedia (DB), FEVER (FE), FiQA (Fi), HotpotQA (HPQ), NFCorpus (NFC), Natural Questions (NQ), Quora Question Pairs (Q), SCIDOCS (SD), SciFact (SF), Touché-2020 (T-v2), et TREC-COVID (T-C). Pour l'évaluation ID, nous affinons nos modèles sur le jeu d'apprentissage de MS Marco. Le choix des points de contrôle sont validés via cette collection également. L'évaluation ID est réalisée sur la partie "dev-set" de la collection MS Marco et sur les deux collections TREC-DL 2019 et 202.

3.4 Résultats

Références. Pour notre analyse, nous commençons par reproduire l'affinage d'un encodeur croisé prêt à l'emploi que nous utilisons comme référence pour contrôler l'effet de nos choix architecturaux.

1. https://github.com/beir-cellar/beir

TABLE 2 – nDCG@10 (moyenne sur 5 initialisations aléatoires) du ré-ordonnancement des 1000 premiers documents retrouvés par BM25. "Notre-ℓX+[Y/all]" correspond à la version de notre architecture avec seulement Y couches d'interactions (toutes sinon) après la couche X. ↑ et ↓ indiquent une différence statistiquement significative entre un modèle Mask ou notre architecture et leur équivalent en encodeur croisé. MSM MS Marco, DL19 et DL20 sont utilisé pour l'évaluation "In-Domain" dont la moyenne apparaît en colonne ID. Pour les évaluations "Out-Of-Domain" (OOD), il s'agit de la moyenne sur les sous-collection BEIR. Les valeurs en **gras** indiquent la meilleure moyenne, "_" la seconde meilleure.

	Domaine d'apprentissage (ID)			Moyennes	
Modèle	MSM	DL19	DL20	ID	BEIR (OOD)
BM25	23.0	51.2	47.7	42.5	43.0
ColBERTv2 (BERT)	45.0	74.6	73.4	**64.3**	47.8
ColBERTv2 (MiniLM)	37.3	67.8	66.3	57.2	42.6
Encodeur Croisé	45.7	75.5	73.6	**64.9**	49.5
Référence - Encodeur Croisé	44.7	73.8	72.9	**63.8**	44.7
Notre-ℓ4+all	43.1 ↓	73.4	70.1	62.2	**50.5**
Notre-ℓ4+3	42.1 ↓	72.6	69.3	61.3	50.4

Bien que nous comparions également directement avec le point de contrôle HuggingFace existant (* dans la table1), nous avons constaté empiriquement que ses performances étaient difficiles à égaler, principalement parce que son ensemble de validation inclut des jeux de données BEIR (Thakur *et al.*, 2021), alors que nous le réservons uniquement pour les évaluations.

Nous comparons notre architecture aussi avec **ColBERTv2** (Santhanam *et al.*, 2022b), un modèle à interaction tardive très performant connu pour son efficacité, mais qui utilise l'opération MaxSim, beaucoup moins expressive, pour modéliser l'interaction entre les tokens de la requête et du document. Comme ColBERTv2 est basé sur un encodeur beaucoup plus grand – basé sur BERT-base avec ∼110M de paramètres – nous incluons également notre propre reproduction, basée sur MiniLM-v2.

Nous rapportons dans la table 2 les résultats pour deux configurations, avec et sans suppression de couches. Nous incluons également notre reproduction de l'encodeur croisé basé sur MiniLM afin d'observer l'effet de nos choix architecturaux.

Premièrement, bien que notre architecture présente une légère baisse de l'efficacité ID par rapport à la référence (voir table 2), il présente des gains de généralisation hors domaine significatifs. Cela valide notre hypothèse architecturale de la Section 3.2, selon laquelle les représentations du document ne nécessitent pas de contextualisation supplémentaire dans les couches d'interaction.

Notre hypothèse selon laquelle les dernières couches d'un modèle de base entraîné en modélisation du langage masquée peuvent être supprimées pour le ré-ordonnancement est également validée par les résultats de la table 2. Le modèle notre-ℓ4+3 basé sur MiniLM (utilisant seulement 7 des 12 couches du modèle de base) produit une performance comparable à la version conservant toutes les couches du modèle de base.

Bien que le modèle ColBERTv2 standard (basé sur `BERT-base`), soit plus performant à la fois en ID et en OOD, notre architecture montre une performance supérieure à taille de modèle équivalente. Il surpasse le MiniLM-ColBERTv2 de 5 et 8 points de nDCG@10 respectivement en ID et OOD

(Table 2).

En résumé, nous montrons que notre architecture obtient les performances ID d'un encodeur croisé standard tout en offrant une meilleure généralisation OOD. La réponse à **??** est donc positive – nous avons réussi à exploiter les enseignements de notre analyse de masquage pour transposer l'efficacité d'un encodeur croisé dans une nouvelle architecture rendue plus efficiente car débarrassée des interactions superflues.

3.5 Analyse du coût computationnel

Nous avons analysé le coût de calcul en comparant le temps d'inférence et l'empreinte mémoire de notre architecture à ceux d'un encodeur croisé standard (inférence complète sur une paire requête-document) et à ColBERT (encodage séparé de la requête et du document avant le calcul d'un `MaxSim` sur leurs représentations compressées). Comme notre architecture peut théoriquement être utilisée dans une configuration où les tokens du document sont pré-calculés hors ligne, nous présentons également des mesures dans cette configuration. En pratique, nous utilisons le même modèle de base `MiniLM-L12-v2` pour chaque architecture. Nous mesurons le temps d'inférence moyenné sur 100 exécutions dans une configuration de charge maximale (512 tokens par document, taille de lot 128 en utilisant un GPU Nvidia TITAN-V de 12 Go) (cf. **??**).

Nous avons trouvé que notre architecture est $2\times$ plus rapide que les encodeurs croisés standards, montant à $4\times$ avec des représentations de document pré-calculées – correspondant effectivement à la latence de ColBERT ($\times 1,15$), tout en offrant une meilleure efficacité (voir section 3.4). Cela est réalisé en utilisant moins de couches et en réduisant l'interaction à l'attention croisée. Comparé à ColBERT, nous avons cependant une empreinte mémoire plus grande ($\times 1,8$).

4 Conclusion

Dans ce travail, motivé par des résultats provenant de l'interprétabilité (Zhan *et al.*, 2020; Lu *et al.*, 2025) et d'expériences préliminaires, nous avons proposé une nouvelle architecture qui commence à combler le fossé entre les modèles à interaction tardive légers et les encodeurs croisés lourds. En combinant stratégiquement la fusion intermédiaire, l'attention croisée et l'élagage de couches, nous avons obtenu un nouveau compromis efficacité-efficience, confirmé par nos expériences : notre architecture divise par quatre le temps d'inférence par rapport aux encodeurs croisés standards. Elle égale les modèles à interaction tardive comme ColBERT, tout en conservant la majeure partie de la performance ID des encodeurs croisés et en ayant des capacités de généralisation supérieures en OOD.

Même si notre architecture permet le pré-calcul des vecteurs de documents, nous n'implantons pas ici le mécanisme complet d'indexation et de recherche. Nous nous concentrons sur l'usage en tant que ré-ordonnanceur. Dans le futur, nous pourrions exploiter des méthodes bien établies pour l'indexation et le stockage de corpus multi-vecteurs (MacAvaney *et al.*, 2020; Santhanam *et al.*, 2022a), pour transformer notre architecture en un encodeur croisé léger d'indexation/recherche. L'évaluation devra être poursuivie pour évaluer la généralisation des résultats.

Références

AMATI G. & VAN RIJSBERGEN C. J. (2002). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, **20**(4), 357–389. DOI : 10.1145/582415.582416.

BAJAJ P., CAMPOS D., CRASWELL N., DENG L., GAO J., LIU X., MAJUMDER R., MCNAMARA A., MITRA B., NGUYEN T. *et al.* (2016). Ms Marco : A human generated machine reading comprehension dataset. *arXiv preprint arXiv :1611.09268*.

BELTAGY I., PETERS M. E. & COHAN A. (2020). Longformer : The long-document transformer. *ArXiv*, **abs/2004.05150**.

CAMPAGNANO C., MALLIA A., PERTSCHUK J. & SILVESTRI F. (2025). E2rank : Efficient and effective layer-wise reranking. In *European Conference on Information Retrieval, (ECIR)*, p. 417–426 : Springer.

CAMPOS D., MARQUES A., NGUYEN T., KURTZ M. & ZHAI C. (2023). Sparse*bert : Sparse models generalize to new tasks and domains. *arXiv preprint arXiv :2205.12452*.

CAO Q., TRIVEDI H., BALASUBRAMANIAN A. & BALASUBRAMANIAN N. (2020). DeFormer : Decomposing Pre-trained Transformers for Faster Question Answering. In D. JURAFSKY, J. CHAI, N. SCHLUTER & J. TETREAULT, Éds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 4487–4497, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.acl-main.411.

DÉJEAN H. & CLINCHANT S. (2025). Reranking with Compressed Document Representation. *arXiv preprint arXiv :2505.15394*. DOI : 10.48550/arXiv.2505.15394.

DEVLIN J., CHANG M.-W., LEE K. & TOUTANOVA K. (2019). BERT : Pre-training of deep bidirectional transformers for language understanding. In J. BURSTEIN, C. DORAN & T. SOLORIO, Éds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, p. 4171–4186, Minneapolis, Minnesota : Association for Computational Linguistics. DOI : 10.18653/v1/N19-1423.

FERRANDO J., SARTI G., BISAZZA A. & COSTA-JUSSÀ M. R. (2024). A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv : 2405.00208*.

FORMAL T., PIWOWARSKI B. & CLINCHANT S. (2021). Splade : Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 2288–2292 : ACM. DOI : 10.1145/3404835.3463098.

FRANKLE J. & CARBIN M. (2019). The lottery ticket hypothesis : Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* : OpenReview.net.

HINTON G., VINYALS O. & DEAN J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv :1503.02531*.

HOFSTÄTTER S., ALTHAMMER S., SCHRÖDER M., SERTKAN M. & HANBURY A. (2020). Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. *ArXiv*.

HUMEAU S., SHUSTER K., LACHAUX M.-A. & WESTON J. (2020). Poly-encoders : Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*.

KARPUKHIN V., OGUZ B., MIN S., LEWIS P., WU L., EDUNOV S., CHEN D. & YIH W.-T. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the*

2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), p. 6769–6781, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.emnlp-main.550.

KHATTAB O. & ZAHARIA M. (2020). Colbert : Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, p. 39–48, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3397271.3401075.

LEI Y., HE S., LI A. & YATES A. (2025). Making large language models efficient dense retrievers. *arXiv preprint arXiv :2512.20612*.

LIN Z., FENG M., DOS SANTOS C. N., YU M., XIANG B., ZHOU B. & BENGIO Y. (2017). A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING. In *International Conference on Learning Representations*.

LU M., CHEN C. & EICKHOFF C. (2025). Pathway to relevance : How cross-encoders implement a semantic variant of BM25. In C. CHRISTODOULOPOULOS, T. CHAKRABORTY, C. ROSE & V. PENG, Éds., *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, p. 25525–25547, Suzhou, China : Association for Computational Linguistics. DOI : 10.18653/v1/2025.emnlp-main.1297.

MACAVANEY S., NARDINI F. M., PEREGO R., TONELLOTTO N., GOHARIAN N. & FRIEDER O. (2020). Efficient Document Re-Ranking for Transformers by Precomputing Term Representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, p. 49–58, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3397271.3401093.

MENG C., ARABZADEH N., ASKARI A., ALIANNEJADI M. & DE RIJKE M. (2024). Ranked list truncation for large language model-based re-ranking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, p. 141–151, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3626772.3657864.

NOGUEIRA R. & CHO K. (2019). Passage re-ranking with bert. *arXiv preprint arXiv :1901.04085*.

ROGERS A., KOVALEVA O. & RUMSHISKY A. (2020). A primer in BERTology : What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, **8**, 842–866. DOI : 10.1162/tacl_a_00349.

ROSA G., BONIFACIO L., JERONYMO V., ABONIZIO H., FADAEE M., LOTUFO R. & NOGUEIRA R. (2022). In defense of cross-encoders for zero-shot retrieval. *arXiv preprint arXiv :2212.06121*.

SANTHANAM K., KHATTAB O., POTTS C. & ZAHARIA M. (2022a). Plaid : an efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, p. 1747–1756.

SANTHANAM K., KHATTAB O., SAAD-FALCON J., POTTS C. & ZAHARIA M. (2022b). ColBERTv2 : Effective and efficient retrieval via lightweight late interaction. In M. CARPUAT, M.-C. DE MARNEFFE & I. V. MEZA RUIZ, Éds., *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 3715–3734, Seattle, United States : Association for Computational Linguistics. DOI : 10.18653/v1/2022.naacl-main.272.

SCHLATT F., FRÖBE M. & HAGEN M. (2024). Investigating the Effects of Sparse Attention on Cross-Encoders. volume 14608, p. 173–190. DOI : 10.1007/978-3-031-56027-9_11.

SCHLATT F., FRÖBE M., SCELLS H., ZHUANG S., KOOPMAN B., ZUCCON G., STEIN B., POTTHAST M. & HAGEN M. (2025). *Rank-DistiLLM : Closing the Effectiveness Gap Between Cross-Encoders and LLMs for Passage Re-ranking*, In *Advances in Information Retrieval*, p. 323–334. Springer Nature Switzerland. DOI : 10.1007/978-3-031-88714-7_31.

SEKULIĆ I., SOLEIMANI A., ALIANNEJADI M. & CRESTANI F. (2020). Longformer for MS MARCO Document Re-ranking Task. *arXiv preprint arXiv :2009.09392*.

TAN H. & BANSAL M. (2019). LXMERT : Learning cross-modality encoder representations from transformers. In K. INUI, J. JIANG, V. NG & X. WAN, Éds., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, p. 5100–5111, Hong Kong, China : Association for Computational Linguistics. DOI : 10.18653/v1/D19-1514.

TAY Y., DEHGHANI M., BAHRI D. & METZLER D. (2022). Efficient transformers : A survey. *ACM Comput. Surv.*, **55**(6). DOI : 10.1145/3530811.

THAKUR N., REIMERS N., RÜCKLÉ A., SRIVASTAVA A. & GUREVYCH I. (2021). BEIR : A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

VAST M., MORAND V., VAN COOTEN B., SOULIER L., MOTHE J. & PIWOWARSKI B. (2026). MICE : Minimal Interaction Cross-Encoders for efficient Re-ranking. DOI : 10.48550/arXiv.2602.16299.

VASWANI A., SHAZEER N., PARMAR N., USZKOEIT J., JONES L., GOMEZ A. N., KAISER L. & POLOSHUKIN I. (2017). Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*. DOI : 10.48550/arXiv.1706.03762.

WANG S., LI B. Z., KHABSA M., FANG H. & MA H. (2020). Linformer : Self-attention with linear complexity. *arXiv preprint arXiv :2006.04768*.

WANG W., BAO H., HUANG S., DONG L. & WEI F. (2021). MiniLMv2 : Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics : ACL-IJCNLP 2021*, p. 2140–2151, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2021.findings-acl.188.

WU C., WU F., QI T. & HUANG Y. (2021). Fastformer : Additive attention can be all you need. *ArXiv*, **abs/2108.09084**.

XU Z., HUANG Z., ZHUANG S. & SRIKUMAR V. (2025). Distillation versus contrastive learning : How to train your rerankers. *arXiv preprint arXiv :2507.08336*.

YATES A., NOGUEIRA R. & LIN J. (2021a). Pretrained transformers for text ranking : Bert and beyond. In *Proceedings of the 14th ACM International Conference on web search and data mining*, p. 1154–1156.

YATES A., NOGUEIRA R. & LIN J. (2021b). Pretrained transformers for text ranking : Bert and beyond. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, p. 2666–2668, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3404835.3462812.

ZAHEER M., GURUGANESH G., DUBEY A., AINSLIE J., ALBERTI C., ONTANON S., PHAM P., RAVULA A., WANG Q., YANG L. & AHMED A. (2020). Big bird : transformers for longer sequences. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA : Curran Associates Inc.

ZHAN J., MAO J., LIU Y., ZHANG M. & MA S. (2020). An analysis of bert in document ranking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, p. 1941–1944, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3397271.3401325.