Des Tokens aux Concepts: Exploiter les SAE pour SPLADE

Yuxuan Zong[1] Mathias Vast[1, 2]
Basile Van Cooten[2] Laure Soulier[1] Benjamin Piwowarski[1]
(1) Sorbonne Université, CNRS, ISIR, Paris, France
(2) Sinequa by ChapsVision, Paris, France
yuxuan.zong[at]isir.upmc.fr
vast[at]isir.upmc.fr
bvancooten[at]chapsvision.com
laure.soulier[at]isir.upmc.fr
benjamin.piwowarski[at]cnrs.fr

RÉSUMÉ

Les modèles de RI neuronaux parcimonieux, tels que SPLADE, offrent un excellent compromis entre efficacité et performance. Cependant, ils reposent sur le vocabulaire du modèle de base, ce qui peut nuire aux performances (polysémie et synonymie) et poser des défis pour les usages multilingues et multimodaux. Pour y remédier, nous proposons de remplacer le vocabulaire du modèle de base par un espace latent de concepts sémantiques appris à l'aide d'auto-encodeurs parcimonieux, ou SAE. Dans cet article, nous étudions la compatibilité de ces deux concepts et montrons que SAE-SPLADE atteint des performances comparables à celles de SPLADE, à la fois sur des tâches dans le domaine et hors domaine, tout en offrant une meilleure efficacité.

ABSTRACT

From Tokens to Concepts : Leveraging SAE for SPLADE

Learned Sparse IR models, such as SPLADE, offer an excellent efficiency-effectiveness tradeoff. However, they rely on the underlying backbone vocabulary, which might hinder performance (polysemanticity and synonymy) and poses a challenge for multi-lingual and multi-modal usages. To solve this limitation, we propose to replace the backbone vocabulary with a latent space of semantic concepts learned using Sparse Auto-Encoders (SAE). Throughout this paper, we study the compatibility of these 2 concepts and demonstrate that SAE-SPLADE achieves retrieval performance comparable to SPLADE on both in-domain and out-of-domain tasks, while offering improved efficiency.

MOTS-CLÉS : Recherche d'Information, Modèles de RI neuronaux parcimonieux, SPLADE, Auto-Encodeur Parcimonieux.

KEYWORDS: Information Retrieval, Learned Sparse Retrieval, SPLADE, Sparse Auto-Encoder.

ARTICLE ACCEPTÉ À : SIGIR 2026.
URL : https://arxiv.org/abs/2604.21511

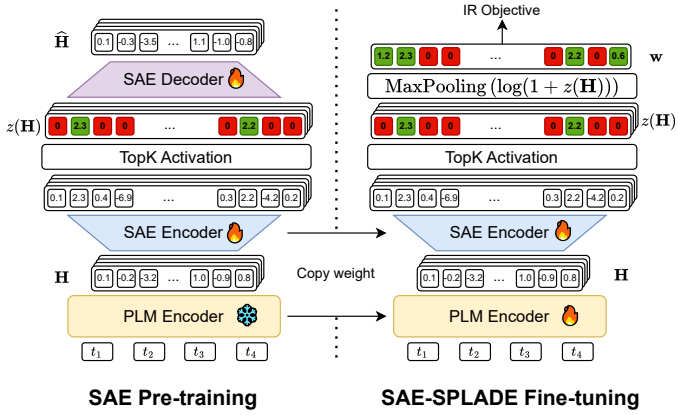

FIGURE 1 – L'architecture de notre modèle SAE-SPLADE. Nous commençons par le pré-entraînement du SAE (gauche). Le SAE-SPLADE est obtenu en supprimant le décodeur SAE et en ajoutant l'agrégation au niveau du document de SPLADE (droite).

1 Introduction

Les modèles de langage contextuels pré-entraînés (PLM) basés sur l'architecture Transformer, tels que BERT (11), ont considérablement amélioré l'efficacité de la recherche par rapport aux méthodes de l'état de l'art précédentes en Recherche d'Information (RI), comme BM25 (38). Suite à leur introduction, une grande variété de méthodes est apparue. Les encodeurs doubles (parcimonieux (14; 17) ou denses (23)), ainsi que les modèles à interaction tardive, tels que ColBERT (41), sont considérés comme les plus efficaces et sont traditionnellement utilisés pour parcourir tout le corpus (*first-stage retrievers*).

Parmi les *first-stage retrievers*, les modèles de représentations parcimonieuses apprises (LSR) comme SPLADE (16; 14) ont montré un grand potentiel à l'ère des PLM. Contrairement aux modèles lexicaux traditionnels (38), les modèles LSR sont capables d'atténuer le problème de décalage lexical (*lexical mismatch*) notamment en effectuant une expansion de termes basée sur des représentations contextualisées. Les modèles LSR conservent en outre les bonnes propriétés des approches sac de mots, telles que la correspondance exacte de termes ou l'interprétabilité, et bénéficient de l'efficacité éprouvée des index inversés, qui permettent une recherche rapide.

Cependant, la plupart des modèles LSR performants s'appuient encore sur le vocabulaire de l'encodeur. Cela a plusieurs implications. Premièrement, l'adaptation à de nouvelles langues (ou modalités) n'est pas directe (8; 33; 25). Deuxièmement, le modèle s'appuie sur l'expansion de termes pour faire face aux *décalages de vocabulaire* (p. ex. les synonymes) et découpe les mentions d'entités en parties (32), ce qui n'est pas entièrement satisfaisant. Troisièmement, le modèle dérive la représentation parcimonieuse d'un texte en utilisant les états cachés de la dernière couche Transformer, mais qui n'est pas nécessairement la plus adaptée à de nombreuses tâches de TAL/RI (43).

Récemment, les Auto-Encodeurs Parcimonieux (SAE) ont été introduits. En projetant des états latents de Grands Modèles de Langage (LLM) dans un vecteur (parcimonieux) de concepts, ils permettent de découvrir les concepts manipulés par ces modèles (18; 37). Dans le contexte des modèles LSR,

les SAE ouvrent une perspective intéressante en permettant de remplacer la couche de projection MLM, vers le vocabulaire d'entrée, par une couche SAE, vers un vocabulaire *appris* de *concepts*. Cette approche est supportée par des travaux récents (35; 22; 27; 46) qui ont montré le potentiel de projeter des requêtes ou documents vers l'espace conceptuel d'un SAE.

Dans ce travail, nous allons plus loin et étudions en profondeur comment les SAE peuvent remplacer la tête MLM de SPLADE pour la tâche de RI dans une nouvelle architecture, SAE-SPLADE. Plus généralement, nous explorons les questions de recherche suivantes :

RQ1 *Est-il possible de remplacer l'espace de sortie par tokens des modèles LSR par un espace de sortie latent, en utilisant des SAE ?*

RQ2 *Quels avantages les SAE apportent-ils aux modèles LSR en termes de coût computationnel et d'efficacité ?*

Dans cet article, nous montrons comment combiner les SAE et SPLADE en un nouveau modèle, SAE-SPLADE. Nous évaluons ce nouveau modèle sur une large gamme de jeux de données, à la fois dans le domaine d'apprentissage (ID) et en dehors (OoD). Nos résultats montrent que SAE-SPLADE peut atteindre des performances à l'état de l'art, avec un coût moindre qu'un modèle SPLADE. Afin d'étudier la balance coût-efficacité entre les modèles, nous introduisons une nouvelle fonction de score $\mathbf{E}^2$ (pour *Efficiency-Effectiveness*), qui pourrait servir de base pour de futurs travaux en RI neuronale parcimonieuse. Nous réferrons le lecteur à l'article original pour de plus amples analyses sur l'impact de différents hyperparamètres clés et de choix de conception sur les performances de SAE-SPLADE. Nous laissons aussi de côté dans le cadre de cette soumission l'étude de l'applicabilité de SAE-SPLADE à la recherche multilingue ou les analyses qualitatives sur les différences de comportement entre SPLADE et SAE-SPLADE. [1]

2 Travaux Connexes et Préliminaires

Les approches LSR désignent une famille de modèles utilisés en tant que *first-stage* car entraînés à prédire des représentations vectorielles parcimonieuses de requêtes et de documents, permettant une recherche efficace. SPLADE (16; 14), dernier né de cette catégorie de modèles, exploite notamment l'architecture Transformer et la tête de projection MLM, qui préconditionne le modèle pour la RI (Formal *et al.*). La dernière itération de SPLADE, SPLADE-v3 (26) est ainsi capable de battre les performances des meilleurs méthodes *first-stage* denses (41), tout en ayant un coût computationnel largement moindre.

Comme argumenté en introduction, nous émettons l'hypothèse que ces modèles souffrent encore de plusieurs limitations liées à la dépendance au vocabulaire des PLM. Si certains travaux ont exploré des moyens de découpler les vocabulaires d'entrée et de sortie (12; 24; 32), le nouveau vocabulaire de sortie s'appuie toujours fortement sur la matrice de projection du vocabulaire d'origine, en raison de la difficulté d'initialiser aléatoirement un nouveau vocabulaire (48). A contrario, nous proposons d'utiliser les SAE pour apprendre un *vocabulaire sémantique* de concepts, afin de se libérer de cette contrainte.

Tout d'abord utilisés afin d'interpréter les comportements des LLMs (44; 47; 5), les SAE (31) sont particulièrement intéressants dans le contexte des LSR car ils peuvent démêler les structures

1. Le code et les points de contrôle seront publiés à la parution.

sémantiques complexes entrelacées dans les représentations denses de ces modèles en un ensemble d'unités conceptuelles distinctes et interprétables (les *latents*).

En pratique, un SAE encode un état caché $\mathbf{h} \in \mathbb{R}^d$ dans un vecteur parcimonieux de latents $z(\mathbf{h}) \in \mathbb{R}^M$. Son encodeur consiste en une transformation linéaire $\mathbf{W}_{enc} \in \mathbb{R}^{M \times d}$, avec un biais $\mathbf{b}_{enc} \in \mathbb{R}^M$, suivie d'une fonction d'activation ϕ, p.ex. ReLU :

$$z(\mathbf{h}) = \phi(\mathbf{W}_{enc}\mathbf{h} + \mathbf{b}_{enc}) \tag{1}$$

Le nombre total de latents M est généralement choisi pour être bien supérieur à la taille de la dimension cachée originale ($M \gg d$), afin de permettre à l'encodeur d'apprendre une représentation plus expressive de l'entrée originale (base surdéterminée).

Pour entraîner cet encodeur, les modèles SAE s'appuient sur un *décodeur* qui prédit $\widehat{\mathbf{h}}$ à partir de $z(\mathbf{h})$, en maintenant $\mathbf{h}$ et $\widehat{\mathbf{h}}$ aussi proches que possible. Formellement, nous avons

$$\begin{aligned} \widehat{\mathbf{h}} &= \mathrm{SAE}(\mathbf{h}) = \mathbf{W}_{dec} z(\mathbf{h}) + \mathbf{b}_{dec} \\ \mathcal{L}_{rsct} &= \|\widehat{\mathbf{h}} - \mathbf{h}\|_2^2 \end{aligned} \tag{2}$$

où $\mathbf{W}_{dec} \in \mathbb{R}^{d \times M}$ et $\mathbf{b}_{dec} \in \mathbb{R}^d$ sont la matrice de poids et le terme de biais du décodeur. Pour garantir que la représentation reste parcimonieuse, une perte de régularisation $\mathcal{L}_{sparsit}$, qui minimise typiquement la norme ℓ_1 du vecteur d'activation parcimonieux, est utilisée. Sans cette contrainte de sparsité, le SAE pourrait maintenir une haute qualité de reconstruction en gardant un grand nombre de latents activés (p.ex. d). La perte globale, $\mathcal{L}_{SAE}$, peut être définie comme :

$$\mathcal{L}_{SAE}(\mathbf{h}) = \mathcal{L}_{rsct} + \alpha_{sp}\mathcal{L}_{sparsit} \tag{3}$$

où α_{sp} contrôle l'importance de la régularisation de sparsité.

Depuis sa création, le SAE (31) a été amélioré – une liste non exhaustive comprend : Gated SAE (36), TopK SAE (18), BatchTopK SAE (6) ou JumpRelu SAE (37). Tous visent à résoudre deux problèmes connus de la méthode originale : (1) le *problème de biais de contraction* (*shrinkage bias*), où la norme ℓ_1 conçue pour imposer la sparsité contribue également à minimiser l'amplitude des activations latentes, réduisant la précision de la reconstruction ; et 2) le *problème des latents "morts"*, où certains latents cessent de s'activer complètement pendant l'entraînement (car leur valeur d'activation est toujours 0). Parmi ceux-ci, le TopK SAE (18) exploite un filtrage top-k sur chaque représentation $z(\mathbf{h})$, évitant de contrôler explicitement la sparsité par régularisation. D'autres travaux ont développé des variantes de TopK SAE qui améliorent la structure de l'espace latent. Par exemple, le Hierarchical TopK SAE (2) apprend un seul modèle tout en optimisant les reconstructions à plusieurs niveaux de sparsité, tandis que le Matryoshka TopK SAE (7) entraîne simultanément plusieurs modèles SAE imbriqués de tailles croissantes. Cet objectif d'entraînement conduit à ce que les sémantiques grossières soient capturées par les k premiers latents, tandis que les informations plus fines sont progressivement encodées dans les dimensions ultérieures.

En RI, les SAE ont été utilisés pour approximer des représentations denses pour la recherche efficace des plus proches voisins (35; 22). En particulier, Park *et al.* (35) montrent que les caractéristiques dérivées des SAE constituent des unités d'indexation efficaces. Cependant, toutes les approches précédentes combinent le SAE avec des modèles de recherche dense déjà entraînés. En revanche, nous proposons d'entraîner un SAE directement à partir d'un modèle de base PLM, sur un corpus de RI général, c'est-à-dire MS MARCO (1), avant de le combiner à un modèle SPLADE. Le SAE-SPLADE

résultant est ensuite affiné sur la tâche de RI, avec le module SAE à la place de sa tête MLM originale, permettant une intégration plus étroite entre la modélisation de la pertinence et la sparsité pendant l'entraînement.

3 Connexion du SAE avec SPLADE

Comme discuté ci-dessus, les multiples lacunes des modèles LSR découlant de leur tête MLM et de sa forte dépendance au vocabulaire d'entrée nous amènent à introduire une nouvelle architecture, qui n'est plus liée par ces contraintes. Bien que prometteur, SNRM a montré que s'appuyer sur une tête de projection aléatoire (c'est-à-dire des vecteurs de vocabulaire aléatoires) ne préconditionne pas suffisamment le modèle pour des performances à l'état de l'art (48) – et nos expériences le confirment. Par conséquent, contrairement aux travaux précédents, nous introduisons un vocabulaire sémantique entièrement nouveau appris avec SAE. Chaque élément de ce vocabulaire – chaque latent – représente un concept sémantique spécifique plutôt qu'un token fixe. Nous conjecturons que cette conception atténue à la fois les décalages de vocabulaire et de sémantique, permettant au modèle de se transférer plus efficacement entre les contextes, y compris vers des langues non vues. En particulier, contrairement au modèle SPLADE original, notre SAE-SPLADE et son vocabulaire basé sur des latents sont censés générer des représentations plus cohérentes entre diverses langues et/ou modalités, améliorant ainsi la transférabilité dans les contextes de recherche multilingue.

Comme illustré dans Figure 1, pour combiner SAE avec un modèle de RI neuronale parcimonieuse, nous utilisons un pipeline d'entraînement en deux étapes. Premièrement, nous entraînons un SAE en reconstruisant les représentations contextualisées des tokens. Nous n'utilisons ensuite que son encodeur pour projeter les représentations contextualisées dans un espace latent (parcimonieux) pour l'estimation de la pertinence. L'encodeur SAE entraîné $z(\cdot)$, qui produit des représentations parcimonieuses dans $\mathbb{R}^M$, projette les représentations contextualisées $\mathbf{H} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_N)$ du PLM, de manière similaire à ce que fait SPLADE avec la tête de projection MLM originale (16). Nous utilisons ensuite les mêmes fonctions d'activation et mécanismes de pooling que SPLADE-max (14).

Entraînement du SAE L'entraînement d'un SAE repose sur des représentations denses contextualisées produites par un modèle pré-entraîné. Dans ce travail, nous utilisons des architectures basées sur l'encodeur telles que BERT (11) ou DistilBERT (39) pour encoder les représentations au niveau des tokens du corpus, avant de les alimenter au SAE. Il est important de noter que les poids de l'encodeur PLM sont maintenus gelés lors de l'entraînement du SAE. Parmi les différentes versions de SAE citées ci-dessus, nous avons d'abord expérimenté avec JumpReLU SAE (37) mais il s'est avéré trop instable [2], et nous avons utilisé le TopK SAE largement répandu (18). Nous avons également expérimenté avec ses variantes, Hierarchical TopK SAE (2) et Matryoshka TopK SAE (7), dans la section B.4.

Entraînement pour la Recherche Parcimonieuse Pour affiner SAE-SPLADE, nous substituons la tête MLM originale de l'encodeur par l'encodeur SAE entraîné et supprimons le décodeur SAE. Contrairement à l'étape précédente, où seul le SAE est entraîné, nous affinons également les paramètres de l'encodeur original.

2. La documentation officielle de SAELens signale ce problème : https://github.com/decoderesearch/SAELens/blob/main/docs/training_saes.md

Nous utilisons le même objectif d'affinage que SPLADEv3 (26), qui combine à la fois les objectifs de distillation par divergence KL ($\mathcal{L}_{KL}$) et MarginMSE ($\mathcal{L}_{MSE}$), mais pas leur stratégie complexe de génération de données – p.ex. l'auto-distillation (15), un cross-encodeur plus puissant pour évaluer les négatifs, et l'approche d'apprentissage par curriculum (49).

Bien que la contrainte top-k du TopK SAE contrôle déjà la sparsité, notre étude empirique montre que la sparsité globale sur la représentation agrégée du document (ou de la requête) est encore relativement faible. Suivant SPLADE (16; 14), nous ajoutons également la régularisation FLOPs $\mathcal{L}_{flops-d}$ et $\mathcal{L}_{flops-q}$ (34), pour contrôler davantage la sparsité de la représentation de SAE-SPLADE lors de l'apprentissage de l'objectif RI.

Au total, notre objectif d'entraînement pour SAE-SPLADE est :

$$\begin{aligned}\mathcal{L}_{\text{SAE-SPLADE}} = \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{MSE}}\mathcal{L}_{\text{MSE}} \\ + \lambda_{\text{flops-d}}\mathcal{L}_{\text{flops-d}} + \lambda_{\text{flops-q}}\mathcal{L}_{\text{flops-q}}\end{aligned} \tag{4}$$

Notons que nous avons également expérimenté l'entraînement conjoint du SAE et de SPLADE, mais avons constaté que c'était préjudiciable. Nous attribuons cela au désalignement entre la perte de reconstruction et l'objectif de la tâche RI, l'un cherchant à compresser autant que possible les informations qui sont essentielles à l'autre. Cependant, comme montré par la suite, le SAE (comme le MLM) fournit un bon préconditionnement pour l'entraînement de SPLADE.

4 Expériences et Résultats

4.1 Configuration Expérimentale

Jeux de données Nous entraînons le SAE sur les 8,8 millions de passages du jeu de données MS MARCO v1 (1) et affinons SAE-SPLADE avec le même jeu de données que ColBERTv2 (41) (distillation basée sur un cross-encodeur). Nous évaluons les modèles *en domaine* sur l'ensemble dev-small de MS MARCO (6980 requêtes), ainsi que sur les ensembles 2019 (43 sujets évalués) et 2020 (54 sujets évalués) de la piste TREC-DL (10; 9) (moyenne des 2 ensembles). Nous rapportons les performances hors domaine sur LoTTE Search (41) (moyenne de 5 sous-ensembles) pour toutes les expériences préliminaires et d'ablation, et comparons en outre SAE-SPLADE aux autres références de base sur les 13 jeux de données publiquement disponibles de BEIR, suivant la pratique courante.

Modèles Nous initialisons notre modèle avec DistilBERT (39) (point de contrôle pré-entraîné depuis HuggingFace [3]), car il est beaucoup plus petit qu'un modèle BERT (gain de vitesse de 2x) tout en étant compétitif, comme reconnu dans SPLADEv2 (14; 15).

Tous les détails sur l'implémentation sont décrits en Annexe A.

3. `distilbert/distilbert-base-uncased`

4.2 Évaluation des Modèles Parcimonieux

En plus des métriques RI standard (p.ex. MRR@10 ou nDCG@10) utilisées pour quantifier l'efficacité des modèles, nous rapportons également leur efficience en utilisant les *FLOPs Requête-Document* moyens (ou QD-FLOPs) entre les représentations de requête et de document. Cette métrique estime le nombre attendu d'entrées de listes de postings, par requête et par document, qui doivent être accédées pendant la recherche (avec une approche de recherche naïve). Formellement [4],

$$\text{QD-FLOPs} = \mathbb{E}_{\mathbf{Q},\mathbf{D}} \left(\sum_t \mathbb{1}(\mathbf{w}_t^{\mathbf{Q}} > 0) \times \mathbb{1}(\mathbf{w}_t^{\mathbf{D}} > 0) \right) \tag{5}$$

où $\mathbb{1}(\mathbf{w}_t^{\mathbf{Q}} > 0)$ est 1 si le token a un poids non nul dans la requête $\mathbf{Q}$ et 0 sinon (idem pour $\mathbf{D}$). Nous estimons les QD-FLOPs à partir d'un grand nombre de requêtes et de documents. De plus, nous quantifions également l'efficience en mesurant le nombre moyen de composantes non nulles dans la représentation des documents (nous le désignons par *Avg. Doc Len* dans les sections suivantes), pour approximer la quantité de stockage nécessaire pour l'index inversé.

Nos expériences impliquent la comparaison de nombreux résultats différents selon les choix d'architecture et d'hyperparamètres. Pour mieux comparer les modèles en termes de compromis entre efficacité et efficience, nous avons conçu une nouvelle métrique, notée $\mathbf{E}^2$ (pour Efficience-Efficacité). Formellement, $\mathbf{E}^2$ est défini comme :

$$\mathbf{E}^2 = \text{MRR} - \mu_1 \text{ QD-FLOPs} - \mu_2 \text{ softplus}_\beta ((\text{QD-FLOPs} - \tau)) \tag{6}$$

où nous utilisons softplus (19) [5], et où μ_1 (avant le seuil QD-FLOPs τ) et $\mu_1 + \mu_2$ (après le seuil τ) contrôlent le compromis efficience-efficacité. Pour illustrer $\mathbf{E}^2$, considérons le cas où β, qui contrôle la douceur (en pratique, nous utilisons $\beta = 2$), est grand, c'est-à-dire $\text{softplus}(x) \approx \max(\cdot, x)$. Dans ce cas, quand QD-FLOPs $= 0$, $\mathbf{E}^2$ vaut 1 (borne supérieure) quand MRR $= 1$ et 0 quand MRR $= 0$. Plus généralement, $\mathbf{E}^2$ définit le compromis entre MRR et QD-FLOPs. Illustrons cela avec $\mu_1 = 0.01$ et $\mu_2 = 0.09$, qui sont les valeurs utilisées dans cet article. Avant le seuil QD-FLOPs $\tau = 5$, $\mathbf{E}^2$ traduit que nous sommes prêts à accepter une baisse de 0,01 MRR en échange d'une augmentation de $0,01 \times \mu_1^{-1} = 1$ QD-FLOPs, et après τ, de $0,01 \times (\mu_1 + \mu_2)^{-1} = 0,1$ FLOPS – nous permettant de définir un seuil QD-FLOPs « doux » au-delà duquel nous considérons le modèle comme trop coûteux. Bien que les valeurs que nous avons choisies (τ, μ_1, μ_2 et β) soient arbitraires, nous pensons qu'elles offrent une métrique concise et solide pour comparer les différents modèles avec lesquels nous avons expérimenté. Enfin, dans les tableaux et Figure 2, nous rapportons $\Delta\mathbf{E}^2$, qui est la différence en $\mathbf{E}^2$ entre les modèles évalués et BM25.

4.3 Résultats Principaux

Sur la base des expériences préliminaires décrites ci-dessus, notre modèle SAE-SPLADE principal est basé sur TopK SAE, avec $M = 2^{16}$ latents, et l'encodeur SAE utilise la sortie de la couche DistilBERT $\ell = 6$ (avant la tête de transformation $\mathcal{T}$). Nous considérons les références de base suivantes pour nos expériences :

4. Dans MS MARCO passage v1, 1 QD-FLOPs correspond à 8,8M d'entrées de posting à traiter pour chaque requête.
5. Nous utilisons sa variante PyTorch $\text{softplus}_\beta(x) = \frac{1}{\beta} \log(1 + \exp(x))$ qui introduit un paramètre de lissage (42).

BM25 (38) – une référence RI standard ;

SPLADE (14) – le modèle parcimonieux appris à l'état de l'art. Nous le reproduisons avec les mêmes données d'entraînement et objectif que SAE-SPLADE pour la cohérence. Nous rapportons en outre le résultat de SPLADEv3 (26) basé sur DistilBERT pour comparaison ;

ColBERTv2 (41) – le modèle à interaction tardive à l'état de l'art. Pour assurer une comparaison équitable, nous rapportons en outre les résultats de ColBERTv2 basé sur DistilBERT de (49) ;

CL-SR (35) – un modèle LSR basé sur les latents sémantiques appris avec un SAE. Contrairement à SAE-SPLADE, CL-SR est basé sur un modèle de recherche dense à vecteur unique, SimLM (45). Par conséquent, ses latents sont appris en reconstruisant les vecteurs de tokens denses `[CLS]`. Nous ne rapportons que les performances de la version la plus efficace du modèle, ainsi que les performances de SimLM.

Comme nos références de base couvrent des modèles avec une très haute efficacité (p.ex. SPLADEv3) et une très haute efficience (p.ex. BM25), nous rapportons les résultats avec les 3 différentes valeurs de k ($k = 4, 8, 16$ pendant à la fois l'entraînement du SAE et l'affinage de SAE-SPLADE) qui ont atteint le meilleur $\Delta\mathbf{E}^2$. Les résultats sont rapportés dans Table 1.

TABLE 1 – Résultats des évaluations. Pour mesurer l'efficacité : MRR@10 pour MS MARCO dev (MSM), nDCG@10 pour la moyenne TREC-DL19 et 20 (TREC-DL) et la moyenne sur BEIR, et Success@5 pour la moyenne sur LoTTE Search (LoTTE). Pour mesurer l'efficience : QD-flops, longueur moyenne des documents, ainsi que $\Delta\mathbf{E}^2$. Les meilleures métriques globales sont en **gras** et les meilleures parmi nos modèles sont soulignées. Parmi nos modèles SAE-SPLADE, nous utilisons $_{\downarrow}$ et $^{\uparrow}$ pour indiquer sur combien de corpus nous sommes significativement moins performants et plus performants par rapport à SPLADE, selon le test-t de Student bi-latéral ($p < 0.05$).

Modèle	Encodeur	MSM	TREC-DL	LoTTE	BEIR	QD-flops↓	Avg. D Len ↓	$\Delta\mathbf{E}^2$ ↑
Denses								
SimLM	$\text{BERT}_{\text{base}}$	**41.1**	70.6	-	-	-	-	-
ColBERTv2	DistilBERT	38.3	73.6	70.6	-	-	-	-
ColBERTv2	$\text{BERT}_{\text{base}}$	39.7	**75.1**	**72.0**	**50.0**	-	-	-
Parcimonieux								
BM25	-	18.3	49.2	51.0	43.7	**0.13**	**39**	0.0
CL-SR	$\text{BERT}_{\text{base}}$	36.8	66.0	-	-	0.74	65	17.9
SPLADEv3	DistilBERT	38.7	74.8	70.3	**50.0**	1.40	165	**19.1**
Nos modèles								
SPLADE	DistilBERT	37.7	72.0	68.9	48.8	1.47	118	18.1
SAE-SPLADE $k = 4$	DistilBERT	$37.1_{\downarrow 1}$	71.5	$68.2_{\downarrow 1}$	$48.5^{\uparrow 1}_{\downarrow 7}$	0.33	66	18.6
SAE-SPLADE $k = 8$	DistilBERT	37.6	72.1	$68.8^{\uparrow 1}_{\downarrow 1}$	$49.2^{\uparrow 4}_{\downarrow 2}$	0.67	109	18.8
SAE-SPLADE $k = 16$	DistilBERT	$38.2^{\uparrow 1}$	72.3	$69.4^{\uparrow 1}$	$49.6^{\uparrow 5}_{\downarrow 1}$	1.37	191	18.7

SAE-SPLADE ($k = 8$) a une efficacité comparable au modèle SPLADE que nous avons entraîné, sur les tâches ID et OoD, mais est légèrement meilleur sur BEIR (significativement meilleur sur 4 jeux de données BEIR, avec $p < 0.05$). En même temps, il atteint une bien meilleure efficience (0,66 vs. 1,47 sur QD-FLOPs). Par rapport aux autres modèles parcimonieux, toutes nos variantes surpassent largement la référence BM25, tandis que notre variante $k = 4$ atteint une taille d'index similaire (Avg. D Len) à CL-SR, mais a une meilleure efficience (QD-FLOPs) et de meilleurs résultats ID, en particulier sur les jeux de données TREC-DL plus difficiles.

Par rapport à la forte référence SPLADEv3 (avec un $\Delta\mathbf{E}^2 > 19$), notre variante $k = 16$ atteint presque les mêmes niveaux de performance : elle a des QD-FLOPs très similaires (1,37 vs. 1,40), une taille d'index légèrement plus grande, tout en correspondant à son efficacité sur tous les jeux de

données que nous considérons, sauf TREC-DL [6].

Nos résultats indiquent que SAE-SPLADE est moins performant sur les tâches ID (notre variante $k = 16$ est à égalité avec DistilBERT-ColBERTv2 sur dev-small mais est en retard sur le plus grand ColBERTv2 basé sur BERT et SimLM) par rapport aux modèles denses, mais est moins susceptible de sur-apprendre (toutes nos variantes battent SimLM sur les jeux de données TREC-DL, et SAE-SPLADE $k = 16$ atteint des résultats très comparables aux modèles ColBERTv2 en OoD). Bien que SAE-SPLADE soit encore légèrement en retard sur les modèles ColBERTv2 à interaction tardive à l'état de l'art en termes d'efficacité, ces derniers nécessitent généralement de stocker des représentations de tokens denses pour tout le corpus, résultant en une utilisation disque significativement plus élevée (3 Gio vs. 20 Gio sur MS MARCO v1) et une latence de recherche plus élevée par rapport à notre méthode (<10 ms en utilisant l'index BMP (29) vs. 58 ms en utilisant PLAID (40)).

Pris ensemble, ces résultats confirment que pré-entraîner un encodeur SAE sur un grand vocabulaire de concepts puis l'exploiter pour construire des modèles LSR efficaces et performants est une méthode solide. Le SAE-SPLADE résultant atteint un meilleur équilibre efficacité–efficience que SPLADEv2, tout en se comparant favorablement à l'approche LSR actuelle à l'état de l'art, SPLADEv3 – répondant à la fois à **RQ1** et **RQ2**.

5 Limitations et Conclusion

Dans cet article, nous présentons SAE-SPLADE, une variante de SPLADE qui remplace le vocabulaire des tokens (lexical) par un dictionnaire de représentations latentes apprises par SAE. En utilisant ces latents et leurs patterns d'activation comme vocabulaire de recherche, notre méthode surmonte plusieurs limitations de SPLADE basées sur les tokens lexicaux, notamment la scalabilité limitée aux contextes multilingues et la dépendance au vocabulaire fixe du Transformer. Nous avons conduit des expériences extensives pour évaluer SAE-SPLADE, montrant qu'il atteint des performances de recherche comparables à SPLADE tout en améliorant significativement l'efficience. De plus, des études d'ablation complètes valident nos choix de conception pour à la fois le SAE et le cadre SAE-SPLADE. Enfin, nos résultats démontrent que l'application d'une contrainte top-k au niveau des tokens est globalement bénéfique pour les modèles de recherche parcimonieuse appris, y compris SPLADE lui-même. Dans l'ensemble, nos résultats montrent que le pré-entraînement SAE est une alternative valide au pré-entraînement MLM, permettant potentiellement d'utiliser moins de couches sur des modèles de base plus grands que DistillBERT, bien que nous n'ayons pas étudié cet aspect.

Malgré ces résultats prometteurs, notre travail présente plusieurs limitations et ouvre des directions pour de futures recherches. Premièrement, bien que nos expériences multilingues démontrent que SAE-SPLADE surpasse SPLADE, un écart de performance subsiste par rapport aux modèles multilingues à l'état de l'art tels que MILCO (33). En raison d'un budget de calcul limité, nous n'avons pas incorporé de techniques plus avancées, telles que l'alignement interlingual ou une architecture de modèle plus complexe, dont il a été démontré qu'elles sont efficaces dans les travaux précédents. De plus, l'augmentation du modèle de base multilingue à des tailles de modèle plus grandes pourrait encore améliorer les performances (8).

6. Nous rappelons au lecteur que les hautes performances de SPLADEv3 DistilBERT ont bénéficié de nombreuses astuces incluant l'apprentissage par curriculum (49) et de meilleurs scores de distillation

Deuxièmement, les SAE ont été initialement conçus pour interpréter les caractéristiques dans les LLMs, et leurs objectifs d'entraînement ne sont pas explicitement alignés avec la RI. Bien que nos expériences montrent qu'une intégration directe de ces deux directions de recherche, SAE et LSR, produit de solides résultats, dans la Section B.4, nous démontrons que certaines variantes de SAE bien conçues pourraient ne pas être bénéfiques pour les performances RI. De futurs travaux pourraient faire avancer cette ligne de recherche en développant des variantes de SAE spécifiquement adaptées aux objectifs RI.

Enfin, SPLADE s'appuie principalement sur des tokens lexicaux pour la correspondance exacte et sur des mécanismes d'expansion pour capturer la similarité sémantique. Une direction prometteuse est d'unifier ces deux signaux en combinant l'apprentissage de concepts basé sur SAE avec un vocabulaire lexical, permettant au modèle de capturer conjointement des concepts sémantiques et des indices de correspondance lexicale fins.

Références

[1] 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*.

[2] Balagansky N., Aksenov Y., Laptev D., Kurochkin V., Gerasimov G., Koriagin N. & Gavrilov D. (2025). Train one sparse autoencoder across multiple sparsity budgets to preserve interpretability and accuracy. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, p. 10182–10190.

[3] Bonifacio L., Jeronymo V., Abonizio H. Q., Campiotti I., Fadaee M., Lotufo R. & Nogueira R. (2021). mmarco : A multilingual version of the ms marco passage ranking dataset. *arXiv preprint arXiv :2108.13897*.

[4] Bricken T., Templeton A., Batson J., Chen B., Jermyn A., Conerly T., Turner N., Anil C., Denison C., Askell A., Lasenby R., Wu Y., Kravec S., Schiefer N., Maxwell T., Joseph N., Hatfield-Dodds Z., Tamkin A., Nguyen K., McLean B., Burke J. E., Hume T., Carter S., Henighan T. & Olah C. (2023). Towards monosemanticity : Decomposing language models with dictionary learning. https://transformer-circuits.pub/2023/monosemantic-features/index.html. Transformer Circuits Thread.

[5] Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A. *et al.* (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[6] Bussmann B., Leask P. & Nanda N. (2024). Batchtopk sparse autoencoders. In *NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*.

[7] Bussmann B., Nabeshima N., Karvonen A. & Nanda N. (2025). Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv :2503.17547*.

[8] Chen J., Xiao S., Zhang P., Luo K., Lian D. & Liu Z. (2024). Bge m3-embedding : Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv :2402.03216*, **4**(5).

[9] Craswell N., Mitra B., Yilmaz E. & Campos D. (2021). Overview of the trec 2020 deep learning track. In *Text REtrieval Conference (TREC)* : TREC.

[10] Craswell N., Mitra B., Yilmaz E., Campos D. & Voorhees E. M. (2020). Overview of the trec 2019 deep learning track. In *Text REtrieval Conference (TREC)* : TREC.

[11] Devlin J., Chang M.-W., Lee K. & Toutanova K. (2019). Bert : Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, p. 4171–4186 : ACL.

[12] Dudek J. M., Kong W., Li C., Zhang M. & Bendersky M. (2023). Learning sparse lexical representations over specified vocabularies for retrieval. In *International Conference on Information and Knowledge Management (CIKM)*.

[Formal *et al.*] Formal T., Lassance C., Piwowarski B. & Clinchant S. Towards Effective and Efficient Sparse Neural Information Retrieval. p. 3634912. doi : 10.1145/3634912.

[14] Formal T., Lassance C., Piwowarski B. & Clinchant S. (2021a). Splade v2 : Sparse lexical and expansion model for information retrieval.

[15] Formal T., Lassance C., Piwowarski B. & Clinchant S. (2022). From distillation to hard negative sampling : Making sparse neural ir models more effective. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, p. 2353–2359 : ACM. doi : 10.1145/3477495.3531857.

[16] Formal T., Piwowarski B. & Clinchant S. (2021b). Splade : Sparse lexical and expansion model for first stage ranking. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, p. 2288–2292 : ACM. doi : 10.1145/3404835.3463098.

[17] Gao L., Dai Z. & Callan J. (2021). Coil : Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, p. 3030–3042 : Association for Computational Linguistics.

[18] Gao L., la Tour T. D., Tillman H., Goh G., Troll R., Radford A., Sutskever I., Leike J. & Wu J. (2025). Scaling and evaluating sparse autoencoders. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[19] Glorot X., Bordes A. & Bengio Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, p. 315–323 : JMLR Workshop and Conference Proceedings.

[20] Godey N., Clergerie É. & Sagot B. (2024). Anisotropy is inherent to self-attention in transformers. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 35–48.

[21] Jang K.-R., Kang J., Hong G., Myaeng S.-H., Park J., Yoon T. & Seo H. (2021). Ultra-high dimensional sparse representations with binarization for efficient text retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, p. 1016–1029.

[22] Kang H., Wang T. & Xiong C. (2025). Interpret and control dense retrieval with sparse latent features. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics : Human Language Technologies (Volume 2 : Short Papers)*, p. 700–709.

[23] 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 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 6769–6781 : Association for Computational Linguistics.

[24] KIM H., LEE T. K. & WON T. (2025). The role of vocabularies in learning sparse representations for ranking. *arXiv preprint arXiv :2509.16621*.

[25] LASSANCE C. (2023). Extending english ir methods to multi-lingual ir. *arXiv preprint arXiv :2302.14723*.

[26] LASSANCE C., DÉJEAN H., FORMAL T. & CLINCHANT S. (2024). Splade-v3 : New baselines for splade.

[27] LASSANCE C., FORMAL T. & CLINCHANT S. (2021). Composite code sparse autoencoders for first stage retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, p. 2136–2140, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3404835.3463066.

[28] LIEBERUM T., RAJAMANOHARAN S., CONMY A., SMITH L., SONNERAT N., VARMA V., KRAMÁR J., DRAGAN A., SHAH R. & NANDA N. (2024). Gemma scope : Open sparse autoencoders everywhere all at once on gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop : Analyzing and Interpreting Neural Networks for NLP*, p. 278–300.

[29] MALLIA A., SUEL T. & TONELLOTTO N. (2024). Faster learned sparse retrieval with block-max pruning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 2411–2415.

[30] MUHAMED A., DIAB M. & SMITH V. (2025). Decoding dark matter : Specialized sparse autoencoders for interpreting rare concepts in foundation models. In *Findings of the Association for Computational Linguistics : NAACL 2025*, p. 1604–1635.

[31] NG A. *et al.* (2011). Sparse autoencoder.

[32] NGUYEN T., CHATTERJEE S., MACAVANEY S., MACKIE I., DALTON J. & YATES A. (2024). Dyvo : Dynamic vocabularies for learned sparse retrieval with entities. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 767–783 : Association for Computational Linguistics. DOI : 10.18653/v1/2024.emnlp-main.45.

[33] NGUYEN T., LEI Y., JU J.-H., YANG E. & YATES A. (2025). Milco : Learned sparse retrieval across languages via a multilingual connector. *arXiv preprint arXiv :2510.00671*.

[34] PARIA B., YEH C.-K., YEN I. E., XU N., RAVIKUMAR P. & PÓCZOS B. (2020). Minimizing flops to learn efficient sparse representations.

[35] PARK S., KIM T. & KO Y. (2025). Decoding dense embeddings : Sparse autoencoders for interpreting and discretizing dense retrieval. *arXiv preprint arXiv :2506.00041*.

[36] RAJAMANOHARAN S., CONMY A., SMITH L., LIEBERUM T., VARMA V., KRAMÁR J., SHAH R. & NANDA N. (2024a). Improving dictionary learning with gated sparse autoencoders. *arXiv preprint library*.

[37] RAJAMANOHARAN S., LIEBERUM T., SONNERAT N., CONMY A., VARMA V., KRAMÁR J. & NANDA N. (2024b). Jumping ahead : Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint library*.

[38] ROBERTSON S. E., WALKER S., JONES S., HANCOCK-BEAULIEU M. & GATFORD M. (1994). Okapi at trec-3. In *Text Retrieval Conference*.

[39] SANH V., DEBUT L., CHAUMOND J. & WOLF T. (2019). Distilbert, a distilled version of bert : smaller, faster, cheaper and lighter. *arXiv preprint library*.

[40] 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.

[41] SANTHANAM K., KHATTAB O., SAAD-FALCON J., POTTS C. & ZAHARIA M. (2022b). Colbertv2 : Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the*

Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), p. 3715–3734.

[42] SHAMIR G. & LIN D. (2022). Reproducibility in deep learning and smooth activations. https://research.google/blog/reproducibility-in-deep-learning-and-smooth-activations/. Google Research Blog, April 5, 2022.

[43] SKEAN O., AREFIN M. R., ZHAO D., PATEL N., NAGHIYEV J., LECUN Y. & SHWARTZ-ZIV R. (2025). Layer by layer : Uncovering hidden representations in language models. *International Conference on Machine Learning (ICML)*.

[44] TOUVRON H., LAVRIL T., IZACARD G., MARTINET X., LACHAUX M.-A., LACROIX T., ROZIÈRE B., GOYAL N., HAMBRO E., AZHAR F., RODRIGUEZ A., JOULIN A., GRAVE É. & LAMPLE G. (2023). Llama : Open and efficient foundation language models. *arXiv preprint library*, **abs/2302.13971**.

[45] WANG L., YANG N., HUANG X., JIAO B., YANG L., JIANG D., MAJUMDER R. & WEI F. (2022). Simlm : Pre-training with representation bottleneck for dense passage retrieval. *arXiv preprint library*.

[46] WEN T., WANG Y., ZENG Z., PENG Z., SU Y., LIU X., CHEN B., LIU H., JEGELKA S. & YOU C. (2025). Beyond matryoshka : Revisiting sparse coding for adaptive representation. *arXiv preprint arXiv :2503.01776*.

[47] YANG A., LI A., YANG B., ZHANG B., HUI B., ZHENG B., YU B., GAO C., HUANG C., LV C. *et al.* (2025). Qwen3 technical report. *arXiv preprint arXiv :2505.09388*.

[48] ZAMANI H., DEHGHANI M., CROFT W. B., LEARNED-MILLER E. G. & KAMPS J. (2018). From neural re-ranking to neural ranking : Learning a sparse representation for inverted indexing. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*.

[49] ZENG H., ZAMANI H. & VINAY V. (2022). Curriculum learning for dense retrieval distillation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 1979–1983.

[50] ZHANG X., THAKUR N., OGUNDEPO O., KAMALLOO E., HERMELO D. A., LI X., LIU Q., REZAGHOLIZADEH M. & LIN J. (2023). Miracl : A multilingual retrieval dataset covering 18 diverse languages. *Transactions of the Association for Computational Linguistics*, **11**, 1114–1131.

A Détails d'implémentation

Nous conduisons toutes nos expériences avec un seul GPU A100 [7].

Pendant l'entraînement, nous utilisons l'optimiseur AdamW avec un taux d'apprentissage de 5e-5, en utilisant 768 documents par lot (≈60k tokens) pour 160k étapes.

Pour initialiser les paramètres du SAE, nous suivons les travaux précédents (18; 37) et fixons les biais $\mathbf{b}_{enc}$ et $\mathbf{b}_{dec}$ à zéro et la matrice de projection de l'encodeur à la transposée de la matrice de projection du décodeur ($\mathbf{W}_{enc} = \mathbf{W}_{dec}^{\top}$), mais ne les lions pas pendant l'entraînement du SAE. Tout au long de l'entraînement, nous re-normalisons les vecteurs latents à la norme unité après chaque étape.

Nous affinons SAE-SPLADE pour 240k étapes, avec un taux d'apprentissage de 2e-5, $\lambda_{\text{KL}} = 1$, $\lambda_{\text{MSE}} = 0.05$, $\lambda_{\text{flops-d}} = 0.04$ et $\lambda_{\text{flops-d}} = 0.06$. Chaque lot d'entraînement est composé de 32 requêtes, avec 8 négatifs difficiles pour chacune. Pour les deux étapes d'entraînement et pour l'évaluation, nous tronquons les requêtes à 32 tokens et les documents à 256 tokens. [8] Nous utilisons le noyau Triton fourni par Gao *et al.* (18) pour maintenir une vitesse d'entraînement et d'indexation comparable, même en augmentant la taille de l'espace latent. Avec cette configuration, l'entraînement du SAE prend environ 35 heures et l'optimisation RI environ 24 heures (≈ 2,5 jours pour entraîner SAE-SPLADE de zéro).

B Étude Préliminaire sur la Conception du SAE pour SAE-SPLADE

Dans cette section, nous conduisons des expériences préliminaires pour sélectionner la meilleure variante de SAE (c'est-à-dire choisir entre TopK SAE ou ses alternatives) et la méthode d'entraînement du SAE à utiliser dans l'article. Pour toutes les variantes, nous fixons le nombre de latents SAE à $M = 2^{16}$ (environ deux fois la taille du vocabulaire BERT pour tenir compte des latents potentiellement morts) et utilisons le dernier état caché du PLM de base comme entrée (ce choix est validé ultérieurement). Nous entraînons le SAE avec $k_{\text{SAE}} = 8$ et affinons SAE-SPLADE avec $k_{\text{SPLADE}} = 8$. Lorsque $k_{SPLADE} = k_{SAE}$, pour réduire l'encombrement, nous utilisons simplement la notation k.

B.1 Comparaison des choix de conception pour l'entraînement des SAE

Les travaux sur les SAE ont exploré diverses techniques d'entraînement de base, parmi les plus importantes étant la normalisation de l'entrée SAE (28; 37; 4; 18) et la diversité des données d'entraînement (30) – nous en avons donc testé l'impact. Pour normaliser l'entrée, nous précalculons une représentation moyenne $\overline{\mathbf{h}}$ et une norme moyenne σ sur un ensemble de représentations de tokens échantillonnées aléatoirement. Pendant l'entraînement du SAE et l'affinage de SAE-SPLADE, nous utilisons $\overline{\mathbf{h}}$ et σ pour normaliser $\mathbf{h}$ comme : $\frac{\mathbf{h}-\overline{\mathbf{h}}}{\sigma}$, et re-mettons à l'échelle la représentation

7. PyTorch 2.8.1, HuggingFace transformers 4.37

8. À l'exception du jeu de données ArguAna dans BEIR, où les requêtes ont également été tronquées à 256 tokens, similairement à ColBERT et SPLADE, car les requêtes elles-mêmes sont de longs documents.

parcimonieuse originale $\mathbf{w}$ pour SAE-SPLADE avec σ pour préserver l'amplitude du vecteur. Pour les données d'entraînement, suivant Park *et al.* (35) qui entraîne le SAE avec des textes de requêtes et de documents, nous évaluons si la reconstruction à la fois des requêtes et des documents lors du pré-entraînement du SAE peut aider SAE-SPLADE.

Nous avons observé qu'aucune modification n'apporte de bénéfice clair, ni en efficacité ni en efficience, à notre modèle. Cela montre que bien que la normalisation et l'utilisation de meilleures données d'entraînement aient toutes deux été prouvées bénéfiques dans d'autres contextes, cela ne se transfère pas à la tâche RI. En particulier, nous observons que la normalisation a induit une légère augmentation du nombre de latents « morts » (environ 5

B.2 Couche cachée optimale de l'encodeur SAE

TABLE 2 – The ablation results for backbone models ($k = 8$, TopK SAE) at various layers.

Layers	MSM	TREC-DL	LoTTE	QD-flops ↓	Avg. D Len ↓	$\Delta\mathbf{E}^2$ ↑	Anisot. ↓	Base Anisot. ↓
Layer 6 + $\mathcal{T}$	37.6	71.0	68.4	0.72	**86**	18.7	0.64	0.13
Layer 6	37.6	**72.1**	**68.8**	**0.67**	109	**18.8**	**0.40**	0.38
Layer 5	**37.7**	72.1	68.8	0.86	118	18.7	0.61	0.61
Layer 4	37.2	71.8	68.0	1.01	121	18.0	0.73	0.57
Layer 3	36.8	70.2	67.3	1.13	114	17.5	0.69	0.49

Skean *et al.* (43) ont montré que différentes couches de l'architecture Transformer jouent des rôles différents dans la correspondance sémantique. Cependant, cela ne pouvait pas être étudié pour les modèles LSR car ils dépendent de la tête MLM, qui contient non seulement une couche de projection de vocabulaire, mais aussi une couche de transformation $\mathcal{T}$ (linéarité suivie d'une unité non linéaire GeLU), entraînée à filtrer les informations non pertinentes pour la Prédiction de Token Masqué à partir de l'état caché de la dernière couche.

En revanche, pour SAE-SPLADE, nous pouvons décider sur quelle couche appliquer le SAE et projeter ses états cachés vers l'espace latent. Plus précisément, au lieu d'entraîner le SAE directement sur les derniers états cachés $\mathbf{h}_i^L$ de l'encodeur de base, nous testons son entraînement en utilisant les états cachés $\mathbf{h}_i^l$ de la couche l et les derniers états cachés « transformés » $\mathcal{T}(\mathbf{h}_i^L)$.

Comme le montre Table 2, nous observons d'abord que l'utilisation de $\mathcal{T}(\mathbf{h}_i^L)$ à la place de $\mathbf{h}_i^L$ nuit à toutes les métriques sauf à la sparsité du document (Avg. Doc Len) – démontrant qu'il est préférable de supprimer entièrement la tête MLM. Deuxièmement, les performances ne sont pas impactées jusqu'à la couche 3 à partir de laquelle une baisse notable se produit. La comparaison du $\mathbf{E}^2$ confirme en outre que l'utilisation de la Couche 6 atteint le meilleur compromis.

En laissant de côté la première configuration (couche 6 + $\mathcal{T}$), nous observons que la réduction du nombre de couches augmente systématiquement les QD-FLOPs du modèle entraîné, ainsi que (mais dans une moindre mesure) la longueur moyenne des documents. Pour mieux comprendre cet effet, nous avons calculé l'anisotropie (20) des représentations de tokens en mesurant la similarité moyenne entre des paires aléatoires de représentations de tokens dans plusieurs contextes pour chaque modèle entièrement entraîné à partir de la couche l. L'hypothèse est que plus l'anisotropie est élevée, plus il est difficile pour l'encodeur SAE de distinguer correctement les différents concepts, ce qui est compensé par une augmentation des QD-FLOPs et une diminution de la sparsité du document. Nous

rapportons dans Table 2 l'anisotropie à la couche utilisée par l'encodeur SAE, et observons que c'est effectivement le cas.

B.3 Impact de la taille du vocabulaire

(21) montrent qu'un nombre plus élevé de latents est corrélé à de meilleures performances RI. Nous avons testé la validité de cette affirmation pour SAE-SPLADE avec trois tailles de vocabulaire latent M, à savoir 2^{15}, 2^{16} et 2^{17}. Nous n'avons observé qu'une légère augmentation d'efficacité sur TREC-DL (environ 0,3 point en doublant la taille du vocabulaire). Nous avons également observé des ratios de *nœuds morts* similaires, ce qui montre que le modèle avec une taille de vocabulaire plus élevée tend à mieux utiliser l'espace latent. Cependant, nous n'avons pas observé d'écart entre les QD-FLOPs et Avg. Doc Len. Enfin, comme un M plus grand entraîne une latence d'encodage plus élevée, nous avons utilisé $M = 2^{16}$ dans toutes les expériences ultérieures.

B.4 Impact de k et Variantes du SAE

Le TopK SAE présente une limitation : chaque vecteur de token de document a au plus k dimensions latentes actives (>0), ce qui peut être sous-optimal pour SAE-SPLADE. En même temps, choisir une bonne valeur pour k est difficile : d'une part, si k est trop petit, le modèle peut ne pas réussir à capturer la sémantique complexe de l'entrée, dégradant les performances RI. D'autre part, si k est trop grand, il peut inclure des dimensions latentes non pertinentes, augmentant ainsi la latence de la requête.

Pour trouver la valeur optimale k_{SAE} pour SAE-SPLADE, nous faisons varier la valeur de k_{SAE} sur une large plage, de $k_{SAE} = 2$ à 32, pour explorer les compromis potentiels entre efficacité et efficience. Chaque SAE-SPLADE est affiné en utilisant $k_{SPLADE} = k_{SAE}$. Nous examinons également le cas où $k_{SPLADE} = M = 2^{16}$ avec $k_{SAE} = 8$ pour maximiser le nombre potentiel de latents actifs pendant l'affinage RI tout en maintenant le nombre d'activés faible pour la reconstruction SAE.

En plus du choix de k, il existe diverses options concurrentes pour entraîner notre encodeur. En plus du TopK SAE, nous analysons également les résultats du Hierarchical TopK SAE (2), qui impose des dimensions latentes désenchevêtrées à plusieurs niveaux de sparsité, et du Matryoshka TopK SAE (7), qui encourage les latents à capturer la sémantique à différentes granularités. L'avantage potentiel de ces modèles SAE est que nous pouvons apprendre l'encodeur avec un k_{SAE} plus élevé avant l'affinage avec un k_{SPLADE} plus bas puisque les latents sont structurés. Ainsi, nous utilisons initialement $k_{\text{SAE}} = 32$ pour le Hierarchical TopK SAE et le Matryoshka TopK SAE [9].

Table 3 montre le rôle crucial du choix de k dans le contrôle à la fois de la sparsité du document (Avg. D Len) et de l'efficience du modèle (QD-FLOPs). Naturellement, la réduction de la sparsité (c'est-à-dire l'augmentation de k_{SAE} et k_{SPLADE}) améliore systématiquement l'efficacité de la recherche, soulignant le compromis entre efficience et performance. Deuxièmement, il n'y a pas d'avantage clair à utiliser des alternatives au TopK : la même valeur de k_{SPLADE} conduit à des performances similaires en termes d'efficacité et d'efficience (meilleur $\Delta\mathbf{E}^2 \approx 18,7$ quelle que soit la variante SAE). Comme le Hierarchical TopK présentait des performances plus prometteuses, nous

9. En suivant le travail original (7), nous utilisons $M_i = (2048, 6144, 14336, 30720, 65536)$, où M_i représente la taille du vocabulaire pour chaque modèle SAE imbriqué.

TABLE 3 – The ablation results of SAE various for different k. The best metrics among the SAE variants are in bold, while we underlined the best metrics sharing the same k_{SPLADE}. We highlight the three model we present in the main result table.

k_{SAE}	k_{SPLADE}	MSM	TREC-DL	LoTTE	QD-FLOPs ↓	Avg. D Len ↓	$\Delta\mathbf{E}^2$ ↑
TopK SAE							
2		35.2	67.6	66.0	**0.15**	**37**	16.9
4		37.1	71.5	68.2	0.33	66	18.6
8		37.6	72.1	68.8	0.67	109	**18.8**
16		**38.2**	72.3	69.4	1.37	191	18.7
32		37.8	72.6	**70.6**	2.67	316	16.9
8	M	38.1	**73.5**	70.3	5.36	369	9.5
Hierarchical TopK SAE							
32	2	34.0	67.0	65.4	**0.13**	**31**	15.7
	4	37.0	71.2	68.1	0.30	62	18.5
	8	37.5	71.2	69.2	0.64	117	**18.7**
	16	37.8	71.6	69.5	1.43	208	18.2
	32	37.9	71.7	70.0	2.71	308	16.9
	M	**38.0**	**71.9**	**70.3**	4.57	387	13.7
128	4	37.0	71.2	67.5	0.32	56	18.5
	8	37.3	71.8	69.0	0.72	106	18.4
	16	38.0	71.9	69.6	1.52	192	18.3
Matryoshka TopK SAE							
32	2	34.5	67.5	65.2	**0.14**	**32**	16.2
	4	37.0	71.5	67.8	0.32	66	18.5
	8	37.5	72.0	68.9	0.73	124	**18.6**
	16	37.7	72.2	69.6	1.50	209	18.0
	32	37.7	72.8	69.6	2.60	298	16.9
	M	**37.7**	**73.1**	**70.0**	4.84	408	12.3

l'avons également entraîné avec un $k_{SAE} = 128$ plus grand pour étudier si un plus grand pool de latents structurés pourrait améliorer les performances de SAE-SPLADE. Nous observons qu'une fois de plus, il n'y a pas d'avantage clair.

Comme illustré dans Figure 2, et sur la base de notre $\Delta\mathbf{E}^2$ proposé, nous observons que l'utilisation de $k = 4$, $k = 8$ ou $k = 16$ atteint le meilleur compromis entre efficacité et sparsité pour le TopK SAE (car ils ont tous un $\Delta\mathbf{E}^2 > 18$).

Pour résumer, bien que les variantes du TopK SAE améliorent théoriquement la structure de l'espace latent, il n'y a aucun impact sur les performances de SAE-SPLADE. Nous utilisons donc la variante la plus simple, à savoir le TopK SAE, dans les sections suivantes. Nous fixons également $k_{SAE} = 8$ dans le reste de cet article car cela atteint le meilleur $\Delta\mathbf{E}^2$.

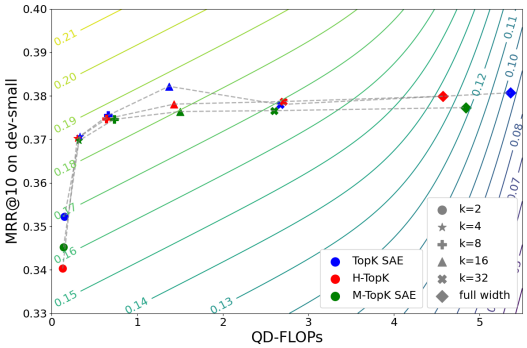

FIGURE 2 – Performance de SAE-SPLADE dépendant de la valeur de k_{SPLADE} et des variantes TopK SAE ; les courbes de niveau correspondent au $\Delta\mathbf{E}^2$ (voir la définition de la métrique dans l'Éq. (6)).

C Ablations

Dans cette section, nous étudions les hyperparamètres qui influencent l'efficacité et l'efficience des modèles SAE-SPLADE, notamment le pré-conditionnement du vocabulaire (MLM, SAE ou aléatoire), le coefficient de régularisation FLOPs et l'effet du passage à un encodeur de base plus grand.

C.1 Effet du Pré-Entraînement du Vocabulaire

Puisque la prédiction de tokens masqués dans les MLM et la reconstruction de représentations de tokens dans les SAE correspondent à des objectifs d'entraînement différents, nous étudions l'apprentissage de modèles LSR construits sur ces deux types de vocabulaires (par exemple, SPLADE vs. SAE-SPLADE) en comparant leur efficacité et leur efficience de récupération. Pour évaluer davantage le rôle du pré-entraînement SAE dans le cadre SAE-SPLADE, nous supprimons l'étape d'entraînement SAE et affinons le modèle à partir de projections aléatoires. Nous appliquons les mêmes contraintes top-k sur les représentations de tokens, lors de l'entraînement et de l'inférence pour les trois vocabulaires ($k = 8$, et sans contrainte pour correspondre à SPLADE). Pour les modèles basés sur le vocabulaire SAE et aléatoire, nous utilisons une taille de vocabulaire de $M = 2^{16}$.

D'après Table 4, nous observons que quel que soit le modèle, l'utilisation de $k = 8$ diminue l'efficacité et augmente l'efficience, comme observé précédemment. Plus intéressant encore, nous pouvons tirer deux conclusions. Premièrement, l'utilisation d'une approche de sparsification top-k est généralement bénéfique pour SPLADE, car elle augmente substantiellement son efficience – en diminuant de 28 le nombre de dimensions actives pour les documents, et de 0.8 QD-FLOPs en moyenne –, et améliore son efficacité hors-domaine (+0.5 pour nDCG@10) au détriment de celle en domaine (-0.5 nDCG@10). Deuxièmement, l'entraînement à partir d'une projection aléatoire n'est possible qu'avec une approche de sparsification top-k – sans quoi les QD-FLOPs deviennent trop élevés (12.88 sans, contre 1.16 avec) pour une utilisation pratique. Néanmoins, le pré-entraînement avec SAE améliore à la fois l'efficacité et l'efficience par rapport au vocabulaire aléatoire.

TABLE 4 – The ablation results of the type of the Vocabulary. "-" represent the model too expansive to evaluate. The best results shared by the same vocabulary type are underlined, while the best results obtained under the same term-masking strategy ($k = 8$ or no mask) are shown in bold.

Model		MSM	TREC-DL	LoTTE	QD-flops ↓	Avg. D Len ↓	ΔE^2↑
MLM	$k = 8$	37.5	71.5	**69.4**	0.67	**90**	18.7
	$k = M$	37.7	72.0	68.9	**1.47**	**118**	**18.1**
Rand.	$k = 8$	36.8	71.6	68.2	1.16	134	17.5
	$k = M$	-	71.8	69.4	12.88	982	-
SAE	$k = 8$	**37.6**	**72.1**	68.8	**0.66**	111	**18.8**
	$k = M$	**38.1**	**73.5**	**70.3**	5.36	369	9.5

C.2 Coefficient de Régularisation FLOPs

La régularisation FLOPs est un mécanisme efficace pour contrôler le compromis efficience-efficacité des modèles LSR, mais son interaction avec k n'est pas triviale. Nous avons donc mené des expériences pour étudier leurs interactions en faisant varier à la fois k et le coefficient de régularisation $\lambda_{flops-d}$ et $\lambda_{flops-q}$. Les résultats sont présentés dans Figure 3.

Nos résultats indiquent que, comme attendu, l'efficacité augmente avec des valeurs de k plus grandes et diminue avec une régularisation FLOPS plus forte, avec la tendance inverse observée pour l'efficience. De plus, la contrainte top-k joue un rôle crucial dans le contrôle à la fois de la sparsité et des QD-FLOPs. Nous constatons que le modèle avec un petit k et un petit coefficient de régularisation FLOPs (0.25x) atteint simultanément une meilleure sparsité que les modèles avec un k plus grand, même avec un coefficient de régularisation FLOPs élevé (4x). Par ailleurs, à mesure que k augmente, l'impact de la régularisation FLOPs sur le modèle devient plus prononcé. En général, la contrainte TopK est plus influente que la régularisation FLOPS, car nous atteignons une meilleure efficacité pour un budget de sparsité/QD-FLOPs donné. Plus précisément, pour des QD-FLOPs similaires, le modèle entraîné avec un k plus petit est plus performant et possède une taille d'index plus réduite (Avg. D Len). Cette observation suggère que, pour contrôler l'efficience du modèle, réduire k est préférable à augmenter le coefficient de régularisation FLOPs.

C.3 Effet de Mise à l'Échelle

Nous comparons également les performances de SAE-SPLADE ($k = 8$) et SPLADE ($k = 8$ et largeur complète) avec différentes tailles de modèle de base, spécifiquement DistilBERT (39) et BERT (11). Dans l'ensemble, le passage du PLM de base à un modèle plus grand produit peu de différence en performance en domaine, tout en apportant des améliorations constantes en contexte hors-domaine. Plus précisément, pour SAE-SPLADE, nous observons une amélioration moyenne de 0.5 points sur LoTTE (68.8 vs. 69.3). En comparaison, SPLADE affiche des gains plus importants lors du passage à un modèle de base plus grand, avec des améliorations dépassant 1.2 points (69.4 vs. 70.2 pour $k = 8$, et 68.9 vs. 70.1 pour $k = M$). Dans tous les cas, l'impact sur l'efficience reste négligeable, avec des différences en QD-FLOPs inférieures à 0.05. Ces résultats suggèrent que le passage à des modèles de base plus grands bénéficie principalement à la généralisation hors-domaine.

En conclusion, à travers toutes les études d'ablation, nous démontrons que SAE-SPLADE est robuste

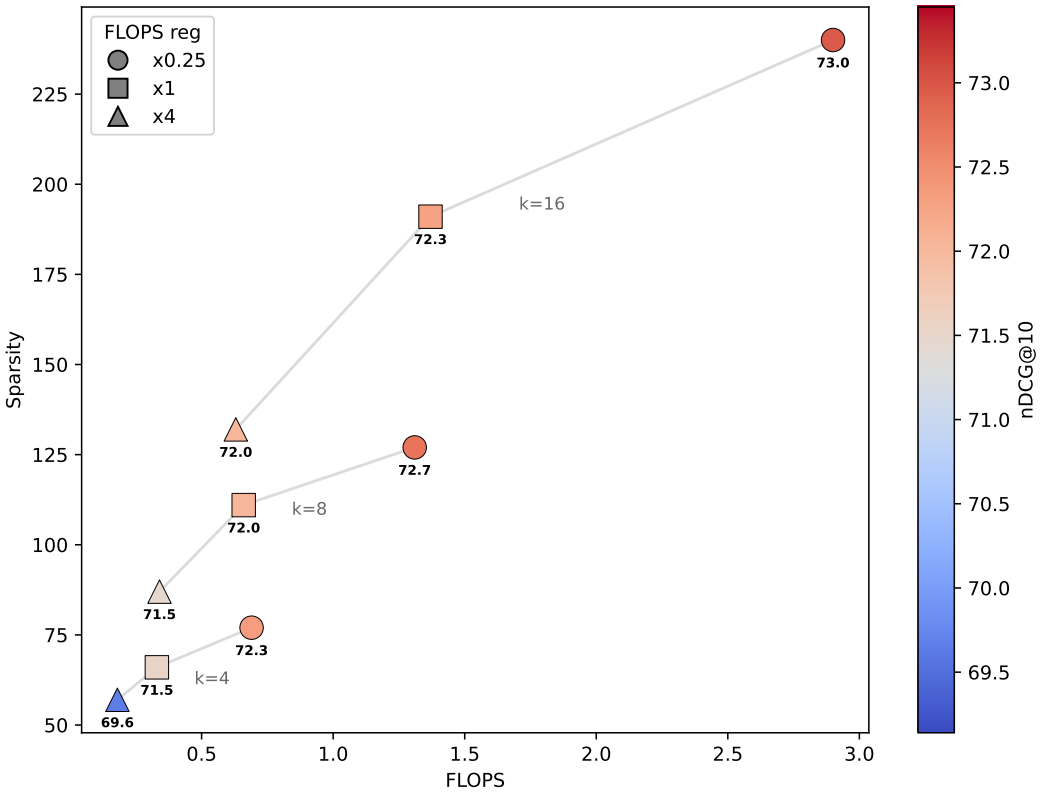

FIGURE 3 – Performance du modèle SAE-SPLADE (avec TopK SAE) en fonction de la valeur de $k = 4, 8, 16$ (en gris, reliés par des lignes) et du coefficient de régularisation FLOPs (forme).

sur un large éventail de paramètres d'entraînement et atteint des performances comparables à celles de SPLADE, en termes d'efficacité et d'efficience. Cela renforce nos affirmations sur **RQ1** et **RQ2**. De plus, nous constatons que l'application d'une contrainte top-k au niveau des tokens lors de l'entraînement et de l'inférence améliore l'efficience et l'efficacité de SPLADE.

D Résultats Multilingues

Pour répondre à **RQ3**, sur la question de savoir si la recherche parcimonieuse basée sur SAE présente une meilleure adaptabilité aux contextes multilingues que les modèles basés sur MLM (p.ex. SPLADE), nous avons conduit un ensemble d'expériences préliminaires. Plus précisément, nous utilisons un modèle de base DistilBERT multilingue [10] et entraînons un modèle SAE-SPLADE-Multilingue en suivant la même procédure que nos expériences principales. Comme référence, nous entraînons également un modèle SPLADE-Multilingue dans des conditions identiques. Pour les deux modèles, nous appliquons une contrainte top-k au niveau des tokens avec $k = 8$. [11] Tous les autres paramètres d'entraînement sont maintenus cohérents avec la configuration monolingue décrite dans la Section 4.1. Nous utilisons également les traductions automatiques de MS MSARCO, mMARCO (3) pour entraîner notre SAE multilingue et pour affiner SAE-SPLADE et SPLADE. Les évaluations sont effectuées sur l'ensemble dev-small de mMARCO et sur MIRACL (50), que nous utilisons comme référence hors domaine pour la recherche multilingue. Nous limitons notre entraînement et nos évaluations aux langues suivantes : {Arabe (ar), Anglais (en), Espagnol (es), Français (fr), Japonais (ja), Russe (ru), Chinois (zh)}, couvrant à la fois les langues latines et non latines. Cette sélection inclut des langues avec des termes lexicaux qui se chevauchent (p.ex. en/es/fr et ja/zh), ainsi que des langues avec des vocabulaires totalement disjoints (p.ex. ru et ar), nous permettant d'évaluer l'adaptabilité dans divers contextes linguistiques.

TABLE 5 – The multilingual performance on mMARCO dev small set. Results of IR performance are in MRR@10. We do not report the ar and ja performance for mColBERT as it is zero-shot in prior work. † : significantly better ($p < 0.05$) compare to our SPLADE baseline under the two-tailed Student's t-test. The best across all the baselines are in **bold** while the best among ours are underlined.

Model		ar	es	fr	ja	ru	zh	en	Avg.
Prior Works									
BM25	MRR@10	11.1	15.8	15.5	14.1	12.4	11.6	18.3	14.1
mColBERT	MRR@10	-	**30.1**	**28.9**	-	**25.0**	**24.6**	**35.2**	-
Ours									
SPLADE k=8	MRR@10	**19.5**	25.6	25.2	**23.9**	21.1	22.9	31.4	24.2
	QDFLOPs	5.22	2.52	2.68	4.14	3.87	3.11	2.14	3.38
SAE-SPLADE k=8	MRR@10	18.8	26.6†	25.8†	23.6	22.0†	23.1	30.7	**24.4**
	QDFLOPs	**3.94**	**1.66**	**1.88**	**3.20**	**3.02**	**2.09**	**1.63**	**2.49**

Les résultats sont rapportés dans Table 5 et Table 6. Sur les deux jeux de données, SAE-SPLADE atteint une efficacité de recherche légèrement meilleure que SPLADE tout en améliorant substantielle-

10. `distilbert/distilbert-base-multilingual-cased`
11. Nous avons également expérimenté avec SPLADE sans contrainte TopK, mais les QD-FLOPs résultants étaient trop élevés pour l'évaluation.

TABLE 6 – The multilingual performance on MICRAL dev set. Results of IR performance are in nDCG@10. † : significantly better ($p < 0.05$) compare to our SPLADE baseline under the two-tailed Student's t-test. The best across all the baselines are in **bold** while the best among ours are underlined.

Model		ar	es	fr	ja	ru	zh	Avg.
Prior Works								
BM25	nDCG@10	39.5	7.7	11.5	31.2	25.6	17.5	22.2
mContriever	nDCG@10	52.5	41.8	31.4	42.4	39.1	41.0	41.4
M3-Sparse (8)	nDCG@10	67.1	38.6	35.3	56.1	44.5	36.1	46.3
MILCO (33)	nDCG@10	**80.4**	**60.9**	**61.7**	**77.2**	**74.6**	**65.5**	**70.1**
Ours								
SPLADE k=8	nDCG@10	59.3	45.0	38.2	49.3	47.3	43.7	47.2
	QDFLOPs	5.61	3.35	2.83	5.75	6.02	3.74	4.55
SAE-SPLADE k=8	nDCG@10	58.6	44.4	38.4	50.8†	47.4	45.9†	47.5
	QDFLOPs	**4.21**	**2.16**	**1.97**	**4.34**	**4.56**	**2.29**	**3.26**

ment l'efficience. Ces résultats soutiennent notre affirmation que l'utilisation de vocabulaires dérivés de SAE améliore l'adaptabilité multilingue des modèles de recherche parcimonieuse par rapport aux vocabulaires basés sur MLM. Par rapport aux autres références de base, SAE-SPLADE surpasse M3-Sparse (8), malgré le fait que ce dernier s'appuie sur un modèle de base substantiellement plus grand. Cependant, un écart de performance subsiste entre notre approche et mColBERT (3), un modèle de recherche dense à vecteurs multiples avec un coût de calcul plus élevé, ainsi que MILCO (33), un modèle parcimonieux multilingue à l'état de l'art qui emploie un modèle de base plus grand et des stratégies d'entraînement et des architectures plus complexes (p.ex. alignement entre langues) [12].

TABLE 7 – The average overlap of vocabulary term over all the languages and the average length of the document representation of our baselines. We report the std using $\pm$.

Model	Voc. Size	Doc Avg. Len	Overlap
SPLADE $k = 8$	119547	$180_{\pm 45}$	$8.6_{\pm 9.0}$
SAE-SPLADE $k = 8$	65536	$163_{\pm 43}$	$11.5_{\pm 8.9}$

Nous analysons en outre les différences de représentation entre SAE-SPLADE et SPLADE. Plus précisément, nous échantillonnons aléatoirement 2 048 documents de MS MARCO, ainsi que les traductions correspondantes dans mMARCO, les encodons en utilisant les deux modèles, et calculons le nombre de termes de vocabulaire ou de latents activés qui se chevauchent, avec une valeur d'activation supérieure à 0, dans toutes les traductions d'un document. Les résultats dans Table 7 mettent en évidence deux observations clés. Premièrement, SAE-SPLADE présente un plus grand chevauchement des latents activés entre les langues tout en maintenant une meilleure sparsité, démontrant son avantage sur SPLADE dans l'apprentissage de représentations multilingues. Cela suggère que l'exploitation des latents dérivés de SAE fournit une meilleure représentation interlinguale que le fait de s'appuyer uniquement sur des vocabulaires lexicaux basés sur MLM. Deuxièmement, le ratio de chevauchement de 11,5 / 163 indique qu'une partie substantielle des latents SAE reste spécifique à la langue. Ce résultat suggère que l'apprentissage de représentations interlinguales en utilisant SAE seul est difficile et que des techniques d'adaptation interlinguale supplémentaires,

12. Nous notons que M3-Sparse et MILCO exploitent également les données d'entraînement de MIRACL pour la distillation, ce qui ne constitue pas une évaluation hors domaine ici.

telles que l'alignement des représentations parcimonieuses anglaises et multilingues utilisé dans MILCO (33), pourraient être très bénéfiques. Nous prévoyons que des stratégies interlinguales plus avancées pourraient être efficacement intégrées dans SAE-SPLADE pour améliorer davantage ses performances multilingues.

En conclusion, tous ces résultats et analyses apportent une réponse positive à **RQ3**.

E Analyse Qualitative

Nous avons vu qu'il est possible de connecter SAE avec des modèles LSR et avons proposé un modèle SAE-SPLADE qui est compétitif en termes d'efficacité et d'efficience avec les modèles SPLADE tout en étant plus efficient. La dernière question de recherche (**RQ4**) qui reste à étudier est de comprendre l'impact du remplacement de la tête MLM par un SAE sur le comportement de SPLADE. Notre hypothèse initiale était que nos latents SAE sont censés encapsuler des unités sémantiques distinctes, contrairement aux tokens, où un seul token pourrait parfois référer à plusieurs significations (polysémie) ou une seule signification pourrait être couverte par plusieurs tokens (synonymie). Dans cette section, nous fournissons une analyse qualitative sur la nature de la relation entre les latents SAE-SPLADE et les tokens SPLADE.

TABLE 8 – First examples (after sorting them by the longest tokens list length) of *synonym* token-latents pairs, grouped by their shared latent.

Latent ID	Synonym tokens list	Concept
6470	['dish', 'soup', 'pepper', 'ingredients', 'sauce',..., 'spices', 'flavors', 'flavour']	Cuisine
42574	['1885', '1880', '1870', '1884', '1881', ...'1879', '1875', '1877', '1874']	1870-1880s
48320	['fry', 'pork', 'stir', 'cooke', 'oven', 'microwave', ..., 'boil', 'grille', 'barbecue', 'roasted']	Cooking methods
24849	['1902', '1901', '1898', '1899', '1895', '1896',..., '1890s', '1900s']	Late 19th century
30794	['fry', 'pork', 'cooke', 'oven', 'microwave',..., 'roast', 'boil', 'grille']	Cooking methods
16100	['moon', 'planet', 'mars', 'orbit', 'planets', 'lunar', 'astronomy',...,'astronaut']	Astronomy
9610	['breath', 'breathing', '##hale', 'respiratory', ..., 'pneumonia', 'coughing']	Respiration

En particulier, nous sous-échantillonnons 10 000 documents de MS MARCO et les encodons avec les modèles SPLADE et SAE-SPLADE (TopK SAE, $k = 8$). Nous utilisons la même règle que pour les représentations multilingues, en équivalant la présence d'un token/concept à un poids supérieur à 0. Pour pouvoir extraire des signaux significatifs, nous énumérons le nombre d'occurrences de chaque token et latent sur l'ensemble du corpus, et supprimons ceux ayant moins de 5 occurrences. Pour chaque token restant t et latent l, nous estimons la distribution des deux probabilités conditionnelles $P(l|T = t)$ et $P(t|L = l)$.

Parmi notre ensemble de paires token-latent, nous avons en outre supprimé toute paire avec au moins une probabilité conditionnelle inférieure à 0,1, pour faciliter notre analyse et supprimer le bruit potentiel. Enfin, nous effectuons un test binomial sur les deux ensembles de distributions $P(L|T = t)$ et $P(T|L = l)$ pour vérifier si les valeurs de co-occurrence observées pour chaque paire sont anormales ou non, en conservant toutes les paires pour lesquelles l'hypothèse H0 est rejetée avec au moins 95% de confiance.

Nous définissons heuristiquement comme « synonymie » une paire token-latent où $P(t|l) \leq 0.4$ et $P(l|t) \geq 0.6$ (c'est-à-dire que le token correspond à l'unité sémantique couverte par le latent, mais n'est pas le seul), comme « polysémie » une paire où $P(l|t) \leq 0.4$ et $P(t|l) \geq 0.6$ (c'est-à-dire que

le token est ambigu et plusieurs latents couvrent ses différentes significations) et comme « identité » une paire où $P(l|t) \geq 0.6$ et $P(t|l) \geq 0.6$. Pour ce dernier, des paires token-latent illustratives impliquent des tokens tels que « canada », « york » et « february ». Nous soutenons que de tels latents SAE sont nécessaires pour maintenir la propriété de correspondance lexicale dans SAE-SPLADE.

Table 8 fournit quelques exemples de tokens synonymes, couverts par un seul latent SAE, tandis que Table 9 fournit des exemples de tokens polysémantiques d'une certaine manière disambiguisés par des latents. Dans l'ensemble, nous constatons que bien que chaque latent individuel dans SAE-SPLADE semble avoir sa propre unité sémantique (comme illustré dans Table 8), il y a des « conflits sémantiques » entre les latents, car certains ont tendance à couvrir la même signification sémantique (voir par exemple les différents latents couvrant les « Méthodes de cuisson » dans Table 8 ou les différents latents couvrant le même aspect du token « bank » dans Table 9).

Nous émettons l'hypothèse que cela peut être causé par quelques facteurs, le premier étant notre choix de SAE. Étant donné que nous avons utilisé l'approche TopK SAE plus simple, il est possible que notre espace latent ne soit pas aussi structuré qu'il pourrait l'être avec une autre variante. Deuxièmement, comme mentionné précédemment, cela pourrait être la conséquence des contradictions entre l'objectif d'apprentissage du SAE et l'affinage de la tâche RI. Bien que nous ne puissions pas tirer de conclusions définitives pour **RQ4**, nos résultats empiriques précédents fournissent tout de même de fortes preuves que leurs distributions respectives ne sont pas alignées, ce qui suggère que les deux modèles ne se comportent pas de la même façon. Nous laissons cette exploration pour de futurs travaux.

TABLE 9 – Illustrative examples of latents associated to different meanings of a single *polysemantic* token and the corresponding MS MARCO passages.

Token	Latent IDs	Passages
'spring'	5162	'Bonita Springs Tourism : Best of Bonita Springs. Bonita Springs, Florida. [...]'
		'[...] All King size mattresses use Split box springs. [...]'
	48743	'Apr Enlightenment may know no weather, but any pilgrimage is best taken in spring. [...]'
		' [...] these invertebrates make up most of a grey fox's diet in the spring. [...]
'bank'	23304, 44077, 5391, 9394	'1 The Bank was founded in 1934 as a privately owned corporation.[...]'
		'1. Check online for your bank's routing number. [...]'
	55773	'In the clip, Banks portrays a sassy real estate-obsessed version of herself [...]'
		'[...] East of Zebulon they provide access to the Outer Banks, US 64 via Rocky Mount, [...]'
		'[...] hurricanes are a major threat to the Outer Banks. [...]'