Actes de CORIA-TALN 2026
Nantes, France, 29 juin au 3 juillet 2026

Tous les tokens sont-ils utiles pour les modèles de langues ?

Eve Sauvage[1, 2] Cyril Grouin[1] Julien Tourille[2]
(1) LISN, 507 rue du Belvédère, 91405 Orsay, France
(2) EDF R&D, 7 Bd Gaspard Monge, 91120 Palaiseau, France
prenom.nom@lisn.fr, prenom.nom@edf.fr

RÉSUMÉ

La qualité des plongements textuels est essentielle pour les tâches en aval de leur utilisation, mais leur utilisation par les modèles Transformer est coûteuse en termes de calcul en raison de la complexité quadratique sur la longueur des séquences. Cela motive les méthodes de réduction des tokens. Parallèlement, des études indiquent que les plongements actuels peuvent représenter de manière sous-optimale les informations sémantiques. Nous étudions une stratégie de réduction des tokens lexicale, qui ne conserve que le premier token de chaque mot. Cette approche s'inspire d'observations linguistiques selon lesquelles les humains sont capables de comprendre un texte malgré l'élision partielle de mots. Nous évaluons notre méthode sur le Massive Textual Embedding Benchmark (MTEB). Nos résultats indiquent que la suppression des tokens de sous-mots finaux ne dégrade pas significativement les performances. Cela implique que ces tokens ajoutent une charge de calcul supplémentaire sans contribuer de manière substantielle à la qualité sémantique, et que leur suppression peut permettre aux modèles de traiter des entrées plus longues

ABSTRACT

Are all Tokens useful for Textual Embeddings ?

High-quality textual embeddings is essential for downstream tasks, but their generation via Transformer models is computationally costly due to quadratic complexity in sequence length. This motivates token reduction methods. Concurrently, studies indicate that current embeddings may sub-optimally represent semantic information. We investigate a linguistically-motivated strategy for token reduction, retaining only the first sub-word token of each word. This approach is inspired by linguistics observations that humans are able to understand text despite partial word elision. We evaluate our method on the Massive Textual Embedding Benchmark (MTEB). Our findings indicate that deleting trailing sub-word tokens does not significantly degrade performance. This implies that these tokens add computational overhead without substantially contributing to semantic quality, an their removal can enable models to process longer inputs.

MOTS-CLÉS : Plongements textuels, Elagage de tokens, Représentation dans les modèles.

KEYWORDS: Textual Embeddings, Token Pruning, Model Representation.

1 Introduction

La qualité et l'efficacité des plongements textuels sont cruciaux pour l'utilisation des modèles de langues. Cette qualité est souvent évaluée à l'aune de la similarité textuelle. Alors que des modèles récents tels que ColBERT (Khattab & Zaharia, 2020; Santhanam *et al.*, 2022) ou Qwen3 (Zhang *et al.*,

Actes des 33ème Conférence sur le Traitement Automatique des Langues Naturelles.
Volume 1 : articles scientifiques originaux, pages 595–609.
Cette œuvre est mise à disposition sous licence Attribution 4.0 International.

2025) font progresser l'état de l'art dans la capture des similitudes sémantiques fines entre les textes, un défi persiste dans le domaine de la recherche d'informations (IR). Ce défi consiste à trouver un compromis entre le coût informatique élevé des modèles neuronaux riches sur le plan sémantique et l'efficacité des méthodes traditionnelles basées sur des statistiques (par exemple BM25) qui manquent de compréhension sémantique profonde. Dans cet article, nous proposons une méthode permettant de préserver la fidélité sémantique des plongements neuronaux tout en réduisant considérablement la charge calculatoire grâce à l'élagage des séquences d'entrée des modèles de langues.

Notre approche s'inspire d'observations issues de la linguistique. Dans plusieurs langues, le sens est préservé malgré des phénomènes tels que l'apocope, où la fin d'un mot est omise (par exemple, *photographe* devient *photo*, *cinématographe* devient *cinéma*). Cela suggère que la plupart des informations sémantiques peuvent être contenues dans un seul morphème. Ce principe trouve un parallèle dans le comportement des modèles. Plusieurs articles de recherche, étudiant les mécanismes d'attention dans les grands modèles linguistiques, montrent que tous les tokens ne contribuent pas de manière égale à la représentation finale d'un modèle (Vig, 2019). Sur la base de ces observations, nous émettons l'hypothèse que l'essentiel de la sémantique d'un texte peut être capturé en ne conservant que les premiers jetons, en anglais *tokens*, de chaque mot. Nous appelons cette stratégie « *entrée apocopée* » (voir figure 1), que nous comparons à une sélection aléatoire des tokens. Nous pensons qu'une telle réduction de la longueur des séquences permettra de réaliser des économies de calcul tout en ayant un impact marginal sur les performances en aval.

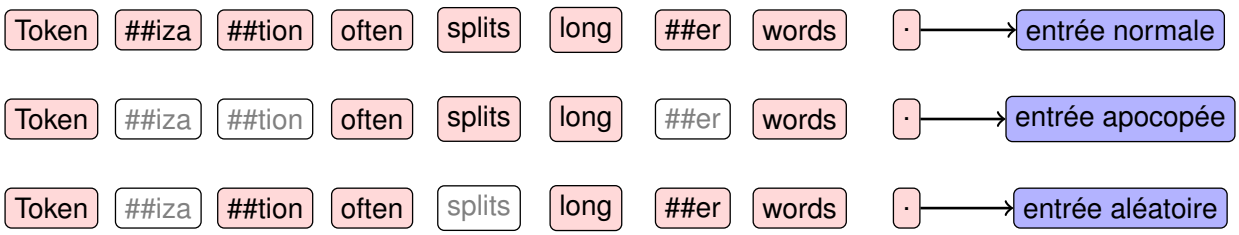

FIGURE 1 – Sélection des tokens selon la configuration d'entrée (normale, apocopée, aléatoire)

Nous évaluons notre méthode à l'aide du benchmark Massive Textual Embedding Benchmark (MTEB) (Muennighoff *et al.*, 2023), qui couvre un large éventail de tâches. Nous comparons nos résultats sur quatre jeux de données en anglais et trois jeux de données en français. Comme présenté dans la figure 1, nous menons des expériences d'ablation en utilisant une sélection aléatoire de tokens afin de nous assurer que le gain de performance est attribuable à la position des tokens sélectionnés.

Nos contributions sont les suivantes : (i) Nous étudions l'impact de l'ablation de tokens pour les modèles de langues, (ii) nous proposons une méthode permettant de réduire efficacement les séquences tout en préservant les performances. Nos résultats montrent des différences de performance minimes par rapport à l'utilisation de séquences complètes, ce qui confirme que notre méthode permet de réduire la longueur des séquences tout en préservant l'intégrité sémantique de l'entrée.

Nous présentons nos travaux en évoquant d'abord les travaux connexes (section 2), puis en présentant la méthodologie (section 3), notamment la méthode d'élagage 3.1, les tâches 3.2 et les modèles choisis 3.3 pour finir par les résultats (section 4) obtenus en terme de réduction de séquence 4.1, en terme de gain de temps de calcul 4.2 et en terme de performances 4.3.

2 Travaux connexes

Le coût de calcul des modèles Transformer, qui évolue de manière quadratique avec la longueur de la séquence (Vaswani *et al.*, 2017), a fait l'objet de plusieurs travaux de recherche sur l'efficacité des modèles. Deux axes de recherche ont été explorés : la modification du mécanisme d'attention et la réduction de la longueur de la séquence d'entrée.

Architectures d'attention efficaces Les modèles tels que BIGBIRD (Zaheer *et al.*, 2021), LINFORMER (Wang *et al.*, 2020) ou LONGFORMER (Beltagy *et al.*, 2020) reposent sur l'hypothèse selon laquelle l'attention « all-to-all » n'est pas essentielle pour traiter des séquences longues. En utilisant une attention éparse, ces architectures permettent de gagner en efficacité computationnelle. En effet, les éléments du mécanisme d'attention peuvent être éliminés tout en conservant des représentations pertinentes pour les tâches en aval, avec une dégradation minimale des performances.

Stratégie de réduction de séquences Une alternative pour atténuer le coût computationnel de la complexité quadratique consiste à réduire le nombre de tokens traités par le modèle. Cela peut être abordé à plusieurs étapes. L'utilisation d'un vocabulaire plus large, comme on le voit dans des modèles tels que TRANSFORMER-XL (Dai *et al.*, 2019), permet d'obtenir des séquences de tokens plus courtes. D'autres travaux récents ont également montré qu'une augmentation de la taille du vocabulaire peut améliorer les performances du modèle (Tao *et al.*, 2024). D'autres méthodes se concentrent sur la réduction de la séquence après tokenisation. Des méthodes basées sur les représentations vectorielles permettent de réduire le nombre de tokens à traiter. Zong & Piwowarski (2025) montrent qu'un sous-ensemble de tokens peut suffire à transmettre la sémantique d'un document pour les modèles d'interaction tardive. D'autres travaux ont exploré la transformation des tokens, en les appariant à leurs voisins les plus proches dans l'espace latent via le clustering (Hanus *et al.*, 2025).

L'élagage de tokens repose sur l'idée que tous les tokens d'une séquence ne contribuent pas de manière égale à la prédiction finale. PoWER-BERT (Goyal *et al.*, 2020) apprend à identifier et éliminer progressivement les tokens redondants au fil des couches d'un modèle de type BERT, en s'appuyant sur les scores d'attention pour estimer l'importance de chaque token. SpAtten (Wang *et al.*, 2021) adopte une logique similaire et supprime en cascade les tokens et les têtes d'attention les moins sollicités, permettant une réduction dynamique du coût de calcul proportionnelle à la complexité intrinsèque de chaque entrée. TR-BERT (Ye *et al.*, 2021) formule quant à lui la décision de suppression comme un problème d'apprentissage par renforcement, entraînant un agent à choisir à chaque couche quels tokens conserver selon un compromis performance/efficacité explicitement défini. D'autres approches se basent sur le mécanisme d'attention pour sélectionner les tokens à élaguer (Kim *et al.*, 2022). Ces méthodes se basent sur des seuils appris pour déterminer quels tokens conserver. De même, certaines méthodes suggèrent de modifier les représentations internes du modèle afin de fusionner les tokens et de réduire la longueur de la séquence (Sauvage *et al.*, 2025).

3 Méthodologie

3.1 Élagage des tokens

Nous voulons estimer les performances des modèles confrontés à la modification de leur entrée sur des critères linguistiques, et en particulier à la suppression des tokens de fin lorsque la tokénisation coupe un mot en plusieurs parties. Nous comparons ce processus à l'apocope et faisons référence à la suppression des tokens de fin sous le nom d'« entrée apocopée ».

Nous déterminons les limites des unités à élaguer sur les espaces, motivés par l'entrainement des tokenizers qui utilisent également cette limite, plutôt que sur les caractères spéciaux ajoutés pour leur compréhension (*e.g.* '##'). Cette méthode nous permet de facilement traiter les différents types de tokenizers sans changer les conditions d'élagage. Nous obtenons des résultats similaire en utilisant les caractères spéciaux marquant tokens subséquent pour `all-MiniLM-L6-v2`.

Nous comparons ces résultats avec une expérience de contrôle afin d'évaluer l'impact de la position des tokens sélectionnés sur la dégradation des performances. Dans cette expérience de contrôle, nous élaguons aléatoirement les tokens dans la séquence en fonction d'un ratio empiriquement déterminé. Nous fixons ce ratio à 0,30 après avoir observé la réduction moyenne de la longueur des séquences dans une expérience préliminaire avec l'élagage des tokens finaux. Nous notons qu'avec cette méthode, des mots entiers peuvent manquer dans la phrase, ce qui peut avoir un impact négatif sur les performances du modèle. Nous appelons cette expérience "entrée aléatoire".

Nous réalisons également l'expérience sans modifier l'entrée du modèle à titre de référence. Nous appelons ces expériences « entrée normale » car les entrées n'ont pas été modifiées. Bien que nous ne modifiions pas l'entrée dans le cadre de l'expérience à entrée simple, celle-ci sera traitée comme une entrée apocopée ou aléatoire afin de garantir l'équité des trois configurations.

3.2 Sélection des tâches

Pour évaluer les performances des plongements de modèles générées à l'aide de la transformation d'entrée décrite dans la sous-section précédente, nous nous appuyons sur le Massive Textual Embedding Benchmark (MTEB) (Muennighoff *et al.*, 2023). Le MTEB est un benchmark complet permettant d'évaluer les modèles de plongements de texte avec une grande variété de tâches. Parmi les huit tâches disponibles dans MTEB, nous en sélectionnons quatre en anglais et trois en français, en utilisant un seul ensemble de données représentatif pour chacune d'entre elles. Conformément au protocole du benchmark, chaque tâche est évaluée à l'aide de son indicateur d'évaluation principal spécifique. Nous utilisons le code d'évaluation officiel afin de garantir que nos résultats soient directement comparables au classement public.

— **Classification** : évalue l'utilité des représentations textuelles pour les tâches de classification en aval. Un classificateur de régression logistique est entraîné sur les plongements de l'ensemble d'apprentissage et évalué sur l'ensemble de test. La principale métrique est la précision. Nous utilisons l'ensemble de données Banking77Classification pour l'anglais et MovieReviewSentimentClassification pour le français. Ces tâches sont évaluées par la précision moyenne (mean accuracy).
— **Classification par paires** : consiste à classer si deux phrases d'une paire sont liées entre elles.

La métrique d'évaluation est la précision moyenne calculée à partir de la similarité cosinus des paires. Nous utilisons l'ensemble de données SprintDuplicateQuestion.

— **Similarité sémantique textuelle** : mesure le degré de similarité sémantique entre deux phrases. La métrique d'évaluation est la corrélation de Spearman entre la similarité cosinus des plongements et les scores annotés par des humains (Reimers *et al.*, 2016). Nous utilisons l'ensemble de données STSBenchmark pour l'anglais et SickFr pour le français. Ces tâches sont évaluées par la valeur cosinus de spearman.

— **Résumé** : évalue la qualité des résumés générés par machine par rapport à des références rédigées par des humains. La métrique est la corrélation de Spearman entre la similarité cosinus des plongements et les scores de qualité humains. Nous utilisons l'ensemble de données SummEval pour l'anglais et SummEvalFr pour le français. Ces tâches sont évaluées par la valeur cosinus de spearman.

Nous présentons les scores obtenus par les modèles sélectionnés et leurs équivalents élagués pour ces quatre tâches.

3.3 Sélection des modèles

Afin d'évaluer les performances par rapport au MTEBenchmark, nous avons sélectionné les principaux modèles de plongement textuel du classement (en août 2025). Tous les modèles sélectionnés sont disponibles sur HuggingFace. Nous avons échantillonné des modèles appartenant à trois catégories distinctes en termes de taille des paramètres :

— Petits modèles (moins de 100M paramètres) :
  — `BAAI/bge-small-en-v1.5`,
  — `prdev/mini-gte`
  — `sentence-transformers/all-MiniLM-L6-v2`
— Modèles moyens (de 100M à 1B paramètres) :
  — `BAAI/bge-large-en-v1.5`
  — `HIT-TMG/KaLM-embedding-multilingual-mini-instruct-v1.5`
  — `mixedbread-ai/mxbai-embed-large-v1`
— Grands Modèles (plus de 1B paramètres) :
  — `Alibaba-NLP/gte-Qwen2-7B-instruct`
  — `BAAI/bge-en-icl`
  — `BAAI/bge-multilingual-gemma2`

Notre sélection comprend deux modèles optimisés pour les instructions : le modèle de taille moyenne `KaLM-embedding-multilingual-mini-instruct-v1.5` et le grand modèle `gte-Qwen2-7B-instruct`. De plus, nous incluons `sentence-transformers/all-MiniLM-L6-v2`, car il sert de référence largement utilisée dans les évaluations sur la qualité des plongements, garantissant la comparabilité avec les recherches antérieures. La plupart utilise le tokenizer WordPiece à l'exception de KaLM-embedding et gte-Qwen2 qui utilisent un tokenizer basé sur BPE, et le tokenizer de mxbai-embedding également basé sur WordPiece mais modifié par la suite. Nous utilisons les modèles multilingues, `all-MiniLM-L6-v2` et `KaLM-embedding` pour effectuer des expériences sur le français.

Dans les sous-sections suivantes, nous faisons référence à ces modèles en utilisant un alias abrégé (par exemple, `bge-small` pour `BAAI/bge-small-en-v.1.5`).

4 Résultats

4.1 Réduction de la séquence

L'élagage des tokens réduit la longueur de la séquence. On peut observer cette réduction sur l'ensemble des données selectionnées dans le tableau 1 sur l'anglais et le tableau 2 sur le français, montrant que le taux d'élagage reste constant même pour les séquences plus longues (comme sur la tâche SummEval). En moyenne, l'élagage des tokens de fin réduit la longueur de la séquence de 25%, tandis que notre expérience de contrôle permet d'obtenir une réduction plus importante, d'environ 30%, comme l'indique le ratio choisi.

Tâche	Tokenizer	Configuration d'entrée		
		normale	apocopée	aléatoire
Banking77Classification	WordPiece	15.07	12.95 -2.12	11.15 -3.92
	Tokie	15.19	12.95 -2.24	11.26 -3.93
	Gemma Tokenizer	15.0	12.95 -2.05	10.98 -4.02
	BPE	13.34	10.95 -2.39	8.83 -4.51
STSBenchmark	WordPiece	14.15	11.81 -2.34	10.54 -3.61
	Tokie	15.44	11.81 -3.63	11.42 -4.02
	Gemma Tokenizer	14.76	11.81 -2.95	3.0 -11.76
	BPE	13.94	9.81 -4.13	8.7 -5.24
SummEval	WordPiece	454.65	361.33 -93.32	318.85 -135.8
	Tokie	501.76	361.33 -140.43	353.39 -148.37
	Gemma Tokenizer	477.55	361.33 -116.22	319.77 -157.78
	BPE	519.85	359.33 -160.52	319.16 -200.69
SprintDuplicateQuestions	WordPiece	16.0	12.5 -3.5	12.05 -3.95
	Tokie	18.26	13.99 -4.27	13.36 -4.9
	Gemma Tokenizer	16.08	13.99 -2.09	11.5 -4.58
	BPE	15.5	11.99 -3.51	10.01 -5.49

TABLE 1 – Nombre moyen de tokens selon le type de tokenizer (WordPiece, Tokie, Gemma Tokenizer, BPE), la configuration d'entrée (normale, apocopée, aléatoire) et les tâches (classification, similarité textuelle, résumé, classification par paire). La différence de taille avec la configuration normale est présentée à droite des valeurs dans une police de caractères plus petite

Tâche	Modèle	Configuration d'entrée		
		normale	apocopée	aléatoire
SickFR	all-MiniLM	5.35	6.07 -0.72	6.66 -1.31
	KaLM-embedding	16.43	16.71-0.28	16.95-0.52
SummEvalFr	all-MiniLM	7.23	7.25 -0.02	7.68 -0.45
	KaLM-embedding	21.44	21.44+0.0	19.52-1.92
MRSentClass	all-MiniLM	142	135 -7	134 -8
	KaLM-embedding	150	151 +1	145 -5

TABLE 2 – Nombre moyen de tokens selon la configuration d'entrée (normale, apocopée, aléatoire) pour chacun des modèles en fonction des tâches sur le français

4.2 Performances en termes de temps de traitement

Contrairement à la réduction du nombre de tokens, visible pour tous les modèles, les améliorations en termes de temps de traitement présentent une grande variété d'une exécution à l'autre (figure 2 et tableau 5 en annexe). Si la méthode permet de gagner beaucoup de temps pour les séquences plus longues, des gains similaires ne sont pas observés sur d'autres jeux de données. Cette différence suggère que l'efficacité de la méthode dépend du jeux de données et indique des domaines potentiels pour des recherches supplémentaires. Néanmoins, cette approche offre des avantages en termes de performances, en particulier pour les modèles plus volumineux.

4.3 Performances des plongements textuels

En anglais Le tableau 3 présente le résultats des différentes configuration d'entrée. Tout d'abord, nous observons que la position des tokens élagués est critique. L'élagage aléatoire entraîne une baisse de performance plus importante que notre méthode d'apocope, qui conserve les tokens du début de l'unité lexicale. En effet, sur toutes les tâches, à l'exception de SummEval, et sur tous les modèles, à l'exception des modèles KaLM-embedding et gte-Qwen2, les ordre de grandeur des différences entre les configurations sont semblables, c'est à dire autour de 2 points pour la différence entre la configuration normale et la configuration apocopée et environ 30 points entre la configuration normale et la configuration aléatoire.

Certaines séries de données (par exemple STSBenchmark et Banking77Classification) évaluent les intégrations sur des phrases courtes, ce qui entraîne une réduction minimale de la séquence pour toutes les méthodes. Par conséquent, les performances ne sont que légèrement affectées par notre approche d'apocope, comme l'indique la faible distance entre les points de référence (circulaires) et apocopiques (triangulaires) pour ces tâches. Sur l'ensemble de données SummEval , la baisse de performance est faible pour les deux méthodes d'élagage. Cet impact minimal peut être attribué à la faible performance de référence de tous les modèles sur cette tâche spécifique, qui masque potentiellement les effets de l'élagage.

Certaines différences peuvent être observées dans le nombre de tokens conservés entre les modèles. Cela peut s'expliquer par le type de tokeniseur et la présence ou l'absence de tokens spéciaux en début et en fin de séquence. La différence est comblée sur l'ensemble de données SummEval, qui est composé de longues séquences.

Une baisse drastique des performances peut également être observée pour le modèle `gte-Qwen2`, même sans modification de la séquence de tokens. Cette baisse de performances est également due à la spécificité du tokeniseur et pourrait être compensée par une méthode adaptée à chaque type de tokeniseur.

En français Les résultats sur les corpus en français donnent des tendances similaires à ce que l'on peut observer sur l'anglais comme le montre le tableau 4.

En effet, les résultats des configurations normale et apocopée différent en moyenne de 1 point tandis que l'écart varie en moyenne de 6 points entre l'entrée normale et l'entrée aléatoire pour `all-MiniLM` sur les deux premières tâches.

Sur la tâche de classification, l'élagage par apocope diminue fortement le nombre de token par rapport

Tâche	Modèle	Configuration d'entrée		
		normale	apocopée	aléatoire
Banking77Classification	all-MiniLM	80.04	78.56 -1.48	51.08 -28.96
	mini-gte	85.72	84.43 -1.29	56.46 -29.26
	bge-small	81.75	80.43 -1.32	50.3 -31.45
	bge-large	84.82	82.03 -2.79	50.26 -34.56
	bge-en	89.99	85.77 -4.22	53.71 -36.28
	bge-multilingual	91.31	88.48 -2.83	59.8 -31.51
	KaLM-embedding	74.13	62.36 -11.77	47.5 -26.63
	mxbai-embed	87.8	86.05 -1.75	54.48 -33.32
	gte-Qwen2	16.49	59.74 +43.25	43.26 +26.77
STSBenchmark	all-MiniLM	82.03	81.61 -0.42	56.56 -25.47
	mini-gte	86.25	85.24 -1.01	57.92 -28.33
	bge-small	85.86	84.57 -1.29	54.82 -31.04
	bge-large	87.52	85.58 -1.94	56.37 -31.15
	bge-en	80.58	73.63 -6.95	42.98 -37.6
	bge-multilingual	78.76	71.1 -7.66	47.23 -31.53
	KaLM-embedding	77.12	58.69 -18.43	53.07 -24.05
	mxbai-embed	89.29	87.41 -1.88	58.14 -31.15
	gte-Qwen2	24.61	49.14 +24.53	40.74 +16.13
SummEval	all-MiniLM	30.81	31.0 +0.19	29.24 -1.57
	mini-gte	30.48	30.95 +0.47	30.36 -0.12
	bge-small	30.14	28.75 -1.39	30.26 +0.12
	bge-large	31.61	28.61 -3.0	28.38 -3.23
	bge-en	27.36	25.55 -1.81	29.71 +2.35
	bge-multilingual	30.35	26.87 -3.48	30.64 +0.29
	KaLM-embedding	30.64	29.3 -1.34	30.67 +0.03
	mxbai-embed	32.71	29.19 -3.52	28.14 -4.57
	gte-Qwen2	27.85	25.23 -2.62	30.06 +2.21
SprintDuplicateQuestions	all-MiniLM	94.55	92.45 -2.1	60.65 -33.9
	mini-gte	95.48	93.58 -1.9	62.97 -32.51
	bge-small	96.67	94.66 -2.01	66.81 -29.86
	bge-large	96.75	94.06 -2.69	63.1 -33.65
	bge-en	96.45	80.85 -15.6	32.57 -63.88
	bge-multilingual	94.08	89.91 -4.17	59.79 -34.29
	KaLM-embedding	89.62	73.3 -16.32	58.68 -30.94
	mxbai-embed	96.82	93.66 -3.16	57.61 -39.21
	gte-Qwen2	5.78	5.51 -0.27	24.95 +19.17

TABLE 3 – Résultats obtenus sur les quatre tâches (classification, similarité textuelle, résumé, classification par paire) selon la configuration d'entrée retenue (normale, apocopée, aléatoire). Le delta observé par rapport à la configuration d'entrée normale est indiqué en petits chiffres après chaque résultat

aux deux autres configurations pour `all-MiniLM`, ce qui entraine un écart de score moins important entre les entrées aléatoires et les entrées apocopées.

Tâche	Modèle	Configuration d'entrée		
		normale	apocopée	aléatoire
SickFR	`all-MiniLM-L6-V2`	62.48	60.62	45.22
	`KaLM-embedding`	76.90	76.54	45.28
SummEvalFr	`all-MiniLM-L6-V2`	28.28	27.86	27.04
	`KaLM-embedding`	29.31	29.35	30.06

TABLE 4 – Résultats obtenus sur les quatre tâches (classification, similarité textuelle, résumé, classification par paire) selon la configuration d'entrée retenue (normale, apocopée, aléatoire) en français. Le delta observé par rapport à la configuration d'entrée normale est indiqué en petits chiffres après chaque résultat

Les résultats de `KaLM-embedding` laissent entendre que les mots du français constituent des tokens entier pour ce modèle, n'induisant qu'une modification minime avec un élagage des tokens non-initiaux.

5 Conclusion et perspectives

Dans cet article, nous avons présenté les expériences que nous avons menées pour alléger les plongements de texte à l'aide de deux méthodes d'élagage (apocope et entrée aléatoire) que nous avons appliquées à des tâches issues de l'ensemble de données MTEB.

Nous montrons que la qualité des plongements textuels n'est que marginalement affectées par la suppression des tokens finaux. De plus, le positionnement des tokens conservés est crucial pour atténuer la dégradation des performances.

Notre étude s'est concentrée sur l'anglais et a exploré des mécanismes similaires sur le français. Toutefois, nous émettons l'hypothèse que ces résultats ne peuvent pas être directement transposés à d'autres langues, en particulier celles qui sont sujettes à une sur-tokenisation par les modèles. Dans de tels contextes, notre méthode pourrait simplifier drastiquement les entrées, ce qui pourrait entraîner des baisses de performances plus importantes.

Par conséquent, nos travaux futurs consisteront à étendre notre étude à des contextes multilingues. Nous prévoyons également de mener des expériences en utilisant d'autres stratégies d'élagage inspirées par des observations linguistiques, notamment en conservant uniquement les mots pleins.

Références

BELTAGY I., PETERS M. E. & COHAN A. (2020). Longformer : The long-document transformer. (arXiv :2004.05150). arXiv :2004.05150 [cs].

DAI Z., YANG Z., YANG Y., CARBONELL J., LE Q. & SALAKHUTDINOV R. (2019). Transformer-XL : Attentive Language Models beyond a Fixed-Length Context. In A. KORHONEN, D. TRAUM & L. MÀRQUEZ, Éds., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 2978–2988 : Association for Computational Linguistics. DOI : 10.18653/v1/P19-1285.

GOYAL S., CHOUDHURY A. R., RAJE S. M., CHAKARAVARTHY V. T., SABHARWAL Y. & VERMA A. (2020). Power-bert : accelerating bert inference via progressive word-vector elimination. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20 : JMLR.org.

HANUS M., GUIGNARD Q. & RODRIGUES C. (2025). Uc-fire : Approche efficace pour la recherche d'informations non supervisée. In *Actes de CORIA-TALN-RJCRI-RECITAL 2025. Actes de la 20e Conférence en Recherche d'Information et Applications (CORIA)*, p. 249–264, Marseille, France : Association pour le Traitement Automatique des Langues.

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.

KIM S., SHEN S., THORSLEY D., GHOLAMI A., KWON W., HASSOUN J. & KEUTZER K. (2022). Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, p. 784–794, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3534678.3539260.

MUENNIGHOFF N., TAZI N., MAGNE L. & REIMERS N. (2023). MTEB : Massive text embedding benchmark. In A. VLACHOS & I. AUGENSTEIN, Éds., *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, p. 2014–2037, Dubrovnik, Croatia : Association for Computational Linguistics. DOI : 10.18653/v1/2023.eacl-main.148.

REIMERS N., BEYER P. & GUREVYCH I. (2016). Task-oriented intrinsic evaluation of semantic textual similarity. In Y. MATSUMOTO & R. PRASAD, Éds., *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics : Technical Papers*, p. 87–96, Osaka, Japan : The COLING 2016 Organizing Committee.

SANTHANAM K., KHATTAB O., SAAD-FALCON J., POTTS C. & ZAHARIA M. (2022). 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.

SAUVAGE E., BOUCHARENC I., GERALD T., TOURILLE J., CAMPANO S., GROUIN C. & ROSSET S. (2025). Plongement des constituants pour la représentation sémantique des phrases. In *Actes de CORIA-TALN-RJCRI-RECITAL 2025. Actes des 32ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN), volume 1 : articles scientifiques originaux*, p. 614–628, Marseille, France : Association pour le Traitement Automatique des Langues.

TAO C., LIU Q., DOU L., MUENNIGHOFF N., WAN Z., LUO P., LIN M. & WONG N. (2024). Scaling laws with vocabulary : Larger models deserve larger vocabularies.

VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł. & POLOSUKHIN I. (2017). Attention is all you need. *Advances in neural information processing systems*, **30**.

VIG J. (2019). A multiscale visualization of attention in the transformer model. In M. R. COSTA-JUSSÀ & E. ALFONSECA, Éds., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics : System Demonstrations*, p. 37–42, Florence, Italy : Association for Computational Linguistics. DOI : 10.18653/v1/P19-3007.

WANG H., ZHANG Z. & HAN S. (2021). Spatten : Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, p. 97–110. DOI : 10.1109/HPCA51647.2021.00018.

WANG S., LI B. Z., KHABSA M., FANG H. & MA H. (2020). Linformer : Self-attention with linear complexity. (arXiv :2006.04768). arXiv :2006.04768 [cs, stat], DOI : 10.48550/arXiv.2006.04768.

YE D., LIN Y., HUANG Y. & SUN M. (2021). TR-BERT : Dynamic token reduction for accelerating BERT inference. In K. TOUTANOVA, A. RUMSHISKY, L. ZETTLEMOYER, D. HAKKANI-TUR, I. BELTAGY, S. BETHARD, R. COTTERELL, T. CHAKRABORTY & Y. ZHOU, Éds., *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 5798–5809, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2021.naacl-main.463.

ZAHEER M., GURUGANESH G., DUBEY A., AINSLIE J., ALBERTI C., ONTANON S., PHAM P., RAVULA A., WANG Q., YANG L. & AHMED A. (2021). Big bird : Transformers for longer sequences. (arXiv :2007.14062). arXiv :2007.14062 [cs, stat].

ZHANG Y., LI M., LONG D., ZHANG X., LIN H., YANG B., XIE P., YANG A., LIU D., LIN J., HUANG F. & ZHOU J. (2025). Qwen3 Embedding : Advancing Text Embedding and Reranking Through Foundation Models. DOI : 10.48550/arXiv.2506.05176.

ZONG Y. & PIWOWARSKI B. (2025). Towards lossless token pruning in late-interaction retrieval models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, p. 2407–2417, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3726302.3730100.

Annexes

Banking77Classification	all-MiniLM	17.78	12.95 +0.22	17.34 -0.44
	mini-gte	20.55	12.95 -0.38	20.01 -0.54
	bge-small	18.75	12.95 -0.16	18.77 +0.02
	bge-large	31.58	12.95 -2.41	28.65 -2.93
	bge-en	331.31	12.95 -28.22	296.09 -35.22
	bge-multilingual	602.54	12.95 -192.99	393.23 -209.31
	KaLM-embedding	36.89	10.95 -2.75	33.74 -3.15
	mxbai-embed	34.24	12.95 -1.67	32.88 -1.36
	gte-Qwen2	290.34	10.95 -25.09	242.23 -48.11
STSBenchmark	all-MiniLM	1.56	11.81 -0.01	1.72 +0.16
	mini-gte	1.46	11.81 +0.03	1.57 +0.11
	bge-small	2.13	11.81 +0.06	2.31 +0.18
	bge-large	4.83	11.81 -0.76	4.12 -0.71
	bge-en	84.06	11.81 -11.07	71.94 -12.12
	bge-multilingual	114.49	11.81 -12.38	97.84 -16.65
	KaLM-embedding	6.78	9.81 -0.49	6.61 -0.17
	mxbai-embed	4.22	11.81 -0.41	3.86 -0.36
	gte-Qwen2	79.64	9.81 -12.09	65.27 -14.37
SummEval	all-MiniLM	3.03	361.33 -0.1	2.43 -0.6
	mini-gte	3.83	361.33 -0.23	2.89 -0.94
	bge-small	3.48	361.33 -0.03	2.94 -0.54
	bge-large	13.81	361.33 -1.39	10.15 -3.66
	bge-en	264.05	361.33 -48.4	210.15 -53.9
	bge-multilingual	230.66	361.33 +49.71	200.73 -29.93
	KaLM-embedding	18.64	359.33 -3.83	12.63 -6.01
	mxbai-embed	13.61	361.33 -1.41	9.95 -3.66
	gte-Qwen2	243.54	359.33 -44.65	185.16 -58.38
SprintDuplicateQuestions	all-MiniLM	6.82	12.5 -0.14	7.23 +0.41
	mini-gte	7.03	12.5 +0.01	7.47 +0.44
	bge-small	8.73	12.5 -0.16	9.13 +0.4
	bge-large	17.38	12.5 -1.25	15.58 -1.8
	bge-en	401.31	13.99 -157.0	385.94 -15.37
	bge-multilingual	380.33	13.99 -40.35	319.31 -61.02
	KaLM-embedding	22.71	11.99 +0.13	23.49 +0.78
	mxbai-embed	16.98	12.5 -1.19	15.36 -1.62
	gte-Qwen2	268.6	11.99 -32.19	216.89 -51.71

TABLE 5 – Temps de traitement des différents modèles selon les configurations

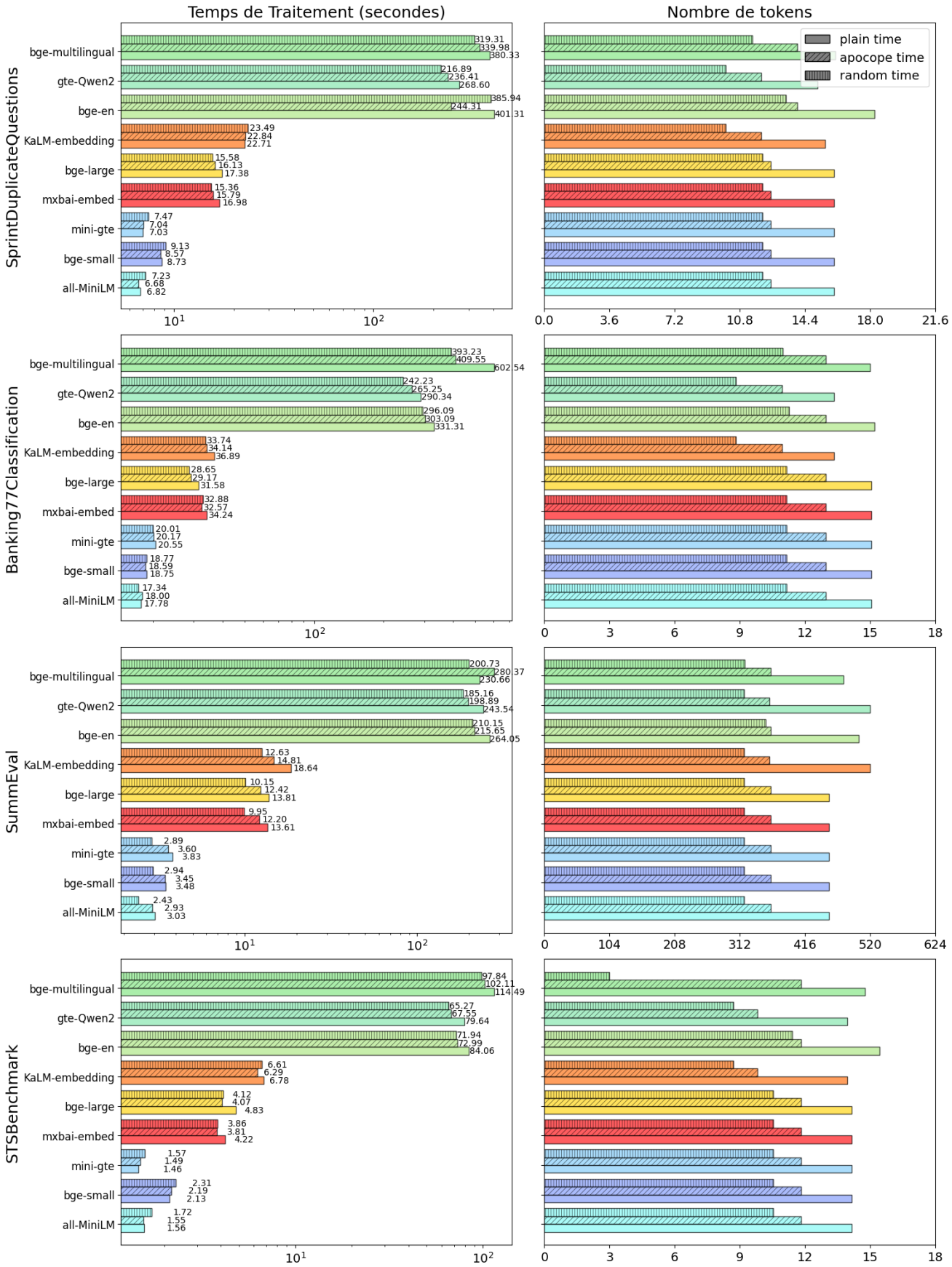

FIGURE 2 – Temps de calculs (à gauche) comparé à réduction de la taille de séquence (à droite). Les expériences avec les entrées apocopée sont représentés par des strilles en diagonal, celles avec les entrées aléatoirement élaguées avec des strilles verticales et celles sans modifications, sans strilles

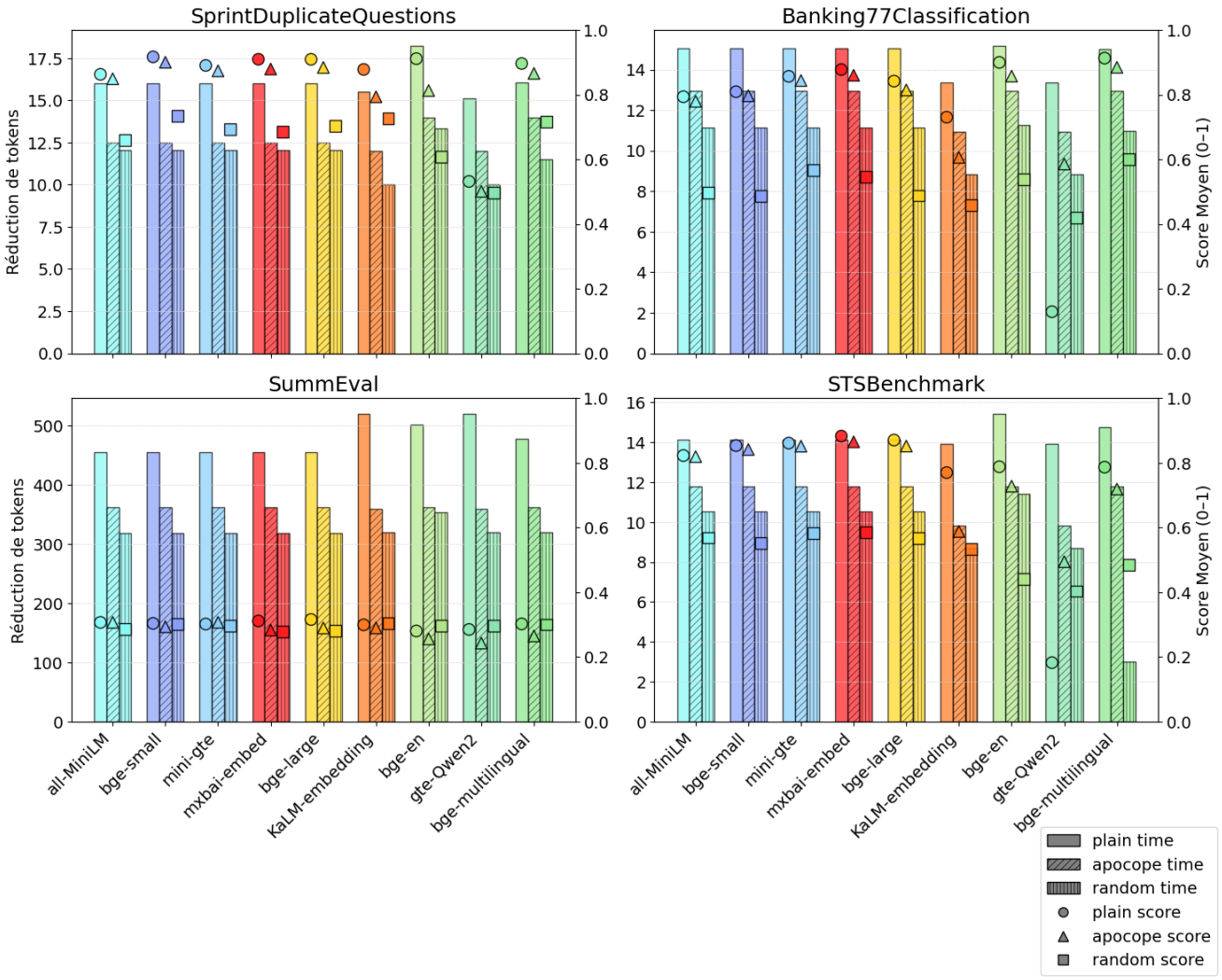

FIGURE 3 – Résultats et nombres de tokens par modèles pour les différents jeux de données.

Tâche	Modèle	normale	apocopée	aléatoire
SickFR	all-MiniLM	62.48	60.62 -1.86	45.22 +17.26
	KaLM-embedding	76.9	76.54 -0.36	45.28 +31.62
SummEvalFr	all-MiniLM	28.28	27.86 -0.42	27.04 +1.24
	KaLM-embedding	29.31	29.35 +0.04	30.06 -0.75
MovieReview	all-MiniLM	56.73	57.18 +0.45	54.48 +2.25
	KaLM-embedding	81.09	80.6 -0.49	57.93 +23.16

TABLE 6 – Performances des modèles selon la configuration d'entrée (normale, apocopée, aléatoire), sur les données en français