Université Sorbonne Nouvelle

Institut de linguistique et phonétique générales et appliquées (ILPGA)

IA frugale pour les modèles français avec l'élagage et le PEFT

MASTER

TRAITEMENT AUTOMATIQUE DES LANGUES

Parcours :

Recherche et Développement

par

Xiaohua CUI

Encadrant de mémoire :

Yoann Dupont

Année universitaire 2023/2024

Remerciement

À l'issue de ce travail de recherche, je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce mémoire.

Tout d'abord, je souhaite remercier sincèrement mon directeur de recherche, Yoann DUPONT, pour son encadrement, ses précieux conseils et son soutien constant tout au long de cette étude. Ses connaissances approfondies et son approche méthodologique ont été vraiment indispensables à la réussite de ce projet.

Je tiens également à exprimer ma gratitude particulière à Philéas CONDEMINE, mon tuteur en entreprise, pour m'avoir guidé avec patience et pour m'avoir transmis ses connaissances pratiques et théoriques, ainsi que des inspirations pour ce projet.

Un grand merci à mes collègues de promotion pour leurs encouragements, leurs échanges enrichissants et leur amitié tout au long de cette aventure académique. Votre soutien moral a été inestimable.

Je souhaite également remercier ma famille pour leur soutien inconditionnel et leur patience durant cette période exigeante. Leur amour et leur encouragement ont été essentiels à ma réussite.

À tous, je vous exprime ma profonde gratitude.

RÉSUMÉ

Avec la transition progressive du *Natural Language Processing* (NLP, Traitement automatique des langues en français) vers l'utilisation des réseaux de neurones et des architectures basées sur les transformers, les modèles pré-entraînés sont de plus en plus utilisés comme point de départ pour diverses tâches. Cette évolution s'accompagne d'une augmentation significative de la charge et des ressources de calcul nécessaires. Pour remédier à ces défis, des méthodes telles que le *pruning* (élagage), la *quantification*, la *distillation* et le *Parameter-Efficient Fine-Tuning* (PEFT) sont employées pour réduire la taille des modèles et les coûts computationnels.

Dans cette recherche, nous explorons comment les techniques d'élagage et de PEFT peuvent être appliquées au modèle CamemBERT afin de diminuer les coûts tout en maintenant les performances acceptables. Nous évaluons l'efficacité de ces méthodes sur différentes tâches, notamment la NER, le POS tagging et l'analyse des sentiments. Notre objectif est d'évaluer la possibilité de réduire la taille des modèles sans trop nuire aux performances, pour qu'ils puissent être utilisés même dans des environnements à ressources limitées. Les résultats de notre étude montrent que l'élagage structuré basé sur les couches peut être appliqué aux modèles français afin de réduire les coûts de calcul pour certaines tâches simples. De plus, l'importance des couches varie en fonction des tâches, ce qui nous amène à adopter des stratégies de compression spécifiques selon la nature de la tâche. Enfin, il a été démontré que LoRA présente une compatibilité limitée avec le pruning structuré, mais nous pouvons compenser les pertes dues à l'élagage en augmentant la durée de l'entraînement. Ces résultats peuvent guider les applications pratiques du modèle CamemBERT et démontrer la viabilité des méthodes frugales pour rendre l'utilisation des LLM plus accessible dans des environnements contraints en ressources.

Mot-clés : IA frugale, élagage, *pruning*, PEFT, NER, POS, *deep learning*, CamemBERT, NLP, TALN, distillation, quantification, *fine-tuning*, transformer

TABLE DES MATIÈRES

Liste des figures

Liste des tableaux

INTRODUCTION

Présentation générale

Depuis la fin des années 2010, l'émergence et la large application de modèles pré-entraînés tels que BERT et XLNet ont marqué des progrès majeurs dans les domaines de l'intelligence artificielle et du NLP. Ces modèles, pré-entraînés sur de vastes ensembles de données non annotées, ont démontré une forte capacité de généralisation sur diverses tâches de NLP.

Ces dernières années, avec la popularisation de ChatGPT, les modèles génératifs pré-entraînés ont montré des avancées spectaculaires en matière de génération de texte et de dialogue, et nous pouvons même les utiliser pour exécuter certaines tâches classiques. Cependant, en raison d'une potentielle insuffisance de performance et d'une manque d'explication des "black box", les LLM comme BERT (*Bidirectional Encoder Representations from Transformers*) sont encore utilisés dans l'industrie.

Bien que ces modèles de langage pré-entraînés permettent de réduire considérablement le coût de développement et d'adaptation en fournissant une base solide, leur déploiement et l'inférence nécessitent toujours beaucoup de ressources computationnelles et de temps en raison de leur taille de plus en plus grande. Par exemple, même un GPU professionnel comme NVIDIA A100, équipé de 80 Go de RAM, ne peut pas répondre aux besoins pour l'inférence du Mixtral-8x7B en pleine précision [JIANG et al., 2023]. Même avec une quantification de 4bits, cela prend toujours presque 5 heures pour une seule inférence. De plus, les différences significatives entre les GPU grand public et commerciaux, que ce soit en termes de nombre de cœurs CUDA, de capacité de VRAM ou de frameworks disponibles, apportent un coût temporel énorme et un seuil élevé pour l'application des grands modèles. Par exemple, pour arriver à équilibrer la performance et le temps d'entraînement, le *fine-tuning* sur BERT nécessite souvent un meilleur GPU que celui des cartes grand public avec plus de RAM / VRAM. Comment rendre ces modèles plus accessibles et épargner le temps de les

employer, même au prix d'une légère perte de performance ? C'est là que l'IA frugale entre en jeu.

Dans cette étude, nous nous appuyons en partie sur les travaux de SAJJAD et al. (2023), qui ont mené des expérimentations sur l'élagage en se concentrant principalement sur les tâches du benchmark GLUE en anglais. Notre approche se distingue par la réalisation des expériences sur un modèle en français. De plus, plutôt que de nous concentrer uniquement sur les tâches GLUE, nous diversifions notre analyse en explorant davantage les stratégies des couches du modèle. Dans le cadre de l'amélioration des modèles transformers en langue française, nous nous posons les questions suivantes : Pouvons-nous réduire le coût de calcul des modèles en utilisant l'élagage et le PEFT, tout en maintenant des performances acceptables ? Plus précisément, pour différents types de tâches de TAL, les stratégies de compression ont-elles des impacts variés, et ces différences peuvent-elles être exploitées pour développer des approches de frugalité optimisées pour des tâches spécifiques ? C'est à ces questions que nous tenterons de répondre dans cette recherche. Dans le cadre de cette rédaction, les outils d'assistance à la traduction (*DeepL* et *FrHepler*) ont été ponctuellement employés afin de garantir une meilleure fluidité et clarté dans l'expression des idées.

Plan de lecture

Notre recherche vise à explorer des approches frugales de l'élagage et PEFT pour économiser le déploiement des modèles Transformers françaises, tout en gardant une performance acceptable. Le mémoire est structuré comme suit :

Le premier chapitre présente un contexte général du sujet d'IA frugale, commençant par une explication plus détaillée du terme et des concepts clés. Dans le deuxième chapitre, nous procéderons à une revue de la littérature autour du sujet étudié. Plus précisément, nous introduirons les concepts du domaine de la compression de modèles et nous concentrerons sur la synthèse des recherches précédentes concernant les deux méthodes que nous avons adoptées : l'élagage et PEFT. Cela nous permettra de mieux positionner notre recherche et de discuter la pertinence et l'importance de notre étude.

La deuxième partie du mémoire se compose du troisième et quatrième chapitre. Le troisième chapitre se concentrera sur les préparatifs nécessaires de notre re-

cherche. Nous y présenterons les modèles et les corpus choisis, ainsi que les stratégies concrètes de nos expériences. Nous présenterons et analyserons nos résultats dans le quatrième chapitre, où sont détaillées les performances des modèles élagués et affinés sur différentes tâches NLP.

Enfin, nous discuterons des résultats obtenus et des perspectives de recherche dans le dernier chapitre, où nous discuterons des avantages et des inconvénients des approches frugales et des pistes d'amélioration pour les futures recherches.

Première partie

Contexte général

CHAPITRE 1

INTRODUCTION À L'IA FRUGALE

Sommaire

1.1 Contexte : deux enjeux majeurs de l'IA frugale

L'IA frugale est une approche qui vise à rendre l'intelligence artificielle plus accessible et durable, en mettant l'accent sur l'efficacité et l'économie des ressources. Dans ce contexte, deux enjeux majeurs se dégagent : la réduction des coûts de calcul et la réduction des coûts de données.

1.1.1 Réduction des coûts de calcul

Avec la popularisation croissante de l'intelligence artificielle, l'empreinte énergétique mondiale des technologies numériques devient de plus en plus dense. Un nombre croissant de personnes commencent à s'intéresser à l'impact macro-environnemental des coûts de calcul dans le domaine de l'IA. Bien que certaines études confirment que l'intelligence artificielle peut aider efficacement à explorer les problèmes climatiques et que sa consommation d'énergie mondiale est relativement

faible (seulement 0.04% selon les enquêtes de de VRIES (2023)), il est nécessaire de prendre en compte les besoins croissants en services d'IA et le développement industriel. En effet, on prévoit que la consommation d'énergie liée à l'IA pourrait être multipliée par 10 dans les quatre prochaines années, ce qui soulève des questions énergétiques importantes qui méritent notre attention [de VRIES, 2023].

Parallèlement, à mesure que l'IA se démocratise, la demande de formation des modèles d'IA et les ressources de calcul personnelles sont déséquilibrées au niveau micro. Par exemple, le *fine-tuning* des modèles nécessite des ressources computationnelles et du temps. Dans des environnements à ressources limitées, il devient crucial de savoir comment entraîner efficacement des modèles performants sans disposer d'une puissance de calcul abondante. C'est le premier enjeu majeur de l'IA frugale : optimiser l'entraînement des modèles pour qu'ils soient accessibles et efficaces même avec des ressources limitées, afin de résoudre à un niveau personnel la question du seuil de formation des modèles pré-entraînés et, au niveau sociétal, la question de la consommation énergétique et du développement durable.

1.1.2 Réduction des coûts de données

L'intelligence artificielle traditionnelle est confrontée à des défis en matière de demande de données, tels que les coûts élevés de l'acquisition, du stockage et du traitement des données. L'obtention de ces données nécessite souvent un processus d'annotation manuel fastidieux et chronophage, ce qui entraîne des coûts de main-d'uvre considérables pour les particuliers et les entreprises. C'est là que se situe le deuxième enjeu : minimiser les coûts associés à la collecte et à l'annotation des données tout en maintenant une haute qualité des modèles.

Dans ces contextes, l'IA frugale a vu le jour. Il s'agit d'une technologie promettant d'assurer la robustesse des modèles d'IA dans des domaines spécifiques tout en utilisant moins de données et de ressources de calcul. Cet article explore principalement l'IA frugale en ce qui concerne les ressources de calcul.

1.2 Concepts clés

Dans la deuxième section de ce chapitre, nous souhaitons introduire quelques points clés de l'IA frugale, y compris certains concepts importants de l'apprentissage

profond liés aux modèles utilisés dans cette étude. L'objectif est de préparer le terrain pour le deuxième chapitre, qui présentera une revue des recherches dans le domaine.

1.2.1 Transformers

Le Transformer est une architecture d'apprentissage profond développée par Google, basée sur le mécanisme d'attention multi-tête de VASWANI et al. (2017), introduite dans l'article de 2017 "Attention Is All You Need". Contrairement aux modèles précédents tels que les RNN et CNN, le Transformer utilise le mécanisme d'attention pour traiter les données en parallèle, ce qui permet de mieux gérer les dépendances à longue distance dans les séquences de texte et de rendre le traitement plus facilement parallélisable, tout en équilibrant l'importance des différentes positions dans la séquence d'entrée lors de la génération de la sortie.

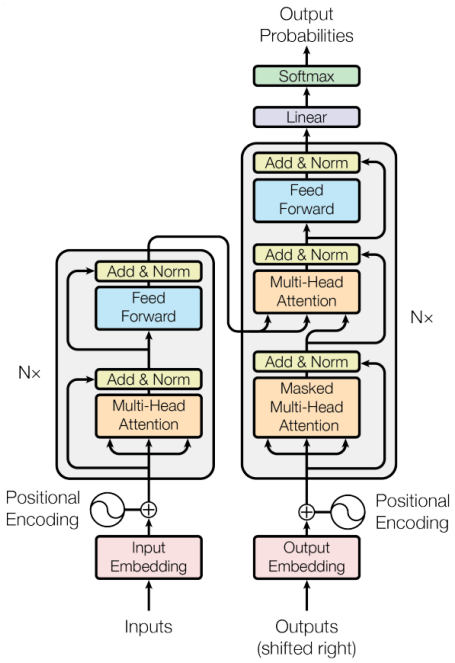

FIGURE 1.1 – Architecture du Transformer (VASWANI et al., 2017)

Concrètement, cette architecture est principalement composée de deux blocs comme illustré à la figure 1.1 : l'encodeur (à gauche) et le décodeur (à droite). L'encodeur traite l'entrée pour construire une représentation intermédiaire, qui capte les caractéristiques pertinentes du texte. Le décodeur, ensuite, utilise cette représentation intermédiaire, enrichie par d'autres entrées, pour générer la séquence cible.

L'apparition du Transformer a révolutionné le domaine NLP et est rapidement devenue l'architecture principale dans ce domaine. Les modèles basés sur le Trans-

former peuvent être classés en trois catégories (voir Hugging Face) : les modèles de type GPT (également appelés modèles Transformers auto-régressifs), les modèles de type BERT (modèles Transformers auto-encodeurs) et les modèles de type BART/T5 (modèles Transformers séquence-à-séquence). Dans cette étude, nous utilisons une variante de BERT — le modèle CamemBERT.

1.2.2 Élagage (Pruning)

Dans le contexte où les modèles d'apprentissage profond deviennent de plus en plus grands et complexes, la technique de l'élagage est largement reconnue comme une méthode importante de compression de modèles. L'élagage identifie et élimine les paramètres ou connexions ayant un impact minimal sur les performances du modèle au cours de l'entraînement, permettant ainsi de simplifier le modèle. Cette opération se situe généralement entre l'entraînement du modèle et le *fine-tuning* et a démontré un potentiel énorme et des perspectives d'application étendues. Ici, nous distinguons principalement l'élagage structuré de l'élagage non structuré.

Dans l'élagage non structuré, l'objet à élaguer est la connexion entre les neurones. En mettant à zéro les poids non importants du modèle, on obtient une matrice clairsemée avec de nombreux éléments nuls. Cette méthode permet de réduire le nombre de paramètres du modèle tout en conservant sa précision, mais est moins supportée par les accélérateurs matériels. L'élagage structuré, en revanche, implique la suppression de filtres ou de couches entières du réseau. Comparé à l'élagage non structuré, l'élagage structuré facilite davantage l'accélération matérielle car la structure du modèle après l'élagage est plus régulière et mieux adaptée au calcul parallèle. L'élagage structuré peut être subdivisé en plusieurs méthodes. Dans cette étude, nous avons choisi d'utiliser l'élagage hiérarchique pour atteindre nos objectifs, en supprimant sélectivement des couches entières en fonction de leur importance pour le modèle.

Nous détaillerons dans le prochain chapitre un aperçu de la recherche sur l'élagage, ainsi que des méthodes d'élagage plus spécifiques.

1.2.3 Modèle pré-entraîné

Le pré-entraînement est une stratégie de formation de modèles en apprentissage profond dont l'idée centrale est d'effectuer d'abord un apprentissage auto-supervisé ou non supervisé sur de grandes quantités de données non étiquetées afin d'ap-

prendre des représentations de caractéristiques générales des données. Ensuite, ces modèles pré-entraînés peuvent être affinés pour des tâches spécifiques. Cette méthode accélère non seulement le processus de déploiement, mais améliore également de manière significative les performances du modèle sur des tâches spécifiques.

1.2.4 PEFT

Le *fine-tuning* d'un modèle pré-entraîné est une méthode d'apprentissage par transfert (Transfer Learning), dont les étapes sont les suivantes :

1. Choisir un modèle pré-entraîné.
2. Préparer un jeu de données spécifique à la tâche pour le *fine-tuning*.
3. Construire une "Model Head" spécifique à la tâche (task-specific head).
4. Initialisation des paramètres : charger les paramètres du modèle pré-entraîné comme paramètres initiaux.
5. Entraînement *fine-tuning* : définir les hyperparamètres d'entraînement et utiliser le jeu de données pour entraîner le modèle de manière supervisée. Les paramètres du modèle sont ajustés pour s'adapter à la tâche spécifique.
6. Évaluation et validation, puis sauvegarder le modèle ajusté.

PEFT (*Parameter-Efficient Fine-Tuning*) est une stratégie de *fine-tuning* visant à n'entraîner qu'un petit nombre de paramètres pour adapter le modèle aux tâches en aval, améliorant ainsi l'efficacité du *fine-tuning*. Cette stratégie inclut LoRA, Adapter Tuning, etc., qui gèlent généralement la plupart des paramètres du modèle pré-entraîné et ne mettent à jour que les matrices de faible rang nouvellement ajoutées, les modules d'adaptateurs ou d'autres structures spécifiques, ce qui est particulièrement utile dans les environnements avec des ressources de calcul limitées. Cependant, PEFT peut parfois ne pas atteindre les niveaux de performance du *fine-tuning* complet. La notion PEFT est étroitement alignée avec notre recherche, puisqu'il contient la méthode LoRA que nous avons employée. Les recherches et les applications de PEFT et de LoRA seront aussi détaillées dans le chapitre suivant.

Ce chapitre introduit l'IA frugale en partant du contexte et des concepts clés. Dans le prochain chapitre, nous procéderons à une revue des recherches pertinentes et détaillerons des méthodes et techniques employées dans notre recherche.

CHAPITRE 2

CONTEXTE D'ÉTUDE

Sommaire

En matière de compression de modèles, il existe généralement quatre grandes catégories selon X. ZHU et al. (2023) : l'élagage, la quantification, la distillation et la factorisation de rang inférieur (*Low-Rank Factorization* en anglais), comme illustré à la figure 2.1. Ces différentes méthodes interviennent de manière distincte sur diverses parties du réseau. Nous commencerons par introduire brièvement la quantification et la distillation, deux techniques qui font partie intégrante d'IA frugale.

2.1 Quantification et Distillation

Nous présentons ici brèvement les concepts de quantification et de distillation, deux méthodes de compression de modèles qui ne sont pas au cur de notre recherche, mais qui sont souvent utilisées en combinaison avec l'élagage et le PEFT.

La quantification est un processus consistant à convertir les valeurs numériques en virgule flottante en valeurs entières à faible largeur de bit, généralement en convertissant les nombres flottants 32 bits en entiers 8 bits [JACOB et al., 2018].

Lors de la quantification, deux paramètres principaux sont utilisés : l'échelle (*scale* en anglais) et le point zéro (*zero-point* en anglais). Le facteur d'échelle est uti-

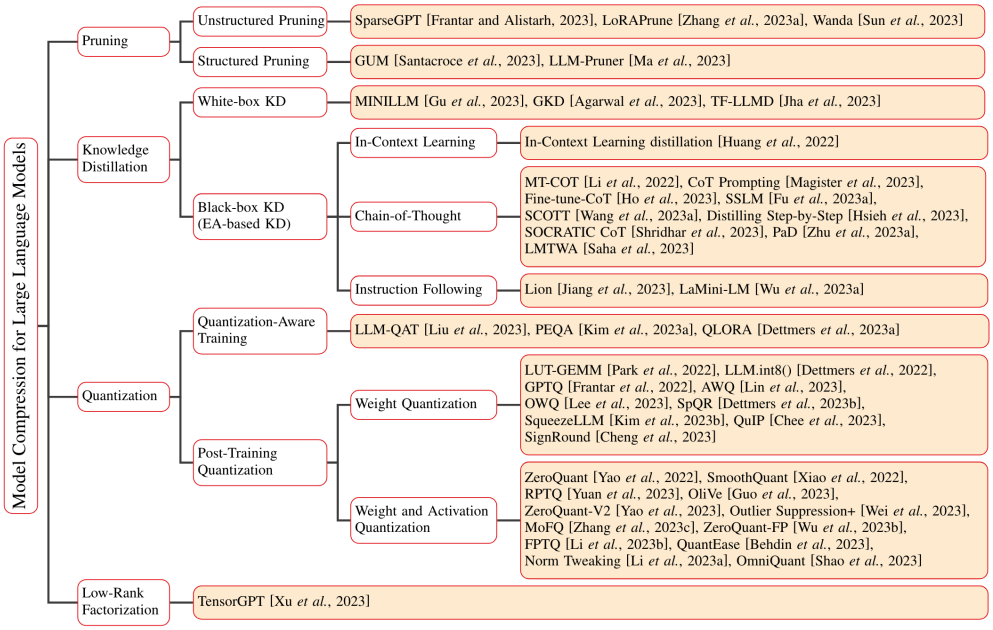

FIGURE 2.1 – Taxonomie des méthodes de compression de modèles (X. ZHU et al., 2023)

lisé pour convertir l'écart entre les unités en virgule flottante et les unités entières, tandis que le zero-point représente la valeur en virgule flottante correspondant à 0 dans l'espace entier quantifié. Mathématiquement, la conversion d'une valeur en virgule flottante r en une valeur entière quantifiée q est exprimée comme suivante :

$$q = \text{round}\left(\frac{r}{S}\right) + Z$$

où S est le *scale* qui définit la relation proportionnelle entre les valeurs réelles et les valeurs entières, Z est le *zero-point* qui représente la valeur entière correspondant à la valeur flottante zéro.

Inversement, pour retrouver la valeur flottante d'origine à partir de la valeur quantifiée, on utilise :

$$r = S \cdot (q - Z)$$

La quantification entraîne inévitablement une perte d'information [HAN et al., 2016]. Nous recourons souvent à des méthodes telles que le *Quantization Aware Training* ou le réentraînement pour minimiser l'impact de la quantification sur les performances du modèle [KRISHNAMOORTHI, 2018].

La distillation (Knowledge Distillation), quant à elle, implique une structure d'entraînement de type enseignant-élève (Teacher-Student) [GOU et al., 2021]. Dans cette

approche, un modèle enseignant préalablement entraîné, souvent complexe et de grande taille, fournit des connaissances à un modèle élève, plus simple et plus léger. L'objectif est de transférer les connaissances du modèle enseignant vers le modèle élève, tout en conservant des performances similaires avec une complexité réduite.

La procédure classique de distillation repose sur la minimisation d'une fonction de perte combinée $\mathcal{L}_{\text{KD}}$. Soit T le modèle enseignant, S le modèle élève, et τ la température utilisée pour adoucir les probabilités de sortie du modèle enseignant. La fonction de perte combinée est donnée par :

$$\mathcal{L}_{\text{KD}} = (1 - \alpha) \cdot \mathcal{L}_{\text{class}}(S, y) + \alpha \cdot \mathcal{L}_{\text{distill}}$$

où $\mathcal{L}_{\text{class}}(S, y)$ est une perte liée à la tâche principale, qui peut être l'entropie croisée ou une autre fonction adaptée à la tâche spécifique. $\mathcal{L}_{\text{distill}}$ est la perte de distillation qui encourage le modèle élève à imiter le modèle enseignant. α est un hyperparamètre qui équilibre les deux types de pertes, et τ ajuste le lissage des probabilités de sortie, facilitant ainsi l'apprentissage par le modèle élève.

Cet article se concentre principalement sur les deux méthodes qui restent : l'élagage et le *Parameter Efficient Fine-Tuning* (PEFT), cette dernière utilisant notamment la méthode *Low-Rank Adaptation* (LoRA). Nous présenterons donc dans la section suivante les recherches qui y sont liées.

2.2 L'élagage des modèles

Dans le premier chapitre, nous avons déjà présenté brièvement l'élagage, en particulier l'élagage structuré et non structuré. Dans cette section, nous allons approfondir les techniques d'élagage en examinant l'état de l'art et la classification des méthodes.

2.2.1 Revue de littérature sur l'élagage

Élagage non structuré

Les recherches sur l'élagage des modèles NLP ont commencé dans les années 1980, remontant à l'article de HANSON et PRATT (1988), qui discutait déjà de la minimisation de la structure des réseaux neuronaux en introduisant différents biais lors de l'entraînement.

Parmi les premiers travaux influents dans ce domaine, nous trouvons celui de LECUN et al. (1989), qui a introduit les premières méthodes d'élagage pour les réseaux de neurones. Dans leur article pionnier, LeCun et ses collègues ont proposé une stratégie qui serait aujourd'hui qualifiée d'élagage non structuré. Cette méthode consistait à calculer la matrice hessienne pour chaque poids du réseau neuronal afin d'identifier ceux ayant le moindre impact sur la performance du réseau, permettant ainsi de les supprimer. Dans leurs recherches, basées sur une tâche de reconnaissance de chiffres manuscrits, ils ont appliqué la technique "Optimal Brain Damage" (OBD) et ont démontré qu'après avoir éliminé jusqu'à 60 % des paramètres, l'erreur sur les ensembles d'entraînement et de test restait pratiquement inchangée, prouvant ainsi que les paramètres supprimés étaient effectivement "superflus" et n'importaient pas à la performance. En tant que première méthode d'élagage non structuré, l'OBD a jeté les bases pour les recherches et applications ultérieures dans ce domaine. Des chercheurs suivants ont construit sur cette base pour proposer une série de méthodes d'élagage, notamment la technique "Optimal Brain Surgeon" introduite par HASSIBI et al. (1993), qui évalue de manière plus précise l'impact des poids et optimise ainsi la stratégie d'élagage.

Au milieu des années 2010, avec l'avènement de l'apprentissage profond et l'utilisation généralisée de modèles à base de RN, les techniques d'élagage ont suscité un regain d'intérêt, déclenchant une nouvelle vague de recherches. HAN et al. (2015) ont proposé une méthode combinant l'élagage, la quantification et le codage de Huffman pour compresser le *Deep Neural Network* (DNN). Leur approche consistait à entraîner d'abord un réseau avec des connexions redondantes, puis à supprimer les poids ayant des valeurs absolues faibles, avant de réentraîner le modèle élagué pour restaurer la précision. Ils ont validé l'efficacité de l'élagage sur le modèle AlexNet [KRIZHEVSKY et al., 2012] et ont réussi à réduire la taille du modèle par un facteur de 9 tout en accélérant le temps d'inférence par un facteur de 3. En 2016, HAN et al. (2016) ont perfectionné leur méthode d'élagage en l'étendant à un cadre complet de compression de modèles, intégrant l'élagage, la distillation et la quantification. Leur méthode, baptisée *Deep Compression*, a ouvert la porte à l'utilisation généralisée de modèles d'apprentissage profonds dans des environnements où les ressources de calcul ou de stockage étaient limitées.

L'élagage non structuré réduit considérablement le nombre de paramètres du mo-

dèle et la complexité théorique des calculs. Cependant, cette approche n'est pas bien adaptée aux architectures matérielles actuelles, telles que les GPU, ce qui limite les gains de vitesse en pratique. Par conséquent, les recherches récentes se sont davantage concentrées sur l'élagage structuré, qui est plus facile à déployer dans des environnements réels.

Élagage structuré

En ce qui concerne l'élagage structuré, WEN et al. (2016) ont proposé une méthode utilisant la régression LASSO, en ajoutant une pénalité au sein de la fonction de perte pour apprendre une sparsité structurée et ainsi permettre la compression du modèle. Contrairement au pruning non structuré traditionnel, qui se concentre sur la suppression de poids individuels, cette pénalité encourage le modèle à réduire à zéro les poids de certaines structures spécifiques du réseau, telles que les canaux de convolution, des noyaux ou même des couches entières ; LI et al. (2017) ont proposé une méthode d'élagage des filtres dans les *Convolutional Neural Networks* (CNN). Leur méthode consiste à calculer la somme des valeurs absolues des poids de chaque filtre de convolution (par régularisation L1), puis à classer ces filtres en fonction de leur importance relative. Les filtres ayant les scores d'importance les plus faibles sont ensuite supprimés ; LUO et WU (2017) ont aussi introduit une approche basée sur l'entropie pour l'élagage des CNN, où ils proposent de mesurer l'incertitude liée à chaque filtre afin de déterminer lesquels peuvent être supprimés sans compromettre la performance du modèle ; la même année, M. ZHU et GUPTA (2017) ont approfondi l'efficacité des méthodes d'élagage. Ils y ont introduit la méthode du *Gradual Pruning* (pruning progressif en français), qui consiste à réduire progressivement les poids au cours de l'entraînement. Cette approche permet de mieux préserver la précision du modèle en optant pour une stratégie d'élagage incrémentale plutôt qu'une élimination massive en une seule fois.

En 2019, FRANKLE et CARBIN (2019) ont formulé l'hypothèse dite du *Lottery Ticket Hypothesis* (Billet de Loterie en français). Ils ont démontré que, dans un réseau de neurones initialisé aléatoirement et surparamétré, c'est-à-dire un réseau trop large contenant un nombre excessif de paramètres, il existe un sous-réseau de très petite taille qui peut, après entraînement, atteindre une performance comparable, voire supérieure, à celle du réseau complet. Concrètement, ils ont d'abord entraîné

un grand réseau de neurones surparamétré jusqu'à convergence, puis ont procédé à un pruning en supprimant les poids ayant une contribution faible à la sortie du modèle (généralement les poids de faible amplitude). Les poids restants du réseau élagué sont ensuite réinitialisés à leurs valeurs initiales d'entraînement. Enfin, ce sous-réseau épars est réentraîné sur les mêmes données. Des expériences menées sur des architectures de réseaux de neurones telles que LeNet et VGG, ainsi que sur des jeux de données comme MNIST et CIFAR-10, ont validé l'hypothèse du billet de loterie, offrant ainsi une nouvelle stratégie pour la compression des modèles.

En plus de l'évolution des techniques et des diverses méthodes existantes, BLALOCK et al. (2020) ont souligné l'absence de critères de référence et d'indicateurs d'évaluation standardisés et ont ainsi proposé *ShrinkBench*, un cadre open-source destiné à standardiser l'évaluation des méthodes d'élagage. Ce cadre vise à offrir une plateforme expérimentale unifiée pour les futures recherches sur l'élagage, garantissant ainsi la comparabilité des résultats des études.

Il en ressort clairement que, bien avant l'avènement des modèles Transformers, les techniques d'élagage ont connu plusieurs décennies de développement. Avec l'essor du deep learning, des méthodes modernes d'élagage, comme le *Deep Compression*, l'hypothèse du billet de loterie et l'élagage automatique ont émergé, fournissant de nouveaux outils pour gérer des modèles complexes à grande échelle.

2.2.2 Élagage structuré au niveau des couches

Dans la section précédente, nous avons retracé l'histoire et le développement des techniques d'élagage, couvrant à la fois les méthodes d'élagage non structuré, les approches d'élagage structuré ainsi que l'émergence de l'élagage automatique. Étant donné que notre recherche se concentre spécifiquement sur l'élagage structuré, plus précisément celui basé sur les couches, cette section explorera les avancées récentes dans ce domaine, ainsi que les techniques et applications connexes. Nous nous focaliserons particulièrement sur les recherches relatives à l'élagage des couches d'attention sur les modèles Transformer.

Concernant l'élagage structuré appliqué aux architectures Transformer, la méthode *LLM Pruner*, proposée par MA et al. (2023), se distingue comme une avancée clé . Contrairement aux méthodes traditionnelles de compression, le LLM Pruner requiert seulement une petite quantité de données et un nombre limité d'étapes d'en-

traînement pour ajuster le modèle élagué, garantissant ainsi sa capacité à s'adapter aux nouvelles distributions de données ; SANTACROCE et al. (2023) et ses collègues ont proposé la méthode *GUM* , qui améliore l'efficacité de l'élagage en renforçant l'individualité et la réactivité des neurones, offrant ainsi des résultats plus optimisés.

Sur le plan des revues de la littérature, plusieurs enquêtes et études ont synthétisé les techniques de compression de modèles appliquées spécifiquement aux Transformers, qui incluent souvent de grands modèles de langage (LLM). Par exemple, X. ZHU et al. (2023) présentent systématiquement les techniques de compression communs et résument les dernières avancées et innovations, tout en introduisant les jeux de données de référence et les tâches standard pour évaluer les modèles compressés, comme GLUE, SuperGLUE et LAMBADA. Cette revue a été fondamentale pour permettre à notre recherche de développer une compréhension globale des études pertinentes et des applications de l'intelligence artificielle frugale ; CHENG et al. (2023) ont publié une synthèse complète de l'état de l'art des techniques d'élagage appliquées aux DNNs. Ils ont catégorisé les différentes méthodes d'élagage et fourni des directives pratiques pour la sélection des techniques les plus appropriées, tout en suggérant des directions de recherche pour l'avenir. Les auteurs ont également maintenu un dépôt GitHub [1], qui regroupe des articles de recherche et des codes source open-source associés.

Pour l'élagage au niveau de couche, SAJJAD et al. (2023) ont étudié l'impact de la suppression de différentes couches dans les modèles Transformers pré-entraînés. Ils ont exploré diverses stratégies telles que la suppression des couches supérieures, la suppression alternée des couches pour offrir des perspectives sur la manière dont ces différentes stratégies d'élagage influencent les performances du modèle dans diverses tâches en aval ; HE et al. (2024) déterminent l'importance des couches en calculant la similarité cosinus entre les entrées et les sorties. Une similarité élevée indique que le module a un faible impact sur la transformation de l'entrée. Ils ont analysé les variations de performances après avoir supprimé un nombre variable de couches d'attention et ont combiné ces suppressions avec des techniques de quantification et d'élagage des couches MLP. Les résultats montrent que la suppression de la moitié des couches d'attention permet de maintenir les performances du modèle, confirmant ainsi l'existence d'une redondance dans ces couches [MICHEL et al., 2019].

1. https://github.com/hrcheng1066/awesome-pruning

Il est intéressant de noter que, pour l'intelligence artificielle frugale, la recherche française a déjà produit plusieurs études et enquêtes existantes. Par exemple, les travaux de PILLET et al. (2024) et de KRICHEN (2024) présentent des synthèses des méthodes dans ce domaine. De plus, AMISSE et al. (2024) ont exploré dans leur étude le concept d'IA frugale et son application dans les politiques publiques, en appelant à une collaboration accrue entre les gouvernements et les institutions de recherche pour promouvoir l'IA frugale afin de répondre aux défis écologiques et énergétiques croissants. Cependant, dans le domaine spécifique de cette recherche — l'élagage structuré au niveau des couches Transformers, les travaux existants ne couvrent pas encore les modèles destinés à la langue française. Par exemple, CamemBERT, une variante de BERT adaptée au français, est l'un des modèles Transformers les plus utilisés pour les tâches NLP françaises. Cependant, les pratiques spécifiques et les recherches sur la compression de ce modèle restent encore limitées.

Cependant, bien que cette recherche s'inspire de l'étude de Sajjad et al. (2023), elle se distingue en plusieurs points essentiels. Tout d'abord, nous travaillons sur le modèle CamemBERT appliqué à la langue française, ce qui peut entraîner des réactions différentes par rapport aux modèles en anglais. Ensuite, nous explorons des tâches plus variées à différents niveaux d'analyse : le POS tagging pour des informations en surface, la NER pour des dépendances lexicales et syntaxiques, et l'analyse sémantique avec Allociné. Là où Sajjad et al. se sont concentrés sur des tâches du benchmark GLUE, nous cherchons également à identifier s'il existe une corrélation entre la nature des tâches en aval et les couches du modèle à conserver ou élaguer. Ces distinctions visent à fournir des insights plus pratiques et approfondis pour l'optimisation des modèles pré-entraînés dans des contextes variés.

Inspirée par l'étude de SAJJAD et al. (2023), cette recherche se distingue en plusieurs points essentiels. Tout d'abord, nous travaillons sur le modèle CamemBERT appliqué à la langue française. Ensuite, au lieu de nous concentrer sur des tâches GLUE, nous explorons des tâches plus variées à différents niveaux d'analyse : le POS tagging, la NER, et l'analyse sémantique avec *Allociné*. Enfin, nous cherchons également à identifier la corrélation entre la nature des tâches en aval et les couches à conserver ou élaguer. Ces distinctions visent à fournir des intuitions plus pratiques et utiles pour l'application du modèle.

2.3 *Parameter Efficient Fine-Tuning* (PEFT)

Pour notre recherche, nous souhaitons obtenir les meilleures performances possibles tout en maintenant la légèreté du modèle. Ainsi, en plus de l'élagage structuré, nous envisageons d'utiliser le technique PEFT [HOULSBY et al., 2019] pour réduire davantage la taille de notre modèle.

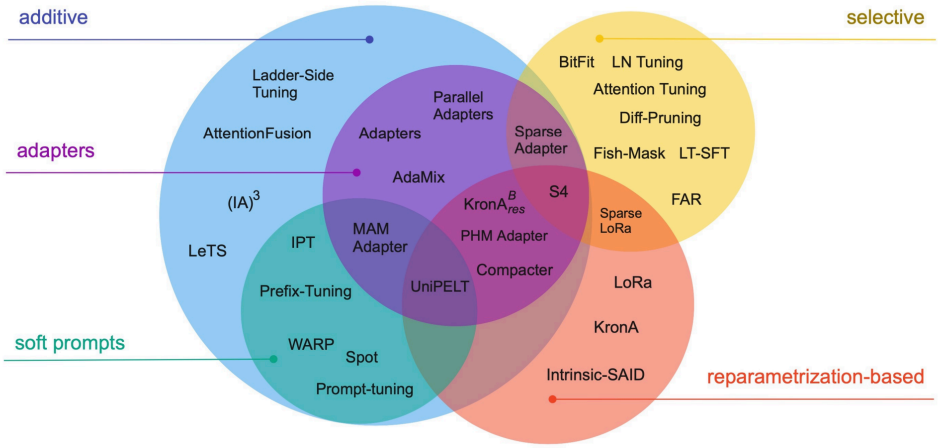

FIGURE 2.2 – Taxonomie des méthodes de PEFT (LIALIN et al., 2023)

Parmi leurs recherches sur le PEFT, LIALIN et al. (2023) ont réalisé une revue systématique et comparative de plus de 40 articles publiés entre février 2019 et février 2023. Ils classifient le PEFT en trois grandes catégories (comme illustré à la Figure 2.2) et ajoutent deux sous-catégories, Adapters et Soft prompts, dans la classification Additive. Pour mieux illustrer ces classifications, en particulier les méthodes de reparamétrisation, nous proposons encore une fois ici une analyse simplifiée de l'architecture des Transformers. Au sein de cette architecture, le bloc de base du Transformer, tel qu'illustré dans la figure 2.3 , se compose principalement de la couche d'attention multi-têtes (MHA) et du réseau de neurones entièrement connecté (FFN, bloc "Feed Forward" dans la figure). Les couches Add & Norm indiquées dans le schéma font référence à l'ajout de résidus et à la normalisation, qui servent à éviter l'extinction du gradient dans les réseaux profonds et à stabiliser l'entraînement. L'attention à produit scalaire pondéré (Scaled Dot-Product Attention) est l'une des composantes essentielles du modèle Transformer, dont la formule est donnée ci-dessous :

$$\text{Att}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right) * V \tag{2.1}$$

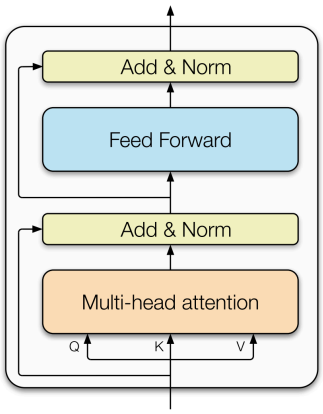

FIGURE 2.3 – Illustration transformer block (VASWANI et al., 2017)

Commençons par les méthodes additives. Cette catégorie se caractérise par l'ajout de réseaux entièrement connectés entre les couches du modèle. Dans la figure 2.4, nous pouvons observer la structure d'un Transformer après l'ajout d'adapters, ainsi que la composition interne de ces derniers. Les adapters y ont été introduits pour la première fois dans deux articles publiés par REBUFFI et al. (2017) et REBUFFI et al. (2018), principalement pour la classification d'images. HOULSBY et al. (2019) et ses collègues ont appliqué cette approche au domaine du NLP. Ils ont ajouté un réseau entièrement connecté après les couches d'attention et les couches FFN, dont la dimension est inférieure à celle de l'entrée. Cette méthode permet d'ajuster moins de 4% des paramètres totaux du modèle tout en atteignant des performances comparables à celles d'un full *fine-tuning*.

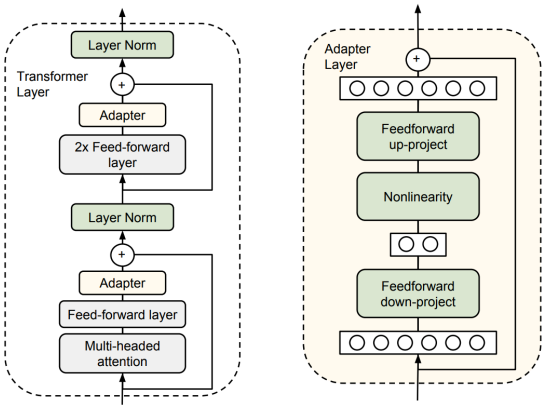

FIGURE 2.4 – Illustration de l'ajout d'adapters dans un modèle Transformer (HOULSBY et al., 2019)

Un autre sous-groupe de la classification additive est celui des *Soft Prompts*. Nous

l'associons souvent à la notion de *Hard Prompts*, qui sont des invites textuels conçus manuellement, constitués de tokens d'entrée discrets, ce qui correspond à la pratique actuelle très répandue du *prompt engineering*. Cependant, la conception manuelle des prompts nécessite souvent une expertise considérable et des essais répétés, ce qui engendre un coût humain important. À l'inverse, les *Soft Prompts* sont des vecteurs continus appris, qui peuvent être optimisés par des méthodes de gradient sur des ensembles de données spécifiques, ce qui les rend beaucoup plus efficaces.

Dans le domaine des *Soft Prompts*, l'une des techniques les plus courantes est le *Prompt Tuning*, un mécanisme proposé par LESTER et al. (2021). Le *Prompt Tuning* se concentre uniquement sur l'entraînement de la couche de représentation des prompts nouvellement ajoutés, tandis que tous les autres paramètres du modèle restent fixes. Ainsi, il n'est pas nécessaire de réajuster l'ensemble du modèle. Lester et ses collègues ont démontré l'efficacité de cette méthode sur plusieurs tâches en aval en utilisant le modèle T5.

Sur la base du *Prompt Tuning*, le laboratoire NLP de l'université Tsinghua a proposé la technique *P-Tuning*, qui inclut les versions v1 et v2. La version p-tuning v1, proposée par LIU et al. (2023), utilise un *prompt encoder* (BiLSTM+MLP) pour encoder des pseudo-prompts en représentations vectorielles continues. Ces embeddings de prompts sont ensuite concaténés aux embeddings d'entrée avant d'être introduits dans le modèle de langage pré-entraîné. Pendant l'entraînement, seuls les paramètres du *prompt encoder* sont optimisés. La version P-Tuning v2 supprime le LSTM présent dans la v1 et introduit des embeddings de prompts entraînables à chaque couche du Transformer pour améliorer davantage l'efficacité du *fine-tuning* [LIU et al., 2022]. Cela permet à P-Tuning v2 de mieux s'adapter aux modèles de différentes tailles et à diverses tâches NLP.

Le **PEFT basé sur la reparamétrisation** est actuellement l'une des approches les plus courantes et les plus pratiques pour le *fine-tuning*. L'idée centrale de ce genre de méthode est de supposer que la matrice des paramètres ajustés, notée ΔW, obtenue lors du *fine-tuning* d'un LLM sur une tâche en aval, est de faible rang. Autrement dit, bien que la matrice soit de haute dimension et contienne des paramètres redondants, la matrice réellement efficace est de dimension inférieure.

La méthode la plus connue de cette catégorie est *LoRA* (Low-Rank Adaptation), proposée par HU et al. (2021). LoRA réalise un *fine-tuning* efficace des paramètres en

décomposant la matrice des paramètres ajustés en un produit de deux matrices de faible rang. Plus précisément :

Dans un *fine-tuning* classique, tous les paramètres du modèle sont optimisés. Par exemple, étant donné une matrice de poids initiale $W_0 \in \mathbb{R}^{d\times k}$, le *fine-tuning* ajoute une matrice d'ajustement ΔW pour obtenir une nouvelle matrice de poids W' :

$$W' = W_0 + \Delta W$$

Pour une entrée donnée x, la sortie du modèle ajusté est alors :

$$h = W'x = (W_0 + \Delta W)x = W_0x + \Delta Wx$$

LoRA suppose que la matrice d'ajustement ΔW peut être approximée par le produit de deux matrices de rang réduit. Au lieu d'apprendre directement ΔW, cette matrice est décomposée en deux matrices $A \in \mathbb{R}^{r\times k}$ et $B \in \mathbb{R}^{d\times r}$, avec $r \ll d$, de manière à ce que :

$$\Delta W = BA$$

Par conséquent, la sortie du modèle LoRA devient :

$$h = W_0x + \Delta Wx = W_0x + BAx$$

Pendant le *fine-tuning*, seules les matrices A et B sont optimisées, ce qui réduit le nombre total de paramètres ajustés de $d \times k$ à $r \times (d + k)$, où r est beaucoup plus petit que d et k.

Aujourd'hui, grâce à son efficacité et à sa facilité en terme d'usage, *LoRA* est largement appliqué dans les domaines de NLP. Il est également intégré dans de nombreux frameworks et outils populaires de deep learning, comme la bibliothèque *Transformers* de Hugging Face. Par conséquent, nous envisageons d'utiliser *LoRA* pour effectuer un *fine-tuning* supplémentaire après l'élagage, dans le but de combiner ces deux techniques et de réduire encore davantage les besoins en ressources du modèle.

2.4 Conclusion

Dans cette section, nous avons exploré d'avantage des techniques de compression de modèles, en mettant particulièrement l'accent sur l'élagage structuré et le *fine-*

tuning efficace en paramètres. Nous avons commencé par une revue approfondie de la littérature sur l'élagage, retraçant son évolution depuis les méthodes pionnières jusqu'aux approches modernes adaptées aux architectures de deep learning. Cette analyse a révélé l'efficacité de l'élagage pour réduire la complexité des modèles ; En parallèle, nous avons examiné les techniques de PEFT, en nous concentrant sur la méthode de *LoRA*. L'élagage structuré, en synergie avec *LoRA*, pourrait être particulièrement avantageux pour les modèles en langue française, sujet que nous explorerons plus en détail dans le chapitre suivant.

Deuxième partie

Expérimentations

CHAPITRE 3

CORPUS ET MÉTHODOLOGIE

Sommaire

Avant de présenter et de discuter nos résultats, nous allons dans ce chapitre introduire de manière systématique les ensembles de données et les méthodes de recherche utilisés dans cette étude.

3.1 Cadres et Scripts

Dans cette section, nous présentons notre modèle étudié, avant de raconter les détails de notre dépôt, y compris la structure des fichiers, leurs fonctionnalités ainsi que les formats d'entrée et de sortie utilisés.

Le modèle NLP que nous avons choisi, intitulé *CamemBERT*, est basé sur RoBERTa et a été entraîné sur 138 Go de textes en français, se concentrant sur les tâches NLP en français. Ce modèle, développé par MARTIN et al. (2020), a surpassé d'autres modèles multilingues dans la plupart des tâches en aval (comme l'annota-

tion POS, l'analyse syntaxique en dépendance, la reconnaissance d'entités nommées etc. Par conséquent, nous l'avons choisi pour mener nos recherches.

Notre bibliothèque se compose principalement de quatre catégories de fichiers :

1. **Fichiers de corpus** : Pour les tâches du POS tagging et de la NER, nous utilisons des fichiers CoNLL locaux. Pour la tâche sémantiques, nous chargeons les données à partir de la bibliothèque `datasets`.
2. **Scripts d'exécution** : Ceux-ci incluent le script principal en Python `train_token_classification.py`, qui contient la majeure partie du code de cette étude. Un script de dataset personnalisé, `custom.py`, est également utilisé conjointement avec le script principal pour charger les bases de données locales.
3. **Fichiers de configuration** : Ces fichiers comprennent le fichier JSON pour la configuration des paramètres LoRA et le fichier `requirements.txt` pour la configuration de l'environnement Python.
4. **Notebooks d'exécution** : Utilisés pour configurer l'environnement, définir les variables selon la stratégie d'élagage, exécuter les scripts, observer et sauvegarder les résultats expérimentaux.

Le script principal est composé de plusieurs modules fonctionnels, notamment :

1. **Prétraitement des données** : Nous utilisons la bibliothèque `datasets` pour charger les données à partir des jeux de données de Hugging Face. Les datasets personnalisés sont chargés via `custom.py`. Après le chargement des données, la fonction `tokenize_and_align_labels` est appelée pour effectuer la tokenisation et l'alignement des étiquettes.
2. **Configuration du modèle et d'élagage** : Ce module utilise la classe `AutoModelForTokenClassification` (ou `SequenceClassification`) de la bibliothèque `transformers` pour charger CamemBERT. Avant l'entraînement, les fonctions `make_layers_indices` et `filter_layers` sont employées pour effectuer l'élagage des couches. Le module de configuration LoRA charge ensuite les paramètres depuis un fichier JSON et applique ces paramètres au modèle à l'aide de la fonction `get_peft_model` de la bibliothèque `peft`.
3. **Entraînement et d'évaluation** : Nous utilisons la classe `TrainingArguments` pour configurer les paramètres d'entraînement, tels

que le taux d'apprentissage, le batch-size, le nombre d'epoch, etc. Ensuite, l'entraînement du modèle est effectué à l'aide de la classe `Trainer`. À la fin de l'entraînement, la fonction `compute_metrics` est utilisée pour calculer les métriques d'évaluation.

4. **Sortie des résultats** : Ce module utilise la fonction `dump` pour sauvegarder les hyperparamètres du modèle, la configuration de l'entraînement et la stratégie d'élagage dans un fichier CSV.

3.2 Stratégies de recherche

Nous allons maintenant nous concentrer sur notre recherche en détaillant spécifiquement nos démarches expérimentales.

Tout d'abord, en ce qui concerne l'élagage, nous avons opté dans cette étude pour une méthode d'élagage au niveau des couches. Cette approche, qui est structurée, fonctionne en supprimant des couches de transformer entières. Plus précisément, nous évaluons d'abord empiriquement l'importance de chaque couche, puis nous classons les couches selon leur importance pour implémenter différentes stratégies de suppression de couches.

3.2.1 Stratégies de l'élagage

Pour chaque tâche, nous employons 5 stratégies d'élagage sur les 12 couches du CamemBERT comme suit :

Conservation d'une seule couche

Cette stratégie, inspirée par l'étude de SAJJAD et al. (2023), consiste à réduire-Nous réduirons le modèle pour ne conserver qu'une seule couche et évaluerons ses performances dans des tâches en aval. Cette approche nous aidera à comprendre l'importance de chaque couche dans des tâches spécifiques. L'hypothèse sous-jacente est que les différentes couches du modèle Transformer apprennent des caractéristiques de diverses importances. En évaluant les performances de chaque couche individuellement, nous pouvons identifier les couches qui contribuent le plus aux tâches en aval, ainsi que celles qui pourraient être redondantes. Cela fournit une base pour les stratégies d'élagage futures.

Conservation des N meilleures couches

Partant du principe que les meilleures couches contiennent les informations les plus critiques du modèle et également inspirés par SAJJAD et al. (2023), nous tenterons de conserver les N couches les plus performantes, basées sur les évaluations individuelles. Le nombre N varie de 2 à 6.

Supression des N pires couches

Considérant que les couches les moins performantes contribuent le moins aux tâches, voire peuvent être redondantes, leur suppression pourrait réduire la charge de calcul et le nombre de paramètres du modèle sans impacter significativement les performances. Ainsi, inspirés par l'approche explorée par SAJJAD et al. (2023), nous supprimerons les N couches les moins performantes, où n variera aussi de 2 à 6.

Conservation des N premières/dernières couches

Inspirés par l'étude de SAJJAD et al. (2023), nous conservons respectivement les n premières couches (par exemple de la couche 0 à 6) et les n dernières couches (par exemple les couches 9, 10 et 11). Cette stratégie repose sur l'hypothèse que les premières couches sont généralement considérées comme captant des caractéristiques syntaxiques et lexicales de bas niveau, tandis que les dernières couches pourraient capter des informations plus abstraites et spécifiques à la tâche. Ainsi, conserver les premières ou les dernières couches aide à évaluer l'importance de ces caractéristiques dans des tâches spécifiques.

Élagage alterné

Comme ce qui a été exploré par SAJJAD et al. (2023), nous supprimons alternativement des couches dans le modèle, telles que les couches 1, 3, 5, 7, 9, 11. Cette stratégie est également basée sur l'hypothèse de redondance d'information dans les réseaux de neurones profonds. Les couches adjacentes peuvent apprendre des caractéristiques similaires, donc supprimer une partie de ces couches ne devrait pas affecter significativement les performances du modèle.

Élagage symétrique

Cette stratégie, inspirée par l'étude de SAJJAD et al. (2023), implique de vider les couches intermédiaires du modèle (telles que les couches 5 et 6, ou 4, 5, 6 et 7) ou d'effectuer une taille symétrique à partir des deux extrémités (par exemple, suppression des couches 0 et 11, ou 0, 1, 10 et 11).

Pour la première méthode, nous supposons que les couches supérieures et inférieures, servant respectivement de points d'entrée et de sortie, sont potentiellement plus importantes que les couches intermédiaires. Mais si la performance du modèle ne diminue pas significativement en élaguant les couches des deux bouts, cela pourrait indiquer que les couches intermédiaires jouent également un rôle crucial dans l'extraction des caractéristiques et le traitement des tâches.

3.2.2 Stratégies sur le LoRA

Augmentation du temps de *Fine-tuning* après élagage

Afin de minimiser la perte de performance après l'élagage, nous proposons une nouvelle stratégie : augmenter le temps de *fine-tuning* sur les modèles élagués. En prolongeant le temps d'entraînement en *fine-tuning*, les modèles élagués ont davantage d'opportunités d'ajuster leurs paramètres et de mieux s'adapter à la nouvelle structure du réseau, afin de restaurer au maximum les performances perdues.

Concrètement, notre approche consiste d'abord à effectuer un *fine-tuning LoRA* standard sur le modèle après l'application de la stratégie d'élagage optimale, en notant le temps d'entraînement nécessaire et les performances obtenues. À ce stade, le temps d'entraînement du modèle est généralement plus court que celui du modèle original, mais cela peut s'accompagner d'une certaine diminution de performance.

Ensuite, nous augmentons le temps de *fine-tuning* pour égaler celui du modèle original non élagué. Par exemple, si le modèle après l'élagage ne nécessite que 10 minutes d'entraînement, mais que le modèle original en nécessite 20, nous poursuivons l'entraînement du modèle élagué jusqu'à 20 minutes. Concrètement, notre approche consiste à ajuster le nombre d'epochs de *fine-tuning* pour le modèle élagué afin de faire correspondre le temps d'entraînement à celui du modèle original non élagué.

Pour ce faire, nous calculons d'abord le temps moyen d'entraînement par époque pour le modèle original et le modèle élagué. Soit baseline_time le temps total d'en-

traînement du modèle original sur n époques, et pruned_time le temps total d'entraînement du modèle élagué sur n époques. Le temps moyen par époque pour chaque modèle est donné par :

$$\text{baseline\_epoch\_time} = \frac{\text{baseline\_time}}{n}, \quad \text{pruned\_epoch\_time} = \frac{\text{pruned\_time}}{n}.$$

Ensuite, nous déterminons le nombre d'époques nécessaires pour le modèle élagué afin que son temps d'entraînement total soit équivalent à celui du modèle original. Le nombre d'époques ajusté pour le modèle élagué, avec le nom *pruned_epochs*, est calculé comme suit :

$$\text{new\_pruned\_epochs} = \frac{\text{baseline\_time}}{\text{pruned\_epoch\_time}}$$

Pour obtenir un nombre d'époques entier, nous arrondissons pruned_epochs à l'entier le plus proche :

$$\text{pruned\_epochs} = \text{round}\left(\frac{\text{baseline\_time} \times n}{\text{pruned\_time}}\right).$$

Enfin, nous continuons l'entraînement du modèle élagué pendant *pruned_epochs* époques supplémentaires pour égaler le temps d'entraînement total du modèle original. Nous comparons (1) les performances du modèle élagué avec temps d'entraînement prolongé à (2) celles du modèle original et du (3) modèle élagué sans prolongation de temps, pour évaluer l'impact de l'allongement du temps d'entraînement sur les performances.

Ajustement des paramètres LoRA

Dans le cadre de notre étude sur l'impact de la compression des modèles, notre recherche générale n'est pas destinée à une tâche spécifique, ainsi, nous ne cherchons pas à réaliser un ajustement extrême des paramètres *LoRA*, que nous avons appelé à partir de la bibliothèque *Hugging Face* en Python, avec les paramètres comme suit :

```
{
    "inference_mode": false,
    "r": 8,
    "lora_alpha": 32,
    "lora_dropout": 0.1,
```

```
    "target_modules": ["query", "value"]
}
```

3.3 Environnement de travail

Cette étude a été réalisée sur la plateforme de serveur cloud AutoDL.

Matériel Nous avons choisi d'utiliser NVIDIA RTX 3080 Ti comme GPU, avec une mémoire de 12GB, afin de concilier l'efficacité expérimentale et le coût.

Logiciel Nous avons utilisé les versions suivantes :

— **PyTorch** 2.3.0

— **Python** 3.12

— **CUDA** 12.1

3.4 Corpus de travail

Dans la section précédente, nous avons présenté les idées et les méthodes spécifiques de notre recherche. L'un des objectifs de notre étude est d'examiner l'impact de l'élagage structuré des différentes couches du modèle sur la performance des tâches. Pour ce faire, nous devons nous baser sur trois tâches de traitement du langage naturel, impliquant plusieurs corpus de données.

3.4.1 WikiNER

Le corpus WikiNER est utilisé pour les tâches de reconnaissance des entités nommées (NER). Ce corpus est publié par NOTHMAN et al. (2013) et marque automatiquement les entités nommées en utilisant les hyperliens entre les articles de Wikipédia et inclut une vérification et une correction manuelle partielle.

Ce corpus comprend neuf langues, dont le français, basé sur 980 773 articles. Comparé à d'autres ensembles de données NE annotés automatiquement, WikiNER présente l'avantage d'avoir été largement utilisé pour entraîner de nombreux systèmes, ce qui facilite la comparaison des performances entre différents modèles. Bien que l'annotation de WikiNER ne soit pas considérée comme un standard d'or, son

utilisation étendue permet d'évaluer efficacement les modèles dans les tâches de reconnaissance des entités nommées, notamment les variations de performances après l'application des techniques d'élagage structurel.

3.4.2 French Treebank

Le French Treebank (FTB) est un projet initié à l'université Paris Diderot en 1996 et publié en 2003 [ABEILLÉ et al., 2003]. Il contient des couches d'annotation morphosyntaxique et syntaxique. Les textes proviennent du journal *Le Monde* de 1989 à 1995. Pour ce jeu de données, nous avons utilisé ses excellentes ressources d'annotation POS [CRABBÉ et CANDITO, 2008] pour tester nos modèles de compression en ce qui concerne l'étiquetage des parties du discours (POS).

En parallèle, nous avons également réalisé des tâches de la NER sur ce corpus. Le corpus de base du FTB ne comporte pas d'annotations NER. Mais SAGOT et al. (2012) ont présenté une nouvelle couche d'annotation d'entités nommées pour le French TreeBank (FTB), alignée avec la version du French TreeBank. L'annotation NER a d'abord été réalisée par pré-annotation automatique, puis corrigée manuellement. Nous avons également choisi ce jeu de données comme l'un de nos jeux de test pour les tâches de NER.

3.4.3 *Allociné*

Le jeu de données *Allociné* [1], créé par Théophile Blard, est un corpus en langue française utilisé pour l'analyse de sentiments. Les textes sont des critiques de films écrites entre 2006 et 2020 par les membres de la communauté du site *Allociné.fr* pour divers films. Il contient 100 000 critiques positives et 100 000 critiques négatives, divisées en ensembles d'entraînement (160 000), de validation (20 000) et de test (20 000). Il supporte les tâches de classification de texte ou de classification de sentiments en polarité (positif ou négatif). La performance du modèle est évaluée en fonction de la précision des étiquettes prédites par rapport aux étiquettes fournies dans le jeu de données. Un modèle basé sur BERT, tf-allociné, atteint une précision de 97,44 % sur l'ensemble de test. Par conséquent, il a été utilisé pour évaluer les performances de notre modèle dans les tâches NLP de type sémantique (classification de sentiments).

1. Lien vers le dataset sur Huggingface : https ://huggingface.co/datasets/tblard/allocine

Cependant, étant donné la taille importante de ce jeu de données et les ressources matérielles limitées à notre disposition, l'entraînement sur l'ensemble complet aurait entraîné un coût en temps considérable. Pour cette raison, nous avons décidé de réduire l'échelle de l'expérience en utilisant seulement 10% des données originales. Cette approche nous a permis de réaliser nos expériences tout en minimisant le temps de calcul requis et en conservant des bonnes performances.

3.4.4 CoNLL

La plupart des jeux de données utilisés dans cet article sont au format CoNLL (*Conference on Computational Natural Language Learning*), un format tabulaire adapté à l'annotation des tokens, souvent structuré au niveau de la phrase. Bien qu'il puisse y avoir des variations, il s'agit généralement d'un tableau avec des lignes vides pour séparer des sections, comme des phrases.

Il existe de nombreux formats CoNLL différents, car chaque tâche partagée annuelle de CoNLL a ses propres spécificités. Parmi ceux-ci, le jeu de données CoNLL-2003 SANG et DE MEULDER, 2003 est un format de référence standard dans le domaine du traitement automatique des langues utilisé pour les tâches de reconnaissance des entités nommées. En outre, nous pouvons également développer les scripts Python ou utiliser des scripts [2] existants pour convertir les ensembles de données au format CoNLL.

Ce chapitre a établi les fondements méthodologiques de notre étude. En examinant l'impact de ces techniques sur différents jeux de données tels que WikiNER, French Treebank, et *Allociné*, nous avons pu identifier les configurations optimales pour divers genre de tâches, en tenant compte de l'équilibre entre performance et complexité. Dans le prochain chapitre, nous aborderons l'analyse des résultats obtenus, qui fournira une évaluation approfondie de l'efficacité de nos approches.

2. https ://github.com/kata-ai/wikiner/blob/master/reader2conll.py

CHAPITRE 4

RÉSULTATS

Sommaire

Dans ce chapitre, nous présenterons ces résultats expérimentaux existants avec des analyses succinctes. Nous présenterons également les résultats de l'application de LoRA et après l'augmentation du temps d'entraînement.

4.1 Résultats d'élagage

Nous allons présenter dans cette section les résultats de nos expériences sur les trois tâches, à savoir l'étiquetage POS, la reconnaissance des entités nommées et la classification des sentiments.

4.1.1 Résultats de la tâche de NER

Pour la tâche de NER, nous avons mené des expériences sur deux corpus : WikiNER et French Treebank. Le tableau 4.1 présente les performances de CamemBERT sans élagage sur les deux jeux de données.

Nous constatons que le modèle obtient de très bonnes performances sur la tâche WikiNER, avec un F1-score atteignant 0.9028. En revanche, sur la tâche FTB_NER,

Model	Dataset	Stratégie	Accuracy	F1-score	Precision	Recall
CamemBERT	WikiNER	full layers	98.58%	90.28%	89.66%	90.92%
CamemBERT	FTB_NER	full layers	98.7%	79.87%	78.28%	81.52%

TABLE 4.1 – Performance du modèle CamemBERT sur la tâche de NER

ses performances sont inférieures, avec un F1-score de 0.7987. Des stratégies d'élagage sont effectué après avoir évalué les performances du modèle de baseline.

Conservation d'une seule couche

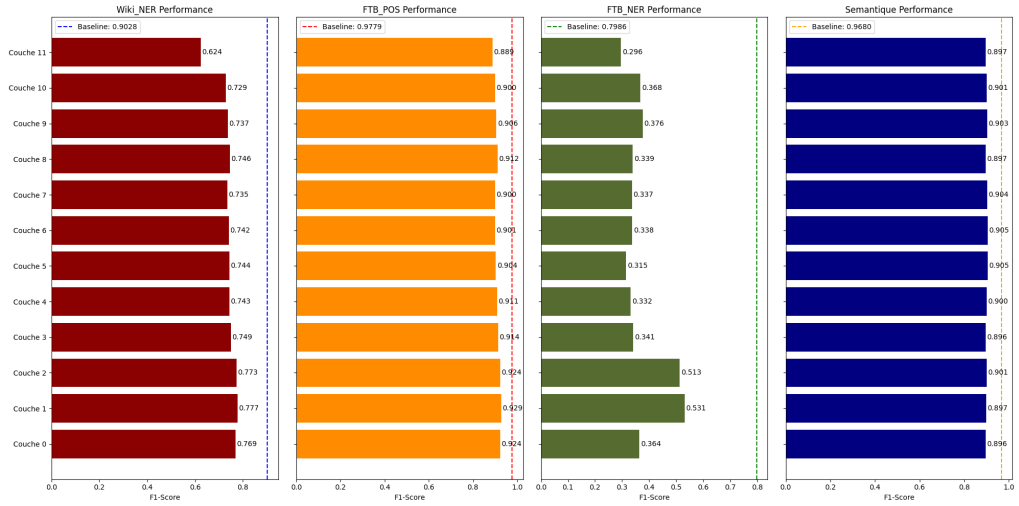

FIGURE 4.1 – Performances du modèle sur les 4 tâches avec conservation d'une seule couche

Nous avons regroupé les performances d'une seule couche sur les quatre dataset dans la figure 4.1. Pour la tâche WikiNER, le F1-score moyen des 12 couches est de 0.7421, avec un maximum de 0.777 et un minimum de 0.6235. En revanche, les performances d'une seule couche sur la tâche FTB_NER sont relativement faibles : le F1-score moyen n'est que de 0.3643, soit moins de la moitié de celui du modèle de baseline. De plus, la variance des performances entre les couches est plus importante, la meilleure performance étant obtenue par la couche 1 avec 0.5313, tandis que la couche 11 a la pire performance, avec seulement 0.2962.

Par ces deux séries d'expériences, nous constatons que les différentes couches ont un impact significativement variable sur les performances des tâches de NER. Nous observons que la couche la plus haute (couche 11) présente des performances nettement inférieures par rapport aux autres couches. De plus, nous avons constaté une relation inverse entre le niveau et son F1-score : les couches inférieures montrent de

meilleures performances, tandis que les couches supérieures affichent des résultats plus faibles.

Conservation de n premières/dernières couches

Une relation potentielle entre la performances d'une seule couche et sa position nous a incités à effectuer un élagage unilatéral sur les deux datasets. Nous avons considéré que, si dans la figure 4.1, les couches inférieures montrent de meilleures performances, alors l'élimination des couches supérieures (Top N) tout en conservant les couches inférieures (Bottom_N) devrait donner de meilleures performances que l'élimination des couches inférieures (Bottom N).

Premièrement, pour le jeu de données WikiNER, les résultats obtenus avec cette stratégie confirment l'hypothèse précédente. En conservant uniquement les couches 0 et 1, puis en augmentant progressivement jusqu'à conserver les couches 0 à 5, les performances continuent d'augmenter. Cependant, à mesure que le nombre de couches conservées augmente, l'incrément du F1-score ralentit, ce qui montre non seulement l'importance des couches inférieures dans l'amélioration des performances du modèle, mais aussi que les informations cruciales sont principalement encodées dans des couches médianes, comme attendu dans les modèles de ce type.

En revanche, la conservation de la partie supérieure des couches (de la couche 6 à la couche 11) ne permet pas d'améliorer les performances de manière significative, confirmant que les couches plus hautes dépendent des informations capturées par les couches inférieures. Cela reflète que, bien que les couches supérieures jouent un rôle dans les représentations finales, en isolation, les couches inférieures restent critiques pour capturer les caractéristiques les plus importantes. Un phénomène similaire est observé sur le jeu de données FTB_NER, comme illustré dans la figure 4.2.

Nous avons également constaté que les courbes ne suivent pas toujours une tendance à la hausse. Par exemple, pour la performance sur le datasset FTB_NER, les performances de Top_N diminuent après l'ajout de la couche 5, étant inférieures à celles obtenues avec $n = 4$ (réduction de 0,3%). Un phénomène similaire est observé sur la courbe Bottom_N du jeu de données WikiNER. Cela nous amène à nous interroger : la présence des couches intermédiaires est-elle vraiment significative pour les tâches de NER ? Nous voudrions tester cette hypothèse en appliquant des stratégies d'élagage symétrique.

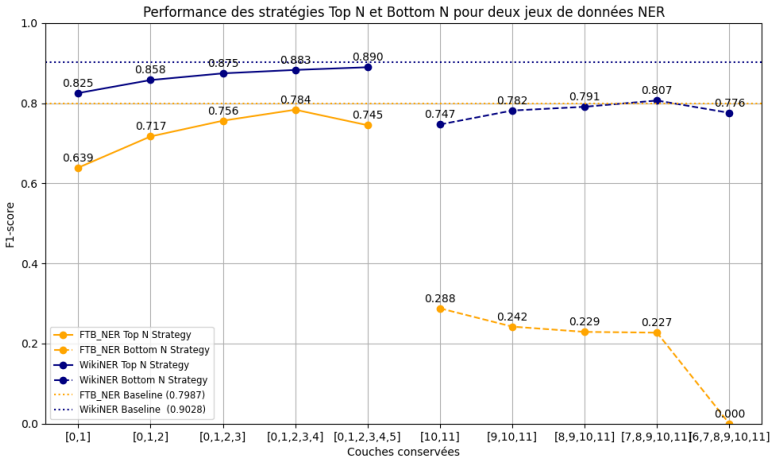

FIGURE 4.2 – Performances des stratégies Top/Bottom_N pour les tâches de NER

Élagage symétrique

Nous avons d'abord effectué un élagage en supprimant les couches centrales tout en conservant symétriquement les couches aux deux extrémités ; parallèlement, nous avons également essayé de supprimer les couches aux extrémités en conservant les couches centrales. La figure 4.3 montre que, pour la tâche WikiNER, conserver les couches centrales pourrais donner de meilleures performances que conserver les couches aux extrémités.

En revanche, pour la tâche FTB_NER, conserver les couches centrales s'avère être une meilleure option. Cependant, nous avons observé que, quelle que soit la stratégie d'élagage symétrique appliquée, le nombre de couches à conserver pour atteindre un niveau de performance équivalent est souvent supérieur à celui requis pour l'élagage unilatéral. Cela indique que, pour les tâches de NER, les stratégies d'élagage symétrique sont moins efficaces que les stratégies d'élagage unilatéral.

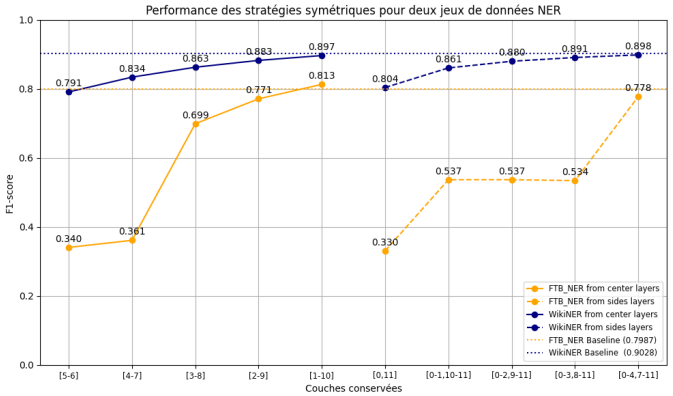

FIGURE 4.3 – Performances des stratégies symétrique pour les jeux de données NER

Conservation de n meilleures couches

Notre stratégie d'élagage unilatérale repose sur l'hypothèse d'une corrélation entre la performance d'une seule couche et sa position. Cependant, les résultats de nos expériences montrent que les performances d'une seule couche ne suivent pas une tendance strictement croissante ou décroissante. Par conséquent, nous avons envisagé de conserver directement les n meilleures couches individuelles, dans l'espoir que cette approche puisse donner de meilleurs résultats que l'élagage basé sur la position. Nos résultats sont présentés dans le tableau 4.2.

D	Stratégie	F1-Score	Precision	Recall	Layers
FTB_NER	best_2	0.644	0.601	0.693	1, 2
	best_3	0.715	0.671	0.764	1,2, 9
	best_4	0.714	0.676	0.756	1, 2, 9, 10
	best_5	0.775	0.748	0.803	0, 1, 2, 9, 10
	best_6	0.792	0.767	0.818	0, 1, 2, 3, 9 10
	del_worst_1	0.823	0.810	0.838	0 - 10
	del_worst_2	0.810	0.796	0.825	0, 1, 2, 3, 4, 6, 7, 8, 9, 10
	del_worst_3	0.828	0.814	0.843	0, 1, 2, 3, 6, 7, 8, 9, 10
	del_worst_4	0.804	0.783	0.825	0, 1, 2, 3, 6, 8, 9, 10
	del_worst_5	0.773	0.744	0.805	0, 1, 2, 3, 8, 9, 10
WikiNER	best_2	0.838	0.821	0.856	1, 2
	best_3	0.858	0.843	0.873	0, 1, 2
	best_4	0.875	0.862	0.887	0, 1, 2, 3
	best_5	0.884	0.875	0.894	0, 1, 2, 3, 8
	best_6	0.892	0.883	0.900	0, 1, 2, 3, 5, 8
	del_worst_1	0.903	0.896	0.909	0 - 10
	del_worst_2	0.901	0.894	0.908	0 - 9
	del_worst_3	0.901	0.893	0.909	0, 1, 2, 3, 4, 5, 6, 8, 9
	del_worst_4	0.897	0.889	0.904	0, 1, 2, 3, 4, 5, 6, 8
	del_worst_5	0.895	0.887	0.903	0, 1, 2, 3, 4, 5, 8
Allociné	best_2	0.901	0.901	0.902	5, 6
	best_3	0.899	0.899	0.899	5, 6, 7
	best_4	0.895	0.896	0.896	5, 6, 7, 9
	best_5	0.896	0.898	0.898	5, 6, 7, 9, 10
	best_6	0.937	0.938	0.937	2, 5, 6, 7, 9, 10
	del_worst_1	0.968	0.968	0.968	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
	del_worst_2	0.963	0.963	0.963	1, 2, 4, 5, 6, 7, 8, 9, 10, 11
	del_worst_3	0.957	0.957	0.958	1, 2, 4, 5, 6, 7, 8, 9, 10
	del_worst_4	0.951	0.952	0.952	1, 2, 4, 5, 6, 7, 9, 10
	del_worst_5	0.950	0.950	0.951	2, 4, 5, 6, 7, 9, 10
FTP_POS	best_2	0.949	0.949	0.949	0, 1
	best_3	0.965	0.965	0.965	0, 1, 2
	best_4	0.971	0.971	0.971	0, 1, 2, 3
	best_5	0.974	0.974	0.974	0, 1, 2, 3, 8
	best_6	0.976	0.976	0.976	0, 1, 2, 3, 4, 8
	del_worst_1	0.979	0.979	0.979	0 - 10
	del_worst_2	0.978	0.978	0.978	0, 1, 2, 3, 4, 5, 6, 8, 9, 10
	del_worst_3	0.979	0.979	0.979	0, 1, 2, 3, 4, 5, 6, 8, 9
	del_worst_4	0.978	0.978	0.978	0, 1, 2, 3, 4, 5, 8, 9
	del_worst_5	0.977	0.977	0.977	0, 1, 2, 3, 4, 8, 9

TABLE 4.2 – Performances des stratégies Best-N et del_Worst-N

Pour le jeu de données WikiNER, nous constatons que la stratégie best-n présente une grande similitude avec la stratégie Top_n, ce qui confirme à nouveau l'importance des couches inférieures. Conserver les n meilleures couches améliore les performances du modèle à mesure que le nombre de couches conservées augmente, tandis que la suppression de plus de couches les moins performantes entraîne une

dégradation des performances. Cela indique que, pour les tâches de NER utilisant Camembert, la stratégie best-n est plus efficace que la stratégie del_worst-n.

En comparant avec d'autres stratégies, nous avons observé que, pour le jeu de données FTB_NER, la stratégie best-n offre des performances supérieures. Par rapport à la performance du modèle de baseline, la performance du modèle de stratégie Top_6 est déjà très proche de celle-là, avec seulement 0,6% de différence en termes de F1-score. Il en est de même pour le jeu de données WikiNER, dont la stratégie best-n surpasse les autres stratégies avec le même nombre de couches conservées, malgré un écart avec le modèle de référence légèrement plus grand, mais très proche (la perte de F1-score n'est que d'environ 1% avec les 6 meilleures couches).

Élagage alterné

Pour la tâche de NER, nous souhaitons finalement étudier si une distribution plus régulière des couches peut maintenir les performances du baseline. Nous avons appliqué différents niveaux de dispersion, comme illustré dans la figure 4.4.

Nous constatons que cette stratégie montre des différences de performances significatives entre les deux jeux de données. Pour le jeu de données WikiNER, à mesure que le niveau de sparsitée augmente, les performances du modèle diminuent progressivement, mais pas de manière très marquée. En revanche, pour le jeu de données FTB_NER, l'augmentation du niveau de sparsitée entraîne une diminution beaucoup plus marquée des performances. Notamment, lorsque le niveau de dispersion atteint 25%, les performances du modèle chutent de 56% par rapport au modèle de référence ; alors que, sous la même stratégie, les performances du modèle pour WikiNER ne diminuent que de 5,38% par rapport à son modèle de référence. Cette divergence dans les pertes de performance pourrait s'expliquer par la nature des annotations. En effet, WikiNER, ayant été annoté de manière semi-automatique, pourrait contenir des régularités ou des structures plus prévisibles, ce qui rend le modèle moins sensible à l'augmentation de la sparsitée, comparé à un jeu de données comme FTB_NER, où les annotations manuelles reflètent une plus grande diversité et complexité.

En outre, aucune des performances des stratégies n'a surpassé le modèle de référence, et, à nombre de couches équivalent, n'est plus performant par rapport aux autres stratégies.

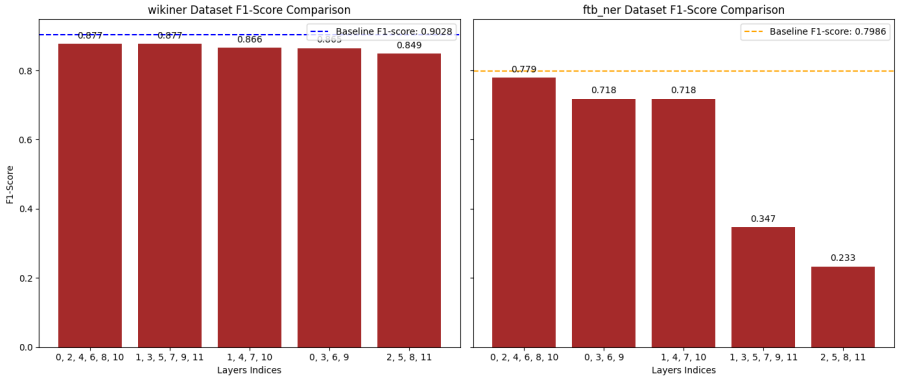

FIGURE 4.4 – Performances des stratégies alternées pour deux jeux de données NER

4.1.2 Résultats de la tâche de POS

Model	Dataset	Stratégie	Accuracy	F1-score	Precision	Recall
CamemBERT	Allociné	full layers	96.80%	96.80%	96.79%	96.84%
CamemBERT	FTB_POS	full layers	97.79%	97.79%	97.79%	97.79%

TABLE 4.3 – Performance du modèle CamemBERT sur les tâches POS et sémantique

Nous constatons dans le tableau 4.3 que le modèle de baseline obtient des performances très élevées sur les tâches de POS tagging et sémantique.

La figure 4.1 donne les qualités de chaque couche en isolation sur la tâche de POS. Comparées aux tâches de NER, les performances d'une seule couche pour la tâche de POS sont en moyenne meilleures, comme le montre la figure 4.1. Pour la tâche FTB_POS, le F1-score moyen en ne conservant qu'une seule couche est de 0.9085, avec un maximum de 0.9288 pour la couche 1 et un minimum de 0.8887 pour la couche 11. Bien que ces scores soient encore inférieurs de 5 à 10% par rapport au modèle de référence, l'écart est plus réduit que celui observé pour les tâches de NER.

En examinant la répartition des performances des différentes couches, on observe une tendance générale à la baisse des performances de la couche inférieure à la couche supérieure, mais cette diminution est également beaucoup moins prononcée que dans les tâches de NER.

Conservation de n premières/dernières couches

Pour la tâche de POS, nous avons également appliqué la stratégie d'élagage unilatéral. Comme le montre la figure 4.5, pour le jeu de données FTB_POS, conserver les couches de la première moitié donne de meilleurs résultats plutôt que de conserver

les couches supérieures. Cela reflète la même tendance observée pour les tâches de NER, mais de manière encore plus prononcée pour les tâches de POS. Cela semble logique, car la tâche de POS tagging est davantage corrélée aux représentations capturées par les couches inférieures.

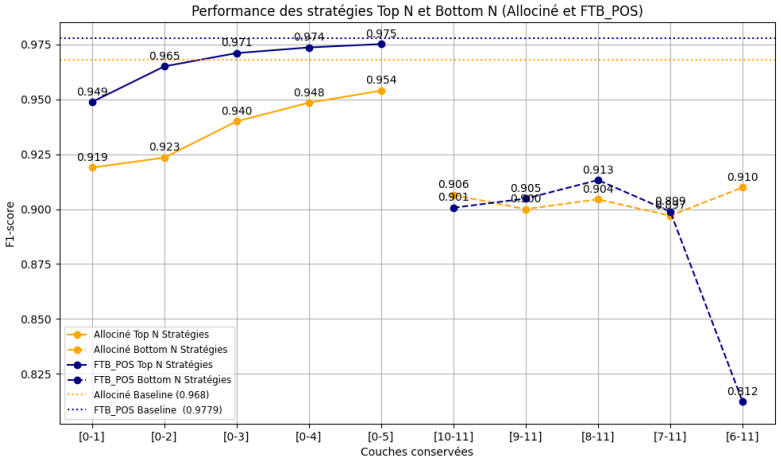

FIGURE 4.5 – Performances des stratégies Top/Bottom_N pour les jeux de données POS et sémantique

Conservation de n meilleures couches

Nous observons que, pour la tâche de POS, les n meilleures couches se situent aussi principalement dans les couches inférieures. Comme indiqué dans le tableau 4.2, pour la tâche de POS, les meilleures couches coïncident aussi très souvent avec les premières couches (Best_4 = Bottom_4). En conservant les 6 meilleures couches, notre modèle a atteint un F1-score de 0.9762, avec seulement 0,0015 de différence de F1-score par rapport au modèle de baseline.

Contrairement aux autres tâches, pour la tâche de POS, la suppression des couches les moins performantes a permis d'obtenir les meilleurs résultats. En supprimant les 1 à 4 couches les moins efficaces, notre modèle a surpassé les performances du modèle de baseline. Ainsi, nous pouvons dire que pour cette tâche, c'est une stratégie qui aboutit potentiellement aux meilleurs résultats.

Élagage symétrique

Dans la figure 4.6, nous constatons que pour la tâche de POS, conserver un petit nombre de couches centrales est relativement moins efficace que de conserver les

couches aux deux extrémités. Globalement, la courbe de droite commence à un niveau plus élevé que celle de gauche, et les points de la courbe de droite sont également répartis à des niveaux plus élevés. Mais lorsque le nombre de couches conservées augmente à 10, les F1-score des deux stratégies convergent vers le même niveau.

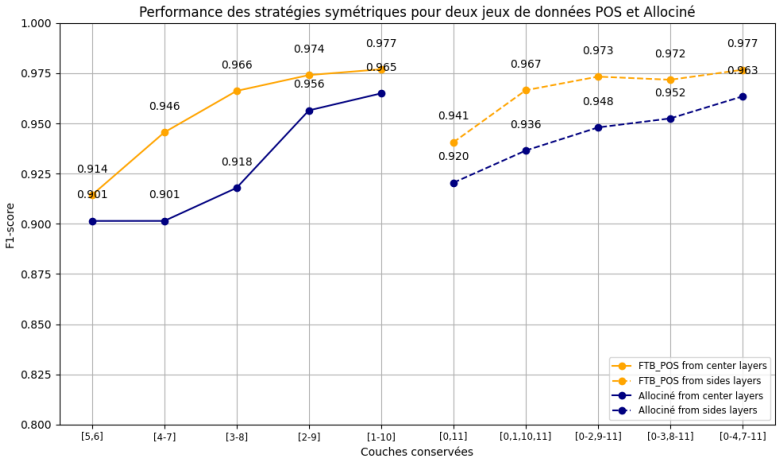

FIGURE 4.6 – Performances des stratégies symétrique pour les jeux de données POS et sémantique

Élagage alterné

Nous présentons dans la figure 4.7 les résultats de l'application de la stratégie d'élagage alterné pour la tâche de POS tagging et d'analyse sémantique. Pour le jeu de données FTB_POS, presque toutes les stratégies d'élagage alterné permettent de compresser le modèle de manière efficace tout en maintenant un niveau de performance proche de l'original. Par exemple, en ne conservant que les couches *0,2,4,6,8*, le modèle conserve encore 99,5% du F1-score baseline. Par conséquent, pour la tâche de POS, l'élagage alterné s'avère également être une stratégie très efficace.

4.1.3 Résultats de la tâche sémantique

Comme le montre la figure 4.1, pour le jeu de données *Allociné*, les modèles avec une seule couche obtiennent encore de bonnes performances sur la tâche d'analyse des sentiments. Le F1-score moyen d'une seule couche atteint également 0.9, avec un maximum de 0.905 pour la couche 6 et un minimum de 0.8958 pour la couche 0, ce qui indique aussi une différence faible de performance entre les couches.

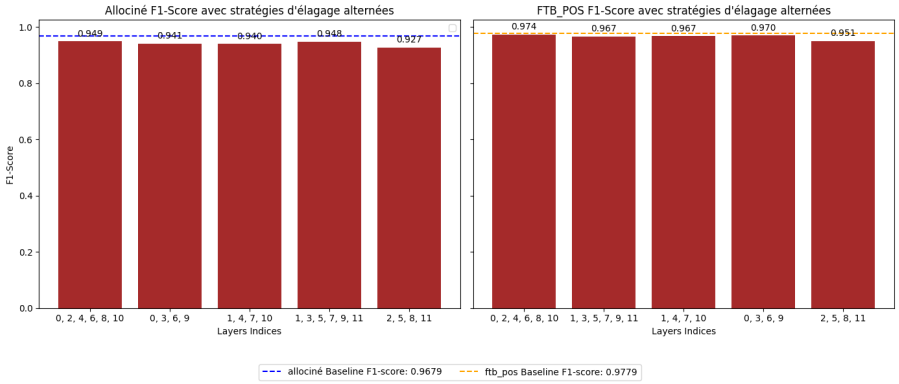

FIGURE 4.7 – Performances des stratégies alternées pour les jeux de données POS et sémantique

Cependant, en examinant la répartition des performances des différentes couches, nous constatons une distribution différente de celle des trois autres tâches précédentes. Parmi les 12 couches, il n'y a pas de tendance claire à la hausse ou à la baisse ; au contraire, les couches intermédiaires ont tendance à avoir de meilleures performances que celles situées aux extrémités.

Conservation de n premières/dernières couches

Nous avons regroupé les résultats de cette stratégie pour FTB_POS et Allociné dans la figure 4.5, parce que ces deux tâches représentent deux types de tâches linguistiques différentes : l'une est basée sur la syntaxe, l'autre sur la sémantique. Nous voulons donc comparer directement les performances des deux types de tâches en fonction de la position des couches conservées. Dans ce graphique, pour le jeu de données *Allociné*, de manière similaire aux deux autres tâches, conserver les couches de la première moitié donne de meilleurs résultats que de conserver les couches de la seconde moitié. Les intervalles de F1-score pour les deux ensembles de données sont distincts, ce qui indique que, pour la tâche d'analyse des sentiments, les combinaisons de couches inférieures produisent des résultats nettement supérieurs à celles des couches supérieures.

Conservation de n meilleures couches

Comme mentionné dans notre analyse des performances d'une seule couche, contrairement aux autres modèles où les meilleures performances se situent généralement dans les couches inférieures ou supérieures, les meilleures couches pour

la tâche sémantique sont principalement situées au milieu (couches 5, 6, 7, 9). Cependant, bien que ces couches affichent individuellement de bonnes performances, lorsque nous les combinons, cela produit une dégradation de performance (Voir tableau 4.2). La performance est moins bonne que les stratégies plus structurées (TOP/-Bottom). Par exemple, en combinant les 5 meilleures couches (5,6,7,9,10), notre modèle n'atteint qu'un F1-score de 0.896, soit une diminution d'environ 7,2% par rapport au modèle de référence.

En outre, nous avons constaté que la suppression des couches les moins performantes est relativement efficace pour la tâche d'analyse des sentiments. En supprimant uniquement la couche la moins performante (couche 11), les performances du modèle ne montrent pratiquement aucune diminution. À mesure que le nombre de couches supprimées augmente, les performances du modèle diminuent progressivement, mais l'écart global reste dans les 2% en terme de F1-score par rapport au modèle de baseline.

Élagage symétrique

Nous nous sommes interrogés sur le fait que si les couches au mileu ont montré de bonnes performances dans les tests précédents, et que les résultats insatisfaisants de la conservation des meilleures N couches pourraient être dus à une discontinuité entre les couches, la stratégie symétrique structurée pourrait-elle être une solution plus efficace ?

Malheureusement, les résultats dans la figure 4.6 montrent que conserver les couches centrales donne des performances encore moins bonnes que conserver les couches aux deux extrémités, et nettement inférieures à celles obtenues avec la stratégie TOP_N avec un même nombre de couches conservées.

Élagage alterné

Comme illustré dans la figure 4.7, nous présentons les résultats de la stratégie d'élagage alterné appliquée au jeu de données *Allociné* : ces stratégies montrent des performances plus diminuées que celles observées pour FTB_POS. Aussi, basés sur les F1-score de chaque combinaison de couche, nous pensons qu'il semble nécessaire de conserver plus de 50% des couches (paires ou impaires), afin de préserver une performance proche de celle de baseline.

4.2 Combinaison d'élagage et de LoRA

Dans cette section, nous voudrions appliquer la méthode de *Fine-tuning* LoRA sur les modèles déjà élagués pour accélérer leur entraînement et améliorer leurs performances. Mais comme nous avons testé de nombreuses stratégies d'élagage, nous devons d'abord sélectionner les meilleures stratégies pour chaque jeu de données, puis les combiner avec LoRA pour obtenir les meilleurs résultats possibles.

Nous considérons que, bien que les stratégies qui conservent un grand nombre de couches (comme 10 ou 11 couches) présentent des performances très proches de celles du modèle de baseline sur certaines tâches, cela n'offre pas de réduction significative de la taille du modèle.

Ainsi, nous estimons qu'il est nécessaire de supprimer au moins 50% des couches pour réaliser une compression significative du modèle. De plus, nous avons constaté que, pour certaines tâches, conserver un nombre réduit de couches suffit déjà à atteindre des performances proches de celles du modèle de référence.

Nous avons collectionné les performances de chaque modèle sous différentes stratégies d'élagage, en prenant pour base les stratégies qui conservent 6 couches et privilégient celles offrant les meilleures performances. Plus précisément, pour les évaluer, nous avons pris en compte deux critères : le nombre de couches conservées et la différence de F1-score par rapport au modèle de baseline. Pour chaque tâche, nous avons constaté que la meilleure performance pour 6 couches ou moins est toujours obtenue avec une combinaison de 6 couches. Par conséquent, nous avons choisi cette meilleure combinaison pour mener des expériences de LoRA. De plus, nous avons sélectionné les meilleures stratégies avec 2 ou 3 couches conservées, puisqu'ils ont aussi une performance proche de celle du modèle de référence, mais avec un nombre de couches plus réduit. Bien entendu, si nous observons qu'une stratégie conservant plus de 6 couches offre des performances nettement supérieures à celles des stratégies conservant 6 couches ou moins, nous incluons également cette stratégie dans les expériences ultérieures. Les meilleures stratégies d'élagage pour chaque jeu de données sont ainsi résumées dans le tableau 4.4.

Étant donné que les performances de FTB_NER ont été médiocres dans les tests précédents, et que la plupart des indicateurs pour FTB_NER après l'application de LoRA sont tombés à 0, nous avons décidé de ne conserver que WikiNER comme jeu

dataset	F1-score	Precision	Recall	Layers	Model_size_mb	Training_time
WikiNER	**0.892**	0.883	0.900	0, 1, 2, 3, 5, 8	257.54	883.83
WikiNER	**0.838**	0.821	0.856	1, 2	149.39	526.64
FTB_NER	**0.792**	0.767	0.818	0, 1, 2, 3, 9, 10	257.56	87.01
FTB_NER	**0.644**	0.601	0.693	1, 2	149.41	42.93
FTB_POS	0.976	0.976	0.976	0, 1, 2, 3, 4, 8	257.60	91.05
FTB_POS	0.965	0.965	0.965	0, 1, 2	176.49	60.15
FTB_POS	0.949	0.949	0.949	0, 1	149.45	50.74
Allociné	0.954	0.954	0.955	0, 1, 2, 3, 4, 5	259.77	614.28
Allociné	0.920	0.920	0.921	0, 11	151.62	239.95

TABLE 4.4 – Meilleures stratégies par dataset avec des couches inférieures ou égales à 6

de données pour nos recherches ultérieures sur les tâches de NER.

Sur la base des stratégies sélectionnées dans la section précédente, nous avons pu compresser le modèle. Ensuite, nous souhaitons étudier si l'application de la technique PEFT sur ces stratégies permet de maintenir les performances tout en accélérant davantage le temps d'entraînement. À titre de comparaison, nous avons d'abord évalué les performances des modèles non élagué après une *fine-tuning* avec PEFT, comme illustré dans le tableau 4.5.

dataset	epoch	F1-score	Precision	Recall	Layers	Model_size_mb	Training_time_sec
FTB_POS	3	0.673	0.673	0.673	full	421.04	112.83
Allociné	3	0.953	0.953	0.954	full	425.39	971.17
WikiNER	3	0.835	0.821	0.849	full	420.92	1202.02

TABLE 4.5 – Performances des modèles non élagués avec LoRA *fine-tuning*

Ensuite, nous appliquons la méthode PEFT sur les modèles élagués, et les résultats sont présentés dans le tableau 4.6.

Nous pouvons constater que les différentes méthodes de *fine-tuning* après le pruning n'ont pas d'impact significatif sur la taille du modèle. De plus, grâce à l'application de LoRA, le temps d'entraînement a été considérablement réduit, diminuant

Dataset	Epoch	F1-score	Layers	Model_name	Model_size_mb	Training_time	LoRA
Allocine	3	**0.905**	[0, 1, 2, 3, 4, 5]	camembert-base	262.596	504.371	1
Allocine	3	**0.954**		camembert-base	259.774	614.277	0
Allocine	3	**0.863**	[0, 11]	camembert-base	154.068	186.889	1
Allocine	3	**0.920**		camembert-base	151.622	239.951	0
FTB_POS	3	**0.620**	[0, 1]	camembert-base	149.721	32.332	1
FTB_POS	3	**0.949**		camembert-base	149.448	50.736	0
FTB_POS	3	**0.668**	[0, 1, 2, 3]	camembert-base	203.985	49.533	1
FTB_POS	3	**0.971**		camembert-base	203.525	71.582	0
FTB_POS	3	**0.716**	[0, 1, 2, 3, 4, 8]	camembert-base	258.248	66.171	1
FTB_POS	3	**0.976**		camembert-base	257.601	91.048	0
WikiNER	3	**0.815**	[0, 1, 2, 3, 5, 8]	camembert-base	258.131	717.911	1
WikiNER	3	0.892		camembert-base	257.542	883.829	0
WikiNER	3	**0.704**	[1, 2]	camembert-base	149.604	326.901	1
WikiNER	3	0.838		camembert-base	149.390	526.642	0

TABLE 4.6 – Comparaison des performances des modèles élagués avec ou sans LoRA

globalement d'environ 10% à 28%. Cependant, l'application de LoRA a un impact notable sur les performances des modèles élagués. En moyenne, les modèles élagués après *fine-tuning* avec LoRA présentent une diminution de performance en termes de F1-score par rapport au *fine-tuning* classique après pruning. Cela est particulièrement vrai pour les tâches de POS, où les performances ont diminué respectivement de 32,9%, 30,3% et 26%.

Dataset	Epoch	F1-score	Layers	Training_time_sec	LoRA
FTB_POS	3	**0.673**	full	112.83	1
	3	0.9779	full	154.42	0
Allociné	3	**0.953**	full	971.17	1
	3	0.968	full	1172.9	0
WikiNER	3	**0.821**	full	1202.02	1
	3	0.903	full	1792.91	0

TABLE 4.7 – Performances des modèles non élagués avec ou sans LoRA *fine-tuning*

Ensuite, nous sommes curieux de savoir si c'est l'élagage qui a causé la diminution des performances lors de l'application de LoRA. Pour cela, nous avons comparé les résultats des deux différents *fine-tuning*s appliqués aux modèles non élagués, comme le montre la figure 4.7.

Nous avons constaté que, pour les tâches NLP non sémantiques, telles que NER et POS, même les modèles non élagués subissent une diminution de performance après un *fine-tuning* LoRA. En revanche, pour la tâche d'analyse des sentiments (ou classification de séquences), les modèles non élagués affichent des performances similaires après un *fine-tuning* avec ou sans LoRA. Cela indique que l'application de LoRA a des effets différents selon le type de tâche NLP, et a un meilleur effet sur des tâches plus complexes et sémantiques.

Dans l'autre aspect, pour les tâches d'analyse des sentiments, la diminution des performances après l'application de LoRA est plus importante pour les modèles élagués que pour les modèles non élagués. Cela suggère que, pour nos tâches sémantiques, les méthodes de compression de modèle telles que LoRA et l'élagage sont efficaces individuellement, mais leur combinaison n'est pas une bonne option. En revanche, pour d'autres tâches NLP, l'effet de LoRA est plus instable, tandis que l'application d'élagage rend le modèle plus proche de la performance de baseline.

4.3 Modèles élagués avec temps d'entraînement équilibré

Jusqu'à présent, nous avons tiré les conclusions suivantes :

1. Pour les tâches de NER et POS, l'application de LoRA ne s'avère pas très efficace, mais certaines stratégies d'élagage seules puissent améliorer les performances.

2. Pour les tâches sémantiques, l'application séparée de LoRA et de l'élagage produit de bons résultats, mais leur combinaison entraîne une dégradation significative des performances.

À ce stade, si nous voulons explorer si l'augmentation du temps d'entraînement peut compenser la perte de performance due à la réduction de la taille du modèle, il serait plus judicieux de se concentrer uniquement sur les stratégies de pruning, sans intégrer LoRA.

Modèle	**T_baseline**	T_élagué	T_élagué/Epoch	**Epoch_ajusté**
wikiner_6	1792.912	**883.829**	294.610	6
wikiner_2	1792.912	526.642	175.547	10
pos_6	154.418	91.048	30.349	5
pos_3	154.418	**60.147**	20.049	8
pos_2	154.418	50.736	16.912	9
allocine_6	1172.899	**614.277**	204.759	6
allocine_2	1172.899	239.951	79.984	15

TABLE 4.8 – Epoch d'entraînement ajusté pour chaque stratégie

Nous avons donc prolongé le temps d'entraînement des modèles élagués optimaux en augmentant le nombre d'epochs. En utilisant la formule présentée précédemment, nous avons calculé le nombre d'epochs supplémentaires nécessaires pour chaque stratégie, en fonction du temps d'entraînement requis par chacune d'elles. Les résultats de nos calculs sont présentés dans le tableau 4.8.

dataset	epoch	F1 (epoch=3)	F1	precision	recall	layers	size	t_time
FTB_POS	5	0.976	0.979	0.979	0.979	[0, 1, 2, 3, 4, 8]	257.6	143.1
FTB_POS	8	0.971	0.974	0.974	0.974	[0, 1, 2]	176.5	139.1
FTB_POS	9	0.949	0.971	0.971	0.971	[0, 1]	149.4	121.3
WikiNER	6	0.892	0.890	0.881	0.900	[0, 1, 2, 3, 5, 8]	257.5	1745.2
WikiNER	10	0.838	0.858	0.843	0.872	[1, 2]	149.4	1367.2
Allociné	6	0.954	0.951	0.952	0.952	[0, 1, 2, 3, 4, 5]	259.8	1240.1
Allociné	15	0.920	0.926	0.926	0.927	[0, 11]	151.6	1161.3

TABLE 4.9 – Performances des modèles élagués avec temps d'entraînement équilibré

Dans le tableau 4.9, nous présentons les résultats des modèles élagués après

réentraînement avec les nouveaux epochs ajustés. Il est important de noter que le temps d'entraînement (training_time) devrait théoriquement être équivalent au temps nécessaire pour le modèle baseline. Cependant, après avoir augmenté le nombre d'epochs, le temps d'entraînement requis n'a pas augmenté proportionnellement comme prévu ; en fait, il est inférieur au temps calculé initialement. Nous supposons que ce phénomène est lié au réglage dynamique du taux d'apprentissage et à la convergence du modèle, mais la différence de temps reste globalement minime.

En examinant les résultats par tâche, nous constatons que, pour les tâches sémantiques sur le jeu de données *Allociné*, conserver un modèle élagué avec 6 couches et effectuer 3 epochs supplémentaires n'a pas apporté d'amélioration de performance. En revanche, pour un modèle réduit à 2 couches et après 12 epochs supplémentaires, une légère amélioration de 6% a été observée par rapport aux performances initiales.

Pour le jeu de données WikiNER, le modèle élagué conservant 6 couches a également été réentraîné avec 3 epochs supplémentaires, mais cela n'a pas non plus permis d'améliorer les performances. Le modèle élagué réduit à 2 couches, après 7 epochs supplémentaires, a vu son F1-score passer de 0.838 à 0.858, indiquant une amélioration qui reste cependant modeste.

Enfin, pour la tâche de POS, nous avons testé trois modèles élagués conservant respectivement 6, 3 et 2 couches. Parmi eux, le modèle conservant la moitié des couches a montré une amélioration notable après 2 epochs supplémentaires, atteignant un F1-score de 0.979, dépassant même celui du modèle baseline. Les deux autres modèles (avec 3 couches et 2 couches) ont également montré des améliorations par rapport à leurs performances initiales.

Dans ce chapitre, nous avons présenté de manière systématique les données expérimentales obtenues, accompagnées des analyses et petites conclusions. Ces données nous ont permis de déterminer quelles stratégies d'élagage offrent le meilleur rapport coût-efficacité. Nous avons également exploré différentes méthodes d'optimisation en terme d'accélération d'entraînement et de performance, ainsi que des combinaisons des méthodes, afin d'évaluer lesquelles se rapprochent le plus des performances du modèle baseline et sont moins coûteux. Dans le dernier chapitre de cette étude, nous approfondirons ces questions et examinerons les facteurs potentiels sous-jacents, pour pouvoir conclure sur l'application frugale mais potentiellement optimale du modèle CamemBERT dans des tâches variées.

CHAPITRE 5

DISCUSSION

Sommaire

5.1 Synthèse et implications des résultats

Jusqu'à présent, nous avons évalué plusieurs stratégies d'élagage appliquées au modèle CamemBERT, en nous appuyant partiellement sur les travaux de SAJJAD et al. (2023). Notre étude se distingue par son application à des tâches plus variées en français, notamment le POS tagging, la NER et l'analyse des sentiments. De plus, nous avons investigué s'il existe une corrélation entre les couches du modèle et les tâches en aval, une approche qui n'a pas été pleinement explorée dans leurs travaux. Nous avons également étudié l'effet de l'application de LoRA après élagage et la prolongation du temps d'entraînement, avec des analyses initiales de certains phénomènes observés. Dans ce chapitre, nous allons approfondir l'analyse de ces résultats, discuter des causes sous-jacentes et évaluer la valeur pratique de ces stratégies.

Reconnaissance des entités nommées (NER)

Pour les tâches de NER, les résultats obtenus varient en fonction des deux corpus utilisés. Le modèle entraîné sur WikiNER a nettement surpassé celui basé sur le corpus FTB_NER, dès la phase de l'entraînement de baseline. Le corpus WikiNER, étant annoté de manière semi-automatique, pourrait contenir des annotations plus

prévisibles, ce qui expliquerait ses performances plus élevées. En plus, les résultats obtenus avec différentes stratégies d'élagage sur FTB_NER montrent des fluctuations plus importantes, alors que sur WikiNER, la courbe de performance est plus stable. Cela suggère que les annotations semi-automatiques de WikiNER pourraient également atténuer les effets des variations induites par l'élagage, en rendant les prédictions plus faciles à généraliser.

En ce qui concerne les stratégies d'élagage Top/Bottom, nos expériences sur les performances des couches individuelles montrent que, pour des tâches comme la NER, il existe une corrélation inverse entre la performance et la position des couches dans le modèle. En d'autres termes, les couches basses (couches 0-6) tendent à produire de meilleurs résultats que les couches plus hautes (couches 7-11). De plus, l'élagage symétrique, qui consiste à supprimer les couches centrales tout en conservant les couches périphériques, s'est avéré plus efficace que la suppression des couches aux extrémités. Cela suggère que, pour les tâches de NER, les couches aux extrémités du modèle contiennent des informations cruciales pour la performance du modèle.

En conclusion, la stratégie d'élagage la plus performante pour NER s'avère être l'élagage Best_N, qui conserve les meilleures couches selon les performances individuelles. Les autres stratégies d'élagage, bien que toujours valables, montrent des résultats inférieurs en comparaison.

Part-of-Speech (POS) Tagging

Pour la tâche de POS tagging, nos expériences montrent que le modèle baseline offre déjà d'excellentes performances. Cela démontre l'intérêt d'appliquer des techniques comme l'élagage pour améliorer les performances tout en allégeant et en accélérant le modèle. Les résultats pour notre tâche de POS tagging présentent plusieurs similarités avec ceux observés pour la NER. Tout d'abord, les couches basses surperforment systématiquement les couches hautes, que ce soit en termes de performances individuelles ou de combinaisons de couches. Les modèles qui conservent les couches basses obtiennent des performances très proches du modèle baseline.

En outre, des stratégies d'élagage, telles que l'élagage symétrique ou l'élagage alterné, parviennent à maintenir des niveaux de performance remarquables même si elles sont plus agressives. Par exemple, dans certaines configurations, un modèle ne conservant que trois couches a montré des performances presque identiques à celles

du modèle baseline. Pour la tâche de POS tagging, le nombre de couches conservées n'a pas un impact aussi significatif sur les performances que pour d'autres tâches.

La meilleure stratégie pour la tâche de POS tagging demeure l'élagage basé sur la conservation des meilleures ou des pires couches (Best_N/Worst_N). En ne conservant que la moitié des couches, les performances restent comparables à celles du modèle baseline. Lorsque les pires couches (1 à 4 couches les moins performantes) sont supprimées, les résultats obtenus sont toujours très proches ou même dépassent ceux du modèle baseline. Cela suggère que la tâche de POS tagging est moins sensible à la quantité de couches conservées et que les couches les plus faibles peuvent être éliminées sans perte de performance.

Analyse sémantique

Pour les tâches d'analyse sémantique, telles que l'analyse séquentielle des sentiments , nos résultats montrent également de bonnes performances avec le modèle de base, établissant ainsi une base solide pour les comparaisons ultérieures. Dans nos expériences d'élagage, nous avons observé certaines similitudes avec les résultats obtenus pour les tâches de NER et de POS tagging. Par exemple, l'importance des couches basses se manifeste également dans notre tâche sémantique. Bien que cela soit moins évident dans les expériences sur les couches individuelles, les comparaisons entre l'élagage des couches basses et des couches hautes montrent que la conservation des couches basses offre de meilleures performances. Cela peut s'expliquer par le fait que les couches supérieures se construisent généralement à partir des informations capturées par les couches inférieures.

De plus, pour l'élagage symétrique, la conservation des couches périphériques s'est avérée être une stratégie efficace pour notre tâche. Cela suggère que, quelle que soit la tâche, les couches aux extrémités du modèle jouent toujours un rôle clé dans la performance globale.

Application de LoRA et prolongation du temps d'entraînement

Nous avons analysé dans le chapitre précédent que, pour notre tâche sémantique, l'application individuelle de LoRA et de l'élagage produit de bons résultats, mais leur combinaison n'a pas permis d'améliorer les performances, voire a entraîné une diminution des résultats. Cela est sans doute dû au fait que l'interférence mutuelle

entre LoRA et l'élagage empêche chacun de ces deux mécanismes de donner leur plein potentiel. L'élagage, en supprimant des capacités de traitement des caractéristiques globales, rend difficile l'application de LoRA, ce qui conduit à une baisse des performances globales.

Pour des tâches comme la NER et le POS tagging, qui sont davantage axées sur le traitement d'informations partielles, LoRA devrait normalement pouvoir ajuster efficacement les paramètres locaux pour améliorer la capture des caractéristiques spécifiques. Cependant, les résultats n'ont pas été à la hauteur des attentes. Nous supposons que cela pourrait être dû à une sensibilité excessive de LoRA aux caractéristiques locales, entraînant une dégradation des performances après ajustement.

Étant donné que notre objectif est de proposer des méthodes efficaces de compression et d'accélération des modèles, l'élagage a déjà prouvé son efficacité. Or, la combinaison avec LoRA n'a pas permis d'améliorer les performances. Par conséquent, nous estimons qu'il serait préférable d'explorer d'autres configurations de LoRA, telles que des réglages de paramètres plus modérés, pour voir si une meilleure synergie entre l'élagage et LoRA peut être atteinte. Si ces ajustements ne s'avèrent pas efficaces, il faudrait envisager d'autres alternatives, telles que l'augmentation du temps d'entraînement, pour compenser la perte de performance induite par l'élagage.

Nous avons testé si l'augmentation du temps d'entraînement des modèles élagués pouvait compenser la perte de performance. Les résultats expérimentaux montrent que, pour la tâche de POS tagging, une augmentation modérée du temps d'entraînement peut effectivement améliorer les performances, et dans certains cas, dépasser les résultats du modèle de référence. Tous nos trois modèles de POS élagués ont atteint un F1-score supérieur à celui du modèle baseline en augmentant le nombre d'epochs d'entraînement. Cependant, cette stratégie n'a pas montré d'améliorations significatives pour la tâche sémantique sur *Allociné*, bien que le nombre d'epochs ait été augmenté de 12, les gains de performance sont restés limités. Cela suggère que l'augmentation du temps d'entraînement ne suffit pas à compenser totalement les pertes de performance dues à l'élagage, en particulier pour les tâches nécessitant une compréhension sémantique globale plus complexe.

5.2 Analyses des observations

En résumant ces observations, nous pouvons dégager des tendances communes à travers les différentes tâches. Tout d'abord, dans presque tous les cas, les couches basses du modèle semblent être les plus importantes pour maintenir des performances élevées. Cela peut être mieux compris en explorant les fondements des modèles Transformer, ainsi que les relations complexes entre la structure des modèles et les types de tâches traitées. Selon VASWANI et al. (2017), chaque couche du modèle Transformer joue un rôle spécifique dans la transformation et l'apprentissage des représentations internes des séquences d'entrée, parmi lesquelles les couches basses sont souvent responsables de capturer des caractéristiques syntaxiques, telles que les relations locales entre les mots (y compris la grammaire et les relations de dépendance), tandis que les couches supérieures sont plus axées sur des relations sémantiques globales, intégrant des informations contextuelles plus larges.

Cela explique en partie pourquoi, notamment dans des tâches comme la NER et le POS tagging, les couches basses du modèle sont cruciales. Ces tâches incluent généralement des processus de nature plus syntaxique. Ce sont souvent les couches 0-6 qui capturent ces informations nécessaires, sans lesquelles des performances dégradent.

Cependant, il est également important de noter les différences observées entre les types de tâches. Comme illustré dans la figure 4.1, la performance des couches basses est particulièrement marquée dans les tâches de la NER et du POS tagging, tandis que dans les tâches sémantiques, les performances sont plus homogènes à travers les différentes couches. Cela montre que les tâches sémantiques dépendent davantage des informations globales du texte, ce qui fait que chaque couche contribue à une part du processus de modélisation de l'information globale.

Nos résultats montrent aussi que la sensibilité d'un modèle à l'élagage dépend du type de tâche et des couches impliquées. Premièrement, pour les tâches bases, comme le POS tagging, les couches basses sont d'une importance capitale. Par ailleurs, le fait de ne conserver qu'un très petit nombre de couches basses permet d'obtenir de bonnes performances, ce qui est largement déterminé par la nature de la tâche elle-même. Le POS tagging est une tâche qui repose principalement sur des dépendances locales et c'est une tâche relativement simple et bien standardisée. Dans les modèles Transformer, les couches basses, qui capturent les caractéristiques lexicales et les relations

syntaxiques locales, suffisent déjà à accomplir efficacement une tâche comme le POS tagging. Par conséquent, la réduction des couches supérieures n'affecte que marginalement les performances pour ce type de tâche.

En revanche, pour les tâches sémantiques, les couches moyennes et hautes semblent jouer un rôle plus crucial et indispensable, comme en témoigne la stratégie Best_N, qui tend à conserver ces couches au lieu des couches inférieures. La raison est que pour ce genre de tâche, le modèle doit être capable de saisir les nuances du texte entier, telles que l'opinion exprimée ou la polarité des sentiments, ce qui dépasse déjà les fonctions des couches inférieures et nécessite une vue d'ensemble de la séquence de texte que les couches supérieures du modèle sont chargées de capturer. Cependant, il est important de noter que, dans nos expériences, l'importance des couches basses pour la tâche sémantique est également évidente. Ce phénomène peut s'expliquer par la nature progressive de la construction des représentations dans un modèle Transformer. À chaque couche, l'information est transmise et enrichie, chaque niveau utilisant l'output de la couche précédente pour ajouter des informations contextuelles. Les couches supérieures s'appuient donc sur les informations syntaxiques et lexicales capturées par les couches inférieures pour effectuer un raisonnement sémantique global plus raffiné. Ainsi, bien que les couches supérieures jouent un rôle crucial dans la compréhension sémantique, les couches inférieures restent indispensables.

Dans l'application pratique, nous pouvons tirer les conclusions suivantes : la stratégie optimale de l'élagage dépend des spécificités de chaque tâche, comme cela a été discuté en détail précédemment. En plus, nous pensons que la stratégie Best_N, qui consiste à conserver les couches les plus performantes, s'avère souvent être la stratégie la plus appropriée pour différents types de tâches. Cela s'explique par le fait que, dans nos expériences, cette stratégie montre une certaine corrélation avec les approches Top_N et Bottom_N, selon les caractéristiques des tâches spécifiques. Ainsi, les stratégies optimales identifiées dans nos expériences peuvent servir de guide pour l'élagage des modèles dans des applications pratiques. De plus, en fonction de la sensibilité du modèle à l'élagage, il est essentiel d'adapter l'intensité de la réduction des couches en fonction de la complexité de la tâche et de la qualité du corpus. Pour les tâches simples, comme le POS, il est possible de procéder à un élagage plus agressif sans compromettre les performances. En revanche, pour les tâches plus complexes, comme l'analyse des sentiments, il est crucial de prendre en compte l'importance des

couches inférieures et supérieures afin de maintenir des performances comparables à celles du modèle baseline.

Tout cela confirme notre hypothèse selon laquelle, même en acceptant une légère perte de performance, la plupart des modèles peuvent être élagués pour réduire leur taille. Pour certaines tâches de nature plus locale, comme le POS, les stratégies d'élagage permettent de réduire de manière significative la taille du modèle, accélérant ainsi à la fois l'entraînement et l'inférence sans affecter considérablement les performances. De plus, pour ces tâches locales, il est possible d'améliorer encore les performances en augmentant le temps d'entraînement. Toutefois, pour les tâches sémantiques plus complexes, l'augmentation du temps d'entraînement semble moins efficace pour compenser les pertes de performance liées à l'élagage.

5.3 Limitations et pistes de recherche future

Dans cette recherfche, nous avons aussi observé plusieurs limitations qui sont autant de pistes pour des recherches futures. La première limitation de notre recherche concerne le modèle. Les résultats de cette étude sont uniquement basés sur le modèle CamemBERT et n'ont pas été validés sur d'autres modèles préentraînés pour le français. Par conséquent, il serait nécessaire que de futures recherches valident les méthodes proposées dans cette étude sur un éventail plus large de modèles préentraînés afin d'évaluer leur universalité.

Aussi, il manque de la comparaison avec d'autres techniques de compression de modèles dans notre recherche. En effet, cette étude se concentre principalement sur l'élagage structuré et les techniques PEFT (telles que LoRA), sans toutefois comparer ces approches à d'autres techniques courantes de compression de modèles, telles que la quantification. Cependant, chacune de ces méthodes pourrait présenter des avantages dans des contextes spécifiques. Il serait donc utile, dans de futures recherches, de comparer notre approche à ces méthodes afin de mieux comprendre les forces et les faiblesses des différentes techniques de compression de modèles.

Enfin, la dernière limitation de la recherche est liée à l'échelle des expériences et aux données. En raison des contraintes matérielles, nos expériences sémantiques ont été menées sur un jeux de donnée de taille modérée et les résultats sont basés sur des exécutions uniques de chaque expérimentation. L'absence de répétitions multiples

pour obtenir une moyenne des résultats peut introduire une certaine variabilité et fragilité dans les conclusions, notamment en ce qui concerne les différentes stratégies d'élagage appliquées à diverses tâches. Pour renforcer la fiabilité de nos conclusions, il serait souhaitable que des expériences futures soient menées sur des ensembles de données plus larges et avec des répétitions multiples, afin de générer des résultats plus représentatifs et stables.

5.4 Conclusion

Dans cette étude, nous avons exploré différentes stratégies d'élagage pour le modèle CamemBERT. Nous avons évalué les performances de ces stratégies sur trois tâches NLP différentes : la NER, le POS tagging et l'analyse des sentiments. Nous avons également étudié l'effet de l'application de la méthode de LoRA et de la prolongation du temps d'entraînement sur les performances des modèles élagués.

En reproduisant et en étendant les travaux précédents sur des corpus et modèle en anglais, cette étude a permis de confirmer l'existence de la redondance des couches dans les modèles de langage préentraînés, ainsi que l'efficacité de l'élagage structuré comme méthode de compression et d'accélération des modèles préentraînés, tout en conservant des performances compétitives, notamment pour des tâches comme le POS. Nos résultats montrent que, pour les tâches comme le POS, qui nécessitent un traitement syntaxique plus direct, l'élagage des couches supérieures peut réduire considérablement la taille du modèle et accélérer l'entraînement, sans affecter significativement la performance. Aussi, les couches inférieures sont importantes pour la plupart des tâches. De plus, nous avons constaté que la stratégie Best_N, qui consiste à conserver les couches les plus performantes, s'est révélée efficace dans différents contextes de tâches. L'étude souligne également l'importance d'adapter la stratégie d'élagage en fonction de la complexité de la tâche et de la qualité du corpus utilisé. Si des tâches simples comme le POS tagging permettent un élagage plus agressif, les tâches plus complexes nécessitent une approche plus prudente pour maintenir des performances. Enfin, l'augmentation du temps d'entraînement peut être une stratégie efficace pour compenser les pertes de performance liées à l'élagage, en particulier pour les tâches de faible niveau, mais l'intervention de LoRA n'a pas permis d'améliorer les performances en combinant avec l'élagage.

BIBLIOGRAPHIE

HANSON, S. J., & PRATT, L. Y. (1988). Comparing Biases for Minimal Network Construction with Back-Propagation, 177-185.

LECUN, Y., DENKER, J., & SOLLA, S. (1989). Optimal Brain Damage. In D. TOURETZKY (Éd.), *Advances in Neural Information Processing Systems* (T. 2). Morgan-Kaufmann.

HASSIBI, B., STORK, D., & WOLFF, G. (1993). Optimal Brain Surgeon : Extensions and performance comparisons. In J. COWAN, G. TESAURO & J. ALSPECTOR (Éd.), *Advances in Neural Information Processing Systems* (T. 6). Morgan-Kaufmann.

ABEILLÉ, A., CLÉMENT, L., & TOUSSENEL, F. (2003). Building a Treebank for French. In A. ABEILLÉ (Éd.), *Treebanks : Building and Using Parsed Corpora* (p. 165-187). Springer Netherlands. https://doi.org/10.1007/978-94-010-0201-1_10

SANG, E. F. T. K., & DE MEULDER, F. (2003). Introduction to the CoNLL-2003 Shared Task : Language-Independent Named Entity Recognition. https://doi.org/10.48550/arXiv.cs/0306050

CRABBÉ, B., & CANDITO, M. (2008). Expériences d'analyse Syntaxique Statistique Du Français. *15ème Conférence Sur Le Traitement Automatique Des Langues Naturelles - TALN'08*, pp. 44-54.

KRIZHEVSKY, A., SUTSKEVER, I., & HINTON, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*.

SAGOT, B., RICHARD, M., & STERN, R. (2012). Annotation Référentielle Du Corpus Arboré de Paris 7 En Entités Nommées. In G. ANTONIADIS, H. BLANCHON & G. SÉRASSET (Éd.), *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, Volume 2 : TALN* (p. 535-542). ATALA/AFCP.

NOTHMAN, J., RINGLAND, N., RADFORD, W., MURPHY, T., & CURRAN, J. R. (2013). Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, *194*, 151-175. https://doi.org/10.1016/j.artint.2012.03.006

HAN, S., POOL, J., TRAN, J., & DALLY, W. (2015). Learning both Weights and Connections for Efficient Neural Network. In C. CORTES, N. LAWRENCE, D. LEE, M. SUGIYAMA & R. GARNETT (Éd.), *Advances in Neural Information Processing Systems* (T. 28). Curran Associates, Inc.

HAN, S., MAO, H., & DALLY, W. J. (2016). Deep Compression : Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *International Conference on Learning Representations (ICLR) 2016*, (arXiv :1510.00149). https://doi.org/10.48550/arXiv.1510.00149

WEN, W., WU, C., WANG, Y., CHEN, Y., & LI, H. (2016). Learning Structured Sparsity in Deep Neural Networks. In D. LEE, M. SUGIYAMA, U. LUXBURG, I. GUYON & R. GARNETT (Éd.), *Advances in Neural Information Processing Systems* (T. 29). Curran Associates, Inc.

LI, H., KADAV, A., DURDANOVIC, I., SAMET, H., & GRAF, H. P. (2017). Pruning Filters for Efficient ConvNets. *International Conference on Learning Representations*. https://openreview.net/forum?id=rJqFGTslg

LUO, J.-H., & WU, J. (2017). An Entropy-based Pruning Method for CNN Compression. (arXiv :1706.05791). https://doi.org/10.48550/arXiv.1706.05791

REBUFFI, S.-A., BILEN, H., & VEDALDI, A. (2017). Learning multiple visual domains with residual adapters. In I. GUYON, U. V. LUXBURG, S. BENGIO, H. WALLACH, R. FERGUS, S. VISHWANATHAN & R. GARNETT (Éd.), *Advances in Neural Information Processing Systems* (T. 30). Curran Associates, Inc.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER,., & POLOSUKHIN, I. (2017). Attention is All you Need. In I. GUYON, U. V. LUXBURG, S. BENGIO, H. WALLACH, R. FERGUS, S. VISHWANATHAN & R. GARNETT (Éd.), *Advances in Neural Information Processing Systems* (T. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

ZHU, M., & GUPTA, S. (2017). To Prune, or Not to Prune : Exploring the Efficacy of Pruning for Model Compression. (arXiv :1710.01878). https://doi.org/10.48550/arXiv.1710.01878

JACOB, B., KLIGYS, S., CHEN, B., ZHU, M., TANG, M., HOWARD, A., ADAM, H., & KALENICHENKO, D. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

KRISHNAMOORTHI, R. (2018). Quantizing Deep Convolutional Networks for Efficient Inference : A Whitepaper. (arXiv :1806.08342). https://doi.org/10.48550/arXiv.1806.08342

REBUFFI, S.-A., VEDALDI, A., & BILEN, H. (2018). Efficient Parametrization of Multi-domain Deep Neural Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8119-8127. https://doi.org/10.1109/CVPR.2018.00847

FRANKLE, J., & CARBIN, M. (2019). The Lottery Ticket Hypothesis : Finding Sparse, Trainable Neural Networks. *International Conference on Learning Representations*. https://openreview.net/forum?id=rJl-b3RcF7

HOULSBY, N., GIURGIU, A., JASTRZEBSKI, S., MORRONE, B., DE LAROUSSILHE, Q., GESMUNDO, A., ATTARIYAN, M., & GELLY, S. (2019, septembre). Parameter-Efficient Transfer Learning for NLP. In K. CHAUDHURI & R. SALAKHUTDINOV (Éd.), *Proceedings of the 36th International Conference on Machine Learning* (p. 2790-2799, T. 97). PMLR. https://proceedings.mlr.press/v97/houlsby19a.html

MICHEL, P., LEVY, O., & NEUBIG, G. (2019). Are Sixteen Heads Really Better than One ? In H. WALLACH, H. LAROCHELLE, A. BEYGELZIMER, F. D'ALCHÉ-BUC, E. FOX & R. GARNETT (Éd.), *Advances in Neural Information Processing Systems* (T. 32). Curran Associates, Inc.

BLALOCK, D., GONZALEZ ORTIZ, J. J., FRANKLE, J., & GUTTAG, J. (2020). What is the State of Neural Network Pruning ? In I. DHILLON, D. PAPAILIOPOULOS & V. SZE (Éd.), *Proceedings of Machine Learning and Systems* (p. 129-146, T. 2).

MARTIN, L., MULLER, B., ORTIZ SUÁREZ, P. J., DUPONT, Y., ROMARY, L., de la CLERGERIE, É. V., SEDDAH, D., & SAGOT, B. (2020). CamemBERT : a Tasty French Language Model. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

GOU, J., YU, B., MAYBANK, S. J., & TAO, D. (2021). Knowledge Distillation : A Survey. *International Journal of Computer Vision*, *129*(6), 1789-1819. https://doi.org/10.1007/s11263-021-01453-z

HU, E. J., SHEN, Y., WALLIS, P., ALLEN-ZHU, Z., LI, Y., WANG, S., WANG, L., & CHEN, W. (2021). LoRA : Low-Rank Adaptation of Large Language Models.

LESTER, B., AL-RFOU, R., & CONSTANT, N. (2021, novembre). The Power of Scale for Parameter-Efficient Prompt Tuning. In M.-F. MOENS, X. HUANG, L. SPECIA & S. W.-t. YIH (Éd.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (p. 3045-3059). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.243

LIU, X., JI, K., FU, Y., TAM, W., DU, Z., YANG, Z., & TANG, J. (2022, mai). P-Tuning : Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks. In S. MURESAN, P. NAKOV & A. VILLAVICENCIO (Éd.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)* (p. 61-68). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-short.8

CHENG, H., ZHANG, M., & SHI, J. Q. (2023). A Survey on Deep Neural Network Pruning-Taxonomy, Comparison, Analysis, and Recommendations. (arXiv :2308.06767). https://doi.org/10.48550/arXiv.2308.06767

de VRIES, A. (2023). The growing energy footprint of artificial intelligence. *Joule*, *7*(10), 2191-2194. https://doi.org/10.1016/j.joule.2023.09.004

JIANG, A. Q., SABLAYROLLES, A., MENSCH, A., BAMFORD, C., CHAPLOT, D. S., de las CASAS, D., BRESSAND, F., LENGYEL, G., LAMPLE, G., SAULNIER, L., LAVAUD, L. R., LACHAUX, M.-A., STOCK, P., SCAO, T. L., LAVRIL, T., WANG, T., LACROIX, T., & SAYED, W. E. (2023). Mistral 7B. https://arxiv.org/abs/2310.06825

LIALIN, V., DESHPANDE, V., & RUMSHISKY, A. (2023). Scaling Down to Scale Up : A Guide to Parameter-Efficient Fine-Tuning. https://doi.org/10.48550/arXiv.2303.15647

LIU, X., ZHENG, Y., DU, Z., DING, M., QIAN, Y., YANG, Z., & TANG, J. (2023). GPT understands, too. *AI Open*. https://doi.org/https://doi.org/10.1016/j.aiopen.2023.08.012

MA, X., FANG, G., & WANG, X. (2023). LLM-Pruner : On the Structural Pruning of Large Language Models. In A. OH, T. NAUMANN, A. GLOBERSON, K. SAENKO, M. HARDT & S. LEVINE (Éd.), *Advances in Neural Information Processing Systems* (p. 21702-21720, T. 36). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2023/file/44956951349095f74492a5471128a7e0-Paper-Conference.pdf

SAJJAD, H., DALVI, F., DURRANI, N., & NAKOV, P. (2023). On the Effect of Dropping Layers of Pre-Trained Transformer Models. *Computer Speech & Language*, *77*, 101429. https://doi.org/10.1016/j.csl.2022.101429

SANTACROCE, M., WEN, Z., SHEN, Y., & LI, Y. (2023). What Matters In The Structured Pruning of Generative Language Models ? https://doi.org/10.48550/arXiv.2302.03773

ZHU, X., LI, J., LIU, Y., MA, C., & WANG, W. (2023). A Survey on Model Compression for Large Language Models. (arXiv :2308.07633). https://doi.org/10.48550/arXiv.2308.07633

AMISSE, M., FAUR, M., GONARD, L., & ORCESI, A. (2024, mars). *Promouvoir des modèles d'intelligence artificielle frugale pour et par les politiques publiques* (rapp. tech.). Ecole des Ponts Paris Tech, Paris-France. https://hal.science/hal-04510171

HE, S., SUN, G., SHEN, Z., & LI, A. (2024). What Matters in Transformers ? Not All Attention Is Needed. https://doi.org/10.48550/arXiv.2406.15786

KRICHEN, M. (2024, février). *Stratégies de Compression et d'Optimisation des Modèles d'Intelligence Artificielle* [working paper or preprint]. https://hal.science/hal-04446898

PILLET, X., VOLKOVA, A., GREFFARD, N., & DUFOUR, R. (2024). Entre performance et frugalité en TAL : Approches pour la réduction de la taille des (L)LMs.