Actes de CORIA-TALN-RJCRI-RECITAL 2025
Marseille, France, 30 juin au 4 juillet 2025

Projeter pour mieux fusionner : une histoire de bandit et de lit

Olivier Ferret
Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
olivier.ferret@cea.fr

RÉSUMÉ

La mise à disposition d'un nombre important de modèles de langue neuronaux affinés pour différentes tâches conduit assez naturellement à se poser la question de l'intérêt de les combiner, en particulier par le biais de la fusion de paramètres, option aboutissant au résultat demandant le moins de ressources. Parmi les nombreuses méthodes existantes, un certain nombre se focalisent sur l'alignement des paramètres en amont de la fusion proprement dite. Dans cet article, nous proposons une nouvelle méthode entrant dans ce champ de recherche, fondé sur l'analyse procustéenne. Nous évaluons cette méthode pour la fusion de modèles affinés pour une même tâche à partir d'un même modèle de base, de type encodeur. En considérant neuf tâches du jeu de données GLUE et six méthodes de fusion de référence, nous montrons que notre proposition est capable d'améliorer les méthodes de fusion existantes dans la plupart des configurations testées.

ABSTRACT

Projecting to better merge : a story of a bandit and a bed.

The availability of a large number of fine-tuned neural language models for different tasks naturally leads to the question of whether it is worthwhile combining them, in particular through parameter fusion, the option that leads to the least resource-intensive result. Among the many existing methods, a number of them focus on parameter alignment, before the actual merging. In this article, we propose a new method in this field of research, based on Procrustes analysis. We evaluate this method for the fusion of fine-tuned models for the same task from the same backbone encoder model. Considering nine tasks from the GLUE dataset and six reference fusion methods, we show that our proposal is capable of improving on existing fusion methods in most of the configurations tested.

MOTS-CLÉS : Modèles de langue neuronaux, fusion de modèles, alignement d'espaces de représentation.

KEYWORDS: Neural language models, model merging, representation space alignment.

ARTICLE : **Contribution originelle.**

1 Introduction

Les modèles de langue neuronaux, en particulier à base de transformeurs (Vaswani *et al.*, 2017), occupent un rôle central dans le traitement automatique des langues actuel. Néanmoins, leur entraînement initial et leur adaptation à un contexte applicatif particulier sont des opérations coûteuses en termes de ressources de calcul. Parallèlement, un nombre de plus en plus important de ces modèles

Actes des 32ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN), volume 1 : articles scientifiques originaux, pages 629–641.

sont mis à disposition, en particulier, sur le dépôt de modèles de la société Hugging Face [1]. Ces modèles regroupent tout à la fois des modèles bruts et des modèles affinés pour certaines tâches. Face à un besoin applicatif spécifique, il est tentant d'évaluer si ce besoin ne peut pas être couvert par un modèle existant, voire par une combinaison de modèles existants. La forme la plus directe d'une telle combinaison est la fusion des modèles au niveau de leurs paramètres. Cette fusion peut porter sur des modèles très comparables afin d'obtenir un modèle plus robuste, à la manière des méthodes d'ensemble, mais elle peut aussi chercher à associer des modèles complémentaires, en particulier sur le plan de leur tâche, favorisant l'élargissement fonctionnel du modèle résultat, capable alors de réaliser plusieurs tâches.

Ainsi que l'illustrent (Li *et al.*, 2023) et (Yang *et al.*, 2024), ce domaine de recherche est particulièrement actif, avec des applications intéressantes notamment pour les modèles décodeurs comme en atteste le succès de la bibliothèque MergeKit (Goddard *et al.*, 2024). Yang *et al.* (2024) répartissent le nombre important de méthodes existantes en deux grandes catégories : les méthodes intervenant en amont de la fusion et les méthodes de fusion proprement dites. Dans cet article, nous nous intéressons plus particulièrement aux méthodes du premier type mais le lecteur pourra se reporter à la section 3.1 pour la présentation d'un certain nombre de méthodes de référence du second type. Au sein du premier type de méthodes, Yang *et al.* (2024) font le distinguo entre les méthodes d'affinage optimisant la possibilité de fusionner les modèles en leur permettant de cohabiter dans un même espace de représentation, les méthodes prenant en compte les différences d'architecture entre modèles et finalement, les méthodes d'alignement de paramètres. C'est sur ces dernières que nous concentrons notre attention. Dans ce champ, un certain nombre d'entre elles (Singh & Jaggi, 2020; Imfeld *et al.*, 2024) se sont appuyées sur la notion de transport optimal pour réaliser l'alignement des paramètres de plusieurs modèles. CCAMerge (Horoi *et al.*, 2024) utilise pour sa part l'analyse canonique des corrélations pour ce faire, plus précisément en maximisant les corrélations entre les combinaisons linéaires des paramètres des modèles. Enfin, Ainsworth *et al.* (2023) proposent trois algorithmes de permutation des paramètres d'un modèle pour les aligner avec les paramètres d'un autre modèle : l'un s'appuie sur l'appariement des activations des modèles, le deuxième sur l'appariement de leurs poids et le dernier sur l'apprentissage d'un alignement.

En nous situant dans ce contexte des méthodes d'alignement de paramètres entre modèles, nous mettons l'accent sur deux contributions principales :

— nous proposons une nouvelle méthode permettant de projeter de façon simple dans un même espace les paramètres de différents modèles de langue neuronaux à fusionner ;
— nous appliquons cette méthode à différentes méthodes de fusion de l'état de l'art et les évaluons en nous focalisant sur la fusion de modèles affinés à partir d'un même modèle de base et pour une même tâche, configuration n'ayant pas toujours fait l'objet d'évaluations aussi systématiques.

2 Méthode

Le problème que nous considérons dans cet article est celui de la fusion de n modèles de langue neuronaux $\{LM_1 \dots LM_n\}$ résultant de l'affinage d'un même modèle préentraîné initial LM pour une tâche donnée T. Ces modèles se différencient en pratique par la fixation d'une graine aléatoire différente pour l'affinage. L'objectif poursuivi est la fusion de ces n modèles afin de produire un seul

1. https://huggingface.co/models

modèle applicable à la tâche T. Il s'agit donc d'une configuration comparable à (Wortsman *et al.*, 2022), avec la différence que nous ne faisons pas appel à un ensemble de validation pour la sélection gloutonne d'un sous-ensemble des n modèles.

Au-delà de l'application d'un certain nombre de méthodes de fusion de modèles de référence à cette configuration dont nous rendons compte à la section 3, nous proposons également une nouvelle méthode permettant, en amont de cette fusion, de réaliser la projection des modèles à fusionner dans un même espace de représentation. Plus précisément, un modèle de langue LM_i est défini ici par une liste de matrices (m_{ik}) correspondant aux différents éléments constitutifs de l'architecture transformeur : matrices constitutives des blocs attentionnels (clés, valeurs et requêtes), matrices des réseaux à action directe (*feed forward network*) suivant les blocs attentionnels mais également les matrices associés aux plongements des jetons textuels (*tokens*). Notre objectif est dans ce cadre de pouvoir projeter, pour un k fixé, toutes les matrices (m_{ik}) dans un même espace de représentation afin d'améliorer toute méthode de fusion appliquée aux modèles LM_i, à l'instar des méthodes d'alignement de poids des modèles neuronaux évoquées à la section 1.

Pour réaliser cette projection, nous proposons de nous appuyer sur l'analyse procustéenne (Gower, 2010). Ce type d'analyse a déjà été appliquée pour aligner des espaces de représentation issus de modèles de langue neuronaux statiques, en particulier dans un contexte d'alignement multilingue (Kementchedjhieva *et al.*, 2018; Taitelbaum *et al.*, 2019), mais aussi dans le cas de plusieurs modèles comparables (Caciularu *et al.*, 2021). En revanche, nous n'avons pas connaissance d'une application à des modèles de langue à base de transformeurs.

De façon générale, l'analyse procustéenne vise à rapprocher une forme d'une autre en faisant appel à trois types de transformations linéaires : translations, mises à l'échelle uniformes et rotations. En adoptant un formalisme matriciel et en supposant que B est la matrice de référence et A, la matrice à rapprocher de B, on cherche donc la transformation T telle que :

$$\min_{T} ||\mathbf{AT} - \mathbf{B}|| \tag{1}$$

Dans le cas qui nous occupe, nous limitons T au champ des rotations, ce qui correspond à une analyse procustéenne orthogonale dans laquelle la matrice de transformation T est orthogonale, configuration la plus répandue pour ce type d'analyse [2]. Schönemann (1966) propose de trouver T en réalisant une décomposition en valeurs singulières de la matrice $M = A^{\mathsf{T}}B = U\Sigma V^{\mathsf{T}}$ et en définissant la transformation T comme $T = UV^{\mathsf{T}}$.

L'analyse procustéenne présentée ci-dessus s'applique au cas de deux matrices, l'une servant de référence à l'autre. Dans notre travail, nous souhaitons ne pas nous limiter à la fusion de deux modèles seulement. Nous nous sommes donc appuyés sur l'analyse procustéenne généralisée (Gower, 1975). Cette extension de l'analyse procustéenne à plus de deux matrices se fait selon une procédure itérative définie par l'algorithme 1 dans lequel à chaque itération, une matrice de référence est calculée à partir de l'ensemble des matrices considérées et chacune de ces matrices est ensuite alignée avec cette matrice de référence grâce l'analyse procustéenne orthogonale de deux matrices. La procédure est poursuivie jusqu'à ce que la différence de distance d'alignement d'une itération à l'autre soit suffisamment petite ou qu'un nombre maximal d'itérations soit atteint. En pratique, dans les expérimentations de la section 3, la convergence se fait le plus souvent en une seule itération.

2. Il est à noter que le produit scalaire, opération particulièrement présente dans l'architecture transformeur, est invariant vis à vis des rotations et des translations mais pas des mises à l'échelle, ce qui est un argument pour laisser de côté ce type de transformations. Par ailleurs, le cas des translations peut être pris en compte en amont en ramenant les vecteurs à l'origine.

Algorithm 1 Analyse procustéenne généralisée

Entrée : M_e, ensemble de n matrices
Soit M_p, l'ensemble des n matrices initiales projetées
Soit m_{ref}, la matrice de référence pour l'analyse procustéenne orthogonale (APO)
Soit apo_dist, la distance entre la matrice de référence et la moyenne des matrices après APO
Soit max_iter, le nombre maximal d'itérations et tol, le critère de convergence sur apo_dist
$m_{ref} \leftarrow$ choix aléatoire parmi M_e ; l : indice de m_{ref} parmi les M_e
for $i = 1, \ldots, n$, avec $i \neq l$ **do**
    $M_p[i] \leftarrow$ APO(m_{ref}, $M_e[i]$)
end for
$m_{ref} \leftarrow$ moyenne(M_p) ; $apo\_dist \leftarrow -\infty$
for $k = 1, \ldots, max\_iter$ **do**
    for $i = 1, \ldots, n$ **do**
        $M_p[i] \leftarrow$ APO(m_{ref}, $M_p[i]$)
    end for
    $m_{moy} \leftarrow$ moyenne(M_p)
    $dist\_nf \leftarrow$ norme_frobenius(m_{ref}, m_{moy})
    if $apo\_dist \neq -\infty$ **and** $|apo\_dist - dist\_nf| < tol$ **then**
        break
    end if
    $m_{ref} \leftarrow m_{moy}$; $apo\_dist \leftarrow dist\_nf$
end for
Sortie : M_p

3 Expérimentations et évaluation

3.1 Cadre d'évaluation

La méthode de projection de différents modèles dans un même espace de représentation que nous avons décrite à la section 2 peut s'appliquer à tout type de modèles de langue neuronaux et peut être appliquée en amont d'un grand nombre de méthodes de fusion de ces modèles. Nous nous sommes plus particulièrement focalisés sur les modèles de type encodeur dans une configuration mono-tâche. La fusion s'opère donc entre plusieurs modèles résultant de l'affinage du même modèle racine pour une même tâche. Nos expérimentations ont été réalisées avec le modèle BERT (Devlin *et al.*, 2019), dans sa version `bert-base-uncased`. À la suite de beaucoup de travaux sur la fusion des modèles de langue, nous avons retenu les neufs tâches suivantes du jeu d'évaluation GLUE (Wang *et al.*, 2018) :

CoLA (Warstadt *et al.*, 2019) tâche d'acceptabilité grammaticale d'une phrase. Métrique : coefficient de corrélation de Matthews ;

MNLI (Williams *et al.*, 2018) classification d'inférences entre phrases (neutre, implication, contradiction). Métrique : exactitude.

MRPC (Dolan & Brockett, 2005) tâche de détection de paraphrases. Métrique : moyenne de l'exactitude et de la f1-mesure ;

QNLI (Rajpurkar *et al.*, 2016) questions et réponses de SQuAD converties en tâche de détection d'inférence textuelle. Métrique : exactitude ;

QQP (Shankar *et al.*, 2017) détection de questions équivalentes. Métrique : moyenne de l'exactitude et de la f1-mesure ;

RTE (Dagan *et al.*, 2005; Haim *et al.*, 2006; Giampiccolo *et al.*, 2007; Bentivogli *et al.*, 2009) tâche de détection d'inférence textuelle. Métrique : exactitude ;

SST-2 (Socher *et al.*, 2013) tâche d'analyse de sentiments. Métrique : exactitude ;

STSB (Cer *et al.*, 2017) tâche de similarité de phrases. Métrique : moyenne des mesures de corrélation de Pearson et de Spearman ;

WNLI (Levesque *et al.*, 2012) tâches du Winograd Schema Challenge converties en tâches de détection d'inférence textuelle. Métrique : exactitude.

Pour chaque tâche, nous avons affiné 5 modèles avec une graine différente pour chaque modèle mais les mêmes hyperparamètres. Pour cet affinage et les hyperparamètres associés, nous avons repris le code et les recommandations fournies par Hugging Face [3]. Les valeurs spécifiques d'hyperparamètres que nous avons adoptées sont données en annexe (cf. section A.1). L'ensemble de test du jeu de données GLUE étant privé, nous avons suivi les pratiques des travaux existants en utilisant 10 % de l'ensemble d'entraînement comme ensemble de test.

Par ailleurs, nous avons considéré les méthodes de fusion suivantes, parmi les plus utilisées comme référence dans les travaux existants impliquant des modèles de type encodeur [4] :

Average (Wortsman *et al.*, 2022) Cette méthode consiste simplement à moyenner les paramètres jouant le même rôles entre les modèles à fusionner ;

Task arithmetic (Ilharco *et al.*, 2023) L'idée de cette méthode est de capturer la spécificité d'un modèle affiné pour une tâche en calculant la différence, pour des paramètres donnés, entre leur valeur dans le modèle affiné et dans le modèle initial préentraîné. On obtient ainsi des formes de vecteurs de tâche que l'on peut combiner par simple addition ;

Fisher (Matena & Raffel, 2022) Cette méthode réalise une fusion des paramètres en s'appuyant sur une estimation de l'importance de ces paramètres dans chacun des modèles par le biais de la matrice d'information de Fisher, estimée à partir de données d'affinage des modèles ;

RegMean (Jin *et al.*, 2023) Cette méthode s'appuie aussi sur des données d'affinage des modèles lui permettant dans ce cas de minimiser la différence entre les prédictions faites par le modèle résultant de la fusion et les modèles à fusionner. Cette optimisation se fait par le biais d'une régression linéaire locale ;

TIES (Yadav *et al.*, 2023) La méthode TIES s'appuie sur Task arithmetic pour réaliser la fusion proprement dite des modèles mais la fait précéder de deux étapes. La première procède à un élagage des paramètres en ne retenant que les 20 % des paramètres ayant l'influence la plus grande sur la performance des modèles. La seconde uniformise le signe des paramètres en s'appuyant sur une procédure tenant compte de l'influence des paramètres concernés ;

3. https://github.com/huggingface/transformers/blob/main/examples/pytorch/text-classification/run_glue.py

4. Les hyperparamètres de ces méthodes de fusion et leurs valeurs adoptées dans nos expérimentations sont donnés en annexe (cf. section A.2).

	cola	mnli	mrpc	qnli	qqp	rte	sst2	stsb	wnli	moy.
min.	56,0	83,6	84,7	90,8	88,8	63,9	92,0	88,8	45,1	77,1
max.	58,3	84,3	85,7	91,3	89,0	70,0	92,5	89,8	56,3	79,7
moy.	57,2	84,0	85,2	91,1	88,9	67,7	92,3	89,2	50,7	78,5
average	54,8	82,3	84,9	90,8	86,5	59,9	92,3	**88,9**	**56,3**	77,4
task arithmetic	59,4	80,7	88,8	90,6	87,6	68,2	92,7	87,8	**56,3**	79,1
regmean	56,8	**83,5**	86,5	**91,3**	**88,7**	69,0	92,1	88,8	**56,3**	79,2
fisher	50,7	79,6	86,2	**91,3**	84,8	60,6	92,4	85,9	**56,3**	76,4
ties	59,8	81,2	**89,0**	91,2	88,1	69,0	92,7	88,4	**56,3**	**79,5**
dare[average]	56,2	81,5	85,1	90,7	86,5	60,6	92,2	87,0	**56,3**	77,4
dare[task arith.]	**59,9**	80,7	**89,0**	90,6	87,6	68,2	92,7	87,8	**56,3**	79,2
dare[regmean]	56,2	83,3	87,2	91,1	**88,7**	**69,7**	92,1	88,7	43,7	77,9
dare[fisher]	53,0	80,5	87,3	91,1	85,5	61,4	92,4	87,7	43,7	75,8
dare[ties]	53,0	72,1	87,4	85,8	83,7	66,8	**92,9**	86,9	**56,3**	76,1

TABLE 1 – Résultats de l'évaluation de différentes méthodes de fusion de référence pour 5 modèles `bert-base-uncased` affinés pour une des 9 tâches du jeu de données GLUE. Les chiffres en gras correspondent à la valeur maximale obtenue par une méthode de fusion pour une tâche donnée. Valeurs x100.

DARE (Yu *et al.*, 2024) À l'instar de la méthode que nous proposons, DARE s'applique en amont de la fusion de modèles et peut donc s'associer à différentes méthodes de fusion. Elle consiste pour l'essentiel à annuler de façon aléatoire l'écart de valeur entre les paramètres d'un modèle affiné et de son modèle de base pour une certaine partie de ces paramètres et à effectuer une mise à l'échelle pour la partie restante.

Du point de vue de la fusion, les modèles de type encodeur posent le problème spécifique de la fusion de la tête de classification ou de régression des modèles [5]. Puisque nous fusionnons des modèles affinés pour une même tâche, nous avons choisi de fusionner les têtes de la même façon que le reste des modèles, avec deux exceptions. Les méthodes Task arithmetic et TIES définissent en effet des vecteurs de tâche par différence avec le modèle préentraîné initial. Or la tête de celui-ci est initialisée de façon aléatoire pour la tâche cible. Définir un vecteur de tâche pour cette tête n'a donc pas de sens. De ce fait, nous avons réalisé la fusion des têtes pour ces deux modèles en faisant appel à un simple moyennage des paramètres. Il est à noter par ailleurs que la méthode de projection présentée à la section 2 est appliquée aussi bien au corps qu'à la tête des modèles.

3.2 Résultats

3.2.1 Fusion sans la méthode proposée

Le tableau 1 donne les résultats de l'évaluation de nos différentes méthodes de fusion de référence pour les 9 tâches du jeu de données GLUE, la dernière colonne à droite fournissant la moyenne pour

5. Dans le cas des modèles de type décodeur ou décodeur-encodeur, la tâche cible est généralement réalisée de façon similaire à la tâche de préentraînement.

ces 9 tâches. Les trois premières lignes correspondent aux valeurs minimales, maximales et moyennes obtenues à l'échelle des 5 modèles considérés pour chaque tâche.

Étant donné que les fusions de modèles réalisées se font de façon non informée, c'est-à-dire sans utilisation d'un ensemble d'évaluation, nous considérons que la fusion de ces modèles s'avère intéressante dès lors que sa performance pour une tâche dépasse la performance du plus mauvais modèle pour cette tâche. Du point de vue statistique, la comparaison à la moyenne fait davantage sens mais dans un contexte applicatif où il faut choisir un modèle individuel sans information particulière de performance, toute solution meilleure que le pire cas est déjà intéressante.

Globalement, la dernière colonne du tableau 1 montre que la supériorité par rapport au plus mauvais modèle est observée pour une majorité de méthodes de fusion mais pas pour toutes. Parmi les méthodes de fusion proprement dites, seule Fisher se situe en moyenne un peu en dessous de *min*. L'association avec DARE dégrade un peu les performances de Fisher et plus franchement celle de TIES, qui est pourtant la meilleure méthode de fusion sans cette association.

Toujours en moyenne, aucune des méthodes de fusion ne dépasse la valeur maximale des modèles individuels mais TIES s'en approche fortement tandis que Task arithmetic, avec ou sans DARE, et RegMean tendent vers ce niveau dans une moindre mesure, toutes ces méthodes dépassant la valeur moyenne des modèles individuels. Enfin, il faut noter que la méthode DARE, qui est, à l'instar de la méthode que nous proposons, une méthode intervenant en amont de la fusion proprement dite et peut donc s'appliquer transversalement à différentes méthodes de fusion, présente un bilan global négatif : elle est neutre ou presque pour Average et Task arithmetic mais dégrade de façon sensible RegMean, Fisher et TIES. Cette observation est aussi en phase avec celle de Huang *et al.* (2024), qui suggère que DARE n'est probablement pas très adaptée à la fusion de plus de deux à trois modèles. Par ailleurs, les évaluations rapportées relatives à DARE concernent la fusion de modèles réalisant des tâches différentes et non la fusion de modèles réalisant la même tâche.

À l'échelle des tâches, aucune méthode de fusion ne permet de dépasser ou d'égaler systématiquement la performance de *min.* pour toutes les tâches, RegMean étant la méthode la plus proche d'atteindre cet objectif. Certaines méthodes de fusion permettent en revanche de dépasser nettement la performance maximale individuelle pour une tâche, comme l'association de Task arithmetic et de DARE pour CoLA ou les méthodes Task arithmetic, Fisher et RegMean, avec ou sans DARE, pour MRPC. Enfin, il faut noter le cas un peu particulier de la tâche WNLI, pour lesquelles toutes les méthodes de fusion en dehors de l'association de DARE et de Fisher ou RegMean égalent la performance maximale individuelle. La petite taille de ce jeu de données explique peut-être ce constat.

3.2.2 Fusion avec la méthode proposée

Le tableau 2 reprend les expérimentations du tableau 1 en appliquant préalablement aux méthodes de fusion utilisées la méthode de projection des modèles dans un même espace proposée à la section 2. À l'échelle globale (dernière colonne à droite), nous pouvons constater que l'impact de cette méthode de projection est positif. Seules les méthodes RegMean et l'association de Task arithmetic et de DARE voient leur performance globale diminuer du fait de cette projection. À l'inverse, cette performance augmente fortement pour Fisher, que ce soit avec ou sans DARE. Dans ce dernier cas, la projection permet même d'obtenir les meilleurs résultats de DARE. Elle permet par ailleurs d'égaler la performance maximale des modèles individuels avec Task arithmetic et fait que la méthode TIES avec DARE n'est plus que la seule méthode ne parvenant pas à dépasser le minimum des performances

	cola	mnli	mrpc	qnli	qqp	rte	sst2	stsb	wnli	moy.
min.	56,0	83,6	84,7	90,8	88,8	63,9	92,0	88,8	45,1	77,1
max.	58,3	84,3	85,7	91,3	89,0	70,0	92,5	89,8	56,3	79,7
moy.	57,2	84,0	85,2	91,1	88,9	67,7	92,3	89,2	50,7	78,5
average	$57{,}3_{2{,}5}$	$83{,}6_{1{,}3}$	$86{,}6_{1{,}7}$	$91{,}3_{0{,}5}$	$88{,}4_{1{,}9}$	$64{,}3_{4{,}3}$	$92{,}0_{-0{,}3}$	$88{,}8_{0{,}0}$	**56,3**$_{0{,}0}$	$78{,}7_{1{,}3}$
task arithmetic	**60,3**$_{0{,}9}$	$81{,}4_{0{,}7}$	$88{,}4_{-0{,}4}$	$89{,}8_{-0{,}8}$	$87{,}4_{-0{,}3}$	$71{,}5_{3{,}2}$	$92{,}9_{0{,}2}$	$88{,}9_{1{,}1}$	**56,3**$_{0{,}0}$	**79,7**$_{0{,}5}$
regmean	$57{,}0_{0{,}3}$	$83{,}5_{0{,}0}$	$87{,}0_{0{,}5}$	$91{,}3_{0{,}0}$	$88{,}5_{-0{,}2}$	$63{,}9_{-5{,}1}$	$91{,}6_{-0{,}5}$	$87{,}6_{-1{,}1}$	**56,3**$_{0{,}0}$	$78{,}5_{-0{,}7}$
fisher	$58{,}3_{7{,}6}$	**83,8**$_{4{,}1}$	$87{,}9_{1{,}7}$	$91{,}0_{-0{,}3}$	**88,6**$_{3{,}8}$	$66{,}1_{5{,}4}$	$92{,}3_{-0{,}1}$	**89,5**$_{3{,}6}$	**56,3**$_{0{,}0}$	$79{,}3_{2{,}9}$
ties	$59{,}1_{-0{,}7}$	$82{,}3_{1{,}1}$	$87{,}5_{-1{,}4}$	$90{,}4_{-0{,}8}$	$87{,}9_{-0{,}1}$	$71{,}5_{2{,}5}$	$92{,}8_{0{,}1}$	$88{,}5_{0{,}1}$	**56,3**$_{0{,}0}$	$79{,}6_{0{,}2}$
dare[average]	$57{,}5_{1{,}3}$	$83{,}7_{2{,}2}$	$86{,}8_{1{,}7}$	**91,4**$_{0{,}7}$	$88{,}5_{2{,}0}$	$64{,}3_{3{,}6}$	$92{,}0_{-0{,}2}$	$89{,}2_{2{,}1}$	**56,3**$_{0{,}0}$	$78{,}8_{1{,}5}$
dare[task arith.]	$45{,}4_{-14{,}4}$	$81{,}4_{0{,}7}$	**88,6**$_{-0{,}4}$	$89{,}7_{-0{,}9}$	$87{,}3_{-0{,}3}$	$71{,}1_{2{,}9}$	$92{,}9_{0{,}2}$	$88{,}9_{1{,}1}$	**56,3**$_{0{,}0}$	$78{,}0_{-1{,}2}$
dare[regmean]	$57{,}0_{0{,}8}$	$83{,}5_{0{,}2}$	$87{,}4_{0{,}3}$	**91,4**$_{0{,}2}$	$88{,}4_{-0{,}3}$	$64{,}3_{-5{,}4}$	$91{,}9_{-0{,}2}$	$87{,}7_{-1{,}1}$	**56,3**$_{12{,}7}$	$78{,}6_{0{,}8}$
dare[fisher]	$58{,}0_{5{,}0}$	$83{,}8_{3{,}3}$	$88{,}4_{1{,}0}$	$91{,}1_{0{,}0}$	**88,6**$_{3{,}1}$	$66{,}1_{4{,}7}$	$92{,}2_{-0{,}2}$	**89,5**$_{1{,}8}$	**56,3**$_{12{,}7}$	$79{,}3_{3{,}5}$
dare[ties]	$51{,}2_{-1{,}8}$	$71{,}3_{-0{,}8}$	$86{,}5_{-0{,}9}$	$85{,}2_{-0{,}5}$	$81{,}9_{-1{,}7}$	**71,8**$_{5{,}1}$	**93,1**$_{0{,}2}$	$86{,}9_{0{,}0}$	**56,3**$_{0{,}0}$	$76{,}0_{0{,}0}$

TABLE 2 – Résultats comparables à ceux du tableau 1 mais en faisant précéder les méthodes de fusion de l'application de la méthode de projection des modèles dans un même espace proposée dans cet article. Les trois premières lignes de résultats sont identiques à celles du tableau 1 et ne sont présentes que pour rappel. Les nombres en indice correspondent au gain ou à la perte de performance par rapport au résultat équivalent du tableau 1. Valeurs x100.

individuelles. Le nombre de méthodes de fusion au-dessus de la moyenne des modèles individuels ou égalant cette moyenne passe quant à lui de quatre à huit et toutes les méthodes sans leur association avec DARE en font partie.

La comparaison plus directe de notre méthode de projection et de DARE (cf. comparaison de la dernière partie du tableau 1 et de la partie centrale du tableau 2) laisse apparaître la supériorité de la première sur la seconde, le gain moyen avec notre méthode étant de 0,83 point tandis que la perte moyenne avec DARE est de 1,1 point. La combinaison notre méthode de projection et de DARE (application de la projection, puis de DARE et enfin d'une méthode de fusion) s'avère quant à elle relativement neutre : dans cette configuration, la perte moyenne enregistrée par l'application de DARE passe en effet de 1,1 point à 1 point.

Au niveau plus individuel des tâches, le dépassement systématique de la performance individuelle minimale reste toujours à obtenir à cause des résultats sur QQP, qui apparaît comme un jeu de données difficile pour les méthodes de fusion testées. La méthode Fisher, qui bénéficie en moyenne la plus fortement de la méthode de projection, se rapproche néanmoins de cet objectif. On peut noter également que l'application de cette projection permet de porter au niveau maximal individuel les résultats pour WNLI et d'avoir des gains très substantiels pour l'association de DARE et de RegMean ou Fisher. À l'inverse, la forte décroissance de DARE associée à Task arithmetic pour CoLA est difficile à interpréter. On constate par ailleurs que deux nouvelles tâches, QNLI et RTE, ont des résultats de fusion au-delà de la performance maximale des modèles individuels et une nouvelle tâche, RTE, dépasse leur performance moyenne.

4 Conclusion et perspectives

Dans cet article, nous nous sommes inscrits dans le contexte des méthodes de fusion de modèles de langue neuronaux en nous concentrant sur les méthodes d'alignement de paramètres entre modèles, en amont des méthodes de fusion proprement dite. Nous avons proposé plus précisément une nouvelle

méthode fondée sur le cadre théorique de l'analyse procustéenne pour projeter les paramètres d'un ensemble de modèles de langue fondés sur l'architecture transformeur dans un espace de représentation commun et faciliter leur fusion.

Nous avons appliqué cette méthode à la fusion de modèles affinés pour une même tâche et issus d'un même modèle de base. Les expérimentations que nous avons menées sur les neuf tâches du jeu de données GLUE et sur six méthodes de fusion de référence montrent l'intérêt de notre proposition, conduisant dans la plupart des cas à une amélioration notable des performances des méthodes de fusion testées. Ces évaluations montrent également que la fusion de modèles est une option intéressante pour minimiser les risques d'adopter un mauvais modèle parmi plusieurs modèles entraînés et permet même dans bon nombre de cas de dépasser la performance moyenne des modèles individuels.

La perspective la plus directe de ce travail est de l'appliquer à d'autres types de modèles, en particulier les modèles de type décodeur et encodeur-décodeur, sachant que les méthodes de fusion sont souvent plus intéressantes pour ce type de modèles que pour les modèles de type encodeur. Une application de la méthode de projection présentée aux paramètres des adapteurs (Rebuffi *et al.*, 2017) ou des modules LoRA (Hu *et al.*, 2022) serait également intéressante à réaliser.

Au-delà de cette simple application de la méthode proposée à d'autres types de modèles, il serait aussi intéressant de comparer les méthodes de fusion précoce considérées ici, et l'amélioration que nous présentons, avec des méthodes de fusion plus tardives de type bagging en mettant en regard leurs performances et leurs coûts respectifs. Enfin, dans une perspective plus explicative, examiner de façon plus précise l'impact de la méthode d'alignement proposée au niveau des paramètres des modèles, à l'instar de ce que Jin *et al.* (2025) ont réalisé pour les LoRA, serait également nécessaire pour mieux caractériser les transformations réalisées et identifier d'éventuelles sources d'amélioration possibles.

Remerciements

Ce travail a bénéficié du soutien partiel du projet PEPR ANR-23-PEIA-0008 SHARP, financé par France 2030. Il a en outre été réalisé grâce au supercalculateur Factory-IA financé par le Conseil Régional d'Île-de-France.

Références

AINSWORTH S., HAYASE J. & SRINIVASA S. (2023). Git Re-Basin : Merging Models modulo Permutation Symmetries. In *The Eleventh International Conference on Learning Representations*.

BENTIVOGLI L., CLARK P., DAGAN I. & GIAMPICCOLO D. (2009). The Fifth PASCAL Recognizing Textual Entailment Challenge. In *TAC*.

CACIULARU A., DAGAN I. & GOLDBERGER J. (2021). Denoising Word Embeddings by Averaging in a Shared Space. In L.-W. KU, V. NASTASE & I. VULIĆ, Éds., **SEM 2021 : The Tenth Joint Conference on Lexical and Computational Semantics*, p. 294–301, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2021.starsem-1.28.

CER D., DIAB M., AGIRRE E., LOPEZ-GAZPIO I. & SPECIA L. (2017). SemEval-2017 Task 1 : Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In S. BETHARD, M. CARPUAT, M. APIDIANAKI, S. M. MOHAMMAD, D. CER & D. JURGENS, Éds., *11th International*

Workshop on Semantic Evaluation (SemEval-2017), p. 1–14, Vancouver, Canada : Association for Computational Linguistics. DOI : 10.18653/v1/S17-2001.

DAGAN I., GLICKMAN O. & MAGNINI B. (2005). The PASCAL Recognising Textual Entailment Challenge. In J. Q. CANDELA, I. DAGAN, B. MAGNINI & F. D'ALCHÉ-BUC, Éds., *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop*, volume 3944 de *Lecture Notes in Computer Science*, p. 177–190 : Springer.

DEVLIN J., CHANG M.-W., LEE K. & TOUTANOVA K. (2019). BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 4171–4186, Minneapolis, Minnesota : Association for Computational Linguistics. DOI : 10.18653/v1/N19-1423.

DOLAN W. B. & BROCKETT C. (2005). Automatically Constructing a Corpus of Sentential Paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

GIAMPICCOLO D., MAGNINI B., DAGAN I. & DOLAN W. B. (2007). The third pascal recognizing textual entailment challenge. In *ACL-PASCAL workshop on textual entailment and paraphrasing*, p. 1–9.

GODDARD C., SIRIWARDHANA S., EHGHAGHI M., MEYERS L., KARPUKHIN V., BENEDICT B., MCQUADE M. & SOLAWETZ J. (2024). Arcee's MergeKit : A Toolkit for Merging Large Language Models. In F. DERNONCOURT, D. PREOŢIUC-PIETRO & A. SHIMORINA, Éds., *2024 Conference on Empirical Methods in Natural Language Processing : Industry Track*, p. 477–485, Miami, Florida, US : Association for Computational Linguistics. DOI : 10.18653/v1/2024.emnlp-industry.36.

GOWER J. C. (1975). Generalized procrustes analysis. *Psychometrika*, **40**(1), 33–51.

GOWER J. C. (2010). Procrustes methods. *WIREs Computational Statistics*, **2**(4), 503–508. DOI : https ://doi.org/10.1002/wics.107.

HAIM R. B., DAGAN I., DOLAN B., FERRO L., GIAMPICCOLO D., MAGNINI B. & SZPEKTOR I. (2006). The second pascal recognising textual entailment challenge. In *Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7, p. 785–794.

HOROI S., OROZCO CAMACHO A. M., BELILOVSKY E. & WOLF G. (2024). Harmony in Diversity : Merging Neural Networks with Canonical Correlation Analysis. In R. SALAKHUTDINOV, Z. KOLTER, K. HELLER, A. WELLER, N. OLIVER, J. SCARLETT & F. BERKENKAMP, Éds., *41st International Conference on Machine Learning*, volume 235, p. 18815–18832 : PMLR.

HU E. J., YELONG SHEN, WALLIS P., ALLEN-ZHU Z., LI Y., WANG S., WANG L. & CHEN W. (2022). LoRA : Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

HUANG C., YE P., CHEN T., HE T., YUE X. & OUYANG W. (2024). EMR-Merging : Tuning-Free High-Performance Model Merging. *Advances in Neural Information Processing Systems*, **37**, 122741–122769.

ILHARCO G., RIBEIRO M. T., WORTSMAN M., SCHMIDT L., HAJISHIRZI H. & FARHADI A. (2023). Editing Models with Task Arithmetic. In *The Eleventh International Conference on Learning Representations*.

IMFELD M., GRALDI J., GIORDANO M., HOFMANN T., ANAGNOSTIDIS S. & SINGH S. P. (2024). Transformer Fusion with Optimal Transport. In *The Twelfth International Conference on Learning Representations*.

JIN R., HOU B., XIAO J., SU W. J. & SHEN L. (2025). Fine-Tuning Attention Modules Only : Enhancing Weight Disentanglement in Task Arithmetic. In *The Thirteenth International Conference on Learning Representations*.

JIN X., REN X., PREOTIUC-PIETRO D. & CHENG P. (2023). Dataless Knowledge Fusion by Merging Weights of Language Models. In *The Eleventh International Conference on Learning Representations*.

KEMENTCHEDJHIEVA Y., RUDER S., COTTERELL R. & SØGAARD A. (2018). Generalizing Procrustes Analysis for Better Bilingual Dictionary Induction. In A. KORHONEN & I. TITOV, Éds., *22nd Conference on Computational Natural Language Learning*, p. 211–220, Brussels, Belgium : Association for Computational Linguistics. DOI : 10.18653/v1/K18-1021.

LEVESQUE H. J., DAVIS E. & MORGENSTERN L. (2012). The Winograd Schema Challenge. In *Principles of Knowledge Representation and Reasoning : Proceedings of the Thirteenth International Conference* : AAAI Press.

LI W., PENG Y., ZHANG M., DING L., HU H. & SHEN L. (2023). Deep Model Fusion : A Survey. *arXiv :2309.15698*. DOI : 10.48550/arXiv.2309.15698.

MATENA M. S. & RAFFEL C. A. (2022). Merging Models with Fisher-Weighted Averaging. In S. KOYEJO, S. MOHAMED, A. AGARWAL, D. BELGRAVE, K. CHO & A. OH, Éds., *Advances in Neural Information Processing Systems*, volume 35, p. 17703–17716 : Curran Associates, Inc.

RAJPURKAR P., ZHANG J., LOPYREV K. & LIANG P. (2016). SQuAD : 100,000+ Questions for Machine Comprehension of Text. In *2016 Conference on Empirical Methods in Natural Language Processing*, p. 2383–2392 : Association for Computational Linguistics.

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, Éds., *Advances in Neural Information Processing Systems*, volume 30 : Curran Associates, Inc.

SCHÖNEMANN P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, **31**(1), 1–10.

SHANKAR I., NIKHIL D. & KORNEL C. (2017). First quora dataset release : question pairs. https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.

SINGH S. P. & JAGGI M. (2020). Model Fusion via Optimal Transport. In H. LAROCHELLE, M. RANZATO, R. HADSELL, M. BALCAN & H. LIN, Éds., *Advances in Neural Information Processing Systems*, volume 33, p. 22045–22055 : Curran Associates, Inc.

SOCHER R., PERELYGIN A., WU J., CHUANG J., MANNING C. D., NG A. Y. & POTTS C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *2013 Conference on Empirical Methods in Natural Language Processing*, p. 1631–1642 : ACL.

TAITELBAUM H., CHECHIK G. & GOLDBERGER J. (2019). A Multi-Pairwise Extension of Procrustes Analysis for Multilingual Word Translation. In K. INUI, J. JIANG, V. NG & X. WAN, Éds., *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, p. 3560–3565, Hong Kong, China : Association for Computational Linguistics. DOI : 10.18653/v1/D19-1363.

VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł. & POLOSUKHIN I. (2017). Attention Is All You Need. In *Advances in neural information processing systems*, p. 5998–6008.

WANG A., SINGH A., MICHAEL J., HILL F., LEVY O. & BOWMAN S. (2018). GLUE : A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In T. LINZEN, G. CHRUPAŁA & A. ALISHAHI, Éds., *2018 EMNLP Workshop BlackboxNLP : Analyzing and Interpreting Neural Networks for NLP*, p. 353–355, Brussels, Belgium : Association for Computational Linguistics. DOI : 10.18653/v1/W18-5446.

WARSTADT A., SINGH A. & BOWMAN S. R. (2019). Neural Network Acceptability Judgments. *Transactions of the Association for Computational Linguistics*, **7**, 625–641. DOI : 10.1162/tacl_a_00290.

WILLIAMS A., NANGIA N. & BOWMAN S. (2018). A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In M. WALKER, H. JI & A. STENT, Éds., *2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 1112–1122, New Orleans, Louisiana : Association for Computational Linguistics. DOI : 10.18653/v1/N18-1101.

WORTSMAN M., ILHARCO G., GADRE S. Y., ROELOFS R., GONTIJO-LOPES R., MORCOS A. S., NAMKOONG H., FARHADI A., CARMON Y., KORNBLITH S. & SCHMIDT L. (2022). Model Soups : Averaging Weights of Multiple Fine-Tuned Models Improves Accuracy without Increasing Inference Time. In K. CHAUDHURI, S. JEGELKA, L. SONG, C. SZEPESVARI, G. NIU & S. SABATO, Éds., *39th International Conference on Machine Learning*, volume 162, p. 23965–23998 : PMLR.

YADAV P., TAM D., CHOSHEN L., RAFFEL C. & BANSAL M. (2023). TIES-Merging : Resolving Interference When Merging Models. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

YANG E., SHEN L., GUO G., WANG X., CAO X., ZHANG J. & TAO D. (2024). Model Merging in LLMs, MLLMs, and Beyond : Methods, Theories, Applications and Opportunities. *arXiv :2408.07666*. DOI : 10.48550/arXiv.2408.07666.

YU L., YU B., YU H., HUANG F. & LI Y. (2024). Language Models Are Super Mario : Absorbing Abilities from Homologous Models as a Free Lunch. In *Forty-First International Conference on Machine Learning*.

A Hyperparamètres

A.1 Entraînement des modèles individuels

Pour toutes les tâches, nous avons adopté les valeurs d'hyperparamètres suivantes pour l'entraînement des modèles :

— nombre d'époques : 3
— taille maximale des séquences : 128
— taille des lots : 32

Par ailleurs, nous avons également utilisé les valeurs suivantes de taux d'apprentissage pour les différentes tâches :

— CoLA : 2e-5
— MNLI : 1e-5

— MRPC : 5e-5
— QNLI : 1e-5
— QQP : 1e-5
— RTE : 5e-5
— SST-2 : 1e-5
— STSB : 5e-5
— WNLI : 1e-5

A.2 Fusion des modèles individuels

Pour l'implémentation des différentes méthodes de fusion, nous sommes appuyés sur la base de code des auteurs de la méthode DARE : https://github.com/yule-BUAA/MergeLM

En l'absence d'ensemble de validation, nous avons retenu pour les différents hyperparamètres des méthodes de fusion les valeurs conseillées comme les plus robustes dans les articles originels présentant ces méthodes :

— Task arithmetic
  — λ : facteur de pondération des différentes tâches, en l'occurrence des différents modèles ; $\lambda = 0,3$
— TIES
  — $\lambda = 1,0$
  — k : pourcentage des plus hautes valeurs de paramètres non réinitialisée zéro ; $k = 20$
— Fisher
  — nombre d'exemples : 1 024
  — taille des lots : 16
  — valeur minimale des coefficients de Fisher : 1e-6
  — facteur de repondération des coefficients de Fisher : 0,3
  — normalisation des coefficients de Fisher : oui
— RegMean
  — nombre d'exemples : 1 024
  — taille des lots : 16
  — α : facteur de pondération des termes non diagonaux des matrices de paramètres ; $\alpha = 0,9$
— DARE
  — stratégie de masquage des paramètres : aléatoire
  — pourcentage de paramètres masqués : 50
  — format des paramètres masqués : différences de valeur des paramètres entre modèles
  — mise à l'échelle des paramètres : oui