Actes de JEP-TALN-RECITAL 2024
Toulouse, France, 8 au 12 juillet 2024

Géométrie des vecteurs de tâches pour l'association et la combinaison de modèles

Loïc Fosse
Orange Innovation, Lannion, France
Aix Marseille Univ., CNRS, LIS, Marseille, France
loic.fosse@orange.com

RÉSUMÉ

Les adaptations de rang faible (LoRA, pour *Low-Rank Adaptation*) sont devenues un standard pour adapter des modèles à un faible coût. Elles sont de plus en plus utilisées que ce soit en traitement du langage ou des images. Plusieurs études utilisent ces adaptations et cherchent à les combiner *a posteriori* de manière à enrichir de manière additive les propriétés d'un modèle. Ces combinaisons suggèrent alors que nous pouvons associer les modèles dans l'espace des paramètres et que nous pouvons donner un sens à cela. Cette propriété n'est que très peu vérifiée dans la pratique et nous proposons ici plusieurs métriques visant à caractériser l'association entre les modèles dans l'espace des paramètres. Nous montrons finalement que nous pouvons corréler ces métriques avec les pertes de performance des modèles lorsque nous réalisons leurs combinaisons.

ABSTRACT

Task vectors geometry for models association and combination.

Low-Rank Adaptations (LoRA) have become a standard for adapting models at low cost. They are increasingly used in both language and image processing. Several studies use these adaptations and seek to combine them *a posteriori* so as to additively enrich the properties of a model. These combinations then suggest that we can associate models in parameter space and make sense of this. This property is only rarely verified in practice, and here we propose several metrics to characterize the association between models in parameter space. Finally, we show that we can correlate these metrics with losses in model performance when we combine them.

MOTS-CLÉS : transformers, adaptation de rangs faibles, similarité de modèles, distance de Grassmann, combinaisons de modèles.

KEYWORDS: transformers, low rank adaptation, models similarities, Grassmann distance, model combinations.

1 Introduction

L'accroissement de la taille des grands modèles de langue, basés sur l'architecture *transformers* (Vaswani *et al.*, 2017), rend de plus en plus complexe et coûteuse leur adaptation sur des tâches précises. Dans l'optique de dépasser les contraintes de ressources pour spécialiser des modèles, deux méthodes se distinguent : les méthodes d'affinages efficaces (*parameter efficient fine-tuning*) et la combinaison de modèles. Les méthodes d'affinage dites efficaces, comme la méthode LoRA (Hu *et al.*, 2021), le *prefix tuning* (Li & Liang, 2021), le *scaling and shifting* (Lian *et al.*, 2022) ou encore

Actes de la 26ème Rencontre des Étudiants Chercheurs en Informatique
pour le Traitement Automatique des Langues, pages 69–84.

le IA3 (Liu *et al.*, 2022), qui sont des dérivées de la notion d'adapteur (Rebuffi *et al.*, 2017), visent à n'affiner qu'une fraction du modèle, basé sur l'hypothèse que pour se spécialiser dans une tâche, seules des modifications mineures du modèle sont nécessaires. Parmi ces méthodes efficaces, les adaptations de rangs faibles (LoRA : *Low-Rank Adaptation*) se détachent du lot, de par les résultats qu'elles permettent d'obtenir en traitement automatique des langues (TAL) (Aleem *et al.*, 2024; Dettmers *et al.*, 2024; Kaddour *et al.*, 2023) et en vision (Luo *et al.*, 2023; Gandikota *et al.*, 2023), ou encore les propriétés intéressantes qui se dégagent de ces méthodes (Fu *et al.*, 2023). La combinaison (ou interpolation) de modèles (Li *et al.*, 2023; Jin *et al.*, 2022; Matena & Raffel, 2022; Falissard *et al.*, 2023; Yu *et al.*, 2023), inspirée des méthodes ensemblistes (Dietterich, 2000), quant à elle, vise à considérer des modèles entraînés sur des tâches et à les combiner dans l'espace des paramètres de manière à créer un nouveau modèle potentiellement meilleur que les modèles précédents sur une tâche donnée (enrichissement), capable de faire plusieurs tâches (multi-tâches) (Yang *et al.*, 2023) ou encore à faire une nouvelle tâche encore jamais vue par les différents modèles (Pfeiffer *et al.*, 2023; Chronopoulou *et al.*, 2023) – dans la littérature, on parle de *Modular Learning* pour ce dernier cas.

Plus récemment, des études cherchent aussi à combiner les modèles adaptés avec la méthode LoRA en TAL. Par exemple, Zhang *et al.* (2024) proposent des méthodes de combinaisons simples, de manière à : supprimer les biais d'un modèle, faire du multi-tâche, supprimer des composantes d'un modèle (notamment la toxicité dans la génération de textes) ou encore faire du transfert de domaine. De la même manière, dans le domaine du traitement des images, certaines études apparaissent aussi sur la composition de modèles adaptés avec LoRA, de manière à additionner les propriétés dans la génération d'images (Shah *et al.*, 2023; Gu *et al.*, 2024).

La combinaison de modèles (affinés complètement ou avec des méthodes de faible rang) semble alors fournir des résultats intéressants, mais elle n'est pas sans poser de questions. En effet, Frankle *et al.* (2020) introduisent la notion de connectivité linéaire et montrent que la combinaison de modèles peut parfois mener à des modèles instables perdant toute notion des tâches sur lesquelles ils ont été entraînés, notamment à cause de problèmes d'interférences entre modèles, qui peuvent être évités en forçant les modèles à partager une partie de leur trajectoire d'entraînement. Ce problème d'interférence entre les modèles est évoqué aussi plus récemment par Yadav *et al.* (2023), et ils proposent un algorithme empirique, pour adresser ce problème, basé sur l'amplitude des poids entre les différents modèles : lors de la comparaison de deux modèles, celui ayant les poids de plus grande amplitude sera le plus important.

La combinaison de modèles soulève alors la question de l'associativité de modèles entraînés sur des tâches différentes, afin de comprendre et d'interpréter les possibles effets d'interférence entre ces modèles. Intuitivement, la combinaison de deux modèles qui sont « proches » résultera en un modèle similaire aux deux premiers (peu d'interférence) et, inversement, la combinaison de deux modèles qui sont « éloignés » résultera en un modèle différent des deux premiers (grande interférence). Deux questions viennent alors naturellement : (1) comment définir la notion de proximité entre les modèles dans l'espace des paramètres ; et (2) pouvons-nous interpréter la proximité entre les modèles par rapport à une proximité empirique des tâches sur lesquelles ils ont été affinés afin d'ainsi comprendre *a posteriori* le comportement de modèles combinés ?

Nous adressons ces questions dans cette étude, à travers le prisme des adaptations de rang faible ainsi que des adaptations complètes de modèles (*full fine-tuning*), sur des tâches classiques de classification en traitement automatique des langues. Dans la suite, la section 2 présente le formalisme et les tâches étudiées, puis la section 3 présente les expériences et les résultats.

2 Formalisme et tâches

Dans un premier temps, nous définissons le formalisme associé aux adaptations de rang faible, les différentes distances utilisées pour calculer les proximités entre modèles, ainsi que les différentes tâches sur lesquelles nous allons travailler.

2.1 Adaptations de rang faible et vecteurs de tâche

Les adaptations de rang faible (LoRA) consistent à re-paramétriser le gradient d'un modèle par une décomposition de rang faible. Lors de l'apprentissage d'un modèle (*full fine-tuning*) paramétré par une matrice $W \in \mathbb{R}^{d \times d}$ (où d représente généralement la dimension des états cachés), nous partons d'un point W_0 (en général, un modèle pré-entraîné) et nous mettons à jour ces poids par descente de gradient pour arriver finalement à $W_f = W_0 + \Delta W$. Cette mise à jour peut être lourde étant donné que ΔW est de la même taille que le modèle de base. L'idée est alors de poser un nouveau paramètre $r << d$ (que l'on choisit) et de considérer $A \in \mathbb{R}^{r \times d}$ et $B \in \mathbb{R}^{d \times r}$ tel que $\Delta W = BA$, puis nous cherchons simplement à apprendre les poids des matrices A et B en laissant intact W_0. Dans la pratique, ces adaptations se réalisent sur les couches linéaires d'un modèle, et nous définissons un couple (A, B) pour chaque couche linéaire que l'on souhaite adapter de cette manière. Typiquement, dans un modèle *transformer*, nous pouvons définir un couple (A, B), pour les différentes projections en requêtes, clés et/ou valeurs [1] (Ghojogh & Ghodsi, 2020).

Ces adaptations légères possèdent des propriétés intéressantes, notamment le fait de régulariser l'entraînement et donc d'éviter les effets de sur-apprentissage (Fu *et al.*, 2023).

De plus, ces adaptations sont étroitement liées avec la notion de vecteurs de tâches introduite par Ilharco *et al.* (2022). Dans cette étude, les auteurs définissent le *vecteur de tâche* comme la différence entre les poids du modèle pré-entraîné et le modèle affiné sur une tâche. Avec les définitions précédentes, le vecteur de tâche correspond donc exactement à la quantité ΔW définie plus haut et donc au produit BA si le modèle est affiné avec des adaptations de faible rang.

Nous décidons alors d'exploiter cette notion de vecteur de tâche et de regarder ce que ces vecteurs peuvent encoder dans leur géométrie. Nous définissons, dans la section suivante, les distances qui vont nous aider à extraire des informations de ces vecteurs de tâches.

2.2 Distance entre vecteurs de tâches

Afin d'estimer des similarités entre tâches, il faut pouvoir être capable de calculer des distances entre des modèles dédiées à ces tâches. Nous proposons ici trois distances (ou *pseudo*-distances), entre deux matrices de paramètres de même taille W_1 (modèle d'une tâche 1) et W_2 (modèle d'une tâche 2), de manière hiérarchique (de la plus contraignante à la moins contraignante) :

Distance L_2 : $||W_1 - W_2||_2$ Cette distance est la plus naturelle et nous donne des informations sur les positions absolues des modèles.

1. Nous pouvons le faire pour tout type de matrices de projection, du modèle sans se limiter aux blocs d'attention, mais une pratique courante est de se limiter aux projections en requêtes et valeurs, comme cela est fait dans (Hu *et al.*, 2021)

Distance du cosinus : $1 - cos(W_1, W_2)$ Cette distance [2, 3] nous donne la corrélation empirique entre les poids des différents modèles (Saporta, 2006), et nous renseigne donc sur les variations communes entre les poids des modèles. Cette distance a notamment été utilisée dans Ilharco *et al.* (2022) pour calculer des similarités entre des modèles de classification d'images.

Distance de Grassmann (Hamm & Lee, 2008) : $d_G(W_1, W_2)$ Cette distance particulière, travaille sur les espaces images des modèles. Elle évaluera la proximité entre les espaces vectoriels images $Im(W_1)$ et $Im(W_2)$. En d'autre termes, elle évaluera la proximité entre les représentations en sortie des modèles. Cette distance a déjà été utilisée dans (Hu *et al.*, 2021) (*cf.* annexe G) pour estimer des dimensions communes entre des matrices (A, B) issues de LoRA avec des rangs différents entraînés sur une même tâche.

La distance de Grassmann s'appuie sur la notion d'angles principaux entre des espaces vectoriels (Afriat, 1957; Miao & Ben-Israel, 1992). Si W_1 et W_2 sont deux matrices de rang r dont les colonnes sont orthonormées [4], nous pouvons alors considérer σ, l'ensemble des valeurs propres de $W_1^T W_2$. Un résultat théorique donné par Björck & Golub (1973) est que ces valeurs propres sont les cosinus des angles principaux entre les espaces images de W_1 et W_2 *i.e.* si on pose θ l'ensemble des angles principaux entre $Im(W_1)$ et $Im(W_2)$, alors $cos(\theta) = \sigma$. Basés sur cette information, nous avons alors :

$$d_G(W_1, W_2) = \sqrt{\sum_{i=1}^{r} (\theta_i)^2} \leq \sqrt{r}\frac{\pi}{2} \tag{1}$$

Une remarque supplémentaire sur cette distance de Grassmann, est que si l'on considère $(W_1, W_2) \in \mathbb{R}^{d \times d}$, deux matrices de rang maximal d, alors $Im(W_1) = Im(W_2) = \mathbb{R}^d$ impliquant donc $d_G(W_1, W_2) = 0$. Cette distance n'est donc intéressante que sur les matrices qui ne sont pas de rang maximal, la rendant particulièrement intéressante dans le cadre de LoRA.

Nous avons ainsi trois distances qui nous permettent d'avoir un niveau d'analyse en cascade : la distance L_2 donne les positions absolues et est donc très stricte ; la distance du cosinus, quant à elle, est plus souple et nous donne la corrélation empirique entre les poids en faisant abstraction de l'amplitude de ces derniers ; et enfin la distance de Grassmann nous donne les distances entre les espaces images et fait abstraction de l'amplitude et de l'orientation, comme on peut le voir sur la figure 1 qui illustre sur un cas simple nos différentes métriques.

2.3 Tâches et entraînements des modèles

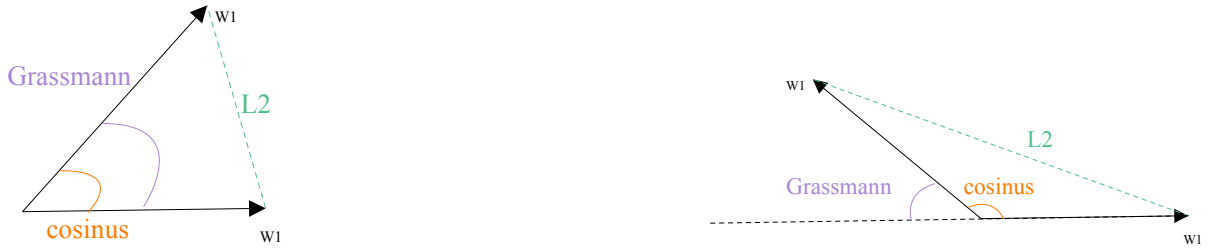
Nous décidons de nous focaliser ici sur des tâches de classification. Nous regardons les tâches suivantes appartenant au *benchmark* GLUE (Wang *et al.*, 2018) :

— COLA : tâche d'acceptabilité linguistique (classification binaire)
— MRPC : détection de paraphrase entre deux énoncés (classification binaire)
— RTE : inférence en langue naturelle (classification binaire)
— QQP : similarité / détection de paraphrase entre deux questions (classification binaire)

2. Le cosinus entre deux matrices est défini comme le cosinus entre les poids applatis de ces deux matrices. Nous considérons également la quantité $1 - cos()$ de manière à avoir une distance et non pas une similarité.

3. Mathématiquement il ne s'agit pas d'une distance. Nous utilisons le terme distance pour traduire le fait qu'une valeur élevée (faible) traduira un éloignement (une proximité).

4. si elles ne le sont pas, une simple décomposition en valeurs singulières des matrices peut nous permettre de le faire



(a) Grassmann = cos

(b) Grassmann ≠ cos

FIGURE 1 – Visualisation des différentes distances sur deux modèles ($W1$, $W2$) à 1 dimension (Vecteurs).

	COLA	MRPC	RTE	QNLI	QQP	MNLI	SST2	SNLI	YELP	IMDB
train	8.55k	3.67k	2.49k	105k	364k	393k	67.3k	550k	560k	25k
dev	1.04k	408	277	5.46k	40.4k	19.65k	872	10k	-	-
test	-	-	-	-	-	-	-	10k	38k	25k

TABLE 1 – Cardinalité des différents jeux de données utilisés

— QNLI : question-réponse sous forme d'inférence en langue naturelle (classification binaire)
— MNLI : inférence en langue naturelle (classification à trois classes)
— SST2 : classification en polarité (classification binaire)

Nous ajoutons la tâche SNLI (Inférence en Langue Naturelle) (Bowman *et al.*, 2015) en complément de la tâche MNLI ainsi que YELP (Asghar, 2016) et IMDB (Maas *et al.*, 2011) en complément de la tâche SST2. Cet ajout est fait pour avoir des jeux de données plus volumineux, contenant des biais différents (notamment pour SNLI), afin d'enrichir nos observations. L'ensemble de ces corpus sont anglais et nous donnons dans la table 1 les tailles respectives des différents jeux de données.

Pour les jeux de données du *benchmark* GLUE, nous n'avons pas à notre disposition un jeu de test annoté [5], les performances sont donc données sur le jeu de validation. Le jeu d'entraînement est lui coupé en deux (90-10) pour faire un jeu d'entraînement et un jeu de validation. C'est cette même stratégie qui est utilisée dans (Zhang *et al.*, 2024).

Tous nos modèles sont affinés à partir du modèle pré-entraîné `roberta-base` (Liu *et al.*, 2019). Pour les affinages complets, nous utilisons pour chaque jeu de données un taux d'apprentissage de 2×10^{-5} avec des lots de taille 20 pendant 20 époques avec une stratégie d'arrêt prématuré fondée sur la fonction de perte de validation avec une patience de 5. Pour les affinages `via` LoRA, nous utilisons un rang $r = 8$ avec un coefficient régularisateur $\alpha = 16$ (2 fois le rang) et un taux d'apprentissage de 10^{-4} (5 fois plus grand que pour les affinages complets), sur le même nombre d'époques et la même stratégie d'arrêt que pour l'affinage complet. Ces adaptations sont réalisées sur les projections de requêtes et de valeurs dans chaque module d'attention. Nous reportons les performances des différents modèles dans le tableau 2. L'ensemble de nos modèles a été entraîné avec les bibliothèques `transformers` et `peft` (pour la méthode LoRA).

5. Les jeux de données tests sont donnés simplement avec le texte et nous devons envoyer nos prédictions aux auteurs pour qu'ils réalisent l'évaluation.

	COLA	MRPC	RTE	QNLI	QQP	MNLI	SST2	SNLI	YELP	IMDB
f-FT	56.36	85.78	69.66	91.87	86.36	85.75	93.69	90.61	97.73	93.95
LoRa	54.98	86.52	71.84	92.28	86.02	86.73	93.92	90.61	98.01	95.34

TABLE 2 – Performances des modèles sur chacune des tâches avec une distinction entre *full-finetuning* (*f-FT*) et adaptations de rangs faibles (LoRa). Les performances sont données pour chaque jeu de données avec la métrique officielle GLUE donnée dans (Wang *et al.*, 2018) et calculée avec la librairie evaluate. Le jeu de données SNLI est évalué avec le F1 de MNLI tandis que YELP et IMDB sont évalués avec la métrique de SST2. Nous ne pouvons pas ici comparer les colonnes entre elles (métrique différente pour chaque jeu de données)

Dans l'ensemble de nos corpus, il est important de noter que les tâches d'inférence (lien logique entre deux énoncés) et les tâches de détection de paraphrase (similarité sémantique entre deux énoncés), peuvent être très similaires dans leurs définitions. La différence majeure entre ces deux étant que la dernière est une tâche symétrique par rapport aux énoncés, contrairement à la première.

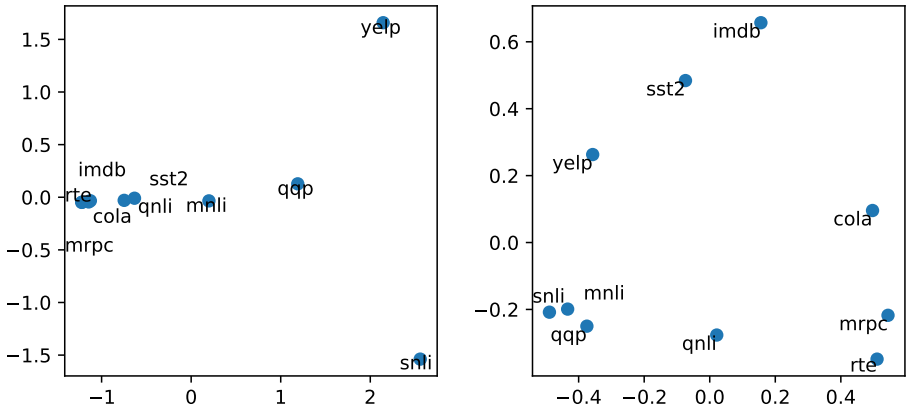
3 Expériences et résultats

Dans cette section, nous utilisons les différentes métriques introduites pour montrer que les distances entre les modèles sont cohérentes par rapport aux tâches sur lesquelles ils sont entraînés. Enfin, nous illustrons le fait que combiner des couples de matrices LoRA (A, B) éloignées, au sens de nos métriques, dégrade les performances de nos modèles allant ainsi dans le sens de nos hypothèses sur les interférences entre ces derniers : combiner des modèles éloignés produira un modèle aux performances dégradées sur les tâches initiales de chaque modèle combiné.

3.1 Distance entre vecteurs de tâches

Pour chacune des distances d évoquées précédemment, nous construisons la matrice 3D T_d telle que $T_d(i, j, k)$ représente la distance d entre le vecteur de tâche des paramètres de la couche k des modèles respectivement entraînés sur la tâche i et j. Pour faciliter les analyses de cet objet, nous considérerons la réduction de T_d par la moyenne sur les couches *i.e.* $\bar{T}_d(i, j) = \frac{1}{|K|} \sum_{k=1}^{|K|} T_d(i, j, k)$. Nous obtenons finalement une matrice symétrique $\bar{T}_d(i, j)$ qui rend compte des distances entre les modèles [6]. Un moyen simple de visualiser cette matrice est alors de réaliser une réduction de dimension par analyse en composantes principales (ACP) (F.R.S., 1901) afin de visualiser les modèles qui sont proches sur un plan. Nous reportons sur la figure 2 cette ACP sur les vecteurs de tâches des modèles affinés complètement. Nous ne présentons sur cette figure que les distances du cosinus et L_2, en effet pour les modèles affinés complètement les rangs des vecteurs de tâches sont très proches des rangs maximaux (rang variant entre 766 et 768 pour un rang maximal de 768) rendant peu informative et très coûteuse la distance de Grassmann (*cf.* section 2.2). Sur la figure 2b, nous observons des comportements intéressants. Les modèles affinés sur les tâches de classification en polarité se retrouvent groupés ensemble (YELP, SST2, IMDB), les modèles affinés sur RTE et MRPC

6. Nous notons que la moyenne à travers les couches peut masquer les comportements différents de chaque couches et nous adressons ceci plus loin dans l'étude

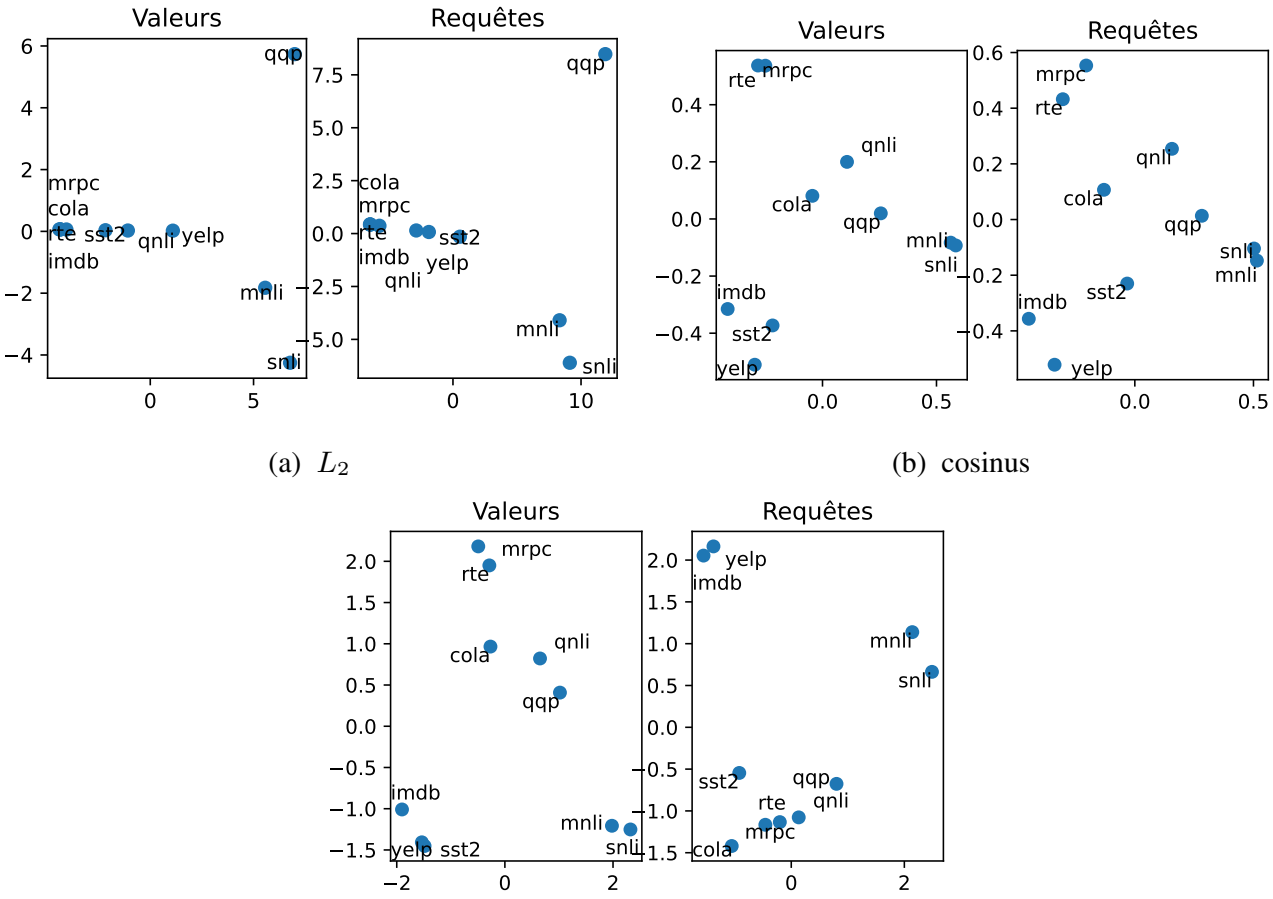


(a) L_2 (b) cosinus

FIGURE 2 – Projection deux dimension par PCA des matrices de distances $\bar{T}_d$, pour les vecteurs de tâches des modèles affinés complètement.

sont aussi groupés ensembles ce qui peut s'expliquer par le fait que les tâches d'inférence et de détection de paraphrases sont très proches dans leurs définitions. Les tâches d'inférence comme MNLI, SNLI et QNLI dans une moindre mesure sont aussi groupées. La tâche QQP, qui semble s'apparenter à la tâche MRPC (détection de paraphrase), se range plutôt du côté des tâches d'inférence (MNLI, SNLI). Enfin, la tâche COLA, seule tâche d'acceptabilité linguistique, ne semble pas montrer d'association particulière avec une autre tâche. La séparation entre le groupe MNLI, SNLI et RTE, MRPC (qui sont 4 tâches très proches dans leurs définitions) peut s'expliquer par les tailles des jeux d'entraînements (*c.f.* table 1) : pour MNLI et SNLI, nous avons des jeux d'entraînements très volumineux (plusieurs centaines de milliers de phrases) tandis que ceux de RTE et MRPC sont plus modestes (quelques milliers). L'entraînement sur les jeux de données volumineux présente donc plus d'étapes d'entraînements (plus de mises à jour du modèle par descente de gradient), pouvant donc expliquer les différences observées. La distance du cosinus nous permet ainsi d'identifier des liens entre les modèles, qui semblent cohérents par rapport à leurs jeux de données d'entraînement. En revanche, les groupements réalisés sur la figure 2a ne donnent lieu à aucune interprétation intéressante traduisant le fait que la distance L_2 est trop restrictive pour bien évaluer des distances entre des modèles en comparaison à la distance du cosinus.

Nous reprenons ainsi cette analyse sur les modèles affinés avec des adaptations de faible rang. Pour rappel, dans le cadre de la méthode LoRA, le vecteur de tâche nous est directement donné par les produits BA. Comme évoqué précédemment, avec cette méthode, nous faisons une adaptation sur les projections en requêtes et en valeurs. Pour l'analyse de ces modèles, nous décidons donc de calculer T_d^r et T_d^v respectivement pour les requêtes et les valeurs (que l'on moyenne sur les couches). Nous pouvons retrouver les résultats de projection sur la figure 3. Nous ajoutons en plus la distance de Grassmann étant donné que cette fois-ci les vecteurs de tâches sont de faible rang (ce sont les produits BA), donnant ainsi un intérêt à cette distance. Sur la figure 3a, nous faisons la même constatation que sur la figure 2a : la distance L_2 est une distance trop restrictive pour apprécier les différences entre les modèles et nous ne distinguons pas de groupements particuliers. Sur la figure 3b, représentant les distance du cosinus entre produits BA, nous faisons des interprétations sensiblement similaires à celles sur la figure 2b avec, cette fois, la tâche COLA qui semble se rapprocher du groupement MNLI-SNLI. On observe en plus, très peu de différences entre les requêtes et les valeurs (excepté un léger déplacement de la tâche SST2), traduisant donc que, vis à vis de la distance du cosinus,



(a) L_2

(b) cosinus

(c) Grassmann

FIGURE 3 – Projection deux dimension par PCA des matrices de distances $\bar{T}_d$, pour les différentes distances.

les modules de requêtes et de valeurs dans les blocs d'attention donnent les mêmes informations sur les différentes tâches. En revanche, sur la figure 3c, représentant les distances de Grassmann, nous remarquons désormais des différences entre les requêtes et les valeurs. L'interprétation sur les valeurs est identique à celle sur les cosinus, avec, en plus, des groupements qui sont mieux définis et mieux espacés sur les axes principaux (*cf.* échelle sur les axes). En revanche, celle sur les requêtes est légèrement différente, le groupement des tâches de polarité n'est plus clairement formé, avec la tâche SST2 qui se rapproche de tâches comme MRPC, RTE ou COLA. Il semblerait alors que, dans les projections de requêtes avec la distance de Grassmann, nous perdons certaines informations relatives à nos différentes tâches. La distance de Grassmann permet ainsi d'identifier, géométriquement, le fait que nous perdons certaines informations relatives aux différentes tâches sur les projections en requêtes dans le module d'attention. Ceci corrobore les travaux de Ethayarajh (2019); Fosse *et al.* (2023, 2022) qui montrent que, dans les modèles *transformers*, c'est la direction prise par les plongements en sortie des modules d'auto-attention qui donne le plus d'informations. Or, comme cela est rappelé dans (Fosse *et al.*, 2023), dans les blocs d'attention, c'est principalement le module de valeurs qui influe sur la direction donnée aux plongements.

	L_2	cos	d_G	cos^{5+}	d_G^{5+}
Requêtes	0.45 (0.21)	0.65 (0.16)	0.65 (0.25)	0.65 (0.16)	0.61 (0.25)
Valeurs	0.48 (0.23)	0.68 (0.21)	0.64 (0.19)	0.74 (0.20)	**0.77 (0.14)**

TABLE 3 – Moyenne (écart type) sur les tâches, des coefficients r de Spearman entre $\delta(i,.)$ et $\bar{T_d}(i,.)$.

3.2 Combinaison de modèles

Nous décidons de poursuivre cette étude sur les distances entre les modèles en réalisant des combinaisons entre ces derniers afin d'inspecter le comportement des modèles combinés en fonction de leur distance dans le but de mieux comprendre les effets d'interférence. Dans cette partie, nous nous concentrons sur les adaptations de rang faible et adoptons la démarche utilisée par Zhang *et al.* (2024). Si on considère les deux couples (A_1, B_1) et (A_2, B_2) de matrices LoRA dont le rang est identique, alors le couple résultant de leur combinaison sera défini par $(\frac{1}{2}(A_1 + A_2), \frac{1}{2}(B_1 + B_2))$ [7]. Ainsi, pour l'ensemble des tâches que nous avons introduites, nous définissons δ tel que $\delta(i,j)$, représente la différence de performance sur la tâche i entre (1) le modèle entraîné sur la tâche i et (2) le modèle entraîné sur la tâche i et combiné avec la tâche j. Pour toutes les tâches i, nous calculons la corrélation de Spearman (Sedgwick, 2014) entre $\delta(i,.)$ et $\bar{T_d}(i,.)$. Nous reportons les résultats de corrélations dans la table 3 (colonnes L_2, cos et d_G). Nous pouvons tout d'abord y voir que l'ensemble des corrélations sont non nulles, ce qui va dans le sens de notre hypothèse de départ, à savoir à nouveau que l'association entre deux modèles éloignés résultera en un modèle moins bon sur les tâches initiales (une grande distance semble être associée à un grand δ et donc une grande interférence). Nous pouvons également voir que les valeurs de corrélations pour la distance L_2 sont plus faibles que pour les autres distances avec aussi des écart-types associés relativement grands. Ceci est à mettre en lien avec nos observations sur la distance L_2 qui était trop stricte et ne nous permettait pas de visualiser des associations intéressantes entre les tâches. Enfin, nous constatons que les corrélations calculées pour les distances de Grassmann et du cosinus sont plus fortes, traduisant le fait que ces métriques semblent plus intéressantes pour estimer les effets d'interférence entre modèles (et aussi les distances entre modèles *cf.* section 3.1).

Les observations de Rogers *et al.* (2021) suggèrent cependant que les différentes couches d'un modèle *transformer* encodent des informations différentes qui sont de plus en plus liées à la tâche à mesure que l'on monte dans les couches. Nous décidons alors de recalculer les corrélations précédentes entre $\delta(i,.)$ et $\bar{T_d^k}(i,.)$, où $\bar{T_d^k}(i,.)$ représente les distances moyennées sur les k *dernières* couches. Dans cette dernière expérience, nous supprimons la distance L_2 qui n'a pas semblé fournir de résultats intéressants jusqu'ici. Nous testons dans un premier temps pour $k = 12$ (c.-à-d. la dernière couche dans un modèle `roberta-base`) et observons une chute des coefficients de corrélation, traduisant le fait que la dernière couche d'un modèle ne semble pas suffire pour estimer les effets d'interférence entre ces derniers. Nous reportons finalement les résultats pour $k = 5$ (qui semble nous fournir les meilleurs résultats) sur la table 3 (colonnes cos^{5+} et d_G^{5+}). Cette fois-ci, pour les modules de valeurs dans les blocs d'attention, nous observons une hausse des coefficients de corrélations et une baisse de la variance associée. La distance de Grassmann semble en particulier donner des corrélations plus fortes. Les modules de requêtes ne montrent pas d'évolution particulière, ce qui est à mettre en lien avec les précédentes observations de la section 3.1 (les modules de requêtes influent peu sur la

7. Nous travaillons avec des modèles de type encodeur, la tête de classification n'est pas modifiée lors de la combinaison de deux modèles

direction des plongements). Il semblerait alors que les couches profondes des modèles *transformers* soient plus intéressantes pour caractériser les tâches et les possibles effets d'interférence entre les modèles, les couches se situant aux extrémités étant trop ou pas assez spécialisées sur les données ce qui semble correspondre à nouveau avec les observations de Rogers *et al.* (2021).

3.3 Interprétation géométrique

À ce stade, nous pouvons faire une interprétation géométrique sur la distance de Grassmann. Durant les expériences précédentes sur les adaptations de rang faible (distance et corrélation suite aux combinaisons), la distance de Grassmann nous a donné des résultats très similaires à la distance du cosinus. Cependant, lorsque nous calculons cette distance entre deux couples (A_1, B_1), (A_2, B_2) LoRA, nous avons l'égalité suivante [8] :

$$d_G(B_1A_1, B_2A_2) = d_G(B_1, B_2)\ . \tag{2}$$

Cette égalité est la conséquence directe du fait que pour tout couple (A, B), nous avons $Im(BA) \subset Im(B)$ ainsi que du fait que A est surjective (vérification empirique). Ainsi, le fait que la distance du cosinus et la distance de Grassmann amènent aux mêmes interprétations suggère que la majeure partie des informations portées par une adaptation LoRA réside dans leurs matrices B. Ceci peut s'expliquer de par le fait que les matrices A projettent des représentations de très grandes dimensions (ici 768) dans un espace de très faible dimension (ici 8). Cette forte compression induit nécessairement une forte distortion et donc une grande perte d'informations dans les représentations. C'est ensuite la matrice B qui se charge de re-projeter les représentations dans un espace de grande dimension et de donner ainsi à chaque représentation sa direction.

4 Conclusion et perspectives

À travers cette étude, nous reprenons différentes métriques afin de calculer l'association entre les modèles dans l'espace des paramètres, basé sur la notion de vecteurs de tâches. Nous montrons que la distance L_2 est trop restrictive et ne permet pas de bien apprécier les différences entre les modèles. Il faut alors se tourner vers des métriques plus vectorielles (sensibles à la direction et non à la norme), comme la distance du cosinus ou celle de Grassmann, lorsque cette dernière fait du sens, qui permettent de mettre en évidence des liens cohérents entre les tâches sur lesquelles les modèles ont été entraînés. Cette association entre les tâches est notamment mise en avant en combinant des modèles adaptés avec LoRA. Nous montrons que combiner des modèles éloignés dans l'espace des paramètres semble perturber ces derniers en les rendant moins bons sur les tâches initiales, répondant ainsi à Ilharco *et al.* (2022), qui évoquaient dans leur discussion que les combinaisons entre modèles orthogonaux, au sens du cosinus, aboutiraient à moins d'interférences entre les modèles. De manière plus précise, les récents travaux de Ortiz-Jimenez *et al.* (2024) suggèrent que cette perte de performance suite à la combinaison des modèles est due au fait que le modèle ne respecte pas la propriété dite d'*arithmétique des tâches*, ce qui semble donc traduire que le régime d'entraînement du modèle n'est pas linéaire. Les auteurs proposent alors une méthode pour rendre linéaire le régime d'entraînement des modèles, afin de faciliter la combinaison des paramètres de ces derniers.

8. Nous justifions théoriquement cette égalité dans l'annexe A

Notre étude possède plusieurs limites. Tout d'abord, à une époque où le paradigme des modèles génératifs est dominant, nous n'avons traité que des tâches de classification relativement simples (3 classes au maximum) avec un modèle de type encodeur. Nous pouvons observer la simplicité des tâches traitées lors des observations de la section 3.1. Dans cette section, les interprétations sur les distances entre les tâches sont relativement similaires entre les modèles affinés complètement et les modèles affinés avec LoRA, suggérant donc que, avec un objet dont la dimension intrinsèque est faible, nous capturons essentiellement les mêmes informations qu'avec un vecteur de tâche en très grande dimension. Cet énoncé n'est pas à prendre pour résultat mais plutôt à mettre en regard avec la simplicité des tâches que nous avons considérée ici. En effet, nous supposons que l'on peut établir une corrélation entre le nombre de dimensions minimum qu'il faut donner à la méthode LoRA pour encoder les mêmes informations qu'un vecteur de tâche complet et la difficulté des tâches traitées : plus il faut de dimensions, plus la tâche est complexe. Une extension possible est alors de reprendre ces travaux avec des modèles de type encodeur/décodeur (Raffel *et al.*, 2020; Lewis *et al.*, 2020) ou décodeur pur (Radford *et al.*, 2019) et sur des tâches génératives (résumé, question réponse, *etc* ...) plus complexes qui demanderaient *a priori* plus de dimensions pour être modélisées correctement. Une seconde limite réside dans notre expérience sur les combinaisons des modèles. Nous nous limitons ici à un cadre très simple qui ouvre certaines pistes pour comprendre la combinaison dans un but de multi-tâches. Nous n'avons cependant pas évoqué le cadre des entraînements modulaires (combiner des modèles pour réaliser une nouvelle tâche). Dans ce cas, combiner des modèles éloignés peut effectivement avoir un intérêt. Il nous faut alors vérifier que le modèle sur la nouvelle tâche peut s'exprimer comme une combinaison linéaire des modèles initiaux. Dans un cas d'affinage complet, cela ne semble pas poser de problèmes étant donné que les vecteurs de tâches sont de très grandes dimensions et donc la probabilité de couvrir tout l'espace avec les combinaisons de ces vecteurs est grande. Cela devient moins évident dans un cas d'adaptation de rang faible où les vecteurs de tâches sont de très faible dimension (relativement à leur taille) et donc, couvrir tout l'espace avec de simples combinaisons linéaires devient moins probable. Avoir des vecteurs de tâches qui sont orthogonaux dans ce dernier cas permettrait d'augmenter cette dernière probabilité. Cette remarque suggère alors que la notion d'interférence entre les modèles lors de leur combinaison dépend du contexte dans lequel on se place. Dans notre étude, nous avons supposé que les interférences se mesuraient *via* la perte de performance dans une tâche sur laquelle il a été entraîné suite à une combinaison avec un autre modèle. Cette dernière ne semble pas en adéquation avec un cadre d'entraînement modulaire où on cherche à adresser une nouvelle tâche.

Références

AFRIAT S. N. (1957). Orthogonal and oblique projectors and the characteristics of pairs of vector spaces. In *Mathematical proceedings of the Cambridge philosophical society*, volume 53, p. 800–816 : Cambridge University Press.

ALEEM S., DIETLMEIER J., ARAZO E. & LITTLE S. (2024). Convlora and adabn based domain adaptation via self-training. *arXiv preprint arXiv :2402.04964*.

ASGHAR N. (2016). Yelp dataset challenge : Review rating prediction. *arXiv preprint arXiv :1605.05362*.

BJÖRCK A. & GOLUB G. H. (1973). Numerical methods for computing angles between linear subspaces. *Mathematics of computation*, **27**(123), 579–594.

BOWMAN S. R., ANGELI G., POTTS C. & MANNING C. D. (2015). A large annotated corpus for learning natural language inference. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, p. 632–642 : Association for Computational Linguistics (ACL).

CHRONOPOULOU A., PFEIFFER J., MAYNEZ J., WANG X., RUDER S. & AGRAWAL P. (2023). Language and task arithmetic with parameter-efficient layers for zero-shot summarization. *arXiv preprint arXiv :2311.09344*.

DETTMERS T., PAGNONI A., HOLTZMAN A. & ZETTLEMOYER L. (2024). Qlora : Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, **36**.

DIETTERICH T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, p. 1–15 : Springer.

ETHAYARAJH K. (2019). How contextual are contextualized word representations ? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* : Association for Computational Linguistics.

FALISSARD L., GUIGUE V. & SOULIER L. (2023). Apprentissage de sous-espaces de préfixes. In *18e Conférence en Recherche d'Information et Applications–16e Rencontres Jeunes Chercheurs en RI–30e Conférence sur le Traitement Automatique des Langues Naturelles–25e Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues*, p. 59–73 : ATALA.

FOSSE L., NGUYEN D.-H., SÉBILLOT P. & GRAVIER G. (2022). Une étude statistique des plongements dans les modèles transformers pour le français. In *Traitement Automatique des Langues Naturelles (TALN 2022)*, p. 247–256 : ATALA.

FOSSE L., NGUYEN D. H., SÉBILLOT P. & GRAVIER G. (2023). Géométrie de l'auto-attention en classification : quand la géométrie remplace l'attention. In *18e Conférence en Recherche d'Information et Applications\\16e Rencontres Jeunes Chercheurs en RI\\30e Conférence sur le Traitement Automatique des Langues Naturelles\\25e Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues*, p. 137–150 : ATALA.

FRANKLE J., DZIUGAITE G. K., ROY D. & CARBIN M. (2020). Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, p. 3259–3269 : PMLR.

F.R.S. K. P. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **2**(11), 559–572. DOI : 10.1080/14786440109462720.

FU Z., YANG H., SO A. M.-C., LAM W., BING L. & COLLIER N. (2023). On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, p. 12799–12807.

GANDIKOTA R., MATERZYNSKA J., ZHOU T., TORRALBA A. & BAU D. (2023). Concept sliders : Lora adaptors for precise control in diffusion models. *arXiv preprint arXiv :2311.12092*.

GHOJOGH B. & GHODSI A. (2020). Attention mechanism, transformers, bert, and gpt : tutorial and survey.

GU Y., WANG X., WU J. Z., SHI Y., CHEN Y., FAN Z., XIAO W., ZHAO R., CHANG S., WU W. *et al.* (2024). Mix-of-show : Decentralized low-rank adaptation for multi-concept customization of diffusion models. *Advances in Neural Information Processing Systems*, **36**.

HAMM J. & LEE D. D. (2008). Grassmann discriminant analysis : a unifying view on subspace-based learning. In *Proceedings of the 25th international conference on Machine learning*, p. 376–383.

HU E. J., WALLIS P., ALLEN-ZHU Z., LI Y., WANG S., WANG L., CHEN W. *et al.* (2021). Lora : Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

ILHARCO G., RIBEIRO M. T., WORTSMAN M., SCHMIDT L., HAJISHIRZI H. & FARHADI A. (2022). Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

JIN X., REN X., PREOTIUC-PIETRO D. & CHENG P. (2022). Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv :2212.09849*.

KADDOUR J., HARRIS J., MOZES M., BRADLEY H., RAILEANU R. & MCHARDY R. (2023). Challenges and applications of large language models. *arXiv preprint arXiv :2307.10169*.

KOPICZKO D. J., BLANKEVOORT T. & ASANO Y. M. (2023). Vera : Vector-based random matrix adaptation. *arXiv preprint arXiv :2310.11454*.

LEWIS M., LIU Y., GOYAL N., GHAZVININEJAD M., MOHAMED A., LEVY O., STOYANOV V. & ZETTLEMOYER L. (2020). Bart : Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 7871–7880.

LI W., PENG Y., ZHANG M., DING L., HU H. & SHEN L. (2023). Deep model fusion : A survey. *arXiv preprint arXiv :2309.15698*.

LI X. L. & LIANG P. (2021). Prefix-tuning : Optimizing continuous prompts for generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, p. 4582–4597.

LIAN D., ZHOU D., FENG J. & WANG X. (2022). Scaling & shifting your features : A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems*, **35**, 109–123.

LIU H., TAM D., MUQEETH M., MOHTA J., HUANG T., BANSAL M. & RAFFEL C. A. (2022). Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, **35**, 1950–1965.

LIU Y., OTT M., GOYAL N., DU J., JOSHI M., CHEN D., LEVY O., LEWIS M., ZETTLEMOYER L. & STOYANOV V. (2019). Roberta : A robustly optimized bert pretraining approach. *arXiv preprint arXiv :1907.11692*.

LUO S., TAN Y., PATIL S., GU D., VON PLATEN P., PASSOS A., HUANG L., LI J. & ZHAO H. (2023). Lcm-lora : A universal stable-diffusion acceleration module. *arXiv preprint arXiv :2311.05556*.

MAAS A., DALY R. E., PHAM P. T., HUANG D., NG A. Y. & POTTS C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics : Human language technologies*, p. 142–150.

MATENA M. S. & RAFFEL C. A. (2022). Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, **35**, 17703–17716.

MIAO J. & BEN-ISRAEL A. (1992). On principal angles between subspaces in rn. *Linear algebra and its applications*, **171**, 81–98.

ORTIZ-JIMENEZ G., FAVERO A. & FROSSARD P. (2024). Task arithmetic in the tangent space : Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, **36**.

PFEIFFER J., RUDER S., VULIĆ I. & PONTI E. M. (2023). Modular deep learning. *arXiv preprint arXiv :2302.11529*.

RADFORD A., WU J., CHILD R., LUAN D., AMODEI D., SUTSKEVER I. *et al.* (2019). Language models are unsupervised multitask learners.

RAFFEL C., SHAZEER N., ROBERTS A., LEE K., NARANG S., MATENA M., ZHOU Y., LI W. & LIU P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, **21**(1), 5485–5551.

REBUFFI S.-A., BILEN H. & VEDALDI A. (2017). Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, **30**.

ROGERS A., KOVALEVA O. & RUMSHISKY A. (2021). A primer in bertology : What we know about how bert works. *Transactions of the Association for Computational Linguistics*, **8**, 842–866.

SAPORTA G. (2006). *Probabilités, analyse des données et statistique*. Editions technip.

SEDGWICK P. (2014). Spearman's rank correlation coefficient. *Bmj*, **349**.

SHAH V., RUIZ N., COLE F., LU E., LAZEBNIK S., LI Y. & JAMPANI V. (2023). Ziplora : Any subject in any style by effectively merging loras. *arXiv preprint arXiv :2311.13600*.

SUN Y., CHEN Q., HE X., WANG J., FENG H., HAN J., DING E., CHENG J., LI Z. & WANG J. (2022). Singular value fine-tuning : Few-shot segmentation requires few-parameters fine-tuning. *Advances in Neural Information Processing Systems*, **35**, 37484–37496.

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

WANG A., SINGH A., MICHAEL J., HILL F., LEVY O. & BOWMAN S. R. (2018). Glue : A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

YADAV P., TAM D., CHOSHEN L., RAFFEL C. & BANSAL M. (2023). Resolving interference when merging models. *arXiv preprint arXiv :2306.01708*.

YANG E., WANG Z., SHEN L., LIU S., GUO G., WANG X. & TAO D. (2023). Adamerging : Adaptive model merging for multi-task learning. *arXiv preprint arXiv :2310.02575*.

YU L., YU B., YU H., HUANG F. & LI Y. (2023). Language models are super mario : Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv :2311.03099*.

ZHANG J., LIU J., HE J. *et al.* (2024). Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, **36**.

A Retour sur la distance de Grassmann

Dans cette section, nous ajoutons des éléments de justifications au fait que si nous considérons deux couples de matrices LoRA (A_1, B_1) et (A_2, B_2), alors nous avons $d_g(B_1A_1, B_2A_2) = d_G(B_1, B_2)$. Comme énoncé dans le corps de l'étude, cette propriété découle directement du fait que dans un premier temps, nous avons $Im(B_iA_i) \subset Im(B_i)$ par définition, puis du fait que les matrices A_i sont de rangs plein [9], induisant ainsi que $Im(B_i) \subset Im(B_iA_i)$ (utilisation du théorème du rang) amenant donc à l'égalité $Im(B_iA_i) = Im(B_i)$.

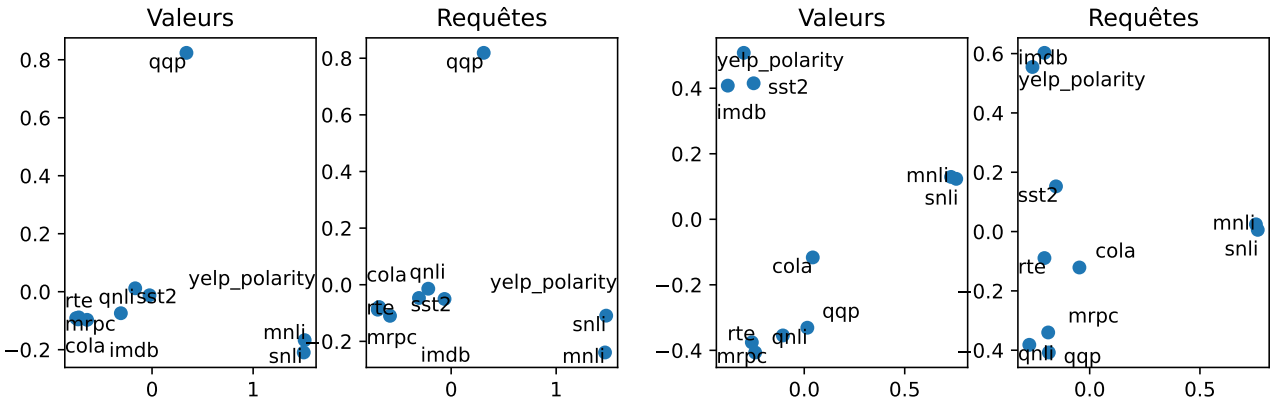
En effet, sachant cette égalité, pour calculer $d_G(B_1A_1, B_2A_2)$ nous devons dans un premier temps trouver, W_1 et W_2, des matrices dont les colonnes sont une base orthonormée de respectivement $Im(B_1A_1)$ et $Im(B_2A_2)$ (*c.f.* équation 1). Or d'après l'égalité ensembliste précédente trouver une base orthonormée de $Im(B_1A_1)$ (*resp.* $Im(B_2A_2)$) est équivalent à trouver une base orthonormée de $Im(B_1)$ (*resp.* $Im(B_2)$), montrant finalement ainsi que :

$$d_g(B_1A_1, B_2A_2) = d_G(B_1, B_2)$$

De cette égalité ainsi que de la proximité des résultats entre les distances du cosinus et de Grassmann, nous en déduisons que c'est principalement la matrice B qui encode des informations relatives aux tâches et non la matrice A dans les décompositions de faible rang. Afin de supporter ceci nous réalisons à nouveau l'opération de regroupement des modèles *via* la distance du cosinus, en séparant cette fois-ci l'analyse entre les matrices A et B, nous reportons les résultats sur la figure 4. Sur cette figure, nous remarquons, effectivement, que le regroupement effectué simplement avec les matrices B est relativement proche de celui observé avec le produit complet sur la figure 3b, tandis que celui observé sur les matrices A semble relativement mauvais pour l'association des différentes tâches. Appuyant ainsi empiriquement notre argument théorique.

Ce dernier point semble intéressant pour l'étude des vecteurs de tâches. En effet, une des difficultés de l'étude de ces vecteurs, est la dimension de ces derniers qui est très grande. Le fait de pouvoir supprimer une matrice réduit considérablement le nombre de paramètres à prendre en compte et réduit ainsi fortement la dimensionnalité. La réduction de dimension des vecteurs de tâches semble un point clé pour l'étude de ces objets. De plus en plus d'études apparaissent pour réduire le nombre de paramètres à mettre à jour dans les décompositions de faible rang (Kopiczko *et al.*, 2023; Sun *et al.*, 2022), permettant ainsi de venir diminuer fortement la dimension du vecteur de tâche. Une réduction de la dimension impliquant nécessairement une meilleure compréhension empirique de l'objet, et donc possiblement une meilleure combinaison de vecteurs de tâches *a posteriori*.

9. vérification empirique



(a) A (b) B

FIGURE 4 – Regroupement *via* la distance du cosinus en séparant l'analyse entre les matrices A et B