Estimation de l'inclusion entre tâches par projection spectrale de vecteurs de tâches

Loïc Fosse[1, 2] Benoît Favre[2] Frédéric Béchet[2]
Géraldine Damnati[1] Gwénolé Lecorvé[1]
(1) Orange Research, Lannion, France
(2) Aix Marseille Université, CNRS, LIS, Marseille, France
{loic.fosse, geraldine.damnati, gwenole.lecorve}@orange.com
{benoit.favre, frederic.bechet}@lis-lab.fr

RÉSUMÉ

L'affinage des modèles a permis la plupart des avancées significatives récentes dans les tâches de TALN. Des études ont exploré les raisons de ces succès en étudiant le mécanisme d'attention, la manière dont les connaissances linguistiques et factuelles sont encodées, *etc*... . Il est cependant difficile d'interpréter les changements causés par l'affinage dans les poids des modèles. Pour mieux comprendre cela, nous proposons une méthode fondée théoriquement pour projeter et comparer les changements de poids (*i.e.* vecteurs de tâches) dans un espace à faible dimension. Cette approche permet de mieux comprendre les connaissances encodées dans un vecteur de tâches, relativement à un autre vecteur de tâche. Nous validons notre méthode en montrant qu'un modèle affiné sur une tâche de résumé encode des informations sur la reconnaissance d'entités nommées.

ABSTRACT

Estimating Task Inclusion Through Spectral Projection of Task Vectors After Fine-Tuning

Fine-tuning pretrained models has enabled most of the recent significant advances on natural language processing tasks. Extensive studies have explored the reasons of these successes by studying the attention mechanism, how linguistic and factual knowledges are encoded, *etc*.). Other studies have also shown that fine-tuning primarily affects the upper layers of models. However, it is still unclear how these pretrained models are modified through the fine-tuning process, i.e., what are the specific changes in the model weights during this process. To better understand this, we propose a procedure with theoretical guarantees to project and compare weight shifts (called *task vectors*) in a low-dimensional space. This approach provides a better understanding of the knowledge encoded in one task vector, relative to another task vector. We validate this approach by showing that a model fine-tuned on a summarization task encodes information about named entity recognition.

MOTS-CLÉS : vecteurs de tâches, spectres, inclusion, entités nommées, résumé.

KEYWORDS: task vectors, spectrum, inclusion, named entities, summary.

1 Introduction

Les *transformers* (Vaswani *et al.*, 2017) ont permis des avancées considérables en traitement automatique du langage naturel (TALN), principalement en raison de la puissance du mécanisme d'attention (Pérez *et al.*, 2021, 2019) qui se trouve au cœur de ces architectures. Ce succès vient

également du pré-entraînement de ces modèles sur des tâches génériques de modélisation du langage comme le langage masqué (Devlin *et al.*, 2018) ou la prédiction du prochain *token* (Radford *et al.*, 2019), ce pré-entraînement pouvant également être suivi, dans certains cas, d'une phase dite d'instruction (Aw *et al.*, 2023) dans les modèles génératifs utilisés aujourd'hui. Ces modèles pré-entraînés peuvent ensuite être adaptés à diverses tâches, par **affinage** sur un corpus annoté. Cette combinaison du pré-entraînement ainsi que de l'affinage, permet d'obtenir les résultats les plus compétitifs : le pré-entraînement permet au modèle d'encoder des connaissances linguistiques et factuelles générales qui facilitent ensuite une meilleure convergence de l'affinage. Cependant, malgré ce principe général, plusieurs questions critiques demeurent. Comment prévenir les effets d'oublis catastrophiques se produisant lors de l'affinage sur une tâche (Li *et al.*, 2024) ? Comment fusionner et/ou démêler les connaissances acquises au cours des différentes phases d'entraînement ? En outre, comment pouvons-nous intégrer des connaissances liées à diverses tâches dans un modèle, en limitant les effets d'interférences entre ces dernières (dans le but de limiter les coûts d'apprentissage) (Ben-David & Schuller, 2003) ? Répondre à ces questions nécessite de comprendre comment sont encodées les connaissances liées à une tâche, dans un modèle. Dans ce contexte, plusieurs études se sont concentrées sur l'évolution des états cachés (ou activations), donnés par les modèles après un affinage. Le résultat principal étant que les couches peu profondes des modèles restent globalement inchangées, tandis que les dernières couches se concentrent davantage sur la tâche sur laquelle le modèle est affiné. Cependant, très peu d'études se concentrent sur l'évolution des paramètres des modèles suite à l'opération d'affinage. L'objectif est de mieux comprendre comment les connaissances liées à une tâche sont encodées dans les poids d'un modèle. Cette étude vise à présenter une nouvelle méthode, fondée théoriquement, qui projette et compare les modifications des poids des modèles apportées par l'affinage (aussi appelées **vecteurs de tâches**) dans un espace à faible dimension. Cette approche va au-delà des observations de similarités entre les vecteurs de tâches (Fosse, 2024), permettant une compréhension plus nuancée des connaissances partagées en mesurant une notion d'**inclusion** entre les tâches au niveau des paramètres des modèles. Les contributions sont les suivantes :

1. **Une méthode de projection**. Une nouvelle méthode de projection et d'analyse des vecteurs de tâches est introduite. Cette méthode présente des garanties théoriques et nous permet de considérablement réduire la dimension des objets étudiés (Bellman, 1966).
2. **Un cadre qui dépasse la similarité**. Une notion d'inclusion entre les tâches est introduite, relativement à un modèle, répondant à la question suivante : Est-ce que un modèle affiné sur une tâche encode des informations sur une autre tâche ?

2 Travaux connexes

Comme mentionné dans l'introduction, l'impact de l'affinage d'un modèle pré-entraîné à une tâche spécifique, au-delà de sa performance sur des ensembles de données de référence, reste mal compris. La sélection de la méthode d'adaptation et l'évaluation de ses effets sur les autres capacités du modèle sont largement basées sur des observations empiriques.

Pour répondre à ces questions, l'une des principales approches proposées est le *probing*, consistant à tester les modèles afin d'évaluer leurs connaissances liées à des tâches spécifiques (souvent linguistiques) (Rogers *et al.*, 2021; Vulić *et al.*, 2020; Zhao *et al.*, 2024; Wallat *et al.*, 2023; Guillaume & Yoshua, 2017; Nikolaev & Padó, 2023), en se concentrant principalement sur les états cachés (ou activations) des modèles. Initialement conçue pour évaluer les connaissances des modèles pré-entraînées, de nombreuses études se sont concentrées sur l'utilisation de cette méthode pour comprendre les

effet de l'affinage sur les modèles de langue de type encodeurs (Durrani *et al.*, 2021; Merchant *et al.*, 2020; Mosbach *et al.*, 2020). Le principal résultat étant que les couches peu profondes des modèles se voient peu modifiées par l'affinage tandis que les dernières couches se spécialisent plus fortement dans la tâche. Un résultat qui est notamment confirmé dans (Durrani *et al.*, 2022) *via* une méthode de *clustering* indépendante du *probing*.

Néanmoins, bien que le *probing* propose un cadre intuitif avec des résultats intéressants, Kunz & Kuhlmann (2020) et, plus formellement, Pimentel *et al.* (2020) ont souligné que son interprétation est délicate, menant dans certains cas à des résultats erronés. Un point qui est également mis en avant dans (Waldis *et al.*, 2024) et repris théoriquement dans (Fosse *et al.*, 2025). Par ailleurs, Darrin *et al.* (2024) ont introduit un cadre théorique basé sur l'information mutuelle (Cover, 1999), qui généralise la notion de *probing* permettant de dépasser les limites théoriques de cette approche.

Parallèlement à l'étude des états cachés *via* le *probing*, certaines études se concentrent sur les modifications dans les poids des modèles suite à l'affinage de ces derniers. Ces modifications de poids, appelées **vecteurs de tâches** (Ilharco *et al.*, 2022), permettent d'obtenir une représentation de la tâche dans l'espace des paramètres d'un modèle. Les vecteurs de tâches ont été utilisés pour combiner les propriétés des modèles par des opérations arithmétiques (Zhou *et al.*, 2024; Zeng *et al.*, 2025; Ortiz-Jimenez *et al.*, 2024; Jin *et al.*, 2024; Zhang *et al.*, 2023) ou pour supprimer les composants indésirables, tels que la toxicité, les informations personnelles ou les biais, d'un modèle de langue (Gao *et al.*, 2024; Zhang *et al.*, 2023; Liu *et al.*, 2024). Plus récemment, Fosse (2024) montre que ces objets peuvent être utilisés pour comparer sémantiquement des tâches : deux tâches proches vont produire deux vecteurs de tâches proches.

Cette étude propose une nouvelle approche pour la comparaison et l'analyse des vecteurs de tâches. Notre objectif étant de dépasser le cadre de la similarité entre ces objets et de comprendre si un vecteur de tâche encode des informations relatives à une autre tâche [1].

3 Préambule : vecteurs de tâches et LoRA

Vecteurs de tâches. Les vecteurs de tâches (Ilharco *et al.*, 2022) font désormais partie intégrante du paysage du TALN (et de l'apprentissage machine d'une manière générale). Ils ont d'abord été définis dans le cadre de l'affinage, où un modèle pré-entraîné est affiné sur différentes tâches, une tâche étant ici uniquement considérée comme une distribution jointe entre les données d'entrée X (texte, images, parole) et les données de sortie Y (classes, texte, *etc* ...) (Maurer *et al.*, 2016). Dans ce cas, le vecteur de tâche a été défini comme suit :

Soit $W_0 \in \mathbb{R}^{m \times n}$, les poids du modèle pré-entraîné, et W_t les poids de ce même modèle après affinage sur une tâche t. Le vecteur de tâche (pour la tâche t) est défini comme :

$$\tau_t \triangleq W_t - W_0. \tag{1}$$

Il est important de noter que Achille & Soatto (2018) ont montré que les vecteurs de tâches, sous certaines hypothèses, sont des statistiques suffisantes d'une tâche au sens de Fisher (Keener, 2010, Définition 3.2 p.43).

1. Il est important de noter que la comparaison de deux vecteurs de tâches, se fait à modèle équivalent *i.e.* les vecteurs sont obtenus à partir du même modèle pré-entraîné *c.f.* définitions suivantes.

Lien avec les adaptations de rang faible (LoRA). Les adaptations de rang faible (ou LoRA) (Hu *et al.*, 2021) sont devenues un standard pour affiner les modèles, que ce soit dans un contexte de TALN (Aleem *et al.*, 2024; Dettmers *et al.*, 2024; Kaddour *et al.*, 2023) ou de vision assistée par ordinateur (Luo *et al.*, 2023; Gandikota *et al.*, 2023). Il s'agit essentiellement de modifier l'opération d'affinage pour lui donner une dimension intrinsèque plus faible, de manière à modifier le moins possible le modèle pré-entraîné afin d'éviter les effets dits d'oublis catastrophiques (Aleixo *et al.*, 2023; Ren *et al.*, 2024). L'affinage d'un modèle consiste essentiellement à modifier les poids du modèle pré-entraîné W_0 en ajoutant un déplacement ΔW_t, qui dépend de la tâche t [2]. Le modèle résultant étant donné par :

$$W_t = W_0 + \Delta W_t.$$

A partir de cela et de la définition de τ_t, nous pouvons en déduire que ΔW_t est exactement τ_t *i.e.* le vecteur de la tâche t. La spécificité du LoRA est d'ajouter une contrainte de faible rang pour ΔW_t, *via* un nouvel hyper-paramètre $r << min(m, n)$ (le rang) et en décomposant ΔW_t en deux nouvelles matrices $A \in \mathbb{R}^{r \times n}$ et $B \in \mathbb{R}^{m \times r}$, telles que $\Delta W_t = B \times A$.

A et B sont alors les nouveaux paramètres à optimiser pendant l'affinage. Ainsi, lors de l'affinage d'un modèle avec LoRA sur une tâche t, nous avons, $\tau_t = B \times A$.

Cette définition permet de considérer le LoRA comme une estimation de faible rang du vecteur de tâche. Si un affinage classique est effectué, le vecteur de tâche τ_t possède alors une dimension importante qui est exactement le nombre de paramètres dans le modèle [3], le rendant difficile à étudier en raison de la malédiction des grandes dimensions (Bellman, 1966). Cette réduction de la taille, combinée aux résultats de (Ortiz-Jimenez *et al.*, 2024; Jin *et al.*, 2024; Jacot *et al.*, 2018), peut notamment expliquer pourquoi les combinaisons de vecteurs de tâches estimés à partir de LoRA conduisent à de meilleurs résultats (Huang *et al.*, 2023; Fosse, 2024; Zhang *et al.*, 2023).

Un point de clarification est nécessaire quant à l'utilisation du LoRA. Le LoRA n'est pas calculé une seule fois sur l'ensemble des poids du modèle, mais sur certaines couches linéaires (paramétrées par des matrices), tandis que les autres couches ne se voient pas modifiées. Un couple (A, B) est donc défini pour chaque couche linéaire modifiée. Dans le cas de l'architecture des *transformers* (Vaswani *et al.*, 2017), une pratique courante consiste à effectuer ces adaptations sur les matrices de projection de requêtes et de valeurs dans les couches d'attention (Ghojogh & Ghodsi, 2020). Selon les définitions précédentes, il n'y donc pas un seul vecteur de tâche, mais plusieurs vecteurs de tâche, un pour chaque matrice du modèle modifiée.

4 Présentation de la méthode

Cette section vise à présenter la méthode de projection des vecteurs de tâches. Soient τ_1 et τ_2 deux vecteurs de tâches. Il existe deux matrices orthonormales U, V et une matrice diagonale à valeurs positives S_1, telles que :

$$\tau_1 = US_1V^h \,. \tag{2}$$

Cette décomposition correspond à la décomposition en valeurs singulières (SVD) (Stewart, 1993) du vecteur de tâche τ_1. Les matrices orthonormées correspondent aux axes principaux du vecteur de tâche τ_1. Les valeurs dans S_1 (également appelé **spectre**) indiquent l'importance attachée à chaque

2. Le déplacement étant obtenu généralement par descente de gradient
3. Si un encodeur de type BERT (Devlin *et al.*, 2018) est utilisé ce vecteur possède alors une dimension de 110M

axe défini par les colonnes de U et V. En d'autres termes, cette décomposition donne l'espace de la tâche 1 (U et V) et la manière dont cet espace est utilisé (S_1). L'objectif ici, est de comprendre si l'espace utilisé pour la tâche 1 peut également être comparé à l'espace pour la tâche 2, et donc, si le vecteur de tâche créé pour la tâche 1 encode des informations liées à la tâche 2, conduisant à une forme d'**inclusion** de la tâche 2 dans la tâche 1. Nous souhaitons aller au-delà des mesures existantes de comparaisons des vecteurs de tâches (Fosse, 2024) qui se contentent d'évaluer la similarité, en introduisant une compréhension hiérarchique des relations entre les vecteurs de tâches. Inspirée des travaux de (Gao *et al.*, 2024), la méthode proposée consiste à utiliser l'espace SVD de τ_1 et à projeter τ_2 dans cet espace. La projection consiste à trouver S_2 comme la solution du problème d'optimisation suivant :

$$S_2 = \arg\min_{S \in \mathcal{S}} \|USV^h - \tau_2\|_F \,, \tag{3}$$

avec $\|A\|_F = \text{tr}(A \times A^h)$ la norme de Frobenius, une norme classique sur l'espace des matrices, et $\mathcal{S}$ étant l'espace des matrices diagonales à valeurs positives. $\mathcal{S}$ est introduit car il est préférable que S_2 ait les propriétés d'un spectre (diagonale à valeurs positives), de manière à être comparable avec S_1 (le coœur de la méthode étant la comparaison de S_1 avec S_2). Grâce à la convexité de la norme de Frobenius ainsi que de $\mathcal{S}$, ce problème d'optimisation est convexe et peut être résolu (de manière optimale) par descente de gradient (ici *via* l'algorithme Adam (Kingma & Ba, 2014)). Une fois la solution obtenue, nous comparons alors S_1 à S_2. Si ces deux vecteurs [4] sont proches, cela signifie que la projection de τ_2 sur l'espace de τ_1 utilise l'espace de la tâche 1 d'une manière très similaire à τ_1, ce qui est interprété comme le fait que τ_1 utilise certaines informations sur τ_2. Il sera alors dit que la tâche 2 est **incluse** dans la tâche 1. Nous mesurerons la proximité *via* les métriques suivantes :

— $L_p(S_1, S_2) = \|S_1 - S_2\|_p = \left(\sum_i |S_1(ii) - S_2(ii)|^p\right)^{1/p}$ avec $p \geqslant 1$

— $\text{REC}(S_1, S_2) = \sum_i \min\left(\frac{S_1(ii)}{\sum S_1}, \frac{S_2(ii)}{\sum S_2}\right)$.

La métrique L_p est la norme p classique et est sensible aux valeurs extrêmes des composantes. La métrique REC (recouvrement) s'inspire du taux d'erreur bayésien (Kaplan & Depaoli, 2013) entre deux mesures de probabilités discrètes. Elle est utilisée pour évaluer le recouvrement entre des mesures de probabilités. Ainsi, elle quantifie le recouvrement entre S_1 et S_2. Il a notamment été montré (Tsybakov, 2009; Gibbs & Su, 2002) que cette métrique est liée à la métrique L_1 entre la version normalisée de S_1 et S_2, et prend ses valeurs entre 0 (faible recouvrement) et 1 (fort recouvrement). Nous pouvons la reformuler comme suit :

$$\text{REC}\,(S_1, S_2) = 1 - \frac{1}{2} L_1 \left(\frac{S_1}{\sum S_1}, \frac{S_2}{\sum S_2}\right) \,,$$

donnant ainsi une métrique moins sensible aux valeurs extrêmes.

Remarque. La figure 1 résume la méthode de projection. Dans la suite, la projection de τ_2 sur l'espace de τ_1 sera notée $\tau_1 \to \tau_2$, car nous partons de la tâche 1 et nous essayons de projeter la tâche 2 (de 1 vers 2). Un résultat supplémentaire peut également être soulevé. Comme mentionné dans la section 1, comparer des vecteurs de tâche peut être difficile étant donné la dimension de ces objets. Cependant, quand τ_1 est de rang r (ce qui est le cas ici grâce à la méthode LoRA), il y a alors exactement r composantes non nulles dans S_1 (Stewart, 1993). Il semble alors logique de forcer S_2 à aussi avoir r composantes non nulles, signifiant que la comparaison de S_1 et S_2, se limite à la comparaison de deux vecteurs de dimension r.

4. Ces matrices sont considérées comme des vecteurs étant donné qu'elles sont diagonales à valeurs positives (le vecteur correspondant simplement à la diagonale).

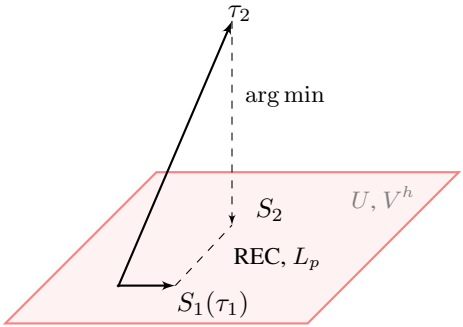


FIGURE 1 – Illustration de la projection de τ_2 sur l'espace de τ_1. Nous notons ceci $\tau_1 \rightarrow \tau_2$, car nous partons de l'espace de τ_1 et nous projetons τ_2 sur cet espace.

5 Protocole expérimental

Jeux de données et tâches. Comme indiqué dans (Zamir *et al.*, 2018), lors de la comparaison de modèles affinés sur différentes tâches, il est important que les conditions d'affinages soient similaires. Les tâches proposées seront donc définies sur le même corpus. Le corpus utilisé ici est constitué de 1544 documents issus de l'ensemble de données OntoNotes (Pradhan & Xue, 2009). Ces documents sont des articles de presse en anglais d'une longueur moyenne de 377 mots. Ils sont accompagnés de multiples annotations linguistiques : entités nommées, syntaxe, sémantique et coréférences. Les tâches traitées sont les suivantes :

— **Résumé / compression (SUM)**, où chaque document est censé être reformulé en un résumé dont la longueur est égale à 10% de la taille du document source. Le résumé de référence a été généré à l'aide de GPT-3.5 en lui imposant la contrainte de longueur (Brown *et al.*, 2020) (*c.f.* Annexe D).

— **Reconnaissance d'entités nommées (NER)**, où les entités nommées sont issues de l'annotation présente dans le corpus.

— **Copie des 10% des première phrases (10% SENT)** du document source. Le rôle de cette tâche, est d'agir comme une tâche de contrôle qui n'a *a priori* aucun lien avec les deux autres.

Le choix des tâches de résumé et de NER est justifié par le fait que, d'un point de vue linguistique, la recherche d'entités nommées dans un texte est une condition préalable au résumé (dans le cadre des articles de presse) : un résumé bien construit doit inclure les entités nommées les plus saillantes du document original. Il a notamment été montré que le fait de se concentrer sur les entités nommées présentes dans un résumé produit par un modèle de langage était un bon moyen d'évaluer les hallucinations de ce dernier (Akani *et al.*, 2023). L'objectif ici est de valider si la métrique d'estimation de l'inclusion de tâches s'aligne sur cette intuition linguistique. Au contraire, le choix de la tâche de copie permet d'avoir une tâche qui n'a pas de liens spécifiques avec les deux premières. Le choix de 10% est guidé par le fait que copier le début d'un document est une méthode classique pour produire un résumé à moindres coûts. L'ensemble de données est divisé en un ensemble d'entraînement, un ensemble de développement et un ensemble de test comprenant respectivement 1344, 100 et 100 documents.

Modèles et procédure d'affinage. Conformément au paradigme dominant du TALN contemporain, les tâches sont traitées en utilisant des modèles de langue génératifs. Par conséquent, les tâches sont soumises au modèle sous forme d'instructions et la réponse est censée suivre un format donné. Un

Entrée	What are the named entities present in the following document : ### Document : {Insérer le document} ### Answer :
Sortie	This document contains the following **number** : . . . and the following list of **person names** : . . . and the following list of **person types** : . . .

TABLE 1 – Formats d'entrée et de sortie pour la tâche NER

	Instruct				Base			
Tâches	R1	R2	RL	RLSum	R1	R2	RL	RLSum
SUM	55.94	33.12	48.71	48.85	56.75	32.77	49.64	49.62
NER	91.36	86.11	86.45	86.28	93.52	88.21	87.80	87.81
10% SENT	94.50	93.58	94.43	94.40	95.10	94.19	95.12	95.11

TABLE 2 – Performances ROUGE obtenues sur les différentes tâches

exemple pour la tâche NER est donné dans la table 1. Pour SUM et 10% SENT, la première phrase d'entrée est adaptée à la tâche, et la sortie est directement constituée du résumé, des entités ou des phrases sélectionnées. Le modèle utilisé ici est `Mistral-7B` (Alexandre *et al.*, 2023), dans sa version Base et Instruct (Jiang *et al.*, 2023) telle qu'elles sont disponibles sur le *hub* `HuggingFace` [5]. Le choix de ce modèle s'explique par le fait qu'il s'agit actuellement de l'un des meilleurs modèles de « taille raisonnable » permettant un affinage. `Mistral-7B` est composé de 32 couches d'attention causale et de 1 couche de plongement non-contextuelle. Dans les expériences présentées, tous les affinages sont effectués avec la même configuration : LoRA de rang 8 et un coefficient de régularisation α de 16 sur tous les blocs de requête et valeurs, pendant 6 époques en utilisant une taille de *batch* de 1 et un taux d'apprentissage constant de $4e-05$. Nous avons choisi LoRA pour affiner nos modèles pour les propriétés intéressantes que cette méthode propose. (Jang *et al.*, 2024) montrent, par exemple, que l'affinage d'un modèle avec la méthode LoRA permet une meilleure convergence lors de l'utilisation de la descente de gradient. En outre, LoRA nous permet d'imposer une contrainte de rang sur les vecteurs de tâches, tirant ainsi pleinement parti de la réduction de la dimension offerte par la méthode de projection. Le choix du rang a été guidé par les travaux de (Hu *et al.*, 2021) et (Fu *et al.*, 2023), qui indiquent que le rang, permet de contrôler les effets de sur-apprentissage [6], ce qui est important dans notre cas compte tenu du faible nombre de données disponibles pour l'entraînement. La table 2 rapporte les performances des modèles affinés pour chaque tâche. La performance est mesurée en termes de score ROUGE entre la sortie prédite et celle attendue (Lin, 2004). Sans surprise, le modèle entraîné sur la tâche de résumé obtient des scores plus faibles que celui sur la tâche de NER ou de recopie. Cependant, il n'y a pas de différences claires entre le modèle Base et le modèle Instruct, leurs performances sont comparables sur les différentes tâches.

5. Nous proposons dans l'Appendix F des expériences avec le modèle Llama-3 8B
6. Plus le rang est faible, plus les effets de sur-apprentissage s'estompent.

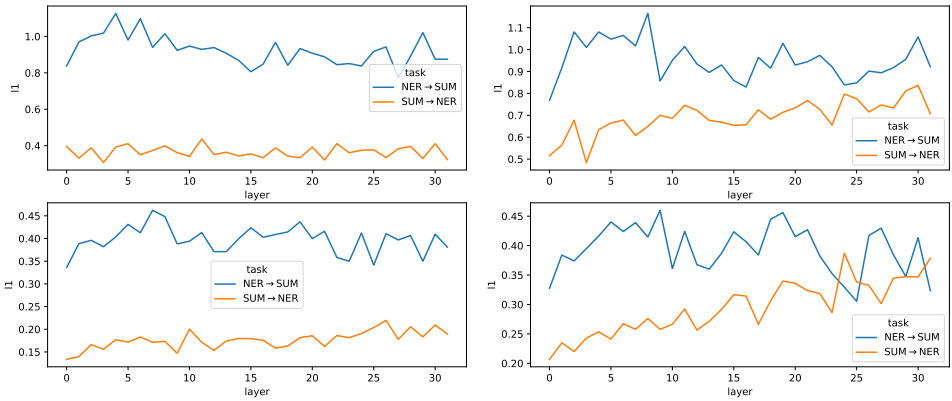
6 Résultats

Cette section présente les résultats obtenus. Pour rappel, les modèles sont affinés avec la méthode LoRA. Comme expliqué précédemment, nous disposons ainsi pour chaque affinage d'un ensemble important de couples (A, B) : un pour chaque couche de projection en requêtes et valeurs. Les métriques L_p et REC associées au calcul de $\tau_1 \rightarrow \tau_2$ peuvent ainsi être calculées pour chacune de ces couches de projection. Les résultats présenteront soit la moyenne de ces métriques, soit leur évolution à travers les couches des modèles. Tout d'abord, les résultats obtenus sur les tâches NER et SUM sont présentés, avant d'introduire la tâche de contrôle de recopie.

NER *v.s.* **SUM.** Nous présentons dans un premier temps l'évolution des métriques de proximité à travers les couches des modèles en séparant l'analyse entre les blocs de requêtes et de valeurs. La figure 2 montre cette évolution pour la métrique L_1. Une première observation claire, entre les couches et pour les deux types de modèles, est que la métrique L_1 est plus faible pour SUM→NER (ligne orange), que pour NER→SUM (ligne bleue). Cela signifie que dans l'espace de la tâche SUM, les tâches NER et SUM utilisent l'espace de manière plus similaire que dans l'espace de la tâche NER. Ce comportement peut s'expliquer par le fait que la tâche NER est plus "spécifique" que la tâche SUM. Lors de l'affinage, le vecteur de la tâche NER sera donc très biaisé / orienté vers cette tâche bien précise. Ainsi, lors du calcul de la projection NER→SUM (que l'on comprend comme la projection de SUM sur l'espace de NER), le vecteur S_1 (associé à la tâche NER) est alors un vecteur très parcimonieux qui n'active que certaines composantes de l'espace, bien spécifiques à la tâche de NER. La tâche SUM n'a, *a priori*, aucune raison de se concentrer sur ces mêmes axes. Ainsi, le vecteur S_2 associé à NER→SUM peut être moins parcimonieux, activant ainsi plus de composantes de l'espace. D'autre part, la tâche SUM est une tâche plus générique, elle peut conduire à un vecteur S_1 plus uniforme lors du calcul de SUM→NER. Il est donc possible que la compétence en matière de NER requise pour la tâche SUM ne soit pas codée sur un seul axe, mais qu'elle soit la composition de plusieurs axes de l'espace de la tâche SUM. Ainsi, lors de la projection du NER sur l'espace SUM, il se répartira sur les différents axes, ce qui conduira à nouveau à un vecteur S_2 uniforme.

Une analyse de l'entropie Shannon normalisée (Shannon, 1948) des différentes projections nous permet de fournir des éléments renforçant ces hypothèses. Pour rappel, si u est un vecteur à n composantes positives, l'entropie de Shannon normalisée du vecteur est donnée par $H(u) = \frac{-1}{\log(n)} \sum_i \frac{u_i}{\sum_k u_k} \log\left(\frac{u_i}{\sum_k u_k}\right)$ et est une quantité comprise entre 0 (distribution de Dirac *i.e* une seule composante non nulle) et 1 (distribution uniforme). La table 3 reporte les valeurs d'entropie des spectres. L'évolution de l'entropie y est conforme à l'explication précédente. La projection SUM→NER, conduit à un vecteur S_1 à haute entropie (la tâche de résumé utilise donc beaucoup d'axes dans son espace), tandis que la projection NER→SUM conduit à un vecteur S_1 à faible entropie (focalisation sur des axes spécifiques). Par ailleurs, peu importe le sens de projection, l'entropie du vecteur S_2 y est haute. Dans la suite, il sera dit que l'espace induit par le vecteur de tâche SUM est plus **diffus** que celui engendré par le vecteur de tâche NER.

Une observation supplémentaire peut être faite sur la figure 2 lors de l'analyse de l'évolution des métriques à travers les couches. Pour le modèle Base (*c.f.* figure 2b) et en particulier pour SUM→NER, la métrique L_1 augmente. Cela signifie que, dans l'espace de la tâche SUM, le vecteur de tâche SUM et le et la projection de NER activeront moins de composantes communes au fur et à mesure que l'on avance dans les couches du modèle. Comme expliqué dans la section 2, lors de l'affinage des modèles, les dernières couches seront davantage biaisée sur les sorties attendues des tâches, (ici les formats de génération) et les premières couches encoderont des connaissances plus génériques. Étant



(a) Instruct (b) Base

FIGURE 2 – Évolution de la métrique L_1 à travers les différentes couches, pour les requêtes (en haut), ainsi que pour les valeurs (en bas)

	Instruct		Base	
	$H(S_1)$	$H(S_2)$	$H(S_1)$	$H(S_2)$
SUM→NER	0.852	0.839	0.868	0.841
NER→SUM	0.774	0.867	0.779	0.872

TABLE 3 – Entropie de Shannon normalisée moyenne des projections entre les différents modules LoRA pour les différentes projections des vecteurs de tâches utilisés.

donné que le résultat du NER consiste uniquement en une liste d'entités nommées alors que le résumé consiste à générer un texte continue, il y a donc bien une différence de format pouvant ainsi expliquer cette augmentation des métriques. Le modèle Instruct, cependant, ne semble pas être affecté par cette variation de la forme de sortie dans les couches finales (figure 2a). Par ailleurs, la différence entre SUM→NER et NER→SUM est beaucoup plus nette dans le cas du modèle Instruct. Il semblerait que le modèle Instruct sépare les différentes tâches (ici les espaces) plus clairement selon cette méthode de projection.

Comparaison avec la tâche de contrôle. Afin de poursuivre l'analyse entre les tâches NER et SUM, le vecteur de tâche 10% SENT est ajouté. Pour simplifier, l'évolution des métriques à travers les couches n'est pas rapportée ici. L'accent est mis sur les mesures moyennes pour tous les couples (A, B) issus de l'affinage LoRA. Les résultats des métriques sont reportés dans la table 4. Dans cette table, les tâches sont triées en fonction de la métrique L_1 du modèle Instruct. Tout d'abord, pour les deux modèles (Instruct et base), dans l'espace de la tâche SUM (deux premières lignes), le vecteur de la tâche NER a plus de composantes communes avec le vecteur de tâche SUM (L_1 plus bas, et REC plus élevé). Afin de valider cette observation, des tests de Student (Student, 1908) sont réalisés. Les probabilités critiques (ou *p-values*) des tests sont reportées dans la table 5. Les probabilités critiques y sont inférieures au seuil classique de significativité de 5% à l'exception d'un test qui est légèrement supérieur à ce seuil, mais toujours inférieur au seuil de 10%. Cela confirme les différences observées sur les métriques moyennes. Cette observation est conforme à l'intuition

	Instruct		Base	
	L_1	REC	L_1	REC
SUM→NER	0.271	0.644	0.493	0.667
SUM→10% SENT	0.277	0.608	0.501	0.602
10% SENT→NER	0.435	0.599	0.459	0.615
10% SENT→SUM	0.440	0.562	0.461	0.580
NER→10% SENT	0.660	0.519	0.676	0.557
NER→SUM	0.660	0.535	0.673	0.582

TABLE 4 – Moyenne des métriques L_1 et REC pour les différentes projections.

	$\mathcal{H}_1$	Instruct	Base
Bi-latéral	$\text{REC}\,(\text{SUM} \to \text{NER}) \neq \text{REC}\,(\text{SUM} \to 10\%\ \text{SENT})$	7.558e-02	1.043e-03
	$L_1\,(\text{SUM} \to \text{NER}) \neq \text{REC}\,(\text{SUM} \to 10\%\ \text{SENT})$	1.440e-05	1.167e-06
Uni-latéral	$\text{REC}\,(\text{SUM} \to \text{NER}) \geqslant \text{REC}\,(\text{SUM} \to 10\%\ \text{SENT})$	3.779e-02	5.214e-05
	$L_1\,(\text{SUM} \to \text{NER}) \leqslant \text{REC}\,(\text{SUM} \to 10\%\ \text{SENT})$	7.200e-06	5.847e-07

TABLE 5 – Probabilités critiques (ou *p-values*) pour la comparaison entre la moyenne des mesures de similarité (REC et L_1). L'hypothèse nulle $\mathcal{H}_0$ étant l'égalité entre les métriques.

linguistique initiale concernant les tâches : le modèle entraîné sur la tâche SUM semble encoder des informations relatives au modèle entraîné sur la tâche NER. Cependant, pour le modèle Instruct, la tâche de copie est insérée entre la tâche SUM et la tâche NER. Une fois de plus, cela peut s'expliquer par l'argument de l'entropie, développé précédemment *i.e.* l'espace induit par la tâche de comptage est plus diffus que l'espace NER, mais pas autant que l'espace SUM. Nous avons vérifié cette affirmation en calculant l'entropie du vecteur S_1 pour 10%SENT sur le modèle Base $(0, 841)$ et le modèle Instruct $(0, 826)$. Cependant, il semble surprenant que l'espace pour cette tâche soit diffus, compte tenu du biais très fort présent dans cette tâche (la réponse est dans l'entrée). Il semble que le modèle ait appris plus que la compétence de comptage. Ce comportement est plus important pour le modèle Base, qui peut à peine distinguer la tâche de copie et la tâche de résumé (métrique L_1 très faible), montrant une fois de plus que le modèle Instruct semble produire des vecteurs de tâches plus démêlés (meilleure séparation des tâches).

7 Conclusion

Cette étude, présente une nouvelle manière de projeter et analyser les vecteurs de tâches allant au-delà de l'analyse des similarités. La méthode présentée, vise à extraire les connaissances partagées encodées dans les vecteurs de tâches dans le but d'établir des relations d'inclusion entre ces derniers. Il est notamment montré qu'un vecteur de tâche produit pour la tâche de résumé encode des informations sur la reconnaissance d'entités nommées, ce qui est conforme aux travaux précédents et à notre première intuition. La méthode a été utilisée dans le contexte de l'inclusion de tâches, mais sa définition n'est pas spécifique à cela et peut largement être utilisée dans d'autres applications. La décomposition SVD est un outil puissant largement utilisé dans le domaine, impliquant que la méthode présentée peut être utilisée dans diverses applications autour des vecteurs de tâches.

Références

ACHILLE A. & SOATTO S. (2018). Emergence of Invariance and Disentanglement in Deep Representations. *Journal of Machine Learning Research*, **19**(50), 1–34.

AKANI E., FAVRE B., BECHET F. & GEMIGNANI R. (2023). Reducing named entity hallucination risk to ensure faithful summary generation. In *Proc. of the 16th International Natural Language Generation Conference*, p. 437–442.

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

ALEIXO E. L., COLONNA J. G., CRISTO M. & FERNANDES E. (2023). Catastrophic forgetting in deep learning : A comprehensive taxonomy. *arXiv preprint arXiv :2312.10549*.

ALEXANDRE S., ARTHUR M., CHRIS B., SINGH C. D., FLORIAN B., GIANNA L., GUILLAUME L., LUCILE S., RENARD L. L., MARIE-ANNE L. *et al.* (2023). Mistral 7b. *arXiv preprint arXiv : 2310.06825*.

AW K. L., MONTARIOL S., ALKHAMISSI B., SCHRIMPF M. & BOSSELUT A. (2023). Instruction-tuning aligns llms to the human brain. *arXiv preprint arXiv :2312.00575*.

BELLMAN R. (1966). Dynamic programming. *Science*, **153**(3731), 34–37.

BEN-DAVID S. & SCHULLER R. (2003). Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines : 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*, p. 567–580 : Springer.

BROWN T., MANN B., RYDER N., SUBBIAH M., KAPLAN J. D., DHARIWAL P., NEELAKANTAN A., SHYAM P., SASTRY G., ASKELL A. *et al.* (2020). Language models are few-shot learners. *Advances in neural information processing systems*, **33**, 1877–1901.

COVER T. M. (1999). *Elements of information theory*. John Wiley & Sons.

DARRIN M., FORMONT P., AYED I. B., CHEUNG J. C. & PIANTANIDA P. (2024). When is an embedding model more promising than another ? *arXiv preprint arXiv :2406.07640*.

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

DEVLIN J., CHANG M.-W., LEE K. & TOUTANOVA K. (2018). Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*.

DURRANI N., SAJJAD H. & DALVI F. (2021). How transfer learning impacts linguistic knowledge in deep nlp models ? *arXiv preprint arXiv :2105.15179*.

DURRANI N., SAJJAD H., DALVI F. & ALAM F. (2022). On the transformation of latent space in fine-tuned nlp models. *arXiv preprint arXiv :2210.12696*.

FOSSE L. (2024). Géométrie des vecteurs de tâches pour l'association et la combinaison de modèles. In *Actes de JEP-TALN-RECITAL 2024.*, p. 69–84 : ATALA & AFPC.

FOSSE L., BÉCHET F., FAVRE B., DAMNATI G., LECORVÉ G., DARRIN M., FORMONT P. & PIANTANIDA P. (2025). Statistical deficiency for task inclusion estimation. *arXiv preprint arXiv :2503.05491*.

FU Z., YANG H., SO A. M.-C., LAM W., BING L. & COLLIER N. (2023). On the effectiveness of parameter-efficient fine-tuning. In *Proc. of the AAAI conference on artificial intelligence*.

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*.

GAO L., NIU Y., TANG T., AVESTIMEHR S. & ANNAVARAM M. (2024). Ethos : Rectifying language models in orthogonal parameter space. *arXiv preprint arXiv :2403.08994*.

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

GIBBS A. L. & SU F. E. (2002). On choosing and bounding probability metrics. *International statistical review*, **70**(3), 419–435.

GRATTAFIORI A., DUBEY A., JAUHRI A., PANDEY A., KADIAN A., AL-DAHLE A., LETMAN A., MATHUR A., SCHELTEN A., VAUGHAN A. *et al.* (2024). The llama 3 herd of models. *arXiv preprint arXiv :2407.21783*.

GUILLAUME A. & YOSHUA B. (2017). Understanding intermediate layers using linear classifier probes. In *ICLR (Workshop)*.

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. *arXiv preprint arXiv :2106.09685*.

HUANG C., LIU Q., LIN B. Y., PANG T., DU C. & LIN M. (2023). Lorahub : Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv :2307.13269*.

ILHARCO G., RIBEIRO M. T., WORTSMAN M., GURURANGAN S., SCHMIDT L., HAJISHIRZI H. & FARHADI A. (2022). Editing models with task arithmetic. *arXiv preprint arXiv :2212.04089*.

JACOT A., GABRIEL F. & HONGLER C. (2018). Neural tangent kernel : Convergence and generalization in neural networks. *Advances in neural information processing systems*, **31**.

JANG U., LEE J. D. & RYU E. K. (2024). Lora training in the ntk regime has no spurious local minima. *arXiv preprint arXiv :2402.11867*.

JIANG A. Q., SABLAYROLLES A., MENSCH A., BAMFORD C., CHAPLOT D. S., CASAS D. D. L., BRESSAND F., LENGYEL G., LAMPLE G., SAULNIER L. *et al.* (2023). Mistral 7b. *arXiv preprint arXiv :2310.06825*.

JIN R., HOU B., XIAO J., SU W. & SHEN L. (2024). Fine-tuning linear layers only is a simple yet effective way for task arithmetic. *arXiv preprint arXiv :2407.07089*.

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*.

KAPLAN D. & DEPAOLI S. (2013). Bayesian statistical methods. *Oxford handbook of quantitative methods*, p. 407–437.

KEENER R. W. (2010). *Theoretical Statistics*. New York, NY : Springer New York, NY, 1st édition.

KINGMA D. P. & BA J. (2014). Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*.

KUNZ J. & KUHLMANN M. (2020). Classifier probes may just learn from linear context features. In *Proc. of the 28th International Conference on Computational Linguistics*, p. 5136–5146.

LI H., DING L., FANG M. & TAO D. (2024). Revisiting catastrophic forgetting in large language model tuning. In Y. AL-ONAIZAN, M. BANSAL & Y.-N. CHEN, Éds., *Findings of the Association for Computational Linguistics : EMNLP 2024*, p. 4297–4308, Miami, Florida, USA : Association for Computational Linguistics. DOI : 10.18653/v1/2024.findings-emnlp.249.

LIN C.-Y. (2004). Rouge : A package for automatic evaluation of summaries. In *Text summarization branches out*, p. 74–81.

LIU S., YAO Y., JIA J., CASPER S., BARACALDO N., HASE P., XU X., YAO Y., LI H., VARSHNEY K. R. *et al.* (2024). Rethinking machine unlearning for large language models. *arXiv preprint arXiv :2402.08787*.

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*.

MAURER A., PONTIL M. & ROMERA-PAREDES B. (2016). The benefit of multitask representation learning. *Journal of Machine Learning Research*, **17**(81), 1–32.

MERCHANT A., RAHIMTOROGHI E., PAVLICK E. & TENNEY I. (2020). What happens to bert embeddings during fine-tuning ? *arXiv preprint arXiv :2004.14448*.

MOSBACH M., KHOKHLOVA A., HEDDERICH M. A. & KLAKOW D. (2020). On the Interplay Between Fine-tuning and Sentence-Level Probing for Linguistic Knowledge in Pre-Trained Transformers. In A. ALISHAHI, Y. BELINKOV, G. CHRUPAŁA, D. HUPKES, Y. PINTER & H. SAJJAD, Éds., *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, p. 68–82, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.blackboxnlp-1.7.

NIKOLAEV D. & PADÓ S. (2023). Investigating semantic subspaces of transformer sentence embeddings through linear structural probing. In *6th BlackboxNLP Workshop : Analyzing and Interpreting Neural Networks for NLP*, p. 142–154 : Association for Computational Linguistics.

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**.

PÉREZ J., BARCELÓ P. & MARINKOVIC J. (2021). Attention is turing-complete. *Journal of Machine Learning Research*, **22**(75), 1–35.

PÉREZ J., MARINKOVIĆ J. & BARCELÓ P. (2019). On the turing completeness of modern neural network architectures. *arXiv preprint arXiv :1901.03429*.

PIMENTEL T., VALVODA J., MAUDSLAY R. H., ZMIGROD R., WILLIAMS A. & COTTERELL R. (2020). Information-theoretic probing for linguistic structure. *arXiv preprint arXiv :2004.03061*.

PRADHAN S. S. & XUE N. (2009). Ontonotes : the 90% solution. In *Proceedings of Human Language Technologies : NAACL 2009*, p. 11–12.

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

REN W., LI X., WANG L., ZHAO T. & QIN W. (2024). Analyzing and reducing catastrophic forgetting in parameter efficient tuning. *arXiv preprint arXiv :2402.18865*.

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.

SHANNON C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, **27**(3), 379–423.

STEWART G. W. (1993). On the early history of the singular value decomposition. *SIAM review*, **35**(4), 551–566.

STUDENT (1908). The probable error of a mean. *Biometrika*, p. 1–25.

TSYBAKOV A. B. (2009). *Introduction to Nonparametric Estimation*. Springer New York, NY.

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**.

VULIĆ I., PONTI E. M., LITSCHKO R., GLAVAŠ G. & KORHONEN A. (2020). Probing pretrained language models for lexical semantics. In *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 7222–7240.

WALDIS A., PERLITZ Y., CHOSHEN L., HOU Y. & GUREVYCH I. (2024). Holmes : Benchmark the linguistic competence of language models. *arXiv e-prints*, p. arXiv–2404.

WALLAT J., BERINGER F., ANAND A. & ANAND A. (2023). Probing bert for ranking abilities. In *European Conference on Information Retrieval*, p. 255–273 : Springer.

ZAMIR A. R., SAX A., SHEN W., GUIBAS L. J., MALIK J. & SAVARESE S. (2018). Taskonomy : Disentangling task transfer learning. In *Proc. of the IEEE conference on computer vision and pattern recognition*, p. 3712–3722.

ZENG S., HE Y., YOU W., HAO Y., TSAI Y.-H. H., YAMADA M. & ZHAO H. (2025). Efficient Model Editing with Task Vector Bases : A Theoretical Framework and Scalable Approach. DOI : 10.48550/arXiv.2502.01015.

ZHANG J., LIU J., HE J. *et al.* (2023). Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, **36**, 12589–12610.

ZHAO H., CHEN H., YANG F., LIU N., DENG H., CAI H., WANG S., YIN D. & DU M. (2024). Explainability for large language models : A survey. *ACM Transactions on Intelligent Systems and Technology*, **15**(2), 1–38.

ZHOU Y., SONG L., WANG B. & CHEN W. (2024). MetaGPT : Merging Large Language Models Using Model Exclusive Task Arithmetic. In Y. AL-ONAIZAN, M. BANSAL & Y.-N. CHEN, Éds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, p. 1711–1724, Miami, Florida, USA : Association for Computational Linguistics.

Tâche	λ	Instruct L_2	Instruct L_1	Instruct REC	Base L_2	Base L_1	Base REC
NER → SUM	0	0.346	0.660	0.538	0.347	0.673	0.576
	1e-6	0.346	0.661	0.551	0.347	0.674	0.601
	1e-5	0.346	0.661	0.547	0.347	0.673	0.595
	1e-4	0.346	0.661	0.556	0.348	0.673	0.563
	1e-3	0.346	0.661	0.563	0.347	0.674	0.598
	1e-2	0.347	0.664	0.543	0.349	0.678	0.556
	1e-1	0.349	0.671	0.497	0.352	0.687	0.524
	1	0.350	0.675	0.521	0.353	0.692	0.525
SUM → NER	0	0.129	0.272	0.649	0.224	0.493	0.667
	1e-6	0.129	0.272	0.635	0.225	0.494	0.662
	1e-5	0.129	0.271	0.665	0.225	0.494	0.664
	1e-4	0.129	0.272	0.642	0.224	0.493	0.656
	1e-3	0.129	0.273	0.637	0.225	0.494	0.666
	1e-2	0.131	0.278	0.597	0.227	0.500	0.638
	1e-1	0.136	0.289	0.533	0.232	0.512	0.568
	1	0.137	0.294	0.568	0.233	0.516	0.583

TABLE 6 – Évolution des métriques L_1, L_2 and REC en fonction de l'hyper-paramètre λ, décrit dans l'équation 3.

A Contrôle de la parcimonie

La méthode de projection présentée était initialement présentée sous la forme du problème d'optimisation suivant :

$$S_2^\lambda = \underset{S \in \mathcal{S}}{\arg\min} \, \|USV^h - \tau_2\|_F + \lambda \|S\|_1 \, , \tag{4}$$

Le terme $\lambda\|S\|_1$ est ajouté pour renforcer la parcimonie de S_2. Ce nouveau terme est accompagné d'un hyper-paramètre λ, qui contrôle l'intensité de la condition de parcimonie. Ceci est motivé par le fait que dans une décomposition SVD, certaines composantes sont plus importantes que d'autres, et nous cherchons à accentuer ce phénomène pour ne conserver que les composantes les plus importantes lors de la projection. Nous avons testé notre méthode entre les tâches NER et SUM avec différentes valeurs du paramètre λ. Nous présentons les résultats dans la table 6. Dans cette table, nous indiquons la valeur moyenne des métriques de recouvrement. Cette table montre qu'avec de grandes valeurs du paramètre λ (1), la projection de τ_2 sur l'espace τ_1 conduit à un vecteur plus éloigné du vecteur initial τ_1. Forcer la projection à être trop parcimonieuse revient à ne conserver que les composantes les plus importantes pour la tâche que l'on souhaite projeter. Chaque tâche étant différente, il semble naturel que les composantes les plus importantes soient différentes d'une tâche à l'autre. En forçant la parcimonie, nous supprimons la possibilité de visualiser les recouvrement qui pourraient avoir lieu au niveau des composantes « moins importantes », perdant ainsi certaines informations. Nous avons décidé d'opter pour des valeurs de λ plus faibles. Cependant, lorsque nous évaluons les mesures pour des valeurs λ relativement faibles, nous constatons très peu de différence (voire aucune) entre les mesures. Nous décidons donc de nous concentrer sur une valeur de $\lambda = 0$ pour le cœur de l'étude.

Rang	Instruct SUM	NER
8	48.71	86.11
16	48.85	90.13
32	47.03	91.07
64	44.26	89.44
128	47.61	90.73

TABLE 7 – Performances ROUGE obtenues pour différents rangs LoRA sur le modèle `Mistral-7B` Instruct

B Procédure d'entraînement

L'algorithme 1, résume la démarche suivie pour réaliser notre méthode de projection.

Algorithm 1 Procédure d'optimisation pour obtenir S_2^λ

Require: ϵ, l, N, τ_1, τ_2, λ
  $U, S_1, V^h \leftarrow SVD(\tau_1)$
  $S_2 \sim \mathcal{N}(0, 1)$
  $t \leftarrow 0$
  $\mathcal{L}_t \leftarrow \infty$
  while $t \leqslant N$ **do**
    $\mathcal{L}_{t+1} \leftarrow ||US_2V^h - \tau_2||_F + \lambda||S_2||_1$
    if $|\mathcal{L}_t - \mathcal{L}_{t+1}| \leqslant \epsilon$ **then**
      $t \leftarrow N$
    else
      $S_2 \leftarrow S_2 - l \times \nabla_S \mathcal{L}_{t+1}$
      $\mathcal{L}_t \leftarrow \mathcal{L}_{t+1}$
      $t++$
    end if
  end while

C Tests sur des rangs différents

Comme mentionné dans l'étude, nous nous sommes concentrés sur un rang de 8 pour l'affinage LoRA. La table 7 présente les résultats obtenus pour d'autres rangs que nous avons testés. Les variations de scores étant faibles, nous décidons de rester sur un rang de 8 dans un soucis d'optimisation.

D Exemples de données

La table 8 présente le prompt que nous avons utilisé pour générer les résumé *via* GPT 3.5.

Summarize in a short paragraph of less than <SIZE> words in English the following text representing a conversation / news report / news article.

TABLE 8 – Prompt utilisé pour générer les résumés avec GPT 3.5. <SIZE> est remplacé par 10% du nombre de mots dans le texte original. Les conversations, *news report*, ou encore articles sont sélectionnés en fonction du type de texte (l'ensemble de données Onto Notes contient ces trois types de texte).

Helen Boehm, the owner of an art porcelain company, casually name-drops her prestigious acquaintances including **Prince Charles**, **Princess Diana**, and **Sarah Ferguson** while mentioning her position on the board of the **Vatican Museum** in **Rome**. This behavior of name-dropping is prevalent in society, from **socialites** to **city officials**, all trying to establish connections and impress others. However, some individuals take it to extreme lengths, strategically planning which names to drop at social events. Name-dropping can have its benefits, such as boosting civic pride by linking a city to famous individuals who have passed through. Yet, there are risks involved, as unsubstantiated name-dropping can backfire and tarnish one's reputation. Ultimately, name-dropping is a complex social phenomenon that can open doors or lead to embarrassment, depending on how it is used.

TABLE 9 – Exemple d'un résumé généré automatiquement par GPT 3.5. Les entitées nommées sont mises en **gras** soulignant la forte présence de ces dernière dans le résumé.

Nous fournissons également dans la table 9, un exemple de résumé généré par le modèle GPT 3.5. Nous pouvons clairement voir dans ce résumé la présence de nombreuses entités nommées, ce qui confirme notre hypothèse sur les liens entre la reconnaissance des entités nommées et la tâche de résumé.

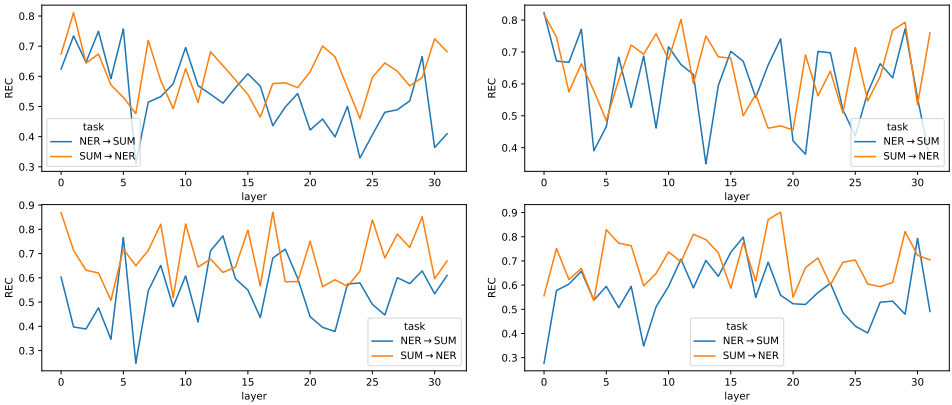
E Figures supplémentaires

Nous présentons également l'évolution de la métrique REC [7] à travers les couches et fournissons les résultats sur la figure 3. Sur cette figure, nous pouvons voir que l'effet de la normalisation du vecteur projeté (*c.f.* section 4) conduit à une variation plus chaotique de la métrique.

F Expériences complémentaires

Afin d'apporter des éléments supplémentaires aux résultats proposés dans la section 6, nous proposons ici de rejouer les expériences avec le modèle Llama3 8B (Grattafiori *et al.*, 2024) une fois de plus dans sa version Base et Instruct. La table 10 propose une réplique de la table 4 dans le cadre de ce nouveau modèle. La table 11 propose également une réplique de la table 5 concernant les tests de significativité des différences observées entre les métriques. Nous tirons globalement les mêmes conclusions en utilisant ce modèle confirmant ainsi les résultats énoncés dans la section 6.

7. Pour cette métrique, une valeur élevée signifie un recouvrement important contrairement à la métrique L_1



(a) Instruct (b) Base

FIGURE 3 – Évolution de la métrique REC à travers les différentes couches, pour les requêtes (en haut) et les valeurs (en bas).

	Instruct		Base	
	L_1	REC	L_1	REC
SUM→NER	0.464	0.589	0.464	0.581
SUM→10% SENT	0.474	0.502	0.474	0.525
10% SENT→NER	0.265	0.583	0.377	0.537
10% SENT→SUM	0.272	0.503	0.380	0.524
NER→10% SENT	0.777	0.452	0.783	0.411
NER→SUM	0.772	0.467	0.780	0.455

TABLE 10 – Réplique de la table 4 dans le cadre du modèle Llama 3 8B

	$\mathcal{H}_1$	Instruct	Base
Bi-latéral	$\text{REC}(\text{SUM} \to \text{NER}) \neq \text{REC}(\text{SUM} \to 10\%\ \text{SENT})$	9.836e-04	0.04
	$L_1(\text{SUM} \to \text{NER}) \neq \text{REC}(\text{SUM} \to 10\%\ \text{SENT})$	1.386e-07	1.648e-08
Uni-latéral	$\text{REC}(\text{SUM} \to \text{NER}) \geqslant \text{REC}(\text{SUM} \to 10\%\ \text{SENT})$	4.918e-04	0.02
	$L_1(\text{SUM} \to \text{NER}) \leqslant \text{REC}(\text{SUM} \to 10\%\ \text{SENT})$	6.928e-08	8.241e-09

TABLE 11 – Réplique de la table 5 dans le cadre du modèle Llama 3 8B