Procrustes Analysis for Improving Language Model Merging

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

Abstract

The availability of many fine-tuned neural language models for different tasks naturally leads to the question of whether it is worthwhile to combine them, particularly through parameter merging, which is the least resource-intensive option. Among the many existing methods, some focus on parameter alignment before actual merging. In this article, we propose a new method within this research area, based on Procrustes analysis. We evaluate this method for merging fine-tuned models for the same task, derived from the same encoder-based model. Considering nine tasks from the GLUE benchmark, three Named Entity Recognition tasks, and six reference merging methods, we show that our proposal can improve upon existing merging methods in most tested configurations.

Keywords: Neural language models, model merging, representation space alignment

1. Introduction

Neural language models, especially transformer-based ones (Vaswani et al., 2017), play a central role in current Natural Language Processing. However, their initial training and adaptation to a specific application context through fine-tuning have a high cost in computing resources. At the same time, an increasing number of these models are being released, typically on Hugging Face's model repository[1]. These models include both raw models and models fine-tuned for specific tasks. When faced with a specific application need, it is quite natural to evaluate whether it can be met by an existing model, or even by a combination of existing models. The most direct form of such a combination is model merging at the parameter level. This merging may involve highly comparable models to obtain a more robust model, in the manner of ensemble methods. However, it may also seek to combine complementary models, especially task-wise, to broaden the functional scope of the resulting model, which can then handle multiple tasks.

As illustrated by (Li et al., 2023) and (Yang et al., 2024), this field of research is particularly active, with notable applications for decoder models, as evidenced by the success of the MergeKit library (Goddard et al., 2024). Yang et al. (2024) categorize the existing methods into two main groups: methods that operate before merging and methods for the merging itself. In this article, we focus more specifically on the first type, though the reader can refer to section 3.1 for a presentation of several reference methods of the second type. Within the first category, Yang et al. (2024) distinguish between i) fine-tuning methods that optimize the ability to merge models by enabling them to coexist in the same representation space, ii) methods that address architectural differences between models, and iii) parameter alignment methods. We focus on this last group. Within this field, several approaches (Singh and Jaggi, 2020; Imfeld et al., 2024) have relied on optimal transport to align the parameters of multiple models. CCAMerge (Horoi et al., 2024) uses canonical correlation analysis, maximizing correlations between linear combinations of model parameters. Finally, Ainsworth et al. (2023) propose three algorithms for permuting model parameters to align one model with another: one based on matching activations, another on matching weights, and a third on learning an alignment.

In this context of parameter alignment methods across models, we focus on two main contributions:

- We propose a new method to project the parameters of different neural language models into the same space before merging.
- We apply this method to different state-of-the-art merging methods and evaluate them, focusing on the merging of models fine-tuned from the same base model for the same task, a configuration that has not always been evaluated systematically.

2. Method

The problem we address in this article is the merging of n neural language models $\{LM_1 \dots LM_n\}$, each obtained by fine-tuning the same initial pre-trained model LM on a given task. These models differ only in the random seed used during fine-tuning. Our objective is to merge these n models into a single model applicable to the target task. This setting is comparable to (Wortsman et al., 2022), with the key difference that we do not rely

[1] https://huggingface.co/models

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 9988–9998
11–16 May 2026.

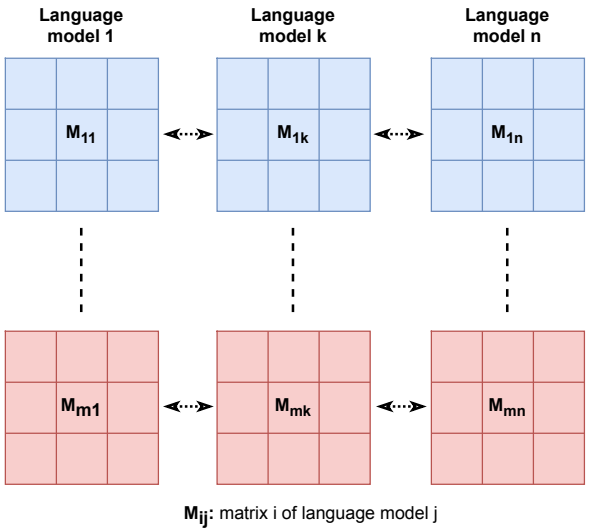

Figure 1: Alignment of language models.

on a validation set for greedy selection of a subset of the n models.

In addition to applying several standard model merging techniques to this setting, as reported in Section 3, we introduce a new approach that, prior to merging, projects the models into a shared representation space. More precisely, a language model LM_j is represented here by a collection of matrices $M_{*j} = \{M_{ij} \mid i \in \{1, \ldots, m\}\}$ corresponding to the different components of the transformer architecture: the attention blocks (keys, values, and queries), the feed-forward networks following the attention blocks, as well as the token embeddings. Within this framework, our aim is to project, for a fixed i, all matrices $M_{i*} = \{M_{ij} \mid j \in \{1, \ldots, n\}\}$ into a common representation space, thereby improving the effectiveness of any merging method applied to the models LM_j, in line with neural model weight alignment approaches discussed in Section 1.

To perform this projection, we draw on Procrustes analysis (Gower, 2010). This technique has previously been used to align representation spaces derived from static neural language models, particularly in multilingual alignment contexts (Kementchedjhieva et al., 2018; Taitelbaum et al., 2019), as well as for multiple comparable models (Caciularu et al., 2021). To our knowledge, however, it has not yet been applied to transformer-based language models.

In general, Procrustes analysis aligns one shape with another by applying three types of linear transformations: translations, uniform scalings, and rotations. Adopting a matrix formalism and assuming that M_{ref} is the reference matrix and W the matrix to be aligned with M_{ref}, the goal is to find the

Algorithm 1 Generalized Procrustes Analysis

Inputs: M_{i*}, a set of n matrices
Hyperparameters: max_iter, the maximum number of iterations; tol, the convergence criterion

Let P_{i*}, the set of projected initial n matrices
Let M_{ref}, the reference matrix for Orthogonal Procrustes Analysis (OPA)
Let opa_dist, the distance between the reference matrix and the mean of matrices after OPA
Let tol, the convergence criterion on opa_dist

$M_{ref} \leftarrow$ random choice among M_{i*}; l: index of M_{ref} within M_{i*}
for $j = 1, \ldots, n$, with $j \neq l$ **do**
  $P_{ij} \leftarrow$ OPA(M_{ref}, M_{ij})
end for
$P_{il} \leftarrow M_{ref}$
$M_{ref} \leftarrow$ mean(P_{i*}); $opa\_dist \leftarrow -\infty$
for $p = 1, \ldots, max\_iter$ **do**
  for $j = 1, \ldots, n$ **do**
    $P_{ij} \leftarrow$ OPA(M_{ref}, P_{ij})
  end for
  $M_{avg} \leftarrow$ mean(P_{i*})
  $dist\_fn \leftarrow$ frobenius_norm(M_{ref}, M_{avg})
  if $opa\_dist \neq -\infty$ **and**
    $|opa\_dist - dist\_fn| < tol$ **then**
    break
  end if
  $M_{ref} \leftarrow M_{avg}$; $opa\_dist \leftarrow dist\_fn$
end for

Outputs: P_{i*}

transformation T such that it minimizes:

$$||\mathbf{WT} - \mathbf{M_{ref}}||_F \quad (1)$$

In the present case, we restrict T to the space of rotations, which corresponds to orthogonal Procrustes analysis where the transformation matrix T is orthogonal, by far the most common configuration for this type of analysis[2]. Schönemann (1966) proposes to compute T by performing a singular value decomposition of the matrix $W^{\mathsf{T}}M_{ref}$ such that $W^{\mathsf{T}}M_{ref} = U\Sigma V^{\mathsf{T}}$ and defining the transformation as $T = UV^{\mathsf{T}}$.

The Procrustes analysis described above applies to the case of two matrices, one serving as the reference for the other. In our work, however, we aim to go beyond merging only two models. We therefore build on Generalized Procrustes Analysis

[2]Note that the dot product, an operation central to transformer architectures, is invariant under rotations and translations but not under scalings, which provides a rationale for disregarding the latter. In addition, translations can be handled beforehand by centering the vectors at the origin.

(GPA) (Gower, 1975). Formally, GPA seeks a set of orthogonal transformations $\{T_1, \ldots, T_n\}$ such that the transformed matrices $\{P_1, \ldots, P_n\}$, with $P_j = W_j T_j$, and the common reference M_{ref} minimize[3]:

$$\sum_{j=1}^{n} \|P_j - M_{ref}\|_F \quad (2)$$

This extension to more than two matrices is implemented via an iterative procedure defined in Algorithm 1. At each iteration, a reference matrix is computed from the entire set of matrices, and each matrix is then aligned with this reference through orthogonal Procrustes analysis of pairs of matrices. The procedure continues until the change in alignment distance between successive iterations becomes sufficiently small, or until a maximum number of iterations is reached. In practice, as shown in the experiments of Section 3, convergence typically occurs after a single iteration.

3. Experiments and Evaluation

3.1. Evaluation Framework

3.1.1. Evaluation Tasks

The projection method for aligning different models within a common representation space, as described in Section 2, is applicable to any type of neural language model and can be employed as a preprocessing step for a wide range of model merging techniques. In this work, we focus specifically on encoder-based models in a single-task setting. Merging is therefore performed across multiple models obtained by fine-tuning the same base model on the same task. Our experiments were conducted using the BERT model (Devlin et al., 2019), specifically the `bert-base-uncased` variant.

Following prior work on language model merging, we carried out these experiments on nine tasks from the GLUE benchmark (Wang et al., 2018):

CoLA (Warstadt et al., 2019). Grammatical acceptability of sentences. Metric: Matthews correlation coefficient.

MNLI (Williams et al., 2018). Natural language inference between sentence pairs (neutral, entailment, contradiction). Metric: accuracy.

MRPC (Dolan and Brockett, 2005). Paraphrase detection. Metric: average of accuracy and F1 score.

QNLI (Rajpurkar et al., 2016). Question answering pairs from SQuAD recast as a textual entailment task. Metric: accuracy.

QQP (Shankar et al., 2017). Detection of duplicate questions. Metric: average of accuracy and F1 score.

RTE (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009). Textual entailment detection. Metric: accuracy.

SST-2 (Socher et al., 2013). Sentiment analysis. Metric: accuracy.

STSB (Cer et al., 2017). Semantic textual similarity. Metric: average of Pearson and Spearman correlations.

WNLI (Levesque et al., 2012). Winograd Schema Challenge recast as textual entailment. Metric: accuracy.

We fine-tuned five models for each task using different random seeds but identical hyperparameters. For fine-tuning and hyperparameter selection, we relied on the implementation and guidelines provided by Hugging Face[4]. The specific hyperparameter values used in our experiments are provided in Section A.1.1 of the appendices. Since the GLUE test set is not publicly available, we followed standard practice by reserving 10% of the training set as a held-out test set.

Since eight of our GLUE tasks are classification tasks, completed with a regression task, we also considered the Named Entity Recognition (NER) task for extending our experiments to sequence annotation tasks. More precisely, we adopted the following three benchmarks for English, with the use of F1 score as a metric:

CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003). Reference benchmark for general domain NER, with four types of entities: person, location, organization, and miscellaneous.

NCBI disease (Doğan et al., 2014). Benchmark in the biomedical domain made of PubMed abstracts annotated for one type of entity: disease.

TweetNER7 (Ushio et al., 2022). Social media benchmark made of tweets annotated with seven types of entities: person, location, event, product, group, corporation, and creative work.

These three benchmarks include both general and specialized content and entity types, as well

[3] In Equation 2, for simplicity, we consider the general case of n matrices to project in a shared space but in our case, we should formally use W_{ij}, T_{ij}, and P_{ij} for taking into account the fact that a language model includes m matrices.

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

as different types of corpora, covering both conventional forms, such as newspaper articles or scientific publications, as well as noisier forms, such as tweets. As with the GLUE benchmark tasks, we fine-tuned five models with different random seeds but identical hyperparameters. The specific hyperparameter values for these models are provided in Section A.1.2 of the appendices.

3.1.2. Merging Methods

In our experiments, we considered the following merging methods, among the most commonly used as baselines in prior work involving encoder-based models:

Average (Wortsman et al., 2022). This method simply averages the parameters with the same role across the models to be merged;

Task arithmetic (Ilharco et al., 2023). The idea behind this method is to capture the specificity of a model fine-tuned for a given task by computing, for each parameter, the difference between its value in the fine-tuned model and in the pretrained base model. This yields task vectors, which can then be combined by simple addition;

Fisher (Matena and Raffel, 2022). This method performs parameter merging by estimating their importance in each model using the Fisher information matrix, computed from fine-tuning data;

RegMean (Jin et al., 2023). This method also leverages fine-tuning data, but in this case to minimize the difference between the predictions of the merged model and those of the original models. The optimization is carried out through local linear regression;

TIES (Yadav et al., 2023). TIES builds upon Task arithmetic for model merging but introduces two preliminary steps. First, it prunes the parameters, retaining only the top 20% that influence model performance most. Second, it aligns parameter signs through a procedure that accounts for their relative importance.

The aforementioned merging methods operate at the level of parameter merging itself. However, some methods, such as the one proposed in this article, intervene before this merging to facilitate it, meaning they can be combined with different merging strategies. In our experiments, we used as a reference the method considered most effective in this regard among recent work:

DARE (Yu et al., 2024), which works by randomly nullifying the parameter differences between a fine-tuned model and its base model for a subset of parameters, while rescaling the remaining ones.

The hyperparameters of these merging methods, along with the values used in our experiments, can be found in Section A.2 of the appendices. Finally, from a merging perspective, encoder-based models raise the specific issue of how to combine their classification or regression heads[5]. Since we merge models fine-tuned on the same task, we chose to merge the heads in the same way as the rest of the model, with two exceptions. The Task arithmetic and TIES methods define task vectors relative to the pretrained base model. However, the head of this model is randomly initialized for the downstream task, making the definition of a task vector for this component meaningless. Therefore, we merged the heads by simply averaging their parameters for these two methods. It is worth noting that the projection method described in Section 2 is applied to both the models' body and head.

3.2. Results for the GLUE benchmark

3.2.1. Base Model Merging

Table 1 reports the evaluation results of the different baseline merging methods on the nine GLUE tasks, with the last column on the right providing the mean performance across these tasks. The first three rows correspond to the minimum, maximum, and mean values obtained over the five individual models considered for each task.

Since the model mergings are performed in an uninformed manner, i.e., without using a validation set, we consider merging to be of interest whenever its performance on a task exceeds that of the worst individual model for the same task. From a statistical perspective, comparing against the mean is more meaningful; however, in an applied context where one must select a single model without prior knowledge of its performance, any solution that improves upon the worst case is already valuable.

Overall, the last column of Table 1 shows that outperforming the worst model is observed for most merging methods, though not for all. Only Fisher performs slightly below *min.* on average among the five merging methods. The combination with DARE slightly decreases Fisher's performance, and more significantly reduces the performance of TIES, which otherwise stands as the strongest merging method without DARE.

On average, none of the merging methods surpasses the maximum performance of individual models, but TIES comes very close, while Task

[5] In the case of decoder or encoder-decoder models, the target task is generally performed in a way that closely resembles pretraining.

	cola	mnli	mrpc	qnli	qqp	rte	sst2	stsb	wnli	mean
min.	56.0	83.6	84.7	90.8	88.8	63.9	92.0	88.8	45.1	77.1
max.	58.3	84.3	85.7	91.3	89.0	70.0	92.5	89.8	56.3	79.7
mean	57.2	84.0	85.2	91.1	88.9	67.7	92.3	89.2	50.7	78.5
average	54.8	82.3	84.9	90.8	86.5	59.9	92.3	**88.9**	**56.3**	77.4
task arithmetic	59.4	80.7	88.8	90.6	87.6	68.2	92.7	87.8	**56.3**	79.1
regmean	56.8	**83.5**	86.5	**91.3**	**88.7**	69.0	92.1	88.8	**56.3**	79.2
fisher	50.7	79.6	86.2	**91.3**	84.8	60.6	92.4	85.9	**56.3**	76.4
ties	59.8	81.2	**89.0**	91.2	88.1	69.0	92.7	88.4	**56.3**	**79.5**
dare[average]	56.2	81.5	85.1	90.7	86.5	60.6	92.2	87.0	**56.3**	77.4
dare[task arith.]	**59.9**	80.7	**89.0**	90.6	87.6	68.2	92.7	87.8	**56.3**	79.2
dare[regmean]	56.2	83.3	87.2	91.1	**88.7**	**69.7**	92.1	88.7	43.7	77.9
dare[fisher]	53.0	80.5	87.3	91.1	85.5	61.4	92.4	87.7	43.7	75.8
dare[ties]	53.0	72.1	87.4	85.8	83.7	66.8	**92.9**	86.9	**56.3**	76.1

Table 1: Evaluation results of different reference merging methods for five `bert-base-uncased` models fine-tuned on one of the 9 tasks from the GLUE benchmark. The first three rows correspond to the highest (*max.*), lowest (*min.*), and average (*mean*) results for the five base models to merge. The following five rows give the results obtained by the merging of these five base models performed by the five merging methods we considered. Finally, the dare[*] rows show the results of the combination of DARE and each of the five merging methods we considered. Bold values indicate the highest score a merging method achieves for a given task. Values ×100.

	cola	mnli	mrpc	qnli	qqp	rte	sst2	stsb	wnli	mean
min.	56.0	83.6	84.7	90.8	88.8	63.9	92.0	88.8	45.1	77.1
max.	58.3	84.3	85.7	91.3	89.0	70.0	92.5	89.8	56.3	79.7
mean	57.2	84.0	85.2	91.1	88.9	67.7	92.3	89.2	50.7	78.5
average	$57.3_{2.5}$	$83.6_{1.3}$	$86.6_{1.7}$	$91.3_{0.5}$	$88.4_{1.9}$	$64.3_{4.3}$	$92.0_{-0.3}$	$88.8_{0.0}$	$\mathbf{56.3}_{0.0}$	$78.7_{1.3}$
task arithmetic	$\mathbf{60.3}_{0.9}$	$81.4_{0.7}$	$88.4_{-0.4}$	$89.8_{-0.8}$	$87.4_{-0.3}$	$71.5_{3.2}$	$92.9_{0.2}$	$88.9_{1.1}$	$\mathbf{56.3}_{0.0}$	$\mathbf{79.7}_{0.5}$
regmean	$57.0_{0.3}$	$83.5_{0.0}$	$87.0_{0.5}$	$91.3_{0.0}$	$88.5_{-0.2}$	$63.9_{-5.1}$	$91.6_{-0.5}$	$87.6_{-1.1}$	$\mathbf{56.3}_{0.0}$	$78.5_{-0.7}$
fisher	$58.3_{7.6}$	$\mathbf{83.8}_{4.1}$	$87.9_{1.7}$	$91.0_{-0.3}$	$\mathbf{88.6}_{3.8}$	$66.1_{5.4}$	$92.3_{-0.1}$	$\mathbf{89.5}_{3.6}$	$\mathbf{56.3}_{0.0}$	$79.3_{2.9}$
ties	$59.1_{-0.7}$	$82.3_{1.1}$	$87.5_{-1.4}$	$90.4_{-0.8}$	$87.9_{-0.1}$	$71.5_{2.5}$	$92.8_{0.1}$	$88.5_{0.1}$	$\mathbf{56.3}_{0.0}$	$79.6_{0.2}$
dare[average]	$57.5_{1.3}$	$83.7_{2.2}$	$86.8_{1.7}$	$\mathbf{91.4}_{0.7}$	$88.5_{2.0}$	$64.3_{3.6}$	$92.0_{-0.2}$	$89.2_{2.1}$	$\mathbf{56.3}_{0.0}$	$78.8_{1.5}$
dare[task arith.]	$45.4_{-14.4}$	$81.4_{0.7}$	$\mathbf{88.6}_{-0.4}$	$89.7_{-0.9}$	$87.3_{-0.3}$	$71.1_{2.9}$	$92.9_{0.2}$	$88.9_{1.1}$	$\mathbf{56.3}_{0.0}$	$78.0_{-1.2}$
dare[regmean]	$57.0_{0.8}$	$83.5_{0.2}$	$87.4_{0.3}$	$\mathbf{91.4}_{0.2}$	$88.4_{-0.3}$	$64.3_{-5.4}$	$91.9_{-0.2}$	$87.7_{-1.1}$	$\mathbf{56.3}_{12.7}$	$78.6_{0.8}$
dare[fisher]	$58.0_{5.0}$	$83.8_{3.3}$	$88.4_{1.0}$	$91.1_{0.0}$	$\mathbf{88.6}_{3.1}$	$66.1_{4.7}$	$92.2_{-0.2}$	$\mathbf{89.5}_{1.8}$	$\mathbf{56.3}_{12.7}$	$79.3_{3.5}$
dare[ties]	$51.2_{-1.8}$	$71.3_{-0.8}$	$86.5_{-0.9}$	$85.2_{-0.5}$	$81.9_{-1.7}$	$\mathbf{71.8}_{5.1}$	$\mathbf{93.1}_{0.2}$	$86.9_{0.0}$	$\mathbf{56.3}_{0.0}$	$76.0_{0.0}$

Table 2: Results comparable to those in Table 1 but with merging methods preceded by the application of the model projection method into a shared space proposed in this article. The first three result rows are identical to those in Table 1 and are included here for reference. Subscript numbers indicate the performance gain or loss relative to the corresponding result in Table 1. Values ×100.

arithmetic (with or without DARE) and RegMean approach this level to a lesser extent. However, all these methods outperform the mean of the individual models. Finally, it should be noted that DARE, which, like our proposed method, intervenes before the merging itself and can therefore be applied to different merging strategies, yields an overall negative outcome: it is neutral or nearly neutral for Average and Task arithmetic, but substantially degrades the performance of RegMean, Fisher, and TIES. This finding is consistent with Huang et al. (2024), who suggest that DARE may not be well-suited for merging more than two or three models. Moreover, prior evaluations of DARE have concerned merging models trained on different tasks, rather than models fine-tuned on the same task.

At the task level, no merging method matches or exceeds the performance of *min.* consistently across all tasks, with RegMean being the closest to achieving this objective. Some merging methods, however, clearly outperform the best individual model for specific tasks: for instance, the combination of Task arithmetic and DARE for CoLA, or Task arithmetic, Fisher, and RegMean (with or without DARE) for MRPC. A noteworthy case is WNLI, where all merging methods, except for the combinations of DARE with Fisher or RegMean, match the best individual model. The small size of this dataset may explain this observation.

3.2.2. Merging with Procrustes Analysis

Table 2 reproduces the experiments from Table 1, but applies beforehand the projection method based on Procrustes analysis described in Section 2, which aligns models into a common representation space. In the specific case of DARE, three different process are thus applied sequentially: our projection method, DARE, and finally, a merging method. At the global level (last column on the right), we observe that the impact of this projection method is overall positive. Only RegMean and the combination of Task arithmetic with DARE show a decrease in overall performance due to the projection. In contrast, Fisher's performance improves substantially, both with and without DARE. In the latter case, projection even yields the best results reported with DARE. It also enables Task arithmetic to match the maximum performance of the individual models, and leads to only TIES with DARE failing to surpass the minimum of individual model performance. Moreover, the number of merging methods exceeding or matching the average of the individual models increases from four to eight, with all non-DARE variants included.

A more direct comparison between our projection method and DARE (see the last part of Table 1 vs. the central part of Table 2), which are quite similar in terms of usage, highlights the former's superiority. The average gain with our projection method is 0.83 points, whereas the average loss with DARE is 1.1. Figure 2 shows a task-by-task analysis of the impact of the two methods: while our Procrustes analysis has a slight negative impact only on QNLI and SST-2, DARE has a slight positive impact only on MRPC and SST-2. Finally, combining our projection method with DARE (i.e., applying the projection, then DARE, followed by a merging method) turns out to be relatively neutral: in this configuration, the average loss due to DARE decreases slightly from 1.1 to 1.0 points.

At the task level, systematically surpassing the minimum individual performance remains out of reach, primarily due to QQP, which appears to be a challenging dataset for the tested merging methods. Nevertheless, Fisher, benefiting the most on average from our projection method, comes close to this objective. Projection also raises WNLI results to the maximum individual level and produces substantial gains for the combination of DARE with RegMean or Fisher. By contrast, the sharp performance drop observed for DARE combined with Task arithmetic on CoLA is difficult to interpret. Finally, our projection method leads to two additional tasks, QNLI and RTE, surpassing the maximum performance of individual models, and one additional task, RTE, surpassing their average performance.

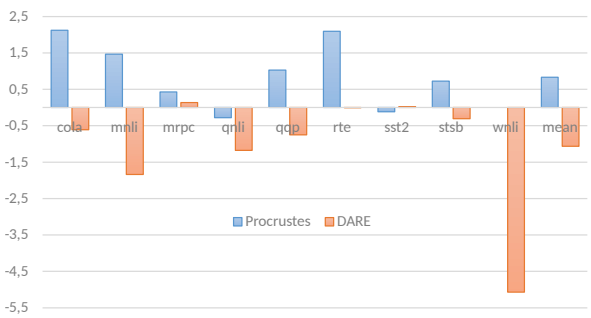

Figure 2: Comparison of performance gains or losses brought by DARE and Procrustes for the tasks of the GLUE benchmark.

3.3. Results for NER benchmarks

Table 3 brings together in a single table the results of the individual models (first three rows), those resulting from their combination using the various methods previously considered (*base* column), as well as those obtained by the same combination methods but preceded by our projection method based on Procrustes analysis (*procr.* column). Regarding the merging methods, we excluded RegMean and Fisher, as they require access to a subset of the training examples[6] and were not among the top two options in Table 2.

The first observation regarding the individual models is that their variance is very low in the case of CoNLL, and much higher, with a similar magnitude, for NCBI and TweetNER7. While the low variance in CoNLL 2003 does not prevent merging methods from yielding gains, these improvements remain modest compared to the individual models. Although TIES and the application of DARE to TIES degrade performance, the other merging methods at least match the average performance of the individual models, and in some cases even achieve the best overall performance. For both NCBI and TweetNER7, TIES is also the worst-performing merging method. This effect is particularly pronounced for TweetNER7. In both cases, merging makes it possible to surpass the worst individual model, and even exceed the average for NCBI, but it cannot outperform the best individual model.

The use of Procrustes analysis is generally beneficial, as indicated by the superscript values in the *procr.* columns. The only negative, albeit minor, impact of this analysis is observed for the Average merging method in the case of NCBI. Conversely, it can sometimes lead to substantial gains, as seen when combined with DARE for the Average method on TweetNER7, enabling a return to competitive performance from an initially very low level. For

[6]These two methods use a subset of the training data for computing some statistics they need, which limits their ability to merge models for which the training data are unavailable.

	CoNLL 2003		NCBI		TweetNER7		mean	
	base	**procr.**	**base**	**procr.**	**base**	**procr.**	**base**	**procr.**
min.	90.4	–	84.2	–	48.9	–	74.5	–
max.	90.6	–	86.1	–	51.7	–	76.1	–
mean	90.5	–	84.9	–	50.6	–	75.3	–
average	90.6	$90.7_{0.1}$	**85.8**	$85.6_{-0.2}$	43.0	$48.9_{6.0}$	73.1	$75.1_{1.9}$
task arithmetic	90.7	$90.8_{0.1}$	85.3	$\mathbf{85.8}_{0.5}$	**49.9**	$51.1_{1.2}$	75.3	$75.9_{0.6}$
ties	90.2	$90.8_{0.5}$	85.1	$85.7_{0.6}$	41.7	$\mathbf{52.0}_{10.3}$	72.3	$\mathbf{76.2}_{3.8}$
dare[average]	90.5	$90.7_{0.2}$	85.0	$85.7_{0.7}$	27.8	$49.4_{21.6}$	67.8	$75.2_{7.5}$
dare[task arith.]	**90.9**	$\mathbf{90.9}_{0.0}$	85.4	$\mathbf{85.8}_{0.3}$	49.8	$51.4_{1.6}$	**75.4**	$76.0_{0.6}$
dare[ties]	89.6	$90.0_{0.4}$	83.5	$84.3_{0.8}$	47.2	$48.8_{1.6}$	73.4	$74.4_{0.9}$

Table 3: Evaluation results for the three NER benchmarks. As with the previous tables, the first three rows refer to the individual models' results, with the best (*max.*) and worst (*min.*) models alongside the mean of all these models. For the following rows and each benchmark, *base* corresponds to the initial results of the merging methods, while *procr.* shows the results of the same merging methods preceded by the Procrustes analysis. Subscript numbers in column *procr.* indicate the performance gain or loss relative to the corresponding result in the column *base*. Values ×100.

this same dataset, it even allows for a significant improvement over the best individual model, and it systematically yields better results than the best individual model for all merging methods, except DARE[TIES], in the case of CoNLL 2003.

Finally, Figure 3 allows for a more specific comparison of the projection method based on Procrustes analysis we propose and DARE. As for the GLUE benchmark, this comparison clearly favors Procrustes analysis, as it yields systematic gains across all three base merging methods and for each of the three benchmarks considered. In contrast, DARE systematically results in performance degradation under the same conditions. However, Table 3 shows that for the NER task, unlike what was observed for GLUE tasks, the combination of DARE with Procrustes analysis yields almost systematic improvements over using DARE alone.

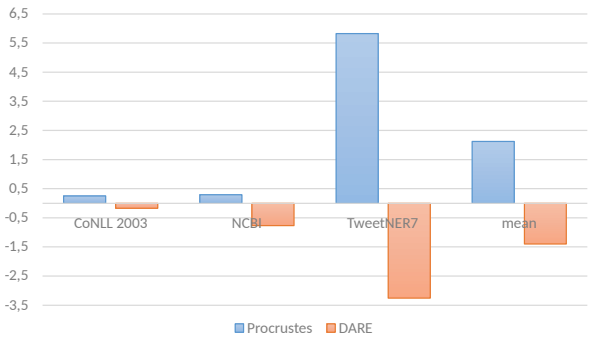

Figure 3: Comparison of performance gains or losses brought by DARE and Procrustes for the NER benchmarks.

4. Conclusion and Future Work

The study presented in this article takes place within the context of neural language model merging methods, focusing on parameter alignment techniques applied before the merging itself. More specifically, we proposed a new method based on the theoretical framework of Procrustes analysis to project the parameters of a set of transformer-based language models into a common representation space, thereby facilitating their merging.

We applied this method to the merging of models fine-tuned on the same task and derived from the same base model. Experiments conducted on the nine GLUE benchmark tasks and six reference merging methods demonstrate the effectiveness of our approach, yielding in most cases a notable improvement in the performance of the tested merging methods. Furthermore, experiments on three complementary NER benchmarks confirmed the trends observed for the GLUE benchmark. All these evaluations also highlight that model merging is a promising option for reducing the risk of selecting a poorly performing model among multiple fine-tuned variants, and in many cases, even allows surpassing the average performance of individual models.

The most immediate direction for future work is to apply our method to other types of models, in particular decoder-only and encoder-decoder architectures, given that merging methods are often more beneficial for these architectures than for encoder-only models. Applying the proposed projection method to adapter parameters (Rebuffi et al., 2017) or LoRA modules (Hu et al., 2022) would also be of considerable interest.

Beyond extending the method to other model types, it would also be worthwhile to compare the

early merging approaches considered here, along with our proposed improvement, against later merging strategies such as bagging, jointly assessing their respective performance and computational cost. Finally, from a more explanatory perspective, a closer examination of the impact of the proposed alignment method at the parameter level, similar to the analysis conducted by Jin et al. (2025) for LoRA, would be necessary to characterize better the transformations performed and to identify potential avenues for further improvement.

5. Ethical Considerations and Limitations

The limitations of this work are closely related to the perspectives mentioned above: the proposed method has so far been validated only with encoder language models focusing on a single task. Its performance with decoder and encoder-decoder models, as well as with models fine-tuned for different tasks still needs to be evaluated. Moreover, our method falls within the category of early fusion approaches. A comparison with late fusion methods would also be of interest, even though early fusion methods can be expected to outperform the latter, a superiority coming at the cost of significantly higher inference time.

Finally, it should be noted that while our experiments show that our method for projecting language models into a shared space improves their merging, they do not provide any insight into its impact on the intrinsic biases of these models. Additional experiments testing model biases before and after projection should therefore be conducted to determine whether the method is neutral in this regard, tends to amplify these biases, or conversely, mitigates them.

6. Acknowledgements

This research work is supported by France 2030 funding managed by the National Research Agency (ANR) as part of the SHARP ANR project ANR-23-PEIA-0008 and the IA CLUSTER program, reference ANR-23-IACL-0003 – DATAIA CLUSTER. It was also made possible by the use of the FactoryIA supercomputer, financially supported by the Ile-de-France Regional Council.

7. Bibliographical References

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. 2023. Git Re-Basin: Merging Models modulo Permutation Symmetries. In *The Eleventh International Conference on Learning Representations*.

Avi Caciularu, Ido Dagan, and Jacob Goldberger. 2021. Denoising Word Embeddings by Averaging in a Shared Space. In **SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*, pages 294–301, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's MergeKit: A Toolkit for Merging Large Language Models. In *2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, Miami, Florida, US. Association for Computational Linguistics.

John C Gower. 1975. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51.

John C. Gower. 2010. Procrustes methods. *WIREs Computational Statistics*, 2(4):503–508.

Stefan Horoi, Albert Manuel Orozco Camacho, Eugene Belilovsky, and Guy Wolf. 2024. Harmony in Diversity: Merging Neural Networks with Canonical Correlation Analysis. In *41st International Conference on Machine Learning*, volume 235, pages 18815–18832. PMLR.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. 2024. EMR-Merging: Tuning-Free High-Performance Model Merging. *Advances in Neural Information Processing Systems*, 37:122741–122769.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing Models with Task Arithmetic. In *The Eleventh International Conference on Learning Representations*.

Moritz Imfeld, Jacopo Graldi, Marco Giordano, Thomas Hofmann, Sotiris Anagnostidis, and

Sidak Pal Singh. 2024. Transformer Fusion with Optimal Transport. In *The Twelfth International Conference on Learning Representations*.

Ruochen Jin, Bojian Hou, Jiancong Xiao, Weijie J. Su, and Li Shen. 2025. Fine-Tuning Attention Modules Only: Enhancing Weight Disentanglement in Task Arithmetic. In *The Thirteenth International Conference on Learning Representations*.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2023. Dataless Knowledge Fusion by Merging Weights of Language Models. In *The Eleventh International Conference on Learning Representations*.

Yova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing Procrustes Analysis for Better Bilingual Dictionary Induction. In *22nd Conference on Computational Natural Language Learning*, pages 211–220, Brussels, Belgium. Association for Computational Linguistics.

Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. 2023. Deep Model Fusion: A Survey. *arXiv:2309.15698*.

Michael S Matena and Colin A Raffel. 2022. Merging Models with Fisher-Weighted Averaging. In *Advances in Neural Information Processing Systems*, volume 35, pages 17703–17716. Curran Associates, Inc.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Sidak Pal Singh and Martin Jaggi. 2020. Model Fusion via Optimal Transport. In *Advances in Neural Information Processing Systems*, volume 33, pages 22045–22055. Curran Associates, Inc.

Hagai Taitelbaum, Gal Chechik, and Jacob Goldberger. 2019. A Multi-Pairwise Extension of Procrustes Analysis for Multilingual Word Translation. In *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3560–3565, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in neural information processing systems*, pages 5998–6008.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model Soups: Averaging Weights of Multiple Fine-Tuned Models Improves Accuracy without Increasing Inference Time. In *39th International Conference on Machine Learning*, volume 162, pages 23965–23998. PMLR.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-Merging: Resolving Interference When Merging Models. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024. Model Merging in LLMs, MLLMs, and Beyond: Methods, Theories, Applications and Opportunities. *arXiv:2408.07666*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language Models Are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. In *Forty-First International Conference on Machine Learning*.

8. Language Resource References

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In *Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7, pages 785–794.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. In *TAC*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: A resource for disease name recognition and concept normalization. volume 47, pages 1–10.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The Third PASCAL Recognizing Textual Entailment Challenge. In *ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The Winograd Schema Challenge. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference*. AAAI Press.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Iyer Shankar, Dandekar Nikhil, and Csernai Kornel. 2017. First quora dataset release: question pairs. https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. ACL.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Asahi Ushio, Francesco Barbieri, Vitor Sousa, Leonardo Neves, and Jose Camacho-Collados. 2022. Named Entity Recognition in Twitter: A Dataset and Analysis on Short-Term Temporal Shifts. In *2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (AACL-IJNLP 2022)*, pages 309–319, Online only. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural Network Acceptability Judgments. volume 7, pages 625–641, Cambridge, MA. MIT Press.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

A. Hyperparameters

A.1. Training of Individual Models

A.1.1. Tasks of the GLUE benchmark

For all the GLUE tasks, we adopted the following hyperparameter values for training models:

- number of epochs: 3
- maximum sequence length: 128
- weight decay: 0
- warm-up ratio: 0
- batch size: 32

In addition, we used the following learning rate values for the different tasks:

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

A.1.2. NER Tasks

For all the NER tasks, we adopted the following hyperparameter values for training models:

- number of epochs: 25
- learning rate: 5e-5
- weight decay: 0.1
- warm-up ratio: 0.1
- batch size: 32

A.2. Merging of Individual Models

For the implementation of the different merging methods, we relied on the codebase provided by the authors of the DARE method: https://github.com/yule-BUAA/MergeLM

In the absence of a validation set, we adopted for the various hyperparameters of the merging methods the values recommended as the most robust in the original papers presenting these methods:

- Task arithmetic
  - λ: weighting factor for the different tasks, in this case, the different models; $\lambda = 0.3$
- TIES
  - $\lambda = 1.0$
  - k: percentage of the highest parameter values not reset to zero; $k = 20$
- Fisher
  - number of examples: 1,024
  - batch size: 16
  - minimum value for Fisher coefficients: 1e-6
  - rescaling factor for Fisher coefficients: 0.3
  - normalization of Fisher coefficients: yes
- RegMean
  - number of examples: 1,024
  - batch size: 16
  - α: weighting factor for the non-diagonal terms of the parameter matrices; $\alpha = 0.9$
- DARE
  - parameter masking strategy: random
  - percentage of masked parameters: 50
  - format of masked parameters: value differences of parameters between models
  - parameter scaling: yes

B. Code and Data Availability

The code for projecting a set of neural language models in a shared representation space via the Procrustes analysis is available here: https://github.com/osf9018/procrustes-merging

Some of the models used for the experiments presented in this article are available here: https://huggingface.co/collections/ferret/procrustes-merging