arXiv:2404.15737v1 [cs.CL] 24 Apr 2024

No Train but Gain: Language Arithmetic for training-free Language Adapters enhancement

Mateusz Klimaszewski[a,b,*], **Piotr Andruszkiewicz**[a] **and Alexandra Birch**[b]

[a]Institute of Computer Science, Warsaw University of Technology
[b]School of Informatics, University of Edinburgh

Abstract. Modular deep learning is the state-of-the-art solution for lifting the curse of multilinguality, preventing the impact of negative interference and enabling cross-lingual performance in Multilingual Pre-trained Language Models. However, a trade-off of this approach is the reduction in positive transfer learning from closely related languages. In response, we introduce a novel method called language arithmetic, which enables training-free post-processing to address this limitation. Inspired by the task arithmetic framework, we apply learning via addition to the language adapters, transitioning the framework from a multi-task to a multilingual setup. The effectiveness of the proposed solution is demonstrated on three downstream tasks in a MAD-X-based set of cross-lingual schemes, acting as a post-processing procedure. Language arithmetic consistently improves the baselines with significant gains in the most challenging cases of zero-shot and low-resource applications. Our code and models are available at https://github.com/mklimasz/language-arithmetic.

1 Introduction

The recent progress of large language models has led to a huge interest in how well they can perform not just in English but across multiple languages and invoked the rise of Multilingual Pre-trained Language Models (MLLMs) [11, 43, 2]. These models serve as general-purpose solutions that can be adapted and applied to various Natural Language Processing tasks. Notably, MLLMs demonstrate zero-shot cross-lingual capabilities, allowing them to generalise effectively to downstream tasks even when trained in a language different from the target language.

The positive transfer of abilities from both related languages and high-quality training data from unrelated languages has meant that MLLMs have reported state-of-the-art performance in low-resourced languages [24]. However, this benefit does not always extend to high-resourced languages [21]. In such cases, the quality of MLLMs tends to decrease compared to their monolingual counterparts [25, 23, among others] due to negative interference phenomena [42]. Additionally, the curse of multilinguality [11] reveals the existence of a trade-off between language coverage and model capacity. Consequently, MLLMs must carefully limit the number of languages included during the pre-training phase.

Modular deep learning (MDL) [32] methods come to the rescue to avoid the abovementioned limitations, enabling the extension of MLLMs to support any number of languages [6, 40, 34, 29, 31]. MDL methods adapt the model to arbitrary tasks and languages by isolating components from each other (and the backbone MLLM) via parameter-efficient extensions. Examples of parameter-efficient modules are LoRA [17] and adapters [37, 16], which are low-budget (in terms of parameters) bottleneck layers that increase an MLLM size by just a fraction. Language adapters especially allow the modularisation of language-specific knowledge by training on a raw, unlabelled corpus in a target language.

The limitation of the MDL and language adapters is their isolation. While they lift the curse of multinguality and prevent negative interference, at the same time, language adapters limit the possible impact of positive transfer. Previous attempts to address these challenges — such as training bilingual [26] or language-family [9] adapters — do not scale effectively. In our work, we tackle this limitation as a post-processing step. Leveraging recent insights from task arithmetic [19], specifically *learning via addition*, we augment language adapters with missing related language knowledge – a concept we term language arithmetic. Remarkably, this training-free approach enhances language adapters across various scenarios, from zero-shot settings to low-resource regimes.

To summarise, our contributions are as follows:

- A novel training-free post-processing method named language arithmetic that enhances language adapters.
- We conduct a cross-lingual evaluation on three downstream tasks and Multilingual Pre-trained Language Models with test cases that include zero-shot and low-resource setups.
- We provide an analysis of language arithmetic internal components.

2 Background

Our research builds upon the contributions of Ilharco et al. [19] and Zhang et al. [45]. The following Section provides the background and serves as a gentle introduction to the concept of task vectors and task arithmetic.

2.1 *Task Vectors & Task Arithmetic*

Let us assume that we have access to a pre-trained model with its weights denoted $\theta_{pre} \in R^d$ and a fine-tuned version of the same

[*] Corresponding Author. Email: mateusz.klimaszewski.dokt@pw.edu.pl

model on a task t represented by $\theta^t_{ft} \in R^d$. The task vector $\tau_t \in R^d$ is an element-wise difference between models' weights.

$$\tau_t = \theta^t_{ft} - \theta_{pre} \tag{1}$$

The task vectors can be part of multiple arithmetic operations, e.g. *learning via addition*. This operation is an addition operation between two task vectors and the base model, i.e. we add two differences between the fine-tuned models and the pre-trained version with weights controlling the impact.

$$\theta_{multi-task} = \theta_{pre} + \lambda_1 \tau_{t_1} + \lambda_2 \tau_{t_2} \tag{2}$$

The lambdas can be further normalised to sum to one, i.e. $\lambda_2 = 1 - \lambda_1$ and simplifying notation with just λ.

$$\theta_{multi-task} = \theta_{pre} + \lambda \tau_{t_1} + (1 - \lambda) \tau_{t_2} \tag{3}$$

While we define learning via addition for two tasks, the same procedure can be applied to multiple tasks.

Task arithmetic allows us to forge a multi-task model from a separate, task-specific set of fine-tuned models, preserving high accuracy (although a shared pre-trained starting point is required, e.g. the same Language Model). Moreover, vectors from different tasks are typically close to orthogonal, and Ilharco et al. [19] speculate that this enables the combination of task vectors via addition with minimal interference.

In the Parameter-Efficient Fine-tuning (PEFT) regime, we can reduce a task vector to only newly introduced PEFT parameters [45]. Another way of looking at this is a task vector that is a zero-valued vector for all the parameters apart from PEFT weights, i.e. a task vector would not modify the base, pre-trained checkpoint.

3 Method

We propose language arithmetic that transitions the task arithmetic concept from a multi-task to a multilingual setup. In this Section, we describe the language arithmetic alongside its application as a training-free, post-processing step to a MAD-X cross-lingual framework [29].

3.1 Language Arithmetic

We formulate a language arithmetic (LA) concept by substituting the task in task vectors and arithmetic with a language. This approach means that instead of merging downstream tasks, we target a problem of cross-lingual performance. We propose to apply *learning via addition* to languages, and in Section 5.3, we demonstrate the discrepancies when comparing language and task vectors. Our study focuses specifically on the language adapters [34, 29]. Due to overlapping abbreviations, we use the LA exclusively as the former, i.e. language arithmetic. In the learning via addition, we limit the parameters to language adapters and simplify the notation that θ represents the adapters' weights and τ is referred to as a language vector. As we operate in a language space instead of a task, the t is replaced with a language, i.e. its language code in the notation. The example equation describes a language arithmetic operation between an English and a Spanish adapter.

$$\theta_{LA} = \theta_{pre} + \lambda \tau_{en} + (1 - \lambda) \tau_{es} \tag{4}$$

Throughout the paper, the equation above is abbreviated as a function: $LA(en, es)$ with lambda as a default parameter. Whenever we use language arithmetic in a zero-shot context, we add a subscript to the function name, i.e. $LA_{0s}(en, es)$ means that language arithmetic between English and Spanish is applied to a different language, e.g. French.

Language arithmetic is a training-free method, taking advantage of already pre-trained modules. The sole requirement is a validation dataset on which the λ parameter can be established. While in our work, we use a pretty fine-grained step (0.05) to determine the λ value (i.e. we run evaluations for $\lambda \in [0, 1]$ with a provided step), our analysis showcased that it is possible to increase the value and limit the computation burden even more (details in Section 5.1).

3.2 Application

We apply the language arithmetic as an enhancement method for language adapters. We assume two scenarios: (i) zero-shot case, where we do not have a language adapter in the desired language and (ii) "regular" enhancement case, where we improve the existing language adapter (in high- and low-resourced languages).

We evaluate our post-processing method as an extension of the MAD-X framework [29] to challenge our solution in a cross-lingual manner. The overview of the schema is presented in Figure 1.

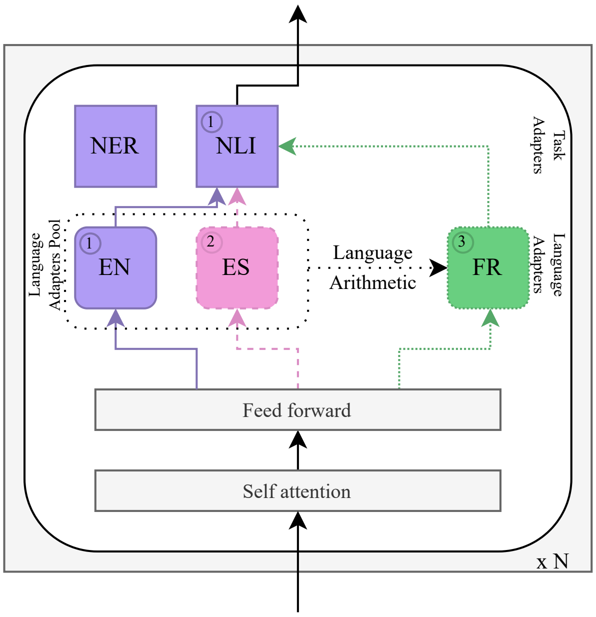


Figure 1. Adapter routing options in a Transformer layer. ① represents the most straightforward case where the task and target language match (EN). ② is a MAD-X case as we train an additional language adapter (ES) and reuse the task adapter (e.g. NLI one). ③ presents the language arithmetic. Depending on the content of the language adapter pool, we can have a zero-shot scenario as in the Figure, where we combine EN and ES to obtain FR, $LA(en, fr)$). The alternative would require taking an adapter from the pool and enhancing it (e.g., improving ES with EN, $LA(es, en)$).

The MAD-X consists of the following:

1. Choosing MLLM
2. Training language adapter ①
3. Training task adapter ①
4. Training more language adapters ②

In our work, we introduce an additional step:

5. Post-processing via Language Arithmetic ③

3.2.1 MLLM

The first step usually assumes the availability of a Multilingual Pre-trained Language Model. In our work, we focus on mBERT[1] [12] and XLM-R[2] [11].

3.2.2 Training language & task adapters

The following two steps depend on the downstream dataset. In the most extreme case, we might have a dataset in one specific language and need to take advantage of cross-lingual performance. Therefore, step number 2 trains a language adapter on raw corpora corresponding to the task language using regular masked language modelling loss in a self-supervised manner. During this step, the MLLM is frozen. The third step freezes both the backbone and language adapter and trains a task adapter on a downstream task dataset. Given a set of tasks or if a new task appears, we can repeat this step as long as a required language adapter exists, i.e. a language adapter that matches the task's language.

3.2.3 Additional language adapters

The last, fourth step is the heart of the MAD-X framework. We can now train language adapters in any language, leverage the pre-trained task adapter(s) and obtain a cross-lingual performance by connecting a new language adapter with a task adapter (i.e. routing first via language adapter and then task) to obtain language-task pair unseen during training. The growing pool of pre-trained adapters can be accessed at public repositories like AdapterHub [28] and reused for further use cases.

3.2.4 Post-processing via Language Arithmetic

First, we assume the use-case where the pool of language adapters does not contain the target languages, i.e. zero-shot scenario. In this study, we add English as a task-specific language to a language related to the target. In the second language adapter enhancement scenario, we apply language arithmetic in its default form, where the goal is to enhance the existing target language adapter (via either English or a related language). At last, we investigate (simulated and actual) low-resource cases.

4 Experiments

4.1 Experimental setup

We use the AdapterHub library [28] for all our experiments. For language adapters, we train on the Wikipedia corpora[3] for 250k steps with a learning rate of 1e-4, an effective batch size of 64 using a single GPU[4] and the same initialisation. For task-specific training, we train for 100 epochs with the same learning rate and a batch size set to 16. We choose the final checkpoint based on validation dataset performance (for language adapters, we evaluate on a held-out subset of Wikipedia).

Downstream evaluation is performed on three tasks: Named Entity Recognition (NER), Natural Language Inference (NLI) and Question Answering (QA), covering jointly 13 languages[5], while the training - to perform cross-lingual evaluation - is performed on the English data. For the NER task, we use the WikiANN [35] dataset and for NLI - XNLI [10]. The QA evaluation is done on XQuAD [5] (we split data 50/50 into valid/test datasets), and the training uses SQuAD 1.1 [36]. Additionally, to evaluate a low-resource scenario for a language not covered during MLLM pre-training, we leverage the Assamese subset from IndicXNLI [1].

4.2 Zero-shot evaluation

The zero-shot evaluation assumes a scenario where the language adapter pool does not contain the desired target language. In such a case, one can use an English adapter as a proxy (i.e. ① in Figure 1). An alternative is the usage of a related language (②). Language arithmetic serves a solution that, instead of choosing, combines the adapter's tuple: $LA_{0s}(en, REL)$, where REL symbolises a related language (we provide the related language list in Appendix B).

Table 1 presents the results of the zero-shot experiment. Language arithmetic consistently outperforms the proxy baselines, reaching up to an improvement of over 3 F1 points in the XLM-R NER setup and almost 1.5 F1 for QA. Additionally, we looked at the λ parameter derived on a validation dataset. While most cases set the value to over 0.5 (i.e. preferring the English side, given $LA_{0s}(en, REL)$), the preferred values did not showcase any consistency and pattern. We analyse that phenomenon in Section 5.1.

4.3 Language Arithmetic evaluation

This evaluation assumes that the target language adapter exists in the adapter pool. We test two cases, i.e. $LA(T, en)$ and $LA(T, REL)$, where REL is once again a related language and T is the target language. The results are presented in Table 2. Compared to the baseline direct application of a target language adapter (i.e. MAD-X method), the gains are not as significant as in the case of the zero-shot scenario. Given that a target language adapter is trained on a significant corpus, it gives less room for improvement (this is not the case in the low-resource regimes, as shown in the following Section). However, considering the cost-to-performance ratio and the minimal fatigue that our post-processing method enforces on existing MAD-X pipelines, we can see a constant but minor gain.

Notably, during the evaluation of XLM-R NLI, we observed that the outcomes remain consistent across different methods (e.g. 67.8 for Swahili). This behaviour is a built-in property of the proposed method. The method has a *fail-safe mechanism*, which keeps the extreme values during the evaluation of $\lambda \in [0, 1]$. Suppose the method does not find a better lambda different than the extreme values (i.e. $\lambda \in \{0, 1\}$). In that case, it means that we are using a proxy language directly, and the language arithmetic did not provide an improvement (but kept the final result the same).

4.4 Low resource evaluation

Training a language adapter might be troublesome for high-resourced languages due to massive corpora requiring significant computa-

[1] bert-base-multilingual-cased

[2] xlm-roberta-base

[3] 20331101.xx checkpoint https://huggingface.co/datasets/wikimedia/wikipedia

[4] Nvidia A100/RTX3090/V100

[5] ar, bg, de, el, es, fr, hi, ru, sw, tr, ur, vi, zh; XQuAD does not cover 4 languages (bg, fr, sw, ur)

Table 1. Zero-shot results of language arithmetic (LA) per language, where one side of the arithmetic equation is the English adapter, and the other is related to the evaluated language adapter (e.g. Italian for Spanish evaluation). As baselines, we report using both adapters in isolation (pEA - proxy via English adapter, pRA - proxy via a related language adapter).

		ar	bg	de	el	es	fr	hi	ru	sw	tr	ur	vi	zh	**AVG**
								NER (F1 score)							
mBERT	pEA	33.45	75.62	78.10	69.80	73.42	78.25	63.03	61.63	63.14	72.56	28.72	68.81	36.99	61.81
	pRA	33.54	74.51	76.76	69.70	71.53	78.37	58.01	62.05	61.85	68.71	25.08	67.44	33.30	60.07
	$LA_{0s}(en, REL)$	37.22	75.83	77.89	73.42	73.42	78.51	64.11	62.95	64.80	72.56	30.39	69.25	37.65	62.77
XLM-R	pEA	31.30	72.80	72.75	65.94	67.39	73.65	62.64	56.17	58.55	65.91	31.74	58.86	21.43	56.86
	pRA	39.07	75.24	73.12	65.00	67.01	72.74	66.43	58.14	57.53	64.60	36.72	55.82	18.91	57.72
	$LA_{0s}(en, REL)$	43.19	76.57	74.27	70.58	73.48	74.75	68.64	60.37	63.02	69.45	37.22	60.68	23.44	61.20
								NLI (Accuracy)							
mBERT	pEA	61.66	65.43	68.40	62.08	72.22	71.84	58.34	65.87	46.25	56.93	56.49	68.46	66.73	63.13
	pRA	53.87	64.49	66.43	61.14	70.84	70.68	54.47	63.21	43.33	54.43	54.51	67.43	66.63	60.88
	$LA_{0s}(en, REL)$	62.06	65.23	68.42	62.44	71.52	72.20	57.96	65.19	46.51	56.95	56.13	69.28	67.33	63.17
XLM-R	pEA	65.99	75.89	73.61	72.50	76.97	75.45	67.47	73.47	58.02	69.10	62.51	73.03	72.91	70.53
	pRA	67.35	76.01	73.67	70.56	69.92	71.46	67.70	74.05	55.89	68.28	62.93	72.36	71.30	69.34
	$LA_{0s}(en, REL)$	68.66	76.69	73.81	73.03	76.97	76.11	68.74	73.65	60.12	70.16	63.87	72.51	73.15	71.35
								QA (F1 score)							
mBERT	pEA	55.93	-	71.93	51.46	72.39	-	50.65	67.46	-	44.99	-	63.51	54.12	59.16
	pRA	41.76	-	71.14	53.03	71.67	-	43.22	66.31	-	43.68	-	65.05	54.97	56.76
	$LA_{0s}(en, REL)$	55.93	-	71.32	53.91	72.61	-	50.40	66.36	-	44.64	-	64.16	54.46	59.31
XLM-R	pEA	61.27	-	72.07	70.22	73.18	-	59.70	69.56	-	64.63	-	70.81	61.31	66.97
	pRA	62.54	-	69.58	64.98	72.60	-	63.00	69.09	-	63.63	-	70.07	59.15	66.07
	$LA_{0s}(en, REL)$	64.87	-	72.69	71.69	74.45	-	64.01	70.11	-	64.98	-	71.55	61.49	68.43

Table 2. Results over all tested languages of the improvement of a target language adapter (denoted as T, i.e. the MAD-X method [29]) via language arithmetic. We use both English (en) and a related language (REL) as a right side of a LA function. Additionally, we report a mix (MIX) results where we choose better arithmetic combination for each language.

	$T \rightarrow$	ar	bg	de	el	es	fr	hi	ru	sw	tr	ur	vi	zh	**AVG**
								NER (F1 score)							
mBERT	T	37.03	79.30	78.50	74.79	74.83	78.32	68.50	65.95	73.65	71.70	40.68	72.64	36.77	65.59
	$LA(T, en)$	37.03	80.11	78.50	75.56	74.64	78.24	66.13	65.95	74.72	72.37	40.68	73.12	39.62	65.90
	$LA(T, REL)$	37.03	79.64	78.50	75.12	74.83	78.26	68.09	66.04	74.50	71.89	40.68	72.88	39.21	65.90
	$LA(T, MIX)$	37.03	80.11	78.50	75.56	74.83	78.26	68.09	66.04	74.72	72.37	40.68	73.12	39.62	66.07
XLM-R	T	50.43	78.71	77.56	78.48	78.62	76.52	71.54	62.37	74.53	76.28	63.96	66.00	25.84	67.76
	$LA(T, en)$	51.32	79.60	77.56	78.65	78.86	76.90	72.27	62.37	74.84	76.28	63.96	66.68	27.55	68.22
	$LA(T, REL)$	51.23	80.16	77.59	78.50	79.06	76.70	71.72	62.34	74.53	76.28	63.96	66.18	28.10	68.18
	$LA(T, MIX)$	51.32	80.16	77.59	78.65	79.06	76.90	72.27	62.37	74.84	76.28	63.96	66.68	28.10	68.32
								NLI (Accuracy)							
mBERT	T	63.41	67.35	67.88	65.23	71.72	72.08	61.24	66.01	58.98	62.99	58.54	68.16	67.64	65.48
	$LA(T, en)$	63.41	67.35	68.46	65.23	72.24	72.24	61.24	66.01	59.16	62.81	58.10	68.54	68.22	65.62
	$LA(T, REL)$	63.41	67.35	68.80	65.09	71.12	72.28	61.08	65.65	58.96	62.99	58.48	68.94	67.90	65.54
	$LA(T, MIX)$	63.41	67.35	68.80	65.23	72.24	72.28	61.24	66.01	59.16	62.99	58.48	68.94	68.22	65.72
XLM-R	T	71.32	76.59	75.45	74.95	77.13	76.97	68.84	74.73	67.80	71.12	65.83	73.83	73.35	72.92
	$LA(T, en)$	70.36	76.59	75.37	74.33	77.41	77.03	69.34	74.73	67.80	71.54	66.03	74.19	73.47	72.94
	$LA(T, REL)$	71.32	76.59	75.35	74.97	77.13	76.97	68.84	74.73	67.80	71.12	66.15	73.85	72.71	72.89
	$LA(T, MIX)$	71.32	76.59	75.37	74.97	77.41	77.03	69.34	74.73	67.80	71.54	66.15	74.19	73.47	73.07
								QA (F1 score)							
mBERT	T	54.46	-	71.17	51.14	73.45	-	48.80	66.92	-	49.27	-	65.85	54.10	59.46
	$LA(T, en)$	55.84	-	71.13	53.41	72.83	-	50.36	67.62	-	48.85	-	63.54	54.15	59.75
	$LA(T, REL)$	55.42	-	70.99	54.71	73.16	-	48.80	67.07	-	49.31	-	64.94	54.94	59.93
	$LA(T, MIX)$	55.84	-	71.13	54.71	73.16	-	50.36	67.62	-	49.31	-	64.94	54.94	60.22
XLM-R	T	67.48	-	71.62	71.99	73.70	-	67.96	70.96	-	61.67	-	74.12	64.85	69.37
	$LA(T, en)$	67.61	-	72.71	73.44	75.60	-	67.96	72.02	-	66.57	-	74.07	63.24	70.36
	$LA(T, REL)$	68.00	-	71.62	73.66	75.71	-	67.96	72.03	-	67.01	-	74.12	64.85	70.55
	$LA(T, MIX)$	68.00	-	72.71	73.66	75.71	-	67.96	72.03	-	67.01	-	74.12	64.85	70.67

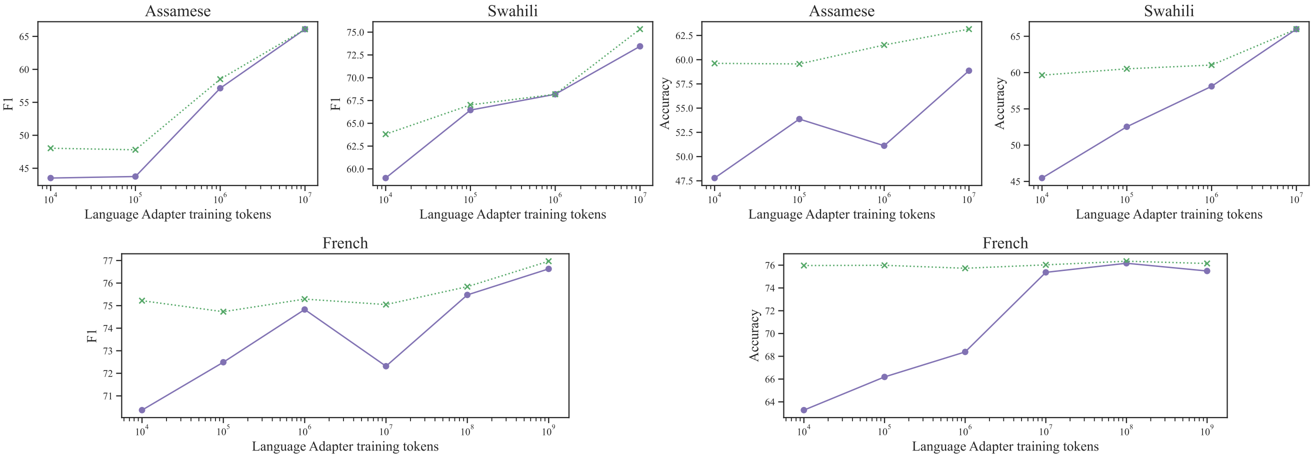


Figure 2. NER (left) and NLI (right) evaluation of a set of adapters trained on a Wikipedia subset showcases that language arithmetic $LA(T, en)$ (green, dotted line) provides significant gains when compared against direct usage of the adapter (violet, solid line), especially in a very low-resource regime. The x-axis represents the token budget of each trained language adapter.

tional resources.[6] On the other hand, in most languages, we lack data to train a strong language adapter, i.e. language-specific corpora might be either too small or unavailable [15]. We investigate whether language arithmetic can help in such cases. We test our solution in three cases and define three (actual and simulated) evaluation scenarios:

- Assamese (as) - low resource language, additionally not used in the pre-training of mBERT and XLM-R,
- Swahili (sw) - low resource language, used in the pre-training,
- French (fr) - high resource, used in the pre-training. We simulate cases from low to high resources.

We train a series of language adapters with different token budgets for each language, from 10k to 10M (or 1B for French; we limit this particular study to the XLM-R model). Afterwards, we compare the usage of such adapters directly against language arithmetic with three adapters (we use $LA(T, en)$, where $T \in \{as, sw, fr\}$).

Figure 2 presents the results of the evaluation performed on the downstream tasks. The most gain is visible in the most challenging scenario, during the evaluation on the Assamese dataset. In this case, the backbone MLLMs did not encounter the language during the pre-training phase. Although the difference becomes less pronounced in the NER task as we approach the limits of available data, there remains a significant margin for NLI - the difference can be explained by the overlap in the corpora (Wikipedia) between NER and language adapter training tasks, following the findings of Gururangan et al. [13]. For Swahili, where the language is part of the pre-training, the flattening effect begins earlier and affects both tasks. Nevertheless, leveraging language arithmetic still yields improvements.

The simulated case of French showcases that even with a relatively weak language adapter (trained on 10k tokens), the language arithmetic can restore existing knowledge and results in high performance for the language. Moreover, comparing the adapters trained with a different token budget, the results remain similar, without significant fluctuations. We believe that this phenomenon happens because the Multilingual Pre-trained Language Model has seen a much higher amount of French in the pre-training procedure than Swahili (over 35 times more tokens in XLM-R pre-training; moreover, French is in the top 15 represented languages). Therefore, even undertrained French adapters have a relatively easy task once they are merged with a robust English adapter. In practical terms, this finding allows us to prototype new languages quicker by estimating the possible end product quality or might serve as an intermediate solution (until the full-corpora adapter is trained).

5 Analysis

5.1 *Lambda impact*

Our study estimates the λ parameter with a small step (0.05). Depending on multiple variables that include, among others, model and evaluation dataset sizes or number of languages, running 20 evaluations might be troublesome (especially when using neural-based metrics, e.g. COMET [38]). Therefore, we analysed the potential impact of choosing a suboptimal lambda with a decrease in evaluation count. The breakdown includes a subset of languages on both tasks (using the XLM-R version). We chose the zero-shot scenario where we performed language arithmetic between English and related language adapters.

In Figure 3, we plot the validation scores with the corresponding baselines, that is, the scores of using directly the adapters. The dotted lines are based on $\lambda = 0$ or $\lambda = 1$ for clarity, meaning we exclusively use the arithmetic equation's left or right side (i.e., a specific language). In most cases, a subset of valid λ values would improve over the baselines. Moreover, the analysis reveals that a coarser evaluation (e.g., with a step of 0.1 or 0.2) would be sufficient, reducing the required number of performed tests up to four times while maintaining most of the improvement. At last, setting the default $\lambda = 0.5$ would be near optimum for the analysed subset.

5.2 *Language Relatedness*

Relatedness of languages is a difficult-to-define concept. At times, in our proposed framework, we might face a choice of multiple, seemingly equally related languages to use for the arithmetic operation. In this analysis, we decided to look at this aspect via a glance at Romance languages. We trained an additional subset of language

[6] Although in our experimental setup, we train each adapter for the same number of steps and choose the best checkpoint based on the validation performance, for low-resourced languages, one could apply an early stopping mechanism in a production-level pipeline.

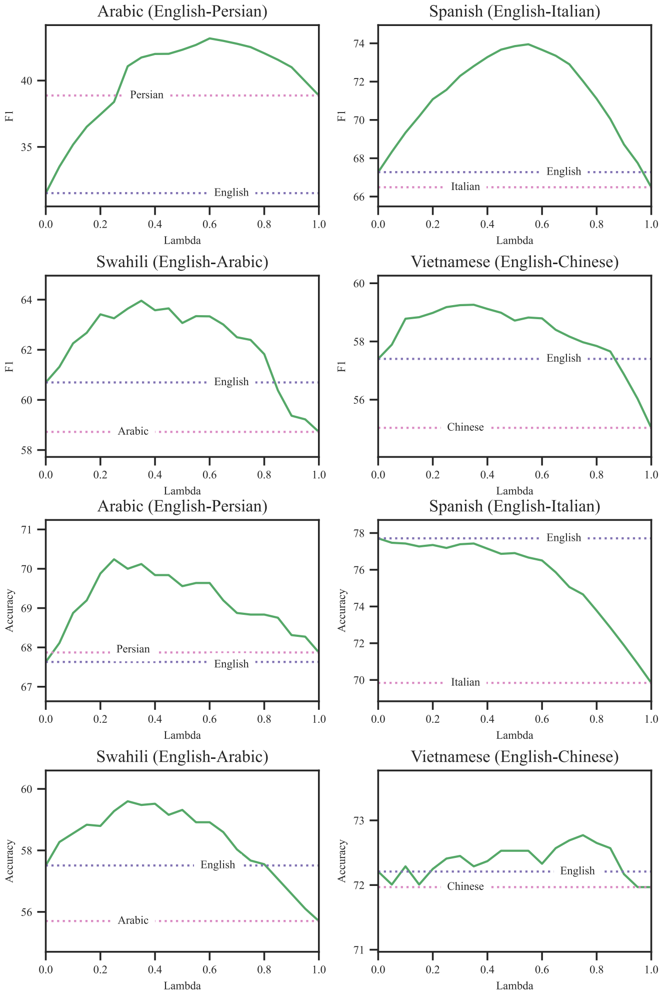


Figure 3. Interpolation of λ values for the zero-shot XLM-R scenario (NER - top, NLI - bottom; for QA see Appendix A) on the validation dataset. The horizontal dashed lines represent the baseline scores for both languages used in language arithmetic.

Table 3. Impact of language relatedness on the language arithmetic. We compare different Romance languages as a right side of LA equation, i.e. $l2$ (both tasks use XLM-R model).

	$LA(l_1 \downarrow, l_2 \rightarrow)$	ca	es	fr	it	pr	ro
				NER			
Eval language	es	78.77	-	78.96	79.06	78.91	78.66
	fr	76.63	76.92	-	76.70	76.80	76.80
				NLI			
	es	77.07	-	77.23	77.13	77.07	76.83
	fr	76.77	76.99	-	76.97	76.95	77.07

adapters and formed a pool of 6 languages: Catalan, French, Italian, Portuguese, Romanian and Spanish. Afterwards, we evaluated languages shared in our NER and NLI tasks (Spanish and French) by arithmetic with the entire Romance languages pool.

The results are presented in Table 3 and show that given a related language (in this case, defined as coming from the same language family), there might be very little difference in which one we will use. The relative difference between the best and the worst language reaches only a 0.40 F1 score in the NER task and the same value in terms of accuracy points in NLI. Nevertheless, the more sophisticated and hand-crafted language choice would improve the results from Table 2.

5.3 *Language vs task vectors*

Task vectors exhibit high sparsity and orthogonality, as Ilharco et al. [19] observed. While the former characteristic can be denoted in language vectors (Figure 4), the latter displays different properties, in contrast to task vectors. In Figure 5, we visualise the cosine similarity between evaluated language vectors of language adapters. Notably, the minimal cosine similarity (0.2) surpasses the maximum (0.18) reported by previous research in the task space [19]. Interestingly, most pairs in the task space oscillate within the range of 0.01 to 0.03. At the same time, language vectors surpass 0.2 in each case, indicating that the orthogonality aspect is an inherent property of task adapters.

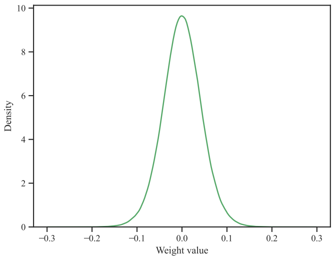
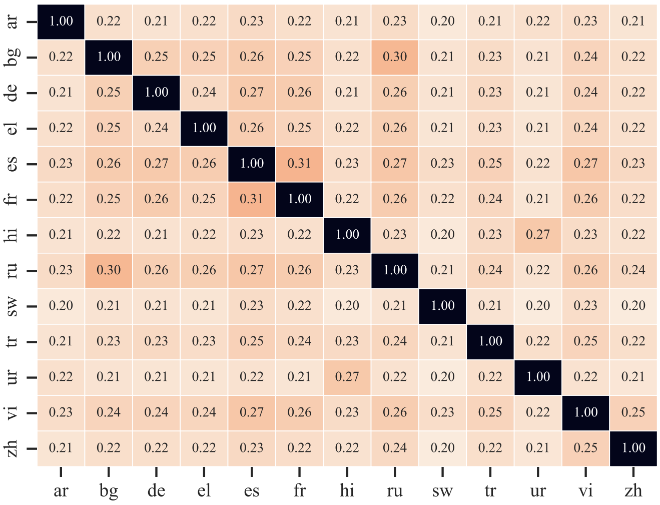


Figure 4. Language vectors, similar to task vectors, are extremely sparse. The kernel density estimate plot presents the weights of a Spanish mBERT adapter. The behaviour is consistent across sampled layers and languages.

Based on the cosine similarity observation, we investigated one of the recent task arithmetic extensions, Ties-Merging [44]. This work introduces a three-step algorithm that prevents different parameter interferences, improving upon task arithmetic. The algorithm *decreases the cosine similarity* via a pruning step and alignment of parameter signs to perform arithmetic only on relevant parameters to the merged tasks. On the experimental details note, as Ties-Merging operates on averaging, not addition, we utilise a different lambda range during validation (as suggested by Yadav et al. [44], $\lambda \in [0.8, 1.8]$), and we set Top-K% to the default value of 20 and additionally to 80.



	ar	bg	de	el	es	fr	hi	ru	sw	tr	ur	vi	zh
ar	1.00	0.22	0.21	0.22	0.23	0.22	0.21	0.23	0.20	0.21	0.22	0.23	0.21
bg	0.22	1.00	0.25	0.25	0.26	0.25	0.22	0.30	0.21	0.23	0.21	0.24	0.22
de	0.21	0.25	1.00	0.24	0.27	0.26	0.21	0.26	0.21	0.23	0.21	0.24	0.22
el	0.22	0.25	0.24	1.00	0.26	0.25	0.22	0.26	0.21	0.23	0.21	0.24	0.22
es	0.23	0.26	0.27	0.26	1.00	0.31	0.23	0.27	0.23	0.25	0.22	0.27	0.23
fr	0.22	0.25	0.26	0.25	0.31	1.00	0.22	0.26	0.22	0.24	0.21	0.26	0.22
hi	0.21	0.22	0.21	0.22	0.23	0.22	1.00	0.23	0.20	0.23	0.27	0.23	0.22
ru	0.23	0.30	0.26	0.26	0.27	0.26	0.23	1.00	0.21	0.24	0.22	0.26	0.24
sw	0.20	0.21	0.21	0.21	0.23	0.22	0.20	0.21	1.00	0.21	0.20	0.23	0.20
tr	0.21	0.23	0.23	0.23	0.25	0.24	0.23	0.24	0.21	1.00	0.22	0.25	0.22
ur	0.22	0.21	0.21	0.21	0.22	0.21	0.27	0.22	0.20	0.22	1.00	0.22	0.21
vi	0.23	0.24	0.24	0.24	0.27	0.26	0.23	0.26	0.23	0.25	0.22	1.00	0.25
zh	0.21	0.22	0.22	0.22	0.23	0.22	0.22	0.24	0.20	0.22	0.21	0.25	1.00

Figure 5. Cosine similarity between language vectors of language adapters.

We report the comparison in the zero-shot setting on the NER task (XLM-R version) in Table 4. The Ties-Merging decreases the results significantly; it is not only outperformed by language arithmetic in every language pair but also by other baselines (see Table 1). Moreover, we note that the pruning operation has the reverse effect; higher pruning (i.e. keeping Top-K% lower) decreases the performance (in contrast to task vectors) by making language vectors more sparse and, hence, closer to orthogonal.

Table 4. Ties-Merging evaluation in the zero-shot setup on the NER task (XLM-R version). In the case of language arithmetic, where the language vectors have a higher overlap (i.e. higher cosine similarity), removing parameter interference decreases the overall performance.

Method	AVG F1 score
LA	61.20
Ties-Merging (Top-K% 20)	56.40
Ties-Merging (Top-K% 80)	59.88

One interpretation of the phenomena can be the different goals of the arithmetic: in the multi-task setup, we try to include multiple, often disconnected, tasks into a single task vector. In contrast, the language vectors' goal is to include the knowledge of the closely related language rather than remove the harmful artefacts. Our findings indicate that language arithmetic has different characteristics than task arithmetic, and the follow-up works that improve upon task arithmetic might not be suited for the multilingual context.

6 Related Work

Knowledge composition from multiple, independently trained adapters has been widely discussed in the literature. However, unlike our work, the solutions require substantial changes to the vanilla adapter setup. The previous work either requires additional parameters to learn a parameterised composition function/a gating module to combine/steer the flow through the suitable adapter(s), or needs a specific training procedure that increases the complexity of the overall solution or, in most cases, both [33, 30, 22, 26, 9, 20, 41]. Moreover, to prevent specifically negative interference, hyper-adapters [7] were proposed using hyper-networks [14], and Ansell et al. [3] applied sparse fine-tuning to compose task and language masks. Unlike the prior studies mentioned earlier, our attempt is training-free and does not modify the base architecture. The most conceptually similar work is proposed by Chronopoulou et al. [8]; however, they operate on the notion of sample similarity to a subset of domains in a domain adaptation regime. Additionally, concurrent to our work, Parović et al. [27] show intial potential of task arithemtic in cross-lingual transfer based on a full fine-tuning setup. At last, we denote the rise of task arithmetic use cases, e.g. vision tasks or cross-task generalisation [39, 18].

7 Conclusion

We have proposed language arithmetic, which enhances language adapters based on task arithmetic learning via addition. It is a training-free method and functions as a post-processing technique for the adapters. Our experiments have shown that LA is particularly beneficial in zero-shot and low-resourced cases. At last, we highlight the differences between language and task arithmetic.

In our future work, we plan to extend language arithmetic by incorporating additional components into the sum. Additionally, we aim to adapt other elements of the task arithmetic framework, initially designed for tasks, i.e. task analogies and forgetting via negation, to a multilingual setup with an analysis of the differences between multi-task and multilingual arithmetic context. Furthermore, we will evaluate LA's performance on various non-classification tasks.

8 Limitations

Our work was tested on English-centric task training and could be extended to different languages with more PEFT methods. Moreover, applying multi-source training based on the work of Ansell et al. [4] could provide better robustness of the task adapters and a more thorough analysis. At last, our analysis of the differences between task and language vectors is preliminary, without definitive conclusions.

References

[1] D. Aggarwal, V. Gupta, and A. Kunchukuttan. IndicXNLI: Evaluating multilingual inference for Indian languages. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10994–11006, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.755.

[2] D. M. Alves, J. Pombal, N. M. Guerreiro, P. H. Martins, J. Alves, A. Farajian, B. Peters, R. Rei, P. Fernandes, S. Agrawal, P. Colombo, J. G. C. de Souza, and A. F. T. Martins. Tower: An open multilingual large language model for translation-related tasks, 2024.

[3] A. Ansell, E. Ponti, A. Korhonen, and I. Vulić. Composable sparse fine-tuning for cross-lingual transfer. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.125.

[4] A. Ansell, M. Parović, I. Vulić, A. Korhonen, and E. Ponti. Unifying cross-lingual transfer across scenarios of resource scarcity. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3980–3995, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.242.

[5] M. Artetxe, S. Ruder, and D. Yogatama. On the cross-lingual transferability of monolingual representations. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.421.

[6] A. Bapna and O. Firat. Simple, scalable adaptation for neural machine translation. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1165.

[7] C. Baziotis, M. Artetxe, J. Cross, and S. Bhosale. Multilingual machine translation with hyper-adapters. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1170–1185, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.77.

[8] A. Chronopoulou, M. Peters, A. Fraser, and J. Dodge. AdapterSoup: Weight averaging to improve generalization of pretrained language models. In A. Vlachos and I. Augenstein, editors, *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2054–2063, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.153.

[9] A. Chronopoulou, D. Stojanovski, and A. Fraser. Language-family adapters for low-resource multilingual neural machine translation. In A. K. Ojha, C.-h. Liu, E. Vylomova, F. Pirinen, J. Abbott, J. Washington, N. Oco, V. Malykh, V. Logacheva, and X. Zhao, editors, *Proceedings of the The Sixth Workshop on Technologies for Machine Translation of Low-Resource Languages (LoResMT 2023)*, pages 59–72, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.loresmt-1.5.

[10] A. Conneau, R. Rinott, G. Lample, A. Williams, S. Bowman, H. Schwenk, and V. Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1269.

[11] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

[13] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740.

[14] D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

[15] B. Haddow, R. Bawden, A. V. Miceli Barone, J. Helcl, and A. Birch. Survey of low-resource machine translation. *Computational Linguistics*, 48(3):673–732, Sept. 2022. doi: 10.1162/coli_a_00446.

[16] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 09–15 Jun 2019.

[17] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[18] C. Huang, Q. Liu, B. Y. Lin, T. Pang, C. Du, and M. Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition, 2024.

[19] G. Ilharco, M. T. Ribeiro, M. Wortsman, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[20] M. Klimaszewski, Z. Belligoli, S. Kumar, and E. Stergiadis. Gated adapters for multi-domain neural machine translation. In *ECAI 2023 - 26th European Conference on Artificial Intelligence*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 1264–1271. IOS Press, 2023. doi: 10.3233/FAIA230404.

[21] T. Kocmi, D. Macháček, and O. Bojar. *The Reality of Multi-Lingual Machine Translation*, volume 21 of *Studies in Computational and Theoretical Linguistics*. Institute of Formal and Applied Linguistics, Prague, Czechia, 2021. ISBN 978-80-88132-11-0.

[22] J. Lee, S.-w. Hwang, and T. Kim. FAD-X: Fusing adapters for cross-lingual transfer to low-resource languages. In Y. He, H. Ji, S. Li, Y. Liu, and C.-H. Chang, editors, *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 57–64, Online only, Nov. 2022. Association for Computational Linguistics.

[23] L. Martin, B. Muller, P. J. Ortiz Suárez, Y. Dupont, L. Romary, É. de la Clergerie, D. Seddah, and B. Sagot. CamemBERT: a tasty French language model. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.645.

[24] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. Le Scao, M. S. Bari, S. Shen, Z. X. Yong, H. Schoelkopf, X. Tang, D. Radev, A. F. Aji, K. Almubarak, S. Albanie, Z. Alyafeai, A. Webson, E. Raff, and C. Raffel. Crosslingual generalization through multitask finetuning. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15991–16111, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.891.

[25] D. Nozza, F. Bianchi, and D. Hovy. What the [mask]? making sense of language-specific bert models. *ArXiv*, abs/2003.02912, 2020.

[26] M. Parović, G. Glavaš, I. Vulić, and A. Korhonen. BAD-X: Bilingual adapters improve zero-shot cross-lingual transfer. In M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1791–1799, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.130.

[27] M. Parović, I. Vulić, and A. Korhonen. Investigating the potential of task arithmetic for cross-lingual transfer. In Y. Graham and M. Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 124–137, St. Julian's, Malta, Mar. 2024. Association for Computational Linguistics.

[28] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vulić, S. Ruder, K. Cho, and I. Gurevych. AdapterHub: A framework for adapting transformers. In Q. Liu and D. Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online, Oct. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.7.

[29] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.617.

[30] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In P. Merlo, J. Tiedemann, and R. Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online, Apr. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.39.

[31] J. Pfeiffer, N. Goyal, X. Lin, X. Li, J. Cross, S. Riedel, and M. Artetxe. Lifting the curse of multilinguality by pre-training modular transformers. In M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.255.

[32] J. Pfeiffer, S. Ruder, I. Vulić, and E. Ponti. Modular deep learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. Survey Certification.

[33] M. Q. Pham, J. M. Crego, F. Yvon, and J. Senellart. A study of residual adapters for multi-domain neural machine translation. In L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, Y. Graham, P. Guzman, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, and M. Negri, editors, *Proceedings of the Fifth Conference on Machine Translation*, pages 617–628, Online, Nov. 2020. Association for Computational Linguistics.

[34] J. Philip, A. Berard, M. Gallé, and L. Besacier. Monolingual adapters for zero-shot neural machine translation. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.361.

[35] A. Rahimi, Y. Li, and T. Cohn. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy, July 2019. Association for Computational Linguistics.

[36] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In J. Su, K. Duh, and X. Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264.

[37] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[38] R. Rei, C. Stewart, A. C. Farinha, and A. Lavie. COMET: A neu-

ral framework for MT evaluation. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.213.

[39] G. Stoica, D. Bolya, J. Bjorner, T. Hearn, and J. Hoffman. Zipit! merging models from different tasks without training, 2023.

[40] A. Üstün, A. Bisazza, G. Bouma, and G. van Noord. UDapter: Language adaptation for truly Universal Dependency parsing. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.180.

[41] J. Wang, Y. Chen, W. Zhang, S. Hu, T. Xu, and J. Zheng. AdapterDistillation: Non-destructive task composition with knowledge distillation. In M. Wang and I. Zitouni, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 194–201, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-industry.20.

[42] Z. Wang, Z. C. Lipton, and Y. Tsvetkov. On negative interference in multilingual models: Findings and a meta-learning treatment. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4438–4450, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.359.

[43] B. Workshop. Bloom: A 176b-parameter open-access multilingual language model, 2023.

[44] P. Yadav, D. Tam, L. Choshen, C. Raffel, and M. Bansal. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[45] J. Zhang, S. Chen, J. Liu, and J. He. Composing parameter-efficient modules with arithmetic operation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

A Lambda impact - QA

Figure 6 presents the analysis of lambda impact for the Question Answering task. For details, refer to Section 5.1.

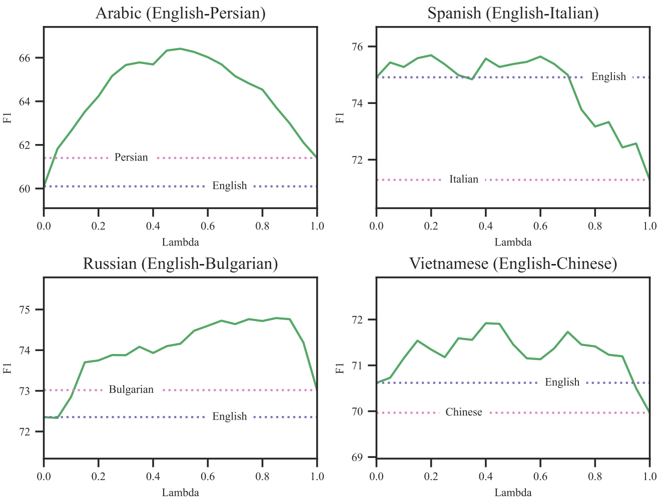


Figure 6. Interpolation of λ values for the zero-shot QA XLM-R scenario on the validation dataset. The horizontal dashed lines represent the baseline scores for both languages used in language arithmetic.

B Related languages choice

We derived related languages for the language adapters based on the available pool, i.e. we reused the adapters from other languages from the task (we trained two bonus ones, Persian and Italian, apart from the analysis in Section 5.2). While the related language could have been more suitable in a few cases, there is always a trade-off between relatedness, criterion for relatedness and available corpora. Moreover, the perfectly related language matching would be an unfair comparison as it would assume the availability of the ideal adapter's pool, which is unlikely in practice. Table 5 presents the related language choice used in our experiments.

Table 5. Languages used in the experiments with corresponding related languages. In the most cases, we used a related language from the already existing pool (i.e. evaluated languages, the first rows labelled "Lang."); however, we trained some additional ones, including Italian and Persian.

Lang.	ar	bg	de	el	es	fr	hi
Related Lang.	fa	ru	fr	de	it	it	ur
Lang.	ru	sw	tr	ur	vi	zh	
Related Lang.	bg	ar	de	hi	zh	vi	