Language Anisotropic Cross-Lingual Model Editing

Yang Xu **Yutai Hou** **Wanxiang Che** **Min Zhang**
Harbin Institute of Technology
{yxu, ythou, car}@ir.hit.edu.cn zhangmin2021@hit.edu.cn

Abstract

Multilingual pre-trained language models can learn task-specific abilities or memorize facts across multiple languages but inevitably make undesired predictions with specific inputs. Under similar observation, model editing aims to post-hoc calibrate a model targeted to specific inputs with keeping the model's raw behavior. However, existing work only studies the monolingual scenario, which lacks the *cross-lingual transferability* to perform editing simultaneously across languages. In this work, we focus on cross-lingual model editing. Firstly, we define the cross-lingual model editing task and corresponding metrics, where an edit in one language propagates to the others. Next, we propose a framework to naturally adapt monolingual model editing approaches to the cross-lingual scenario using parallel corpus. Further, we propose *language anisotropic* editing to improve cross-lingual editing by amplifying different subsets of parameters for each language. On the newly defined cross-lingual model editing task, we empirically demonstrate the failure of monolingual baselines in propagating the edit to multiple languages and the effectiveness of the proposed *language anisotropic* model editing. Our code is publicly available at https://github.com/franklear/LiME.

1 Introduction

Pre-trained language model based approaches have become the best practice in many fields, including multilingual NLP (Che et al., 2021; Tunstall et al., 2022). During training, Transformer-based (Vaswani et al., 2017) models can embed language abilities (Geva et al., 2021) and memorize facts (Dai et al., 2022) in the parameters. Though, models inevitably make undesired predictions with specific inputs, such as mistake labels or outdated facts. Moreover, the performance of multilingual models is unbalanced across languages, leading to inconsistency predictions over the same input in different languages. However, the high cost of training and data collecting makes it unrealistic to re-train the models using calibrated data in all languages. Therefore, there is a pressing need for an approach to calibrate multilingual pre-trained models across all languages of interest simultaneously.

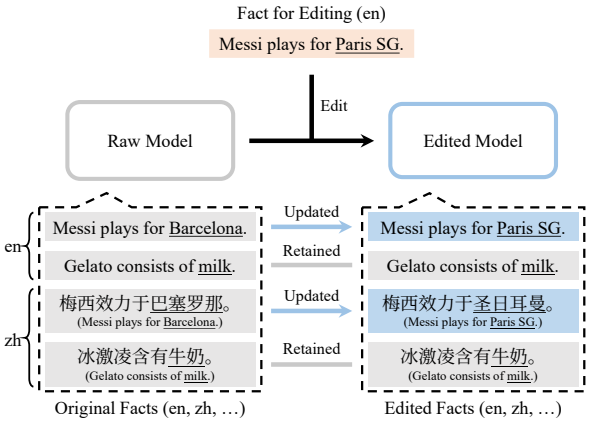

Figure 1: An example of cross-lingual model editing for updating facts, where we represent facts inferred from models as sentences in the dashed boxes. The goal is to update the given fact while retaining unrelated facts. Further, cross-lingual editing requires the edit in one language (e.g., en) to affect all languages (en, zh, . . .).

As an emerging research area, model editing allows us to calibrate the behavior of pre-trained language models targeted to specific inputs (Sinitsin et al., 2020; Cao et al., 2021; Mitchell et al., 2022a,b; Meng et al., 2022a,b; Hase et al., 2021). However, challenges emerge when applying model editing to the cross-lingual scenario, due to the two features of multilingual pre-trained models:

The first is *cross-lingual transferability*. Based on prior research conducted on pre-trained multilingual models like XLM (Conneau and Lample, 2019) and InfoXLM (Chi et al., 2021), it is well-established that incorporating diverse language data during training leads to advantageous cross-lingual transfer effects. Thus, input with the same meaning can be expressed in multiple languages as completely different sentences. The editor has

Findings of the Association for Computational Linguistics: ACL 2023, pages 5554–5569
July 9-14, 2023

to be aware of this feature in case it suffers from editing failure in unseen languages.

The second is *language anisotropy*. Recent work reveals that language-specific and language-universal parameters exist in the multilingual pre-trained model (Wang et al., 2020). This finding means the model tends to mainly activate a subset of its parameters depending on the language to be processed, which we call *language anisotropy*. An editor which treats all parameters identically for all languages is not *language anisotropic*, potentially harming other languages when editing.

In this work, we propose for the first time cross-lingual model editing on multilingual pre-trained language models. Different from existing model editing, an edit in a single language propagates to the others in cross-lingual model editing. As is shown in Figure 1, with cross-lingual model editing, editing a fact in English also affects the Chinese version, while retaining unrelated facts.

We propose a simple yet effective framework to adapt existing monolingual model editing approaches to the cross-lingual scenario using the parallel corpus. Specifically, we replace the inputs for editor training with their parallel expressions in random languages. For example, the editor can be asked to edit model predictions on English input. The edited model is then supervised to enforce that the predictions are updated on parallel Chinese input and retained on unrelated French inputs. The next time, the above languages randomly change. To this end, the cross-lingual training formula helps the editor gain *cross-lingual transferability*.

Besides, we leverage the *language anisotropy* nature of the multilingual models to further improve cross-lingual model editing. Specifically, we propose to add a group of L_0 constrained language-specific masks as the editor's parameters. During editing, the masks are used to instruct the editor to focus on different parts of the raw model's parameters according to the inputs' language. Training along with the masks, the editor gains the skill of making *language anisotropic* edits.

Our primary contributions are as follows:

- We define the cross-lingual model editing task and corresponding evaluation metrics.
- We propose a simple yet effective framework to adapt the monolingual editing approaches to the cross-lingual scenario.
- We propose *language anisotropic* model editing to improve cross-lingual model editing significantly.

2 Background: Model Editing

Sinitsin et al. (2020) propose Editable Training (model editing) as an efficient approach to modify the behavior of a trained model on specific inputs, where three core requirements are highlighted:

- *Reliability*: the edited model acquires the desired behavior on specific inputs.
- *Locality*: the edit influences the model on other inputs of interests as little as possible.
- *Efficiency*: the editing approach should be computationally efficient.

Reliability and *locality* are essential attributes of the model editing task, while *efficiency* is required to make the editor usable.

Recent work explores several ways to solve the model editing problem (Sinitsin et al., 2020; Mitchell et al., 2022a; Meng et al., 2022a,b). Despite the variety of algorithms, their training formulas are similar, i.e., training the editor end-to-end on editing data under the condition of *reliability* and *locality*. Specifically, a training step of the editor contains two stages. 1) Editing stage: the editor is used to edit desired predictions into the raw model $f(\cdot;\boldsymbol{\theta})$, producing the edited model $f(\cdot;\boldsymbol{\theta}_u)$. 2) Editor training stage: the edited model is then constrained under the requirements of *reliability* and *locality*, corresponding to two core objectives respectively.

For *reliability*, the edited model need to make the desired prediction y_e in response to the input x_e. This requirement refers to the task loss L_{task}, e.g., cross-entropy or L_2. So we have

$$L_{\text{rel}} = \lambda_{\text{rel}} L_{\text{task}}\left(f(x_e;\boldsymbol{\theta}_u), y_e\right). \qquad (L_{\text{rel}})$$

For *locality*, the edited model needs to retain predictions of unrelated inputs, which means that for an unrelated input x_r, the output $f(x_r;\boldsymbol{\theta})$ should be kept. Though a similar loss like L_{rel} can work in theory, the stronger KL divergence loss is used to minimize the side effect on unrelated labels

$$L_{\text{loc}} = \lambda_{\text{loc}}\,\text{KL}\left(f(x_r;\boldsymbol{\theta}_u) \parallel f(x_r;\boldsymbol{\theta})\right). \quad (L_{\text{loc}})$$

In addition, other auxiliary objectives can be utilized which do not affect the training formula.

Note that the goal is to train the editor instead of the raw model. During training, the gradients propagate through the edited model to the editor. At test time, only the editing stage is needed. Overall,

the training of the editor is a meta version of the model training because the "data" that the editor processes is the model (plus the input-prediction pair to be edited).

3 Cross-Lingual Model Editing

3.1 Task Definition

Following the work on monolingual model editing, we continue taking the idea of making an edit with *reliability* and *locality* (Sinitsin et al., 2020), while introducing *cross-lingual transferability*.

Assuming we have a model f parameterized by $\boldsymbol{\theta}$ that maps the input x to the prediction $p = f(x; \boldsymbol{\theta})$. An update is needed when we want the model change its prediction from p to y. Here the requirement of *cross-lingual transferability* brings the key difference. The same input can be represented in multiple languages, producing parallel sentences. Therefore, the edit with *reliability* for x should affect the parallel inputs, denoted as $I(x)$. As the example in Figure 1, "Messi plays for Paris SG." in English is parallel to its Chinese translation. For *locality*, the side effect should be as low as possible, which means the prediction of input $x' \notin I(x)$ is retained.

Note that under this setting, one edit is always independent of another. The editor revisits the $\boldsymbol{\theta}$ for every edit, then produces the corresponding $\boldsymbol{\theta}_u$. Formally, the goal of the editor is to

$$
\begin{aligned}
&\textit{find} \quad \boldsymbol{\theta}_u, \\
&\textit{s.t.} \quad \begin{cases} f(x_p; \boldsymbol{\theta}_u) = y & \forall x_p \in I(x) \\ f(x_n; \boldsymbol{\theta}_u) = f(x_n; \boldsymbol{\theta}) & \forall x_n \notin I(x) \end{cases}, \\
&\textit{given} \quad x, I(x) = \{x' | x' \text{ is parallel to } x\}, y, f, \boldsymbol{\theta}.
\end{aligned}
$$

3.2 Cross-Lingual Editing Based on Monolingual Approaches

Dispite the *cross-lingual transferability*, the requirements of *reliability* and *locality* stay the same with monolingual model editing, which are defined by the training data. To fully leverage the monolingual editing approaches and build reasonable baselines, we propose a framework to adapt them to the cross-lingual scenario using the parallel corpus as illustrated in Figure 2.

What we need is a slight change in the training formula of the monolingual editing approaches, namely aligning inputs in different languages. Given x_e in the editing language l_e as the input to be edited and the corresponding desired prediction y_e, the inputs used in the training objectives are sampled over the parallel inputs set $I(x_e)$.

For *reliability*, the edited model is asked to update the prediction to y_e on the sampled input $x_u \in I(x_e)$ in the updating language l_e. Thus the *reliability* loss (L_{rel}) is modified by replacing x_e with x_u. For *locality*, the sampled input $x_r \notin I(x_e)$ in the retaining language l_r is used as input, and the *locality* loss (L_{loc}) remains the same.

Monolingual editing is a degenerate case where only a single language is considered, i.e., $l_e = l_u = l_r$. When the languages differ, the editor trained under the above sampling strategy acquires *cross-lingual transferability*.

Intuitively, the editor functions as updating on identical inputs while not affecting unrelated inputs. In the above cross-lingual adaptation, *reliability* loss tells the editor what should be identical, and *locality* loss tells what should be unrelated. Thus, the two losses illustrate a semantically equivalent range for the editor across multiple languages, deriving the *cross-lingual transferability*. Therefore, the adaptation we make leverages the parallel corpus to inspire the potential of transferability that comes with the model editing task.

3.3 Language Anisotropic Editing

A multilingual pre-trained model like mBERT (Devlin et al., 2019) can integrate over one hundred languages in a single model. However, a known phenomenon called the curse of multilinguality (Conneau et al., 2020) exposes the trade-off between the number of languages and the model capacity, implying the languages tend to compete for the shared parameters. Further, it is revealed that language-specific and language-universal parameters exist in the multilingual model, which potentially harm its *cross-lingual transferabillity* (Wang et al., 2020). All this evidence indicates that the multilingual model is *language anisotropic* in the perspective of the parameters. Therefore, we introduce a priori, i.e., the update should focus more on a certain subset of parameters according to the language of the input to edit. Nevertheless, identifying which language prefers which parameters is not so direct. Our idea is to drive the editor to find the important parameters during training.

As shown in the top-right part of Figure 2, we realize the idea with a group of learnable language-specific masks. The model editor produces new

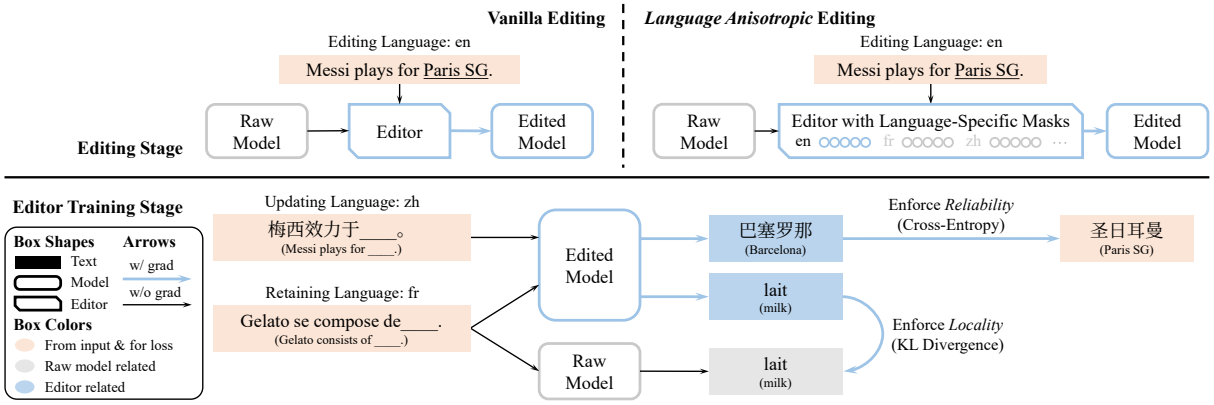

Figure 2: The overall framework of the proposed cross-lingual model editing. Each training step consists of two stages. The editor edits the model at first, then losses for *reliability* and *locality* are obtained from the outputs of the edited model to supervise the editor. Languages of editing/updating/retaining are randomly sampled in each training step to endow the editor with *language transferability*. Our novel *language anisotropic* model editing applies soft masks according to the editing language, which are supervised using the re-parameterized L_0 loss.

parameters to update the raw model, so we mask the input/output of the editor to apply an adaptive weighting. For an update in language l, we mask each parameter (tensor) $\boldsymbol{W}$ to be updated with $\boldsymbol{m}^l_{\boldsymbol{W}} \in [0,1]^{\dim \boldsymbol{W}}$ through

$$\text{mask}(\boldsymbol{W}, \boldsymbol{m}^l_{\boldsymbol{W}}) = \boldsymbol{W} + \boldsymbol{m}^l_{\boldsymbol{W}} \odot \boldsymbol{W},$$

where $\odot$ computes the element-wise production. The mask operation bypasses the whole parameter firstly, then increases the weight of the selected part. We also add an auxiliary L_0 loss

$$L_{\text{mask}} = \lambda_{\text{mask}} \sum_{l,\boldsymbol{W}} \|\boldsymbol{m}^l_{\boldsymbol{W}}\|_0,$$

which is a sparsity penalty to make the mask filter only the important components in a parameter. We follow Louizos et al. (2018) to optimize L_0 with their re-parametrization approach. It should be noted that the mask is only aware of and applied to the editing language because we aim to update all the languages simultaneously, making any assumption on the updating or retaining languages meaningless.

Unfortunately, the element-wise masks for each language may contain as many parameters as the raw model, causing over-parameterization and a waste of computation. Say h is the hidden size. If predicting the $O(h^2)$ updated parameters (or their gradients), the editor's parameters will inflate to unacceptable $O(h^4)$. Inspired by the capacity of the low-rank updating demonstrated in previous model editing work (Cao et al., 2021; Mitchell et al., 2022a), we factorize the full mask matrix into two low-rank matrics, then constructing the updated raw parameters with non-parameterized operations.

The proposed *language anisotropic* model editing can work with various model editing approaches, while the implementation is specific to the algorithm details. Taking a parameter matrix $\boldsymbol{W} \in \mathbb{R}^{n \times m}$ in an MLP layer for example. By the chain rule, its gradient on the loss L is

$$\nabla_{\boldsymbol{W}} L = \boldsymbol{x}^\top \boldsymbol{\delta},$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the layer's input, and $\boldsymbol{\delta} \in \mathbb{R}^m$ refers to the gradient of the layer's output (i.e., the "input" in the backward pass).

For hyper-network based approaches (Cao et al., 2021; Mitchell et al., 2022a), a network g is built to conduct gradient transform. Hence, we insert the language masks $\boldsymbol{m}^l_{\cdot}$ here as

$$\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\delta}} = g\left(\text{mask}(\boldsymbol{x}, \boldsymbol{m}^l_{\boldsymbol{x}}), \text{mask}(\boldsymbol{\delta}, \boldsymbol{m}^l_{\boldsymbol{\delta}})\right).$$

For other approaches that do not manipulate gradients (Sinitsin et al., 2020), the g is an identical transformation, and the language masks do not affect the rest part of the editing algorithm.

Finally we construct the full sized gradient using the rank-1 predictions

$$\tilde{\nabla}_{\boldsymbol{W}} L = \tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{\delta}}.$$

The extra parameters and computation is in the order of $O(h|\mathcal{L}|)$. Since the size of the language set $\mathcal{L}$ is likely to be tens while the hidden size h can easily reach the thousand level, the extra time-space cost is tiny compared to the original $O(h^2)$

order. To this end, we obtain an approach to make *language anisotropic* model editing.

4 Experiments

4.1 Evaluation

To evaluate cross-lingual model editing approaches, we focus on *cross-lingual transferability*, while continuing to keep our eyes on *reliability* and *locality*. Suppose that the languages we focus on make up $\mathcal{L}$, and the corpus is $\mathcal{D}_{\mathcal{L}}$. For $l \in \mathcal{L}$, each monolingual subset $\mathcal{D}_l$ of the corpus contains a number of tuples (x_k, y_k), which means we desire the model to predict y_k to the input x_k. The y_k does not need to be different from the raw prediction $f(x_k; \boldsymbol{\theta})$. Taking the union of datasets in all the languages, we have the cross-lingual model editing dataset $\mathcal{D}_{\mathcal{L}} = \cup_{l \in \mathcal{L}} \mathcal{D}_l$.

Inspired by Cao et al. (2021), we propose three cross-lingual model editing metrics. Overall, we distinguish the languages where inputs are to be edited from where predictions are to be updated. Let $\mathcal{D}_{\text{edit}}$ be the set of (input, desired prediction) pairs fed to edit the model, which cause model predictions to inputs in $\mathcal{D}_{\text{update}}$ updated. In addition, $I(x) = \{x' | x' \text{ is parallel to } x\}$ refers to parallel inputs of a specific input x across languages of interest.

To measure *reliability* under *cross-lingual transferability*, we use editing accuracy. We calculate the ratio of the predictions successfully updated

$$\text{acc} = \mathbb{E}_{\substack{(x_e, y_e) \sim \mathcal{D}_{\text{edit}} \\ x_u \sim \mathcal{D}_{\text{update}} \cap I(x_e)}} \left[\mathbb{1}_{f(x_u; \boldsymbol{\theta}_u(x_e, y_e)) = y_e} \right].$$

To measure *locality* under *cross-lingual transferability*, we use editing consistency which reflects the retaining rate of predictions to unrelated inputs

$$\text{con} = \mathbb{E}_{\substack{(x_e, y_e) \sim \mathcal{D}_{\text{edit}} \\ x_r \sim \mathcal{D}_{\text{update}} \setminus I(x_e)}} \left[\mathbb{1}_{f(x_r; \boldsymbol{\theta}_u(x_e, y_e)) = f(x_r; \boldsymbol{\theta})} \right].$$

The above two metrics are not necessarily consistent or even conflicting, similar to precision and recall in classification. Thus, we define the editing success rate as the harmonic mean

$$\text{succ} = \frac{2 \times \text{acc} \times \text{con}}{\text{acc} + \text{con}}.$$

Since evaluating over the full set for each edit has a huge overhead of enumerating every two inputs, we follow existing work on model editing (Cao et al., 2021; Mitchell et al., 2022a,b) to estimate it with mini-batched expectation. Notably, in this work $\mathcal{D}_{\text{edit}}$ and $\mathcal{D}_{\text{update}}$ are finite datasets. Thus we enumerate each $(x_e, y_e) \in \mathcal{D}_{\text{edit}}$, and uniformly sample a subset in certain size of testing inputs x_u from $I(x_e)$ or x_r from complement set of $I(x_e)$ (for acc or con, respectively) to make a pair in order to calculate the metrics.

To obtain an average metric over all the languages, we calculate the macro average over editing languages. Specifically, to avoid enumerating all language pairs, we mix all the languages into $\mathcal{D}_{\text{update}} = \mathcal{D}_{\mathcal{L}}$, and use edit sets of single language from $\{\mathcal{D}_l\}_{l \in \mathcal{L}}$ as $\mathcal{D}_{\text{edit}}$ one by one, then finally calculate the macro average. Finally, the success rate is calculated using the averaged accuracy rate and consistency rate.

4.2 Baselines

Finetuning As the most common baseline of model editing, we use finetuning (degenerated editor). With no editor to train, finetuning has no cross-lingual variant and makes no use of parallel corpus since no editor is to be trained.

Learned Editors Considering the proposed approaches are compatible with various learned editors, we use three monolingual editors as the basis: Editable Training (Sinitsin et al., 2020), KnowledgeEditor (Cao et al., 2021), and MEND (Mitchell et al., 2022a). We compare the editing performance of each editor with and without our approaches.

4.3 Datasets

Following the widely used setting, we construct synthetic editing datasets using existing data (Sinitsin et al., 2020; Cao et al., 2021; Mitchell et al., 2022a,b). We use the knowledge-intensive task mLAMA (Kassner et al., 2021) for fact editing, which is natural because predictions involve only specific knowledge, which is prone to change. Nevertheless, a usable dataset with a parallel corpus of another task, like classification, is lacking due to the difficulty in translating entities. Therefore, to demonstrate the generic task-agnostic editing ability of cross-lingual model editing, we also use a semantics-focused dataset XNLI (Conneau et al., 2018) for error correction.

mLAMA is a multilingual dataset of knowledge probing task through (masked) language modeling, providing facts expressed as masked sentences in 53 languages. Each fact is a triple $\langle$[X], type, [Y]$\rangle$ including two entities, e.g., $\langle$Messi, play-for, Paris SG$\rangle$. To produce the textual

Approach	Training Languages	mLAMA			XNLI		
		acc%	con%	succ%	acc%	con%	succ%
Finetuning	n/a	21.94	55.69	31.48	47.53	98.24	64.06
Editable Training	en only	51.13	17.33	25.88	71.02	**95.24**	81.36
Editable Training	all	**99.78**	**24.45**	**39.27**	**89.45**	93.04	**91.21**
KnowledgeEditor	en only	37.18	50.19	42.72	69.96	**96.79**	81.22
KnowledgeEditor	all	**64.69**	**53.00**	**58.26**	**86.20**	95.08	**90.42**
MEND	en only	24.76	61.09	35.24	84.90	94.87	89.61
MEND	all	**99.58**	**75.76**	**86.05**	**98.16**	**97.75**	**97.95**

Table 1: Experiment results to compare monolingual and cross-lingual model editing approaches. During evaluation, the editing language is limited to en, while the updating and retaining languages contains all languages.

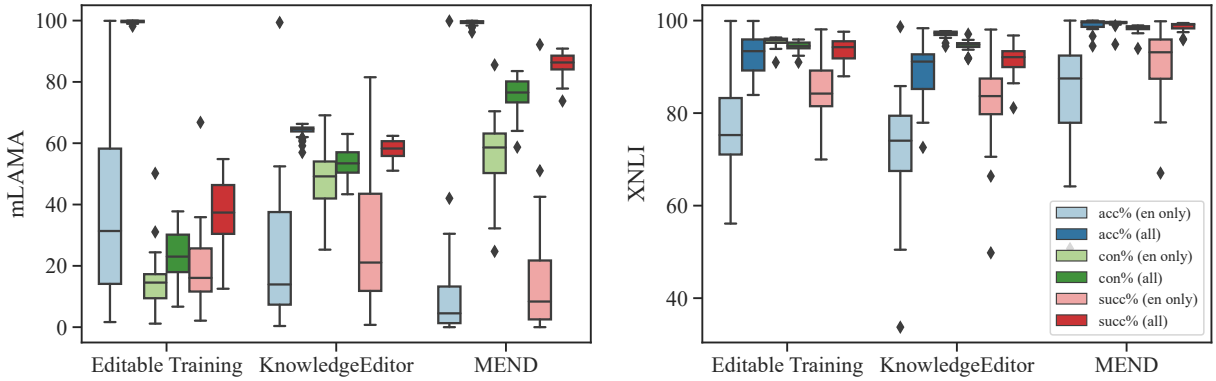

Figure 3: Editing performance varies across different languages. Training editors with parallel data improves overall editing performance, while decreasing the performance variance among languages.

expression from triples, mLAMA provides one template for each type ("play-for") of fact like "[X] plays for [Y].".

In the original setting of mLAMA, they fill the real [X] and replace [Y] with [MASK] tokens to probe the pre-trained language model. In our model editing setting, to construct the editing input, we also keep the [X] but uniformly sample an entity within the same type as [Y]. To measure the *locality*, we replace the [Y] as [MASK] tokens in a row, where the number of [MASK] tokens is sampled from the length distribution of entity name in the corresponding language. Note that translation of an entity may be invisible for the edited model or even nonexistent. Consequently, editing with entity names, which involves the entity linking problem, can be intractable in pure cross-lingual model editing. Therefore, we always treat the entity in the edit input as the desired prediction.

XNLI is a parallel corpus of natural language inference in 15 languages, which can be modeled as a three-way sentence pair classification task, where we ask the model to predict the relation between a premise-hypothesis pair in {entailment, neutral, contradiction}.

In the model editing scenario, we treat the premise-hypothesis pair as a whole input sentence to classify. Unfortunately, since the raw model has already been finetuned using the training and dev set, a dedicated training setting for error correction cannot be built. Thus, we train the editor to edit arbitrarily, which implies the error correction ability. During training, we sample edit input over the training set and give a uniformly random label as the desired prediction. To evaluate an editor on *reliability*, we use data in the test set that the raw model gives wrong predictions and use corresponding gold labels as the desired predictions. As for *locality*, we continue to sample inputs to be retained over the whole test set.

4.4 Cross-Lingual Model Editing

In this part, we demonstrate that the cross-lingual scenario exceeds the capability of the monolingual model editing approach. Specifically, we compare the editing performance of the monolingual approaches and the proposed cross-lingual variants. Recall that we use $\mathcal{L}$ to represent the full language set, i.e., the 15 languages for XNLI and the 53 for mLAMA. In the case of XNLI, the data is inher-

Approach	mLAMA			XNLI		
	acc%	con%	succ%	acc%	con%	succ%
Fintuning	10.14	48.68	16.79	56.48	98.54	71.81
Editable Training	97.39	21.90	35.75	90.02	93.58	91.76
w/ *Language Anisotropic* Model Editing	**97.87**	**24.41**	**39.08**	**91.79**	**93.68**	**92.72**
KnowledgeEditor	47.30	49.32	48.29	83.88	**95.79**	89.44
w/ *Language Anisotropic* Model Editing	**55.91**	**51.00**	**53.34**	**86.88**	95.45	**90.96**
MEND	94.83	67.59	78.92	98.16	97.44	97.80
w/ *Language Anisotropic* Model Editing	**96.12**	**69.20**	**80.47**	**98.42**	**98.02**	**98.22**

Table 2: Experiment results show that all three editors benifit from *language anisotropic* model editing on both datasets. All of the approaches are trained and evaluated in all languages.

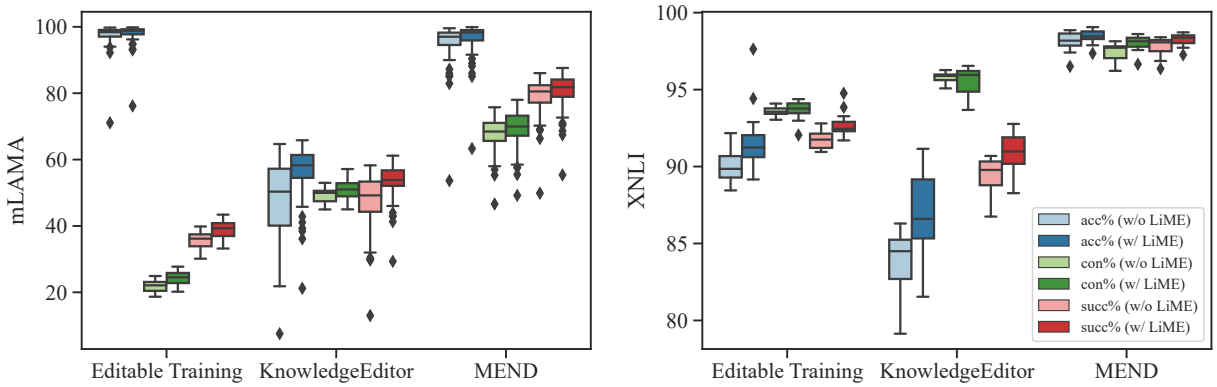

Figure 4: Distribution of editing performance across languages. *Language Anisotropic* Model Editing (LiME for short) provides overall performance improvement and closes the performance gap across languages.

ently parallel, while in mLAMA, each language, excluding English, relies on a translated subset of English. Given this scenario, we train the editors using the English subset to ensure uniform exposure to knowledge during training, thereby mitigating potential issues arising from training set disparities. More specifically, we select en as the editing language and expect the approaches to update predictions across all the languages. Hence, we have l_e = en, $l_u, l_r \in \mathcal{L}$ during evaluation. Table 1 shows the averaged results of en $\rightarrow$ all the languages, while Figure 3 illustrates the distribution of results across languages.

Finetuning suffers from severe cross-lingual underfitting according to its low editing accuracy, causing a low overall success rate.

The monolingual editors work much better than finetuning. Although never seen other languages, the editors demonstrate partial *cross-lingual transferability*. Moreover, the editor acquires the ability to perform updates of *locality*, almost reaching all the highest editing consistency on XNLI.

However, only editors with the proposed cross-lingual editing training framework truly generalizes the desired prediction to inputs in other languages. On XNLI, editors trained cross-lingually on all languages improves the editing accuracy by a large margin, with much less loss of editing consistency, resulting in a large growth in editing success rate. On mLAMA, where the model faces a much larger output space, editors trained on all languages reveals its high consistency and improves all three metrics significantly. Moreover, Figure 3 shows that the performance gap across languages is closer under the cross-lingual training framework.

4.5 Language Anisotropic Model Editing

After confirming the effectiveness of the cross-lingual model editing, we conduct experiments to study how the proposed *language anisotropic* model editing improves performance. Here we always train and evaluate approaches in all languages ($l_e, l_u, l_r \in \mathcal{L}$). Table 2 shows the averaged all $\rightarrow$ all results, with the per-language distribution plotted in Figure 4. Editors using parallel training data in Table 1 are the same as the editors without *language anisotropic* model editing in Table 2. The difference is that we no longer limit the editing language, thus the editing task becomes harder, making the results in Table 2 lower.

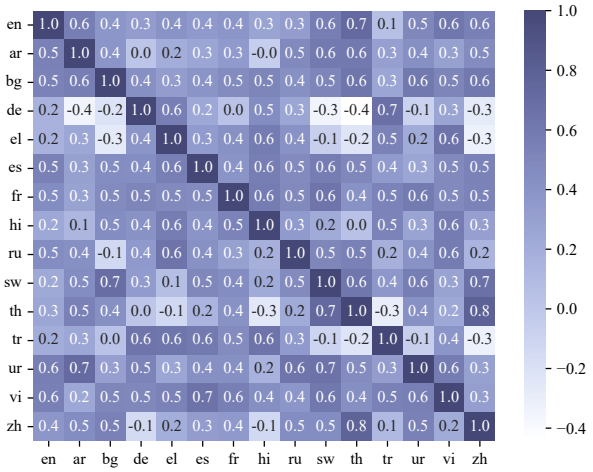

Figure 5: Cosine similarities of learned parameters of language-specific masks. In each row, we inspect the top-1% preferred dimensions of a certain language l by value, on which dimensions we calculate the cosine similarities between l and every languages.

Finetuning still falls into underfitting across languages, performing similar to the situation of single editing language.

With *language anisotropic* model editing, the performance of editors reach a new high in both datasets. Note that on XNLI, the small growth ($97.80\% \rightarrow 98.22\%$) corresponds to the large error reduction ($2.20\% \rightarrow 1.78\%$, 19% relatively).

Though trained with parallel data, a performance gap still exists between some languages and the others. *Language anisotropic* model editing helps the editors close the performance gap and increases the overall editing success rates.

To illustrate the function of the language-specific masks, we conduct analyses using one of the final MEND based checkpoints on XNLI. We observe that the parameters of the masks are very close in most dimensions across all languages. However, masks for different languages show preferences in small but different dimension subsets. Therefore, we plot the cosine similarities of learned parameters in the masks as a heatmap in Figure 5, where we limit the size of the preferred subset to 1% of the full hidden size. The heatmap of cosine similarities demonstrates that *language anisotropic* model editing captures the *language anisotropy* feature of the multilingual pre-trained language model. Through adaptively re-weighting gradients of a small subset of parameters for each language, *language anisotropic* model editing improves the performance of cross-lingual model editing.

5 Related Work

Model Editing Sinitsin et al. (2020) initially presents the model editing problem and proposes a MAML-like method, called Editable Training. Our cross-lingual model editing problem definition and metrics mostly extend their work. The proposed *language anisotropic* model editing approach can be applied to Editable Training by using the rank-1 masks to construct a full gradient/parameter mask.

A series of work models editing as a learning-to-update problem and develops the hyper-network based approaches, such as KnowledgeEditor (Cao et al., 2021), MEND (Mitchell et al., 2022a), and SLAG (Hase et al., 2021). They build the editor to constrain gradients during finetuning. We gain a lot of inspiration from their work when designing our methods.

A category of approaches regard the language model as a knowledge base, and utilize a wider range of editing fomulars (Santurkar et al., 2021; Meng et al., 2022a,b; Geva et al., 2021; Dai et al., 2022; Mitchell et al., 2022b). We can obtain the cross-lingual variants using the parallel corpus, while whether the *language anisotropic* model editing works depends on the algorithm details.

Cross-Lingual Transferability In recent work, multilingual pre-trained language models show their *cross-lingual transferability* (Devlin et al., 2019; Conneau et al., 2020; Xue et al., 2021; Chi et al., 2021), where multiple languages included in the training corpus benifit each other. Opposite to the positive cross-lingual transfer, Wang et al. (2020) study the negative interference phenomenon. They show the existence of language-specific parameters, which is also a theoretical basis of our work. Based on this priori, their work and our proposed *language anisotropic* model editing have similar underlying ideas: identifying the language-specific parameters and using them to improve the *cross-lingual transferability*.

Though, our work differs from theirs in method and task. They leverage language-specific pruning to identify the preferred parameter subset of different languages. Then they propose an iterative second-order meta-optimizing algorithm to improve pre-training. Our approach does not perform prune, where the masks play the role of reweighting coefficients. Our approach also makes no change in the training algorithm, maintaining maximum compatibility with various model editing approaches.

6 Conclusion

In this work, we define the task and metrics of cross-lingual model editing. After summarizing the training formula of various monolingual model editing approaches, we naturally extend the formula to a cross-lingual variant using the parallel corpus. Further, we propose *language anisotropic* model editing to improve cross-lingual model editing. We conduct experiments to verify that the cross-lingual model editing problem is necessary and find that the proposed approaches are effective.

Limitations

Our work depends mainly on parallel data. Although tasks focusing on language abilities can leverage machine translation to obtain parallel data (Hu et al., 2020), it is much harder for tasks about knowledge and facts. Using parallel data to train cross-lingual model editors is like doing full supervision, while we need to leverage weakly labeled data to mitigate data scarcity.

On the other hand, whether monolingual or cross-lingual, model editing still struggles with the continual learning problem. In the real world, knowledge constantly emerges and fades, disabling the stop of learning. However, most studies, including our work, focus on a single or a batch of inputs. Thus, an effective solution of continuously updating a series of inputs is necessary before model editing becomes a practical technic.

Note that our work focuses on the editor's generalized cross-lingual editing ability. We expect the editor to perform the editing honestly. This target potentially offers the possibility to modify model behavior maliciously. Though editing may not soon become a practical technic, the potential risk does exist.

Acknowledgement

This work was supported by the National Key R&D Program of China via grant 2020AAA0106501 and the National Natural Science Foundation of China (NSFC) via grant 62236004 and 61976072.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6491–6506. Association for Computational Linguistics.

Wanxiang Che, Jiang Guo, and Yiming Cui. 2021. *Natural Language Processing: A Pre-trained Model Approach*. Publishing House of Electronics Industry, Beijing, China.

Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2021. Infoxlm: An information-theoretic framework for cross-lingual language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 3576–3588. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8440–8451. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7057–7067.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2475–2485. Association for Computational Linguistics.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. 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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5484–5495. Association for Computational Linguistics.

Peter Hase, Mona T. Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. 2021. Do language models have beliefs? methods for detecting, updating, and visualizing model beliefs. *CoRR*, abs/2111.13654.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *CoRR*, abs/2003.11080.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021. Multilingual LAMA: investigating knowledge in multilingual pretrained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 3250–3258. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual knowledge in GPT. *CoRR*, abs/2202.05262.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *CoRR*, abs/2210.07229.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2463–2473. Association for Computational Linguistics.

Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. 2021. Editing a classifier by rewriting its prediction rules. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 23359–23373.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry V. Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Lewis Tunstall, Leandro von Werra, and Thomas Wolf. 2022. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media, Incorporated.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Zirui Wang, Zachary C. Lipton, and Yulia Tsvetkov. 2020. On negative interference in multilingual models: Findings and A meta-learning treatment. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4438–4450. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November*

16-20, 2020, pages 38–45. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 483–498. Association for Computational Linguistics.

A Datasets Preprocessing

A.1 mLAMA

Along with the raw English data from LAMA (Petroni et al., 2019), mLAMA provides translations of the facts in the other 52 languages if possible. mLAMA is organized into two-level. The first level is relations, and the second is facts. Facts in the same relation share the same template and can be identified by the ⟨`[X]`, `[Y]`⟩. Thus, we split at the fact level for consistency across different data splits. Specifically, we split the whole dataset to train/dev/test with a ratio of 8:1:1, resulting in 628,612/78,555/78,600 facts in the train/dev/test set.

In our setting, the template with filled `[X]` is used as input, and then we test if the edited model predicts `[Y]` (§4.3). Thus, to avoid leakage and keep large output space, we ensure that an `[X]` can only appear in one split while not limiting the label `[Y]`. Take an example of relation P19 (`[X]` was born in `[Y]` .) where `[X]` is a person's name and `[Y]` is a location. In the training set, if an input is "`Allan Peiper` was born in `[MASK]` .", there cannot exist an input in the dev/test set with `[X]` = `Allan Peiper`. On the contrary, a `[Y]` (like "`Alexandra`") can be used as the desired prediction during both training and testing, because we use it as a label. We first exclude samples in the test set having overlap `[X]` with the training set, then exclude samples in the dev set overlapping with the train/test. Finally, we obtain 628,612/23,718/53,993 facts in the train/dev/test set after preprocessing summing up all languages.

A.2 XNLI

XNLI dataset contains completely parallel data in fifteen languages. We use Huggingface Datasets to access to XNLI dataset, and follow the official split with 392,702/2,490/5,010 samples in train/dev/test set in each language.

B Experiment Details

We conduct all experiments three times and use the mean of editing success rates as the final performance metric, including main experiments and hyperparameter tuning.

B.1 Model and Implementation

We use `bert-base-multilingual-cased` from Hugging Face Transformers (Wolf et al., 2020) as the basic model. As the basic model editors, we use MAML-like Editable Training (Sinitsin et al., 2020), and hyper-network based KnowledgeEditor (Cao et al., 2021) and MEND (Mitchell et al., 2022a). For XNLI, the pre-trained model finetuned on the en training set is used as the raw model to edit in all the following experiments. For mLAMA, we use the pre-trained language model directly.

We work on the official MEND codebase, together with HuggingFace Transformers and Datasets. During preliminary experiments, we find that MEND in their implementation suffers from low computation efficiency and sub-optimization due to the token-level BatchNorm (Ioffe and Szegedy, 2015) variant used by the editor. Thus, we replace the token-level BatchNorm in MEND editor with LayerNorm (Ba et al., 2016) in our implementation.

B.2 Hyperparameters

Finetuning We use Adam (Kingma and Ba, 2015) optimizer with learning rate of 5×10^{-6}. For each input to be edited, the maximum step is set to 100.

Learned Editors We follow Mitchell et al. (2022a) in most of the hyperparameter settings of the three monolingual editing approaches we use. For Editable Training and KnowledgeEditor, we set the learning rate to 5×10^{-5}. For MEND, the editor is initialized to an identical mapping and trained by Adam optimizer with the learning rate of 1×10^{-6}. For the inner gradient decent updating, the learning rate is set to 1×10^{-4}. For the coefficients of losses, we set $\lambda_{\text{rel}} = 0.1$ and $\lambda_{\text{loc}} = 1.0$.

Since `bert-base-multilingual-cased` is used as the raw model, we follow the setting of `bert-base` of the original MEND, i.e., editing MLPs in the last three layers of the encoder, leaving the other parameters frozen.

Language Anisotropic Model Editing The *language anisotropic* varients inherit hyperparameters

for architectures and training from their corresponding base editors.

The newly introduced training hyperparameters include the learning rate of masks and λ_{mask}. We tune the learning rate of masks in $\{1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}\}$, and λ_{mask} in $\{0.01, 0.1, 1\}$. We pick the best values and apply them to all main experiments, i.e., learning rate of masks of 1×10^{-3}, and λ_{mask} of 1.0.

B.3 Training Details

We utilize the early-stopping strategy along with up to $500,000$ training steps. When training on the full datasets, we evaluate the model every $100,000$ steps and finalize training when the editing success rate is not improved over $200,000$ steps. When training on the English only subset, the validation interval is set to $20,000$ and the early stop patience is $40,000$ steps.

All experiments fit in one NVIDIA RTX 2080Ti GPU, where a single run takes one to three days.

C Additional Results

The large versions with raw data points of Figure 3 and Figure 4 are as follows.

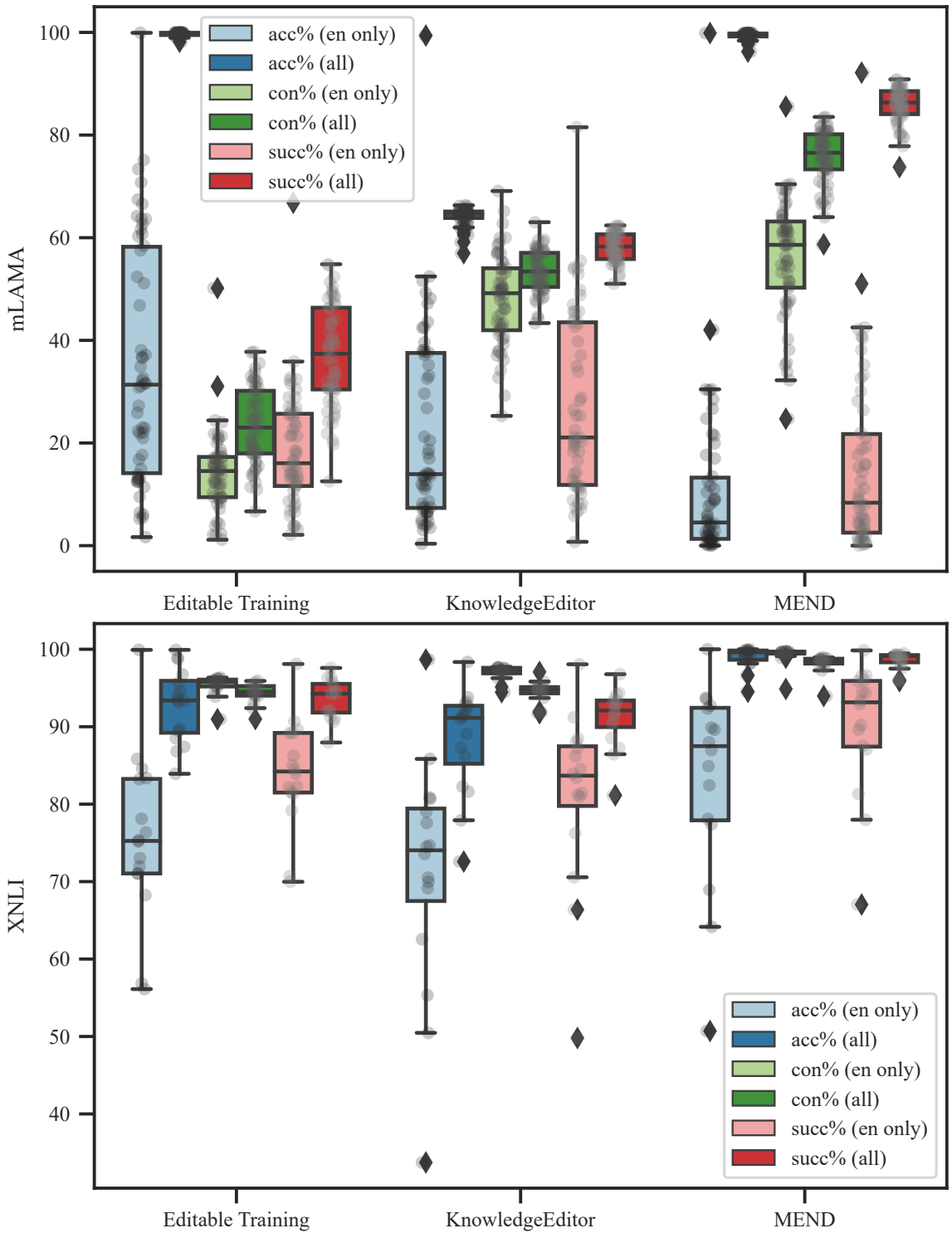

Figure 6: A large version of Figure 3, where each grey point corresponds to an en $\rightarrow l$ result averaged over three runs.

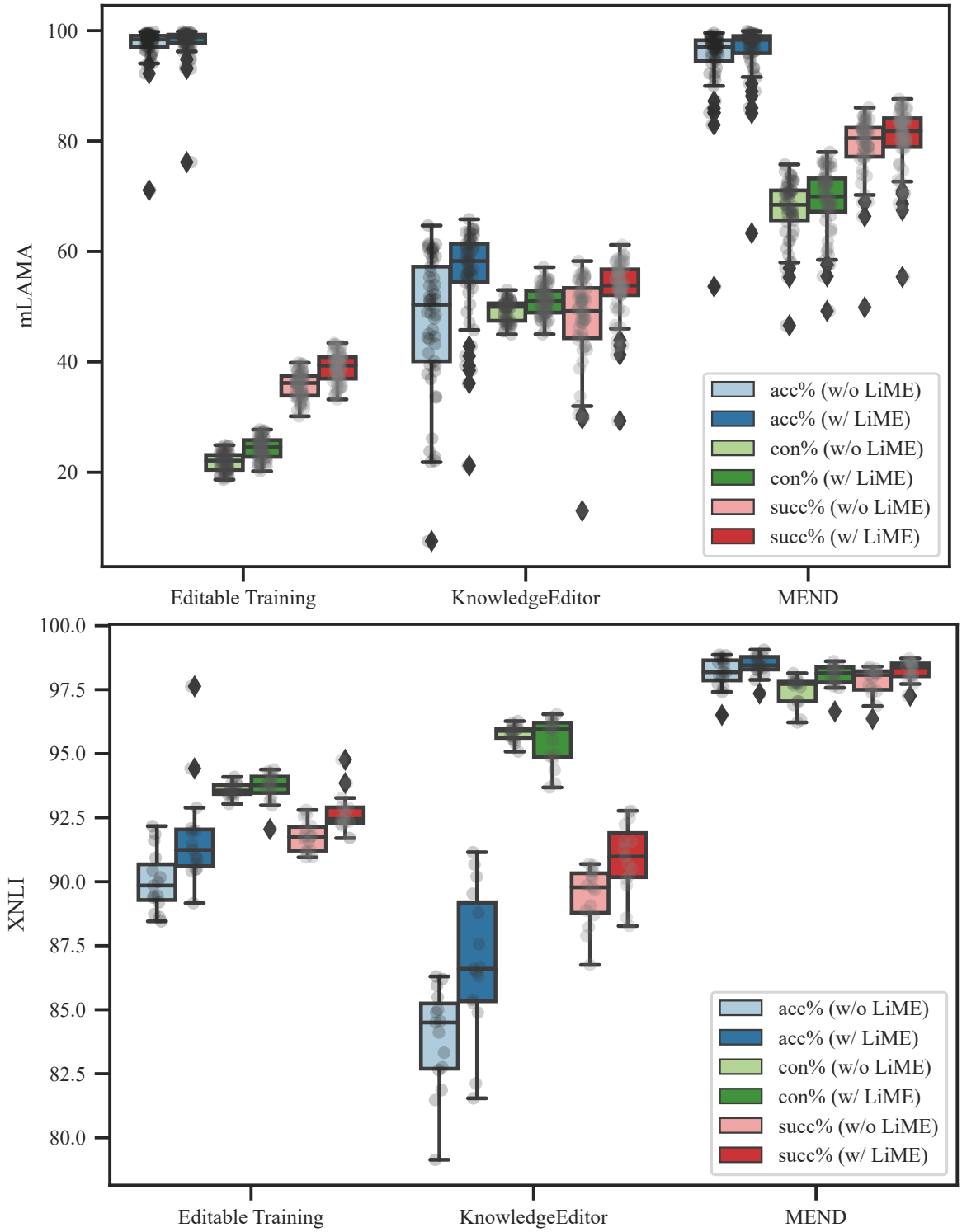

Figure 7: A large version of Figure 4, where each grey point corresponds to an $l \rightarrow$ all result averaged over three runs. LiME for *Language Anisotropic* Model Editing.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☑ A2. Did you discuss any potential risks of your work?
Limitations

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4 and Appendix A

☑ B1. Did you cite the creators of artifacts you used?
Section 4 and Appendix A

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
License and terms will be included in the code repository to be released.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 4 and Appendix A

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4 and Appendix A

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix B

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix B

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4 and Appendix B

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix B

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.