electronics

MDPI

Article

Merging Methods for Multilingual Knowledge Editing for Large Language Models: An Empirical Odyssey

Kunil Lee [1,2], Ki-Young Shin [2], Jong-Hyeok Lee [1,3] and Young-Joo Suh [4,*]

[1] Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), 77 Cheongam-ro, Nam-gu, Pohang 37673, Republic of Korea; skywalker@postech.ac.kr (K.L.); jhlee@postech.ac.kr (J.-H.L.)
[2] AI Lab, Designovel Co., Ltd., CHANGeUP GROUND Pohang, 87 Cheongam-ro, Nam-gu, Pohang 37673, Republic of Korea; shinky@designovel.com
[3] LLSOLLU, 5 Mabang-ro 10-gil, Seocho-gu, Seoul 06771, Republic of Korea
[4] Graduate School of Artificial Intelligence, Pohang University of Science and Technology (POSTECH), 77 Cheongam-ro, Nam-gu, Pohang 37673, Republic of Korea
* Correspondence: yjsuh@postech.ac.kr

Abstract

Multilingual knowledge editing (MKE) remains challenging because language-specific edits interfere with one another, even when locate-then-edit methods succeed in monolingual settings. We study whether vector merging—combining independently computed per-language updates into a single edit—can mitigate this interference. We evaluate six merging variants with two backbone large language models, two base knowledge editing methods, and 12 languages on the MzsRE benchmark under a large-scale batch-editing setting, and we examine how the weight scaling factor and the rank compression ratio affect editing performance. Summation with shared covariance proves the most reliable strategy overall, whereas naive summation without shared covariance performs poorly. Task Singular Vectors for Merging (TSVM) helps only in specific settings, so its ability to reduce multilingual interference is limited. Performance is also sensitive to both weight scale and rank ratio, with larger-than-default scaling and relatively low rank often yielding the best results. When the results are analyzed by language-resource level, the choice of merging method matters most for the relatively low-resource languages, such as Thai and Vietnamese. These findings clarify the practical strengths and limits of current vector merging methods for MKE and provide guidance for future multilingual knowledge editing research.

Keywords: large language models; multilingual knowledge editing; locate-then-edit; vector merging

Check for updates

Academic Editors: Colin Layfield and John Abela

Received: 4 May 2026
Revised: 31 May 2026
Accepted: 18 June 2026
Published: 22 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Modern large language models (LLMs) [1–4] contain billions to trillions of parameters, enabled by the scalability of the Transformer architecture [5]. Training such models requires substantial GPU compute and energy, and it produces considerable greenhouse gas emissions. To update knowledge stored in LLM parameters, efficient fine-tuning methods [6] have been developed, but they are still time-consuming [7]. Fortunately, knowledge editing (KE) provides a framework for modifying target knowledge while preserving unrelated knowledge at a much lower cost [8]. In particular, *locate-then-edit* KE methods can directly manipulate LLM parameters without additional learning [9–11]. However, most prior

work has been conducted in English, and performance degradation has been observed in cross-lingual settings (e.g., edit in English → test in Chinese) [12]. Moreover, interference between languages has been reported in multilingual settings (e.g., edit and test in multiple languages simultaneously) [13]. In particular, multilingual knowledge editing (MKE) performs substantially worse than monolingual KE given the same requests. Recently, a vector merging method called Task Singular Vectors for Merging (TSVM) was shown to effectively reduce task interference [14], which suggests a possible analogy to our problem.

Multilingual knowledge editing matters whenever a model is deployed across many languages. Modern LLMs power global products and multilingual assistants, so a single factual update—a newly elected head of state, a change in corporate ownership, or a corrected date of birth—must remain consistent in every language the model serves. Editing only English leaves the model factually wrong when the same question is asked in Chinese, Thai, or Czech [12], which is unacceptable for user-facing assistants and for regulatory or regional factual corrections. A natural remedy is to run locate-then-edit independently for each language, but the resulting per-language updates must eventually be combined into a single deployed model, and combining them naively causes exactly the cross-language interference we study. MKE therefore asks how to apply many factual edits in many languages at once while keeping them mutually consistent, which motivates the research questions below.

In this paper, we explore the possibility of applying vector merging methods to multilingual knowledge editing (MKE). It is worth clarifying the difference between cross-lingual KE and MKE. In cross-lingual KE, evaluation is performed in languages that are not available at editing time. In MKE, by contrast, both editing and evaluation are performed in multiple languages simultaneously. We begin with the following research questions (RQs):

- RQ1: How do vector merging methods perform in MKE?
- RQ2: Can TSVM effectively mitigate multilingual interference?
- RQ3: How do factors such as weight scale and rank ratio affect performance? We raise this question because both factors are fixed to defaults in prior locate-then-edit work, and their effect on MKE has not been measured.
- RQ4: Do merging methods benefit low-resource languages such as Thai and Vietnamese, and which merge operator best preserves their edits when many languages are edited jointly? We raise this question because low-resource languages score lowest in the aggregate results, and a single deployed model must serve them alongside high-resource languages.

Vector merging methods, however, originate from a different problem: combining separately fine-tuned weights into a single weight update. We therefore design a systematic study of several merging methods while controlling for backbone type, base KE method, shared covariance, weight scale, and rank compression ratio. Our contributions are as follows:

- To the best of our knowledge, this is the first systematic study of locate-then-edit MKE with mass editing in 12 languages in parallel (batch size $= 700 \times 12$). Previous work [13,15] used the same dataset [15], but considered only one editing request across 12 languages at a time (batch size $= 12$).
- We evaluate six merging methods with two backbones and two base KE methods, and observe that vector summation with shared covariance achieves the strongest overall performance. We also show that TSVM can reduce interference under limited conditions, but in general, none of the tested merging methods effectively closes the gap between MKE and monolingual KE.
- We are the first to analyze the weight scaling factor in MKE, and find that the optimal scaling factor exceeds the default value of 1.0 in most settings.
- We also investigate the effect of the rank compression ratio on TSVM performance and find that relatively low rank often leads to better results.

https://doi.org/10.3390/electronics15122747

- We analyze performance by language-resource level and find that the choice of merging method has the largest effect on the relatively low-resource languages, where shared covariance summation and the orthogonalizing TSVM merge diverge most (RQ4).

2. Related Work

2.1. Knowledge Editing for Large Language Models

Knowledge editing aims to update factual associations in large language models without retraining the entire model. Compared with conventional fine-tuning methods such as LoRA [6], KE seeks to modify targeted knowledge while preserving unrelated behavior at a substantially lower cost. Existing KE methods span several paradigms, including early hyper-network-based approaches such as KnowledgeEditor [16], learned editors such as MEND [7], semi-parametric methods such as SERAC [17], and in-context editing methods such as IKE [18]. Among these lines of work, our study is most closely related to direct parameter-editing approaches.

More specifically, locate-then-edit methods are motivated by the view that factual associations are stored in transformer feed-forward layers as key-value memories [19]. ROME [9] first showed that factual knowledge can be localized and modified through closed-form updates to specific layers. MEMIT [10] extended this idea to the mass editing setting, PMET [20] improved editing precision, and AlphaEdit [11] further refined the approach by constraining updates with a null-space projection. Complementary work has also examined knowledge localization at the neuron level: Knowledge Neurons [21] identified neurons associated with factual recall and demonstrated small-scale factual editing through neuron intervention. Although these methods achieve strong results in monolingual settings, multilingual knowledge editing remains relatively underexplored. Prior work has shown that cross-lingual KE degrades when editing and evaluation are performed in different languages [12], and more recent work has extended this challenge to cross-lingual multi-hop settings [22]. More directly related to our problem, MzsRE [15] provides a multilingual benchmark covering 12 languages, and LU-LAFNs [13] reduces interference by identifying language-agnostic factual neurons. In contrast, we focus on mass multilingual editing and ask whether vector merging can mitigate interference among language-specific edits.

2.2. Task Vectors and Model Merging

Model merging studies how to combine parameter updates from multiple tasks or fine-tuned models into a single model without retraining from scratch. The simplest strategies sum or average parameter differences, but these approaches often suffer from task interference when the underlying updates are not well aligned. This issue is closely related to our setting, where language-specific editing vectors may contain both shared information and conflicting components.

Representative work in this area includes Model Soups, which showed that averaging weights from multiple fine-tuned models can improve robustness when the solutions lie in a shared basin [23]; Task Arithmetic, which demonstrated that task vectors extracted from fine-tuned models can be composed through simple algebraic operations [24]; and TIES-Merging, which explicitly addresses redundant updates and sign conflicts across task vectors [25]. Our work is most directly inspired by Task Singular Vectors for Merging (TSVM) [14], which introduces a low-rank view of task updates and shows that singular-vector-based compression can reduce interference during model merging. Although TSVM was originally proposed for merging task-specific model updates rather than multilingual knowledge edits, its underlying motivation is highly relevant to MKE. This connection motivates our study of summation-, averaging-, and TSVM-based rules for merging language-wise editing vectors. At the same time, our setting differs from standard model

merging: rather than combining independently fine-tuned task models, we merge editing vectors produced by locate-then-edit KE methods. Our work therefore brings together two previously separate lines of research, namely, direct knowledge editing for LLMs and interference-aware vector merging.

3. Preliminaries

3.1. Large Language Models

Structurally, a large language model (LLM) is an architecture built from stacked Transformer [5] blocks. Within each block, two distinct modules operate in tandem. First, an attention layer transforms contextual input into short-term memory. This is paired with a position-wise feedforward network (FFN) that selects active slots from massive vector stores, effectively functioning as the model's long-term memory. Finally, all outputs from these modules are summed with the previous layer's activations in a residual fashion. More specifically, the output activation of $\boldsymbol{x}$ at layer l within the model, denoted as $\boldsymbol{h}^l$, can be defined as follows:

$$\boldsymbol{h}^l = \boldsymbol{h}^{l-1} + \boldsymbol{a}^l + \boldsymbol{m}^l, \tag{1}$$

$$\boldsymbol{m}^l = \boldsymbol{W}^l_{\text{out}}\,\sigma(\boldsymbol{W}^l_{\text{in}}\,\gamma(\boldsymbol{h}^{l-1} + \boldsymbol{a}^l)\,), \tag{2}$$

where $\boldsymbol{a}^l$ and $\boldsymbol{m}^l$ denote the outputs of the attention layer and the FFN layer, respectively; $\boldsymbol{W}^l_{\text{in}}$ and $\boldsymbol{W}^l_{\text{out}}$ denote the weight matrices of the FFN layers; σ denotes the non-linear activation function; and γ denotes the layer normalization. Here we follow expressions defined in [9,11].

3.2. Linear Associative Memory

A Linear Associative Memory (LAM) receives a key $\boldsymbol{k}$ and recalls its corresponding value $\boldsymbol{v}$ as follows:

$$\boldsymbol{v} = \boldsymbol{W}\boldsymbol{k}, \forall(\boldsymbol{k}, \boldsymbol{v}) \in \boldsymbol{S}, \tag{3}$$

where $\boldsymbol{W}$ is the weight matrix of the LAM, designed to memorize $(\boldsymbol{k}, \boldsymbol{v})$ associations within a finite set $\boldsymbol{S}$. Under this framework, an FFN layer from Equation (2) can be interpreted as follows [19]:

$$\underbrace{\boldsymbol{m}^l}_{\boldsymbol{v}} = \boldsymbol{W}^l_{\text{out}}\,\underbrace{\sigma(\boldsymbol{W}^l_{\text{in}}\,\gamma(\boldsymbol{h}^{l-1} + \boldsymbol{a}^l)\,)}_{\boldsymbol{k}}, \tag{4}$$

where the down-projection matrix $\boldsymbol{W}^l_{\text{out}}$ serves as the LAM weight matrix.

3.3. Locate-Then-Edit Methods

In knowledge editing, we study *knowledge* (like facts) of the form (subject s, relation r, object o). For example, in the sentence "The president of the U.S. is Joe Biden," the subject is "the United States," the relation is "the president," and the object is "Joe Biden." An editing request is also a triple $(s, r, o \rightarrow o')$, where o' is the new object we want to modify; for example, ("the United States", "the president", "Joe Biden" $\rightarrow$ "Donald Trump").

The *locate-then-edit* framework assumes critical information used to predict object o given subject s and its relation r, which is stored within the key-value structure of the FFNs defined in Equation (4). Like a standard LAM, the FFNs recall facts from a certain input (such as a prompt about a subject and a type of relation) and contribute significantly to predicting the next tokens associated with the correct object.

Under this framework, the overall KE process is divided into two steps: (1) identifying the specific Transformer layers responsible for storing knowledge, and (2) applying a KE algorithm to these layers sequentially in bottom-to-top order. The identification of these layers is conducted via causal tracing techniques [9,10]. Once the identification has been

established for a specific model architecture, further recomputation is generally unnecessary; consequently, subsequent research typically adopts these standardized layer ranges for editing tasks. For each target layer, the KE algorithm computes a matrix perturbation $\boldsymbol{\Delta}$ based on editing requests for the weight matrix $\boldsymbol{W}_{\text{out}}$ defined in Equation (4) (we omit the layer index l in the rest of this paper for brevity). The resulting updated weight matrix, $\boldsymbol{W}'_{\text{out}} = \boldsymbol{W}_{\text{out}} + \boldsymbol{\Delta}$, is intended to encode the newly modified knowledge.

Formally, suppose we have n editing requests and a set of corresponding keys and values for newly updated knowledge, $\boldsymbol{S}_1 = \{(\boldsymbol{k}_i, \boldsymbol{v}_i) | 0 < i \leq n\}$, and concatenate these keys and values into matrices, which are denoted as follows:

$$\boldsymbol{K}_{req} = [\boldsymbol{k}_1 | \boldsymbol{k}_2 | \ldots | \boldsymbol{k}_n] \in \mathbb{R}^{h \times n}, \quad \boldsymbol{V}_{req} = [\boldsymbol{v}_1 | \boldsymbol{v}_2 | \ldots | \boldsymbol{v}_n] \in \mathbb{R}^{d \times n}; \tag{5}$$

and similarly, $\boldsymbol{K}_{const}$ and $\boldsymbol{V}_{const}$ denote keys and values for other knowledge which should be preserved. Then we have the following:

$$\boldsymbol{\Delta} = \arg\min_{\tilde{\boldsymbol{\Delta}}} (\left\| (\boldsymbol{W}_{out} + \tilde{\boldsymbol{\Delta}}) \boldsymbol{K}_{req} - \boldsymbol{V}_{req} \right\|^2 + \left\| (\boldsymbol{W}_{out} + \tilde{\boldsymbol{\Delta}}) \boldsymbol{K}_{const} - \boldsymbol{V}_{const} \right\|^2). \tag{6}$$

We can obtain a closed-form solution for this problem by applying a normal equation [10,26]:

$$\boldsymbol{\Delta}_{\textbf{MEMIT}} = \boldsymbol{R} \boldsymbol{K}_{req}^T \left(\lambda \boldsymbol{C}_{const} + \boldsymbol{C}_{req} \right)^{-1}, \tag{7}$$

where the error term $\boldsymbol{R} = \left(\boldsymbol{V}_{req} - \boldsymbol{W} \boldsymbol{K}_{req} \right)$; λ is a hyperparameter; and covariance matrices $\boldsymbol{C}_{const} = \boldsymbol{K}_{const} \boldsymbol{K}_{const}^T$ and $\boldsymbol{C}_{req} = \boldsymbol{K}_{req} \boldsymbol{K}_{req}^T$. This method is often called **MEMIT**. If we further apply a null-space constraint [11], we have the **AlphaEdit** method as follows:

$$\boldsymbol{\Delta}_{\textbf{AlphaEdit}} = \boldsymbol{R} \boldsymbol{K}_{req}^T \boldsymbol{P} \left(\lambda \boldsymbol{I} + \boldsymbol{C}_{req} \boldsymbol{P} \right)^{-1}, \tag{8}$$

where $\boldsymbol{P}$ is the null-space projection matrix (null-space of $\boldsymbol{C}_{const}$); λ is a hyperparameter. In the error term $\boldsymbol{R}$, $\boldsymbol{V}_{req}$ means ideal values that a perfectly modified model can recall for keys $\boldsymbol{K}_{req}$, which is usually calculated with approximation by "network hooking" methods [9,10].

In this paper, we only consider a one-step batch editing scenario, which means applying KE algorithms at once with all editing requests. The sequential editing problem is out of the scope of this work.

4. Merging Methods for Multilingual Knowledge Editing

4.1. Problem Definition

In a multilingual knowledge editing (MKE) scenario, suppose that we have n editing requests and each request was written in m languages; we can calculate Δ using Equation (7) or Equation (8) for each language and apply it to the original model as follows:

$$\boldsymbol{W}^{new} = \boldsymbol{W}^{old} + \alpha \boldsymbol{\Delta}^{merged}, \tag{9}$$

$$\boldsymbol{\Delta}^{merged} = f(\boldsymbol{\Delta}^1, \boldsymbol{\Delta}^2, \ldots, \boldsymbol{\Delta}^m), \tag{10}$$

where $f(\cdot), \underbrace{\mathbb{R}^{d \times h} \times \ldots \mathbb{R}^{d \times h}}_{m} \Rightarrow \mathbb{R}^{d \times h}$ denotes the merging function and α denotes the scaling factor. For each $\boldsymbol{\Delta}^i$ of i-th language, we call it the knowledge editing vector for the i-th language.

4.2. Merging Functions

Table 1 summarizes the six merging functions studied in this work, together with the rationale for each. In this paper, we study the following merging functions for Equation (10).

To calculate the covariance matrix in Equation (7) or Equation (8) for each language, we can either (1) gather covariance only in that language: $\boldsymbol{C}_{req}^{i} = \boldsymbol{K}_{req}^{i}(\boldsymbol{K}_{req}^{i})^{T}$; or (2) gather across all languages: $\boldsymbol{C}_{req}^{i} = \sum_{j=1}^{m} \boldsymbol{K}_{req}^{j}(\boldsymbol{K}_{req}^{j})^{T}$. We first introduce merging methods based on $\boldsymbol{\Delta}^{i}$s calculated under the former case as follows:

- Sum:

$$\boldsymbol{\Delta}^{merged} = \sum_{i=1}^{m} \boldsymbol{\Delta}^{i}. \tag{11}$$

- Mean:

$$\boldsymbol{\Delta}^{merged} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{\Delta}^{i}. \tag{12}$$

- TSVM [14]: First, we decompose each $\boldsymbol{\Delta}^{i}$ into $\boldsymbol{U}^{i}$, $\boldsymbol{\Sigma}^{i}$, $\boldsymbol{V}^{i}$ using Singular Value Decomposition (SVD), then select the top-k highest singular values and their corresponding column and row vectors from $\boldsymbol{U}^{i}$ and $\boldsymbol{V}^{i}$:

$$\tilde{\boldsymbol{U}}^{i} = \boldsymbol{U}^{i}[:, 1:k], \tilde{\boldsymbol{\Sigma}}^{i} = \boldsymbol{\Sigma}^{i}[:k], \tilde{\boldsymbol{V}}^{i} = \boldsymbol{V}^{i}[1:k]. \tag{13}$$

  The value k is determined as follows:

$$k = \lfloor r \cdot d \rfloor, \tag{14}$$

  where the factor r is a hyperparameter. It reflects the degree of rank compression: if r is equal to 1, that means no compression; if r is close to zero, the $\boldsymbol{\Delta}$ is highly compressed into a low-rank matrix. Subsequently, we concatenate resulted components across all languages as follows:

$$\begin{aligned} \boldsymbol{U}^{concat} &= \left[\tilde{\boldsymbol{U}}^{1} \mid \tilde{\boldsymbol{U}}^{2} \mid \ldots \mid \tilde{\boldsymbol{U}}^{m}\right], \\ \boldsymbol{\Sigma}^{concat} &= \begin{bmatrix} \tilde{\boldsymbol{\Sigma}}^{1} & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{\Sigma}}^{2} & \ldots & \boldsymbol{0} \\ \ldots & \ldots & \ldots & \ldots \\ \boldsymbol{0} & \boldsymbol{0} & \ldots & \tilde{\boldsymbol{\Sigma}}^{m} \end{bmatrix}, \\ \boldsymbol{V}^{concat} &= \left[\tilde{\boldsymbol{V}}^{1} \mid \tilde{\boldsymbol{V}}^{2} \mid \ldots \mid \tilde{\boldsymbol{V}}^{m}\right]^{T}. \end{aligned} \tag{15}$$

  To make $\boldsymbol{U}^{concat}$ orthogonal, we decompose $\boldsymbol{U}^{concat}$ to $\boldsymbol{P}$, $\boldsymbol{D}$, $\boldsymbol{Q}$, by applying SVD, and set $\boldsymbol{U}^{merged} = \boldsymbol{PQ}$ (similarly for $\boldsymbol{V}^{concat}$). Finally, we have the following:

$$\boldsymbol{\Delta}^{merged} = \boldsymbol{U}^{merged}\boldsymbol{\Sigma}^{concat}\boldsymbol{V}^{merged}. \tag{16}$$

In addition, we introduce merging methods based on $\boldsymbol{\Delta}^{i}$s calculated under the latter case (gather covariance across all languages), and name these methods with the suffix "-Cov" to distinguish them from the former ones.

- Sum-Cov: The same as Sum, but with $\boldsymbol{\Delta}^{i}$s calculated by shared covariance. This is similar to the method used in the previous work [13] that adapted MEMIT to MKE. However, they apply the editing algorithm with one request (in multiple languages) at a time; by contrast, we apply the editing algorithm with massive requests (in multiple languages) simultaneously in one batch.
- Mean-Cov and TSVM-Cov: The same as Mean and TSVM, but with $\boldsymbol{\Delta}^{i}$s calculated by shared covariance.

Table 1. Summary of the six merging functions studied in this work. The methods factor along two design axes: the operation used to merge the per-language editing vectors $\mathbf{\Delta}^i$, and whether the covariance matrix in Equations (7) and (8) is gathered per language or shared across all languages. We select these six variants to isolate the effect of each axis.

Method	Covariance	Merge Operation	Rank-Comp.	One-Line Rationale
Sum	Per-language	Summation	No	Naive additive merge; serves as the baseline.
Mean	Per-language	Averaging	No	Sum rescaled by $1/m$ to single-edit magnitude.
TSVM	Per-language	Low-rank SVD + orthogonal concat.	Yes	Tests whether low-rank structure reduces interference.
Sum-Cov	Shared	Summation	No	Shared covariance implicitly normalizes the update scale.
Mean-Cov	Shared	Averaging	No	Shared-covariance counterpart of Mean.
TSVM-Cov	Shared	Low-rank SVD + orthogonal concat.	Yes	Combines low-rank merging with shared covariance.

5. Experiments

5.1. Experimental Settings

5.1.1. Metrics

We follow [7,9–11] and evaluate token-level accuracy over four aspects of each edit: efficacy (on editing requests), generalization (on paraphrases), specificity (on unrelated questions), and portability (on one-hop reasoning questions) [13,15]. Throughout the paper, we report a single unified metric, the averaged accuracy, defined as the arithmetic mean of these four aspects; we do not report the four aspect scores separately. The formal definition of the accuracy function and of the four aspects is given in Appendix A.

This token-level, teacher-forced accuracy is widely used in typical locate-then-edit methods such as ROME, MEMIT, and AlphaEdit [9–11]; we adopt the same metric because MEMIT and AlphaEdit are the base editing methods in our study, so editing Efficacy is measured consistently with the methods we build on: it measures whether the edited model assigns the highest probability to each gold answer token, conditioned on the prompt and the preceding gold tokens. As a stricter complement, we additionally report exact-match (EM) results—where a request counts as correct only if every gold token is predicted exactly—for all main-table settings in Appendix E; the relative ordering of the merging methods is preserved under EM.

5.1.2. Dataset

We use the test split of MzsRE [15], which is built upon the zsRE relation-extraction benchmark [27], in our experiments. It contains 700 samples, each including one editing request, one paraphrase of an editing request, one unrelated question and answer, and one one-hop altered question and answer of an editing request, which are written in 12 languages: English (en), Chinese (zh), Czech (cz), Vietnamese (vi), Turkish (tr), French (fr), Spanish (es), German (de), Russian (ru), Dutch (du), Portuguese (pt), and Thai (th). Note that we follow the language-code notation of the original MzsRE benchmark [15]. In ISO 639-1, these correspond to cz = cs (Czech) and du = nl (Dutch).

5.1.3. Backbones and Base Methods

We use Llama3.1-8B-Instruct [2] and Qwen2.5-7B-Instruct [3] as our backbones. Following related work, we target the critical layers 4–8 for editing in both backbones. The choice of these layers follows the causal tracing analysis of ROME and MEMIT [9,10] and

the early-to-middle critical-layer band subsequently adopted by AlphaEdit [11] and related locate-then-edit work for 7–8B decoder LLMs. The two backbones are decoder transformers of comparable depth (Llama3.1-8B has 32 layers and Qwen2.5-7B has 28), so their factual-association layers fall in the same range; using a common, literature-standard band also removes per-model layer tuning as a confound when comparing merging methods. We treat layer-range sensitivity as a limitation and leave a dedicated ablation to future work.

For each backbone, we conduct our merging methods to two base methods: MEMIT [10] and AlphaEdit [11]. We choose MEMIT and its successor AlphaEdit because they are the most representative work under the locate-then-edit framework.

5.1.4. Monolingual Editing

We also evaluate monolingual editing performance for each language. This is done by applying editing algorithms with only one language. We compare these results with multilingual counterparts to analyze the interference that occurs in the multilingual setting. They are named as "Mono" in the result tables.

5.1.5. Other Details

We use several NVIDIA A100 PCIe 80GB GPUs for our experiments and use pytorch [28] as a development framework. For calculations of $\mathbf{\Delta}$, merging results are performed with *float32* precision, estimating value matrix $\boldsymbol{V}_{req}$, and evaluation is performed with *bfloat16* precision.

5.2. Experimental Results

5.2.1. Comparison of Averaged Accuracies of Merging Methods on MzsRE (RQ1, RQ2)

We make the following observations from Tables 2 and 3.

- Observation 1: Sum-Cov outperforms the other methods in most cases. Sum-Cov attains the best averaged accuracy in three of the four settings (59.62, 57.96, and 61.00 for Llama + MEMIT, Llama + AlphaEdit, and Qwen + MEMIT), exceeding the next-best merging method by +4.18, +3.80, and +5.48 points, respectively. By contrast, plain summation without shared covariance (Sum) records almost zero accuracy. Mean, a magnitude-rescaled version of Sum, recovers to 46–49% but remains 11–13 points below Sum-Cov, and Mean-Cov is worse still (39–42%). Sharing covariance therefore helps when updates are summed but not when they are averaged.
- Observation 2: TSVM substantially improves overall performance, while TSVM-Cov fails to achieve higher scores than Sum-Cov in most cases. TSVM reaches 54–56% in the first three settings—far above Sum and Mean but still below Sum-Cov. However, as shown in Table 3, both TSVM and TSVM-Cov achieve higher accuracies than Sum-Cov with the Qwen backbone and AlphaEdit algorithm: TSVM attains 58.17 versus 53.65 for Sum-Cov (+4.52), and TSVM-Cov attains 57.42 (+3.77). In this setting, Sum-Cov is also unstable—its Turkish score (39.93) is a clear outlier relative to the 50–60% range of the other languages.
- Observation 3: There is a substantial performance gap between monolingual and multilingual editing. The best multilingual method trails the monolingual upper bound (Mono) by 4.23, 4.24, 6.09, and 10.35 points across the four settings, and no merging method closes this gap. Similarly, previous work [13] reports the performance gap between MKE and monolingual editing with experimental settings that differ from ours. They conduct editing and evaluation with one request at a time while we conduct experiments with all requests at once. We further investigate the possibility of mitigating multilingual interference with merging methods, but unfortunately, none of them effectively achieve the goal.

Table 2. Comparison of merging methods on MzsRE dataset with backbone Llama3.1-8B. Averaged accuracies are reported; unit is percentage. The rank ratio r for TSVM and TSVM-Cov are 0.125, 0.292 for MEMIT, 0.125, 0.063 for AlphaEdit, respectively. Weight scale is set to 1.0.

Methods	Edit & Test Languages												
	en	**zh**	**cz**	**vi**	**tr**	**fr**	**es**	**de**	**ru**	**du**	**pt**	**th**	**avg**
Llama3.1-8B, MEMIT.													
Sum	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean	47.66	46.12	47.44	40.85	43.60	46.63	46.14	50.93	46.01	46.88	48.27	43.53	46.17
TSVM	62.72	51.73	56.72	50.80	52.86	57.12	56.44	61.78	53.40	56.80	57.68	47.24	55.44
Sum-Cov	66.41	55.23	60.01	54.76	56.89	62.03	60.94	65.28	60.37	60.73	60.92	51.83	59.62
Mean-Cov	39.72	43.28	39.20	33.95	36.91	38.16	37.87	40.84	41.75	38.45	38.90	41.56	39.22
TSVM-Cov	54.33	49.38	51.05	44.25	47.81	51.28	50.77	55.67	50.85	50.98	51.76	45.32	50.29
Mono	70.42	61.14	64.63	60.17	61.62	66.17	65.16	69.21	64.45	63.23	65.25	54.75	63.85
Llama3.1-8B, AlphaEdit													
Sum	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean	48.10	46.62	47.40	41.43	44.04	46.74	46.53	50.97	46.21	47.13	48.07	43.82	46.42
TSVM	61.08	50.64	55.60	49.68	51.78	55.97	55.06	60.44	52.12	55.33	55.97	46.28	54.16
Sum-Cov	65.22	54.39	58.44	53.56	55.36	60.26	59.31	64.01	57.55	59.03	59.09	49.27	57.96
Mean-Cov	39.73	43.20	39.16	34.03	36.91	38.26	37.78	40.88	41.68	38.38	39.08	41.64	39.23
TSVM-Cov	52.29	49.27	48.16	43.23	46.06	48.99	48.53	53.13	49.02	48.95	49.25	44.68	48.46
Mono	69.52	59.03	62.74	58.49	59.43	64.50	63.71	67.92	63.21	61.42	63.66	52.76	62.20

Table 3. Comparison of merging methods on MzsRE dataset with backbone Qwen2.5-7B. Averaged accuracies are reported; unit is percentage. The rank ratio r for TSVM and TSVM-Cov are 0.188, 0.208 for MEMIT, 0.188, 0.396 for AlphaEdit, respectively. Weight scale is set to 1.0.

Methods	Edit & Test Languages												
	en	**zh**	**cz**	**vi**	**tr**	**fr**	**es**	**de**	**ru**	**du**	**pt**	**th**	**avg**
Qwen2.5-7B, MEMIT													
Sum	0.01	0.00	0.05	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.02	0.01
Mean	48.94	53.16	47.50	43.06	44.36	47.52	47.01	49.81	48.44	46.38	48.03	49.04	47.77
TSVM	63.25	61.46	53.48	48.20	47.60	56.47	57.74	58.43	55.87	54.99	56.42	52.38	55.52
Sum-Cov	66.90	66.69	59.77	53.60	53.58	60.88	62.94	62.59	63.61	59.84	61.95	59.66	61.00
Mean-Cov	40.38	49.45	40.87	37.19	39.10	40.45	39.31	40.51	45.17	38.49	40.69	47.38	41.58
TSVM-Cov	62.73	61.67	53.08	49.18	47.32	55.91	57.33	57.68	55.89	53.64	56.08	52.81	55.28
Mono	71.92	72.10	65.85	61.95	63.37	68.25	69.57	67.24	67.11	65.85	67.95	63.96	67.09
Qwen2.5-7B, AlphaEdit													
Sum	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.02	0.00	0.00	0.02	0.00
Mean	52.18	53.82	49.45	44.46	45.31	49.50	49.07	52.37	49.33	48.88	49.93	49.17	49.46
TSVM	65.07	64.13	57.10	51.26	51.12	58.16	59.92	60.58	59.05	57.64	59.04	55.01	58.17
Sum-Cov	63.07	59.50	50.53	46.68	39.93	56.00	58.96	55.05	54.92	52.09	55.94	51.13	53.65
Mean-Cov	41.05	50.05	41.36	37.48	39.59	41.09	39.77	40.97	45.43	38.83	41.16	47.54	42.03
TSVM-Cov	65.36	62.69	55.49	50.62	49.73	58.92	59.98	59.85	57.04	56.56	59.41	53.43	57.42
Mono	72.29	66.29	68.37	64.16	65.40	70.46	71.32	68.56	69.75	67.88	69.60	68.16	68.52

5.2.2. Effect of Weight Scale (RQ3)

We also investigate the effect of weight scaling factor α in Equation (9), which is ignored in previously published work in the field of the locate-then-edit framework. This study is the first to investigate the effect of the scaling factor in MKE. The results are shown in Figure 1. Properly scaled weights achieve higher performance: all curves are unimodal

with a single interior maximum, and the optimum lies slightly above the default scale of 1.0 in most settings.

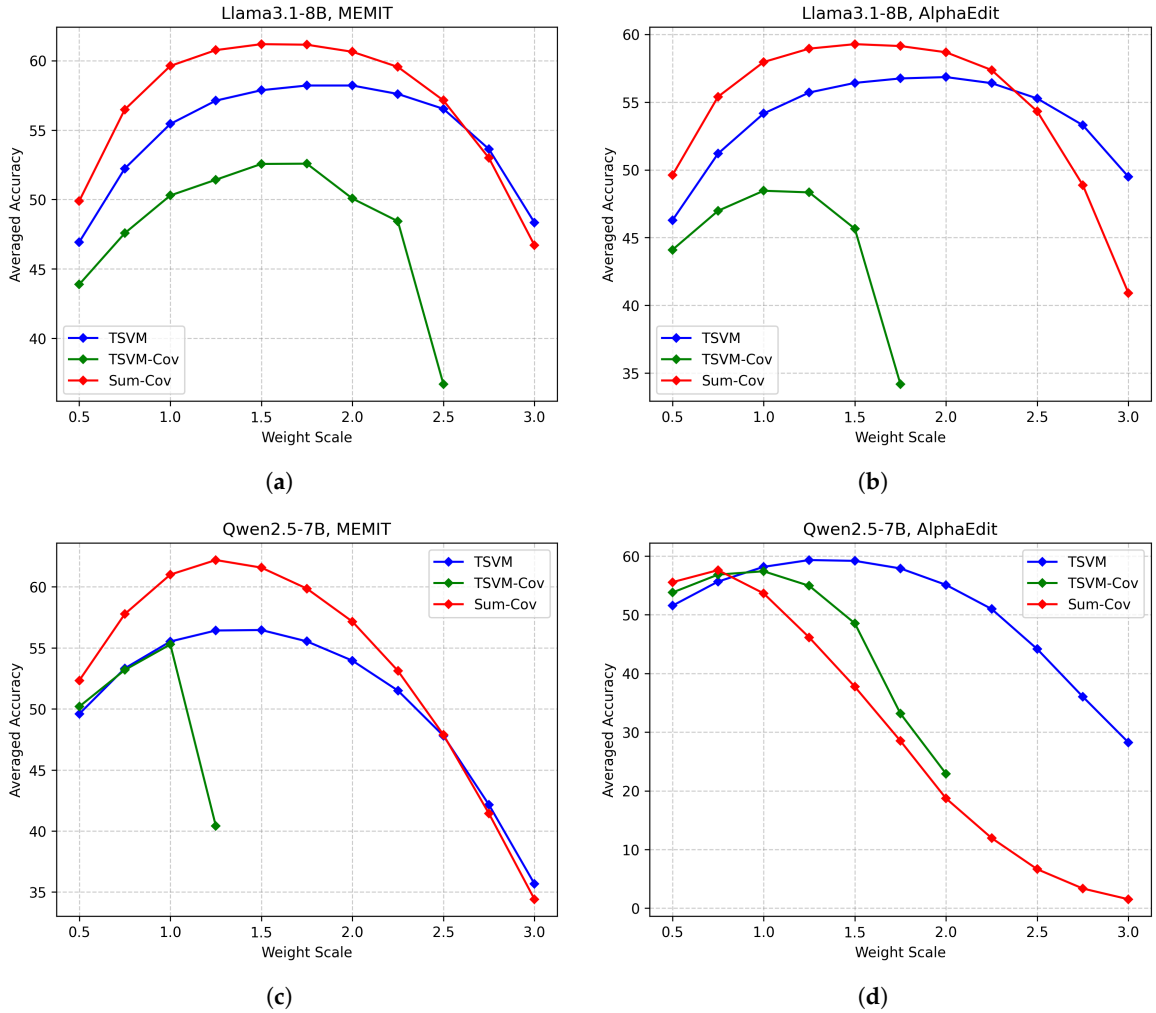


Figure 1. Effect of scaling factor α on TSVM, TSVM-Cov, and Sum-Cov. Accuracies are averaged across all languages. The horizontal axis is the scaling factor α in Equation (9) and the vertical axis is the averaged accuracy; the four panels (**a**–**d**) correspond to the four backbone × base method settings (Llama + MEMIT, Llama + AlphaEdit, Qwen + MEMIT, Qwen + AlphaEdit). Each curve rises then falls, with its maximum slightly above the default $\alpha = 1.0$, indicating that the closed-form update is mildly under-scaled by default.

5.2.3. Effect of Rank Ratio r (RQ3)

The rank ratio r in Equation (14) determines the compression ratio of TSVM-based methods. In the original work of TSVM [14], they use a fixed ratio (0.083 in our case). Here we investigate the effect of rank ratio r and the results are shown in Figure 2. We make the following observations:

- Observation 1: In most cases, the curve of TSVM is unimodal while that of TSVM-Cov is multimodal. Specifically, the accuracy of TSVM-Cov drops sharply for rank ratios in the narrow range $r \in [0.15, 0.20]$ and then recovers, whereas TSVM stays unimodal throughout the evaluated range. We attribute this non-monotonic behavior to an interaction between the shared covariance and the low-rank truncation: the shared covariance solve already couples the per-language updates, and truncating it at a particular rank can discard directions that this coupling depends on, so accuracy degrades in a narrow band of r before higher ranks restore the missing directions. We analyze this shared covariance/low-rank interaction further in Section 6.4.

- Observation 2: Properly low rank ratio tends to achieve optimal performance. It is not surprising because of the low-rank nature of knowledge editing vectors [9]. In much prior work on model merging [14], they report rank reduction can boost merging performance. In the MKE, this result is the first analysis of the relation between the rank of $\mathbf{\Delta}$ and the performance.

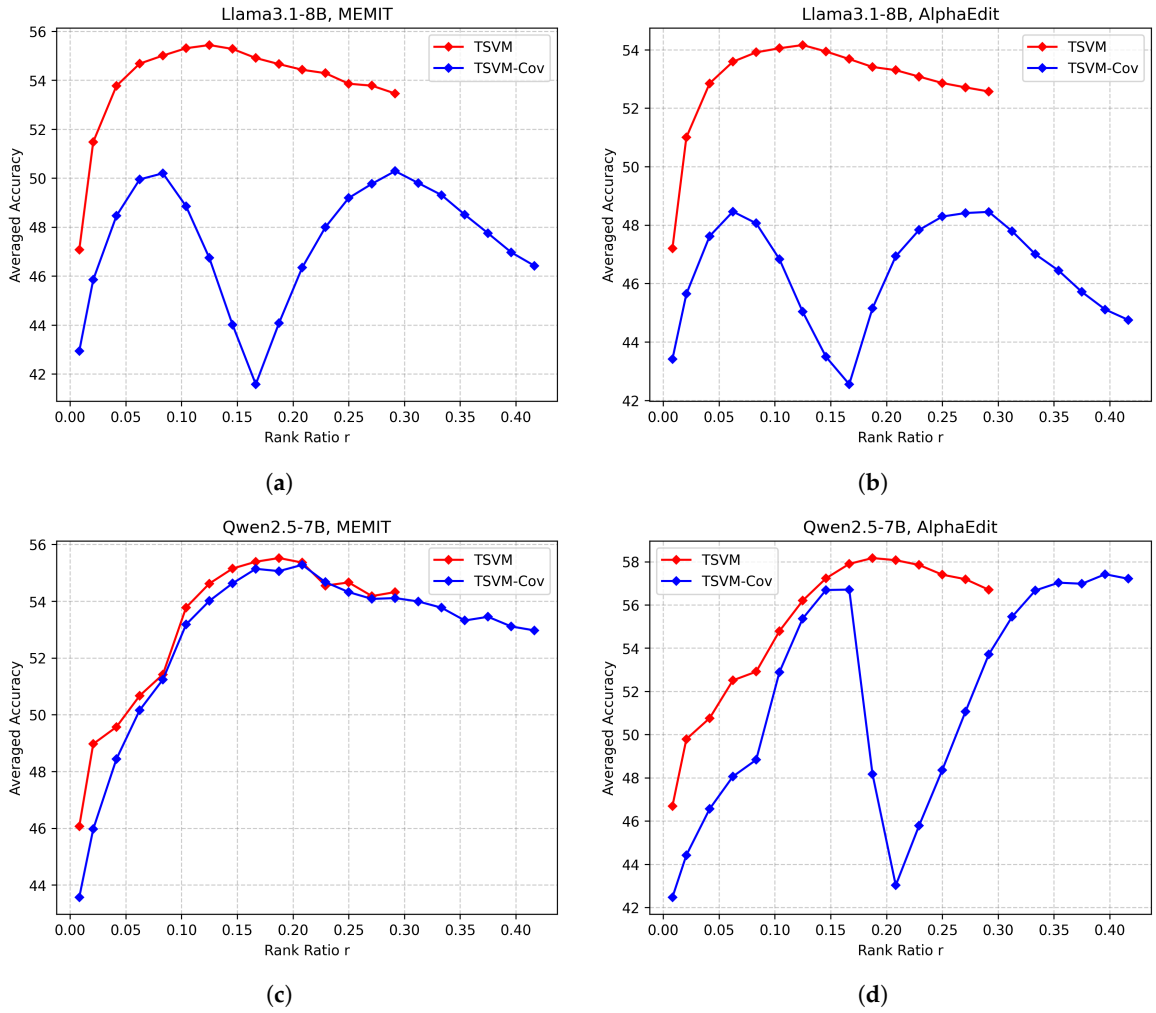


Figure 2. Effect of rank ratio r on TSVM, TSVM-Cov. Accuracies are averaged across all languages. The horizontal axis is the rank ratio r in Equation (14) (the fraction of singular components retained) and the vertical axis is the averaged accuracy; the four panels (**a**–**d**) correspond to the four backbone $\times$ base method settings. TSVM stays unimodal and peaks at a relatively low rank ratio, whereas TSVM-Cov is multimodal, dropping sharply for $r \in [0.15, 0.20]$ before recovering.

5.2.4. Qualitative Error Analysis (RQ2)

We now examine per-sample predictions to see how multilingual interference manifests in practice. Using the Qwen2.5-7B + AlphaEdit setting, we compare the three merged methods that yield Table 3 (Sum-Cov, TSVM, TSVM-Cov) on identical edit requests across the 12 languages. The dominant failure mode is *cross-lingual divergence*: for 87.9% of edits under Sum-Cov, at least one language recalls the new object correctly while at least one other reverts to the old object, even though all 12 share the same fact. Table 4 shows a representative request—for the subject *Blood, Sweat and T-shirts*, the network is edited from "BBC Two" to "CBS", queried as "What network did Blood, Sweat and T-shirts air on?"—where Sum-Cov succeeds in only half of the languages while the low-rank methods are more consistent.

Two secondary patterns recur. First, *target leakage*: the edited object is occasionally emitted in the wrong script for the target language, most often for Chinese (13.9% of Chinese

predictions under Sum-Cov are non-CJK). Second, *low-resource degradation*: per-language exact-match rates fall sharply for the lowest-resource languages (English 68.9% versus Thai 9.0%, Turkish 13.1%, and Vietnamese 24.7%), mirroring their lowest scores in Table 3. The low-rank methods partly offset the worst case: of the edits where Sum-Cov fails in Turkish, TSVM recovers 13.8% and TSVM-Cov 11.8%, consistent with the spectral analysis in Section 6.2. Appendix D provides additional per-language error-case tables that exhibit the same patterns.

Table 4. Qualitative cross-language error analysis (Qwen2.5-7B, AlphaEdit) for a single edit request applied across all 12 languages: for the subject *Blood, Sweat and T-shirts*, the network is edited from "BBC Two" to "CBS", queried in each language as "What network did Blood, Sweat and T-shirts air on?". Each cell marks whether the merged model exactly recalls the new object when queried in that language (✓ correct, × incorrect). Sum-Cov succeeds in only 6/12 languages on this single fact, illustrating cross-lingual interference; the low-rank methods are more consistent (TSVM 8/12, TSVM-Cov 7/12). A dash (—) in the "New object" column marks a gold object written in a non-Latin script (here Thai) that is not typeset in this table; the correctness marks are unaffected.

Language	New Object	Sum-Cov	TSVM	TSVM-Cov
English	CBS	✓	✓	✓
Chinese	CBS	✓	✓	✓
Czech	CBS	×	×	×
Vietnamese	CBS	✓	✓	×
Turkish	CBS	×	×	×
French	CBS	×	✓	✓
Spanish	CBS	✓	✓	✓
German	CBS	×	×	✓
Russian	CBS	✓	✓	✓
Dutch	CBS	×	✓	×
Portuguese	CBS	✓	✓	✓
Thai	—	×	×	×

Table 5 makes the low-resource effect of Section 6.5 concrete on a single fact. For the edit *Sunnyside Hospital* (→ "1956"), the shared covariance merge recalls the new object in only one of the five relatively low-resource languages (Thai, Vietnamese, Turkish, Czech, Dutch), whereas the orthogonalizing TSVM merge recovers four of five; the same pattern recurs across the additional cases in Appendix D. This is the per-fact signature of the aggregate gap in Table 6: the joint shared covariance solve sacrifices precisely the lowest-resource languages, and the per-language orthogonal merge restores them. Languages are listed low-resource first.

Table 5. Cross-language error analysis (Qwen2.5-7B, AlphaEdit) for the edit request *Sunnyside Hospital*: the object is edited from "1955" to "1956", queried as "In which year did Sunnyside Hospital cease to exist?". Languages are grouped into the relatively low-resource block (Thai, Vietnamese, Turkish, Czech, Dutch) and the high-resource block; each cell marks the exact recall of the new object (✓ correct, × incorrect). On the low-resource block, Sum-Cov recovers only 1/5, versus TSVM 4/5 and TSVM-Cov 4/5 (all languages: 6/12, 8/12, 8/12). A dash (—) marks a gold object in a non-Latin script, not typeset here; the marks are unaffected.

Language	New Object	Sum-Cov	TSVM	TSVM-Cov
Relatively low-resource				
Thai	1956	×	✓	✓
Vietnamese	1956	×	✓	✓
Turkish	1956	×	✓	✓
Czech	1956	✓	✓	✓
Dutch	1956	×	×	×

Table 5. *Cont.*

Language	New Object	Sum-Cov	TSVM	TSVM-Cov
High-resource				
English	1956	✓	×	×
Chinese	—	✓	✓	✓
French	1956	×	✓	✓
Spanish	1956	✓	✓	✓
German	1956	✓	✓	✓
Russian	—	×	×	×
Portuguese	1956	✓	×	×

Table 6. Relatively low-resource view of the editing results: averaged accuracy (%, arithmetic mean over the four aspects and 700 edits) for the five relatively low-resource languages (Thai, Vietnamese, Turkish, Czech, Dutch), their mean (LRL), and the mean of the seven high-resource languages (HRL: en, zh, fr, es, de, ru, pt). Same runs as Tables 2 and 3; Mono is the monolingual upper bound (edits one language at a time). The best merging method per block on the LRL average is in bold. Sum and Mean are omitted here (Sum collapses to 0 in every language); both appear in Tables 2 and 3.

Backbone + Method	Merge	th	vi	tr	cz	du	LRL	HRL
Llama3.1-8B + MEMIT	TSVM	47.2	50.8	52.9	56.7	56.8	52.9	57.3
	Sum-Cov	51.8	54.8	56.9	60.0	60.7	56.8	61.6
	TSVM-Cov	45.3	44.3	47.8	51.0	51.0	47.9	52.0
	Mono	54.8	60.2	61.6	64.6	63.2	60.9	66.0
Llama3.1-8B + AlphaEdit	TSVM	46.3	49.7	51.8	55.6	55.3	51.7	55.9
	Sum-Cov	49.3	53.3	55.5	58.5	59.0	55.1	60.0
	TSVM-Cov	44.7	43.2	46.1	48.2	49.0	46.2	50.1
	Mono	52.8	58.5	59.4	62.7	61.4	59.0	64.5
Qwen2.5-7B + MEMIT	TSVM	52.4	48.2	47.6	53.5	55.0	51.3	58.5
	Sum-Cov	59.7	53.6	53.6	59.8	59.8	57.3	63.7
	TSVM-Cov	52.8	49.2	47.3	53.1	53.6	51.2	58.2
	Mono	64.0	61.9	63.4	65.9	65.8	64.2	69.2
Qwen2.5-7B + AlphaEdit	TSVM	55.0	51.3	51.1	57.1	57.6	54.4	60.8
	Sum-Cov	51.1	46.7	39.9	50.5	52.1	48.1	57.6
	TSVM-Cov	53.4	50.6	49.7	55.5	56.6	53.2	60.5
	Mono	68.2	64.2	65.4	68.4	67.9	66.8	69.8

6. Discussion

6.1. Effect of Vector Merging Framework (RQ1)

Our results show that the choice of merging framework has a substantial effect on MKE performance. Among the evaluated methods, Sum-Cov consistently provides the strongest overall results across most backbone and base method combinations. This finding suggests that sharing covariance across languages is more important than applying a sophisticated merging rule after language-specific updates have already diverged. In contrast, naive summation without shared covariance almost completely fails, indicating that multilingual editing vectors are not directly compatible when computed independently.

Another important observation is that averaging-based methods do not offer a reliable compromise. Although Mean improves dramatically over Sum, it still remains clearly below Sum-Cov, and Mean-Cov performs even worse in most settings. This pattern indicates that simply reducing the magnitude of updates cannot resolve multilingual interference by itself. Instead, the interaction among language-specific key distributions appears to be a more fundamental source of the problem.

More broadly, the results show that current vector merging methods do not eliminate the gap between multilingual and monolingual editing. Even the best multilingual setting remains noticeably below the monolingual upper bound in all four experimental configurations. Therefore, the main challenge in MKE is not merely how to merge language-specific updates, but how to construct updates that are mutually compatible across languages before or during merging.

6.2. Effect of TSVM (RQ2)

TSVM was motivated by prior evidence that low-rank task representations can reduce interference in model merging. In our setting, TSVM is clearly more effective than simple Sum and often competitive with strong baselines, which suggests that low-rank structure is indeed relevant for multilingual editing vectors. However, TSVM does not consistently outperform Sum-Cov, and TSVM-Cov improves over TSVM only in limited cases. In other words, rank-aware merging alone is insufficient to solve multilingual interference.

One notable exception appears in the Qwen + AlphaEdit setting, where TSVM and TSVM-Cov both outperform Sum-Cov. This result suggests that the usefulness of TSVM depends on the geometry of the underlying editing vectors, which in turn may depend on both the backbone model and the base KE algorithm. However, because this behavior is not reproduced in the other settings, we interpret it as evidence of conditional effectiveness rather than a generally superior strategy.

To probe this exception, we measure how well the per-language editing vectors $\mathbf{\Delta}^i$ align across languages (Figure 3; computation in Appendix B). Qwen + AlphaEdit shows the lowest mean cross-language subspace overlap (0.10, versus 0.18–0.19 for the two Llama settings). Summing such poorly aligned updates under a shared covariance accumulates their conflicting directions, seen as the outlier Turkish score in Table 3 (39.93). TSVM orthogonalizes the per-language components before recombining them, which is why it helps most in Qwen + AlphaEdit, while the better-aligned Llama updates are already well served by Sum-Cov. The advantage is conditional, not universal: the same low alignment does not occur in the other three settings.

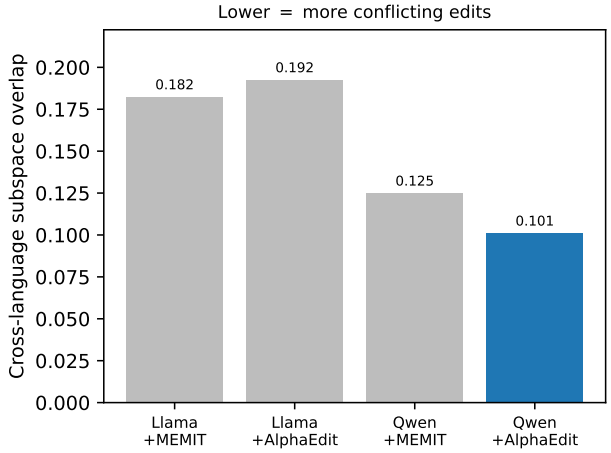


Figure 3. Mean cross-language subspace overlap of the per-language editing vectors $\mathbf{\Delta}^i$ (top-64 left singular subspaces), averaged over all language pairs at a representative edited layer. Lower values indicate more conflicting per-language directions; Qwen + AlphaEdit is the lowest, which explains why shared covariance summation suffers most there while TSVM's orthogonalized merge avoids the accumulated interference.

The subspace overlap analysis above is correlational; we test the mechanism directly with a conditioning analysis of the shared covariance solve (Table 7; quantities defined in Appendix C). The shared covariance merge solves all twelve languages jointly, whereas

TSVM solves each language independently. For Qwen2.5-7B + AlphaEdit, the joint solve reproduces each language's edit 4.3× less faithfully than the per-language solve, against only 1.4–1.9× in the other three settings. Two structural factors explain this: AlphaEdit's null-space projection retains only 60% of the edit-key energy for Qwen2.5-7B (versus 91% for Llama3.1-8B), and the inverted matrix is correspondingly more ill-conditioned. Routing all twelve languages jointly through this attenuated, ill-conditioned subspace forces them to compete for a small shared capacity, so the shared covariance merge degrades—the regime in which the orthogonalizing TSVM merges, which never forms the joint inverse, wins.

Taken together, these findings answer RQ2 conservatively: TSVM can mitigate multilingual interference to some extent, but only under limited conditions.

Table 7. Why the shared covariance (Sum-Cov) merge collapses only for Qwen2.5-7B + AlphaEdit. Per (backbone, base method), all quantities are averaged over the five edited layers. cond($\boldsymbol{M}$) is the condition number of the matrix the merge inverts ($\boldsymbol{P}\boldsymbol{K}_{req}\boldsymbol{K}_{req}^{\top} + \lambda\boldsymbol{I}$ for AlphaEdit, $\lambda\boldsymbol{C}_{const} + \boldsymbol{K}_{req}\boldsymbol{K}_{req}^{\top}$ for MEMIT). "$\boldsymbol{PK}$ kept" is the fraction of edit-key energy surviving AlphaEdit's null-space projection, $\|\boldsymbol{P}\boldsymbol{K}_{req}\|_F^2 / \|\boldsymbol{K}_{req}\|_F^2$ (AlphaEdit only). $\text{fid}_{\text{joint}}$ and fid_{solo} are the edit-reproduction errors of the shared 12-language solve (=Sum-Cov) and the per-language solve (=the TSVM building block); lower is more faithful. The last column is their ratio—the penalty for sharing the covariance across languages.

Backbone + Method	cond(M)	PK Kept	$\text{fid}_{\text{joint}}$	fid_{solo}	Joint/Solo
Llama3.1-8B + MEMIT	1.1×10^4	–	0.72	0.49	1.5
Llama3.1-8B + AlphaEdit	1.7×10^3	0.91	0.77	0.56	1.4
Qwen2.5-7B + MEMIT	3.0×10^9	–	0.63	0.34	1.9
Qwen2.5-7B + AlphaEdit	6.8×10^4	0.60	0.39	0.09	4.3

6.3. Effect of Weight Scaling Factor (RQ3)

Weight scaling has a measurable effect on performance. In most settings, the best performance is obtained not at the default scale of 1.0, but at a slightly larger value. This result shows that the magnitude of the closed-form update is not automatically well calibrated, even when the editing direction itself is useful. From an optimization perspective, the scaling factor plays a role analogous to a learning rate, although locate-then-edit methods are derived analytically rather than through iterative training.

One possible explanation is that the scale of the computed update is sensitive to several handcrafted components in the closed-form solutions of Equations (7) and (8). In particular, both the covariance-related term and the estimated target-value term $\boldsymbol{V}_{req}$ can vary substantially in magnitude. As a result, two editing methods with similar directional quality may still differ in performance because one produces an under-scaled or over-scaled update.

This finding also has practical implications. If scaling is tuned appropriately, practitioners can improve performance without changing the editing algorithm itself. At the same time, exhaustive searching over the scaling factor is expensive. A useful direction for future work is therefore to develop principled or data-driven rules for choosing the scaling factor automatically.

6.4. Effect of Rank Compression Ratio (RQ3)

The rank compression ratio controls how much information from each language-specific editing vector is retained during TSVM. Our results show that relatively low rank often leads to the best performance, which is consistent with the view that the most important multilingual editing signals are concentrated in a compact subspace.

At the same time, the behavior of TSVM-Cov is more complex than that of TSVM. In several settings, TSVM-Cov exhibits a non-smooth performance curve with abrupt drops in a narrow range of rank ratios. This instability suggests that shared covariance and low-rank reconstruction interact in a nontrivial way.

6.5. Do Merging Methods Help Relatively Low-Resource Languages? (RQ4)

MzsRE spans twelve languages of uneven representation in LLM pre-training corpora. Following the resource-level taxonomy of Joshi et al. [29], we group the five least represented—Thai (th), Vietnamese (vi), Turkish (tr), Czech (cz), and Dutch (du)—as the relatively low-resource group (LRL), and the seven best represented—English, Chinese, French, Spanish, German, Russian, and Portuguese—as the high-resource group (HRL). The split is internal to this benchmark: even the LRL languages are better resourced than the world's genuinely under-served languages [29], but they are the least represented here. Table 6 reports per-tier averaged accuracy; per-language values are the columns of Tables 2 and 3.

Two findings are specific to the low-resource tier. First, the best merge is nearly as effective for LRL as for HRL: in the three settings where Sum-Cov wins, it retains 93–95% (Thai) and 87–91% (Vietnamese) of monolingual accuracy, comparable to the 92–93% it retains for HRL, so editing eleven additional languages jointly costs the low-resource languages little beyond their already-lower monolingual ceiling. Second, the merge operator matters most for LRL: in Qwen2.5-7B + AlphaEdit—the one setting where TSVM beats Sum-Cov—Sum-Cov underperforms on LRL disproportionately (LRL 48.1% vs. HRL 57.6%, a 9.6-point tier gap, against 3.0 for Mono; Turkish falls to 39.9%, 61% of its monolingual accuracy), and TSVM restores the tier (+6.4 points on LRL vs. +3.2 on HRL, narrowing the gap to 6.4). The shared covariance degradation analyzed in Section 6.2 thus lands hardest on the relatively low-resource languages, where an orthogonal per-language merge is the practical remedy.

6.6. Limitations of This Work

This study has several limitations. First, we evaluate only two backbone models and two locate-then-edit base methods, so our conclusions may not transfer uniformly to other architectures or editing paradigms. Second, our experiments focus on one-step batch editing on MzsRE, which means the results do not directly address sequential editing, continual editing, or other multilingual editing benchmarks. In addition, our search over weight scales and rank ratios is empirical rather than theoretically grounded. Although this is sufficient to reveal important trends, it does not yet provide a predictive framework for selecting hyperparameters.

7. Conclusions

In this paper, we investigated whether vector merging methods can improve multilingual knowledge editing for large language models. We evaluated six merging variants across two backbone models and two locate-then-edit base methods under a large-scale multilingual batch-editing setting covering 12 languages. Our results show that shared covariance summation is the most reliable overall strategy, while TSVM can be beneficial only in limited settings. A per-resource-level analysis shows that the choice of merging method has the largest effect on the relatively low-resource languages (RQ4).

The study also yields two broader findings. First, multilingual interference remains unsolved: none of the evaluated merging methods closes the gap between multilingual and monolingual editing. Second, the scaling factor and rank compression ratio affect performance, showing that the quality of a closed-form edit depends not only on direction but also on magnitude and spectral structure.

These findings suggest that future progress in MKE will likely require methods that model cross-lingual compatibility more explicitly, rather than relying only on post hoc merging of independently computed updates. We hope this work provides an empirical basis for that direction and serves as a reference for future research on multilingual locate-then-edit methods.

Author Contributions: Conceptualization, K.L. and K.-Y.S.; methodology, K.L.; software, K.L.; validation, K.L., J.-H.L., K.-Y.S. and Y.-J.S.; formal analysis, K.L.; investigation, K.L.; resources, K.L.; data curation, K.L.; writing—original draft preparation, K.L.; writing—review and editing, K.-Y.S.; visualization, K.-Y.S.; supervision, J.-H.L. and Y.-J.S.; project administration, Y.-J.S.; funding acquisition, Y.-J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partly supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (RS-2022-NR070870), by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (MSIT) (No.RS-2019-II191906, Artificial Intelligence Graduate School Program (POSTECH)), and by the Gyeongsangbuk-do RISE (Regional Innovation System and Education) project (B0080527002599).

Data Availability Statement: The data supporting the results presented in this paper are publicly available online.

Acknowledgments: We are grateful for the technical and financial support that was provided by Designovel Co., Ltd.

Conflicts of Interest: Author Kunil Lee was employed by the company AI Lab, Designovel Co., Ltd., CHANGeUP GROUND Pohang, 87 Cheongam-ro, Nam-gu, Pohang 37673, Republic of Korea. Additionally author Jong-Hyeok Lee was employed by the company LLSOLLU, 5 Mabang-ro 10-gil, Seocho-gu, Seoul 06771, Republic of Korea. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LLM	Large Language Model
KE	Knowledge Editing
MKE	Multilingual Knowledge Editing
TSVM	Task Singular Vectors for Merging
RQ	Research Question
FFN	Feed-Forward Network
LAM	Linear Associative Memory
SVD	Singular Value Decomposition
MEMIT	Mass Editing Memory In Transformer
LRL	Low-Resource Language
HRL	High-Resource Language

Appendix A. Evaluation Metric Definitions

This appendix gives the formal definitions of the metrics summarized in Section 5.1.1. For an autoregressive model P_θ, an input prompt $\boldsymbol{p}$, and an anticipated answer $\boldsymbol{a}$, the token-level, teacher-forced accuracy is

$$Acc(\boldsymbol{p}, \boldsymbol{a}, P_\theta(t \mid \cdot)) = \mathbb{E}_{o_i \in tokenize(\boldsymbol{a})}\{o_i = \arg\max_{t_i} P_\theta(t_i \mid \boldsymbol{p} + t_{\leq i})\}, \tag{A1}$$

where the expectation is taken over the gold answer tokens o_i and each token is scored conditioned on the prompt and the preceding gold tokens. The four aspects instantiate Equation (A1) on different evaluation sets drawn from $\boldsymbol{D}_{req}$:

- **Efficacy** is the accuracy on the editing requests themselves:

$$Acc_{eff} = \mathbb{E}_{\boldsymbol{p},\boldsymbol{a} \sim \boldsymbol{D}_{req}}\{Acc(\boldsymbol{p}, \boldsymbol{a}, P_\theta(t \mid \cdot))\}. \tag{A2}$$

- **Generalization** is the accuracy on paraphrases of the edit requests:

$$Acc_{gen} = \mathbb{E}_{\boldsymbol{p},\boldsymbol{a}\sim \boldsymbol{D}_{req}}\{\mathbb{E}_{\boldsymbol{p}'\sim rephrase(\boldsymbol{p})}\{Acc(\boldsymbol{p}', \boldsymbol{a}, P_\theta(t \mid \cdot))\}\}. \tag{A3}$$

- **Specificity** is the accuracy on questions that are unrelated to the editing requests:

$$Acc_{spe} = \mathbb{E}_{\boldsymbol{p},\boldsymbol{a}\sim \boldsymbol{D}_{req}}\{\mathbb{E}_{\boldsymbol{p}'\sim unrelated(\boldsymbol{p})}\{Acc(\boldsymbol{p}', \boldsymbol{a}, P_\theta(t \mid \cdot))\}\}. \tag{A4}$$

- **Portability** is the accuracy on one-hop reasoning questions derived from the editing requests:

$$Acc_{por} = \mathbb{E}_{\boldsymbol{p},\boldsymbol{a}\sim \boldsymbol{D}_{req}}\{\mathbb{E}_{\boldsymbol{p}'\boldsymbol{a}'\sim onehop(\boldsymbol{p},\boldsymbol{a})}\{Acc(\boldsymbol{p}', \boldsymbol{a}', P_\theta(t \mid \cdot))\}\}. \tag{A5}$$

The averaged accuracy reported throughout the paper is the arithmetic mean of these four quantities.

Appendix B. Computation of the Cross-Language Subspace Overlap in Figure 3

This appendix details how the cross-language subspace overlap in Figure 3 is computed. For each of the four backbone×base method settings, we load the per-language editing vectors $\boldsymbol{\Delta}^i \in \mathbb{R}^{d\times h}$ produced by the closed-form solution (Equation (7) or Equation (8)) for each of the 12 languages and each of the five edited layers, and take the singular value decomposition $\boldsymbol{\Delta}^i = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_R > 0$.

To measure how much the per-language updates share a common low-rank subspace, we take the top-k left singular subspace $\tilde{\boldsymbol{U}}^i \in \mathbb{R}^{d\times k}$ of each language ($k = 64$) and, for every language pair (i, j), compute the mean squared cosine of the principal angles between their subspaces,

$$\text{overlap}(i, j) = \frac{1}{k}\left\|(\tilde{\boldsymbol{U}}^i)^\top \tilde{\boldsymbol{U}}^j\right\|_F^2, \tag{A6}$$

which equals 1 for identical subspaces and 0 for orthogonal ones. For each setting, overlap is averaged over all $\binom{12}{2}$ language pairs at a representative edited layer; these averages are the values plotted in Figure 3. All decompositions are computed in *float32* on the saved editing vectors, so the analysis requires no re-editing of the models.

Appendix C. Conditioning of the Shared-Covariance Solve

This appendix defines the quantities reported in Table 7. Let $\boldsymbol{K}_{req}$ be the matrix of edit keys (one column per request) and, for the shared covariance ("-Cov") merge, let $\boldsymbol{M}$ be the matrix the closed-form solve inverts:

$$\boldsymbol{M}_{\text{AlphaEdit}} = \boldsymbol{P}\boldsymbol{K}_{req}\boldsymbol{K}_{req}^\top + \lambda \boldsymbol{I}, \qquad \boldsymbol{M}_{\text{MEMIT}} = \lambda \boldsymbol{C}_{const} + \boldsymbol{K}_{req}\boldsymbol{K}_{req}^\top, \tag{A7}$$

where $\boldsymbol{P}$ is the null-space projection of AlphaEdit and $\boldsymbol{C}_{const}$ the covariance term of MEMIT (Equations (7) and (8)). The shared covariance merge forms $\boldsymbol{M}$ once over all twelve languages jointly, whereas TSVM solves each language ℓ independently with its own $\boldsymbol{M}_\ell$ built from that language's keys $\boldsymbol{K}_{req}^{(\ell)}$ only.

We report three quantities, each averaged over the five edited layers:

- **Condition number** $\text{cond}(\boldsymbol{M}) = \sigma_{\max}(\boldsymbol{M})/\sigma_{\min}(\boldsymbol{M})$, the ratio of the largest to smallest singular value of $\boldsymbol{M}$.
- **Null-space retention** (AlphaEdit only), the fraction of edit-key energy that survives the null-space projection, $\|\boldsymbol{P}\boldsymbol{K}_{req}\|_F^2 / \|\boldsymbol{K}_{req}\|_F^2$.

https://doi.org/10.3390/electronics15122747

- **Edit-reproduction error**. For each language ℓ, we form the matrix $\boldsymbol{T}_\ell = \boldsymbol{K}_{req}^{(\ell)\top} \boldsymbol{M}^{-1}(\cdot) \boldsymbol{K}_{req}^{(\ell)}$, i.e., the keys of language ℓ passed through the solve evaluated at their own keys; a faithful edit gives $\boldsymbol{T}_\ell \approx \boldsymbol{I}$. We measure $\|\boldsymbol{T}_\ell - \boldsymbol{I}\|_F / \sqrt{n}$ and average over languages. With the shared $\boldsymbol{M}$, this gives $\text{fid}_{\text{joint}}$ (the Sum-Cov solve); with the per-language $\boldsymbol{M}_\ell$, it gives fid_{solo} (the per-language solve that TSVM merges). The last column of Table 7 is their ratio $\text{fid}_{\text{joint}}/\text{fid}_{\text{solo}}$, the fidelity penalty incurred by sharing the covariance across languages.

Appendix D. Additional Qualitative Error Cases

This appendix supplements the error analysis of Section 5.2.4 with additional edit requests from the Qwen2.5-7B + AlphaEdit setting, comparing the three merged methods (Sum-Cov, TSVM, TSVM-Cov) on identical requests across the 12 languages. As in the main text, each request is correctly recalled in some languages but reverts to the old object in others under Sum-Cov, illustrating cross-lingual divergence, and the low-rank methods are somewhat more consistent. A dash (—) in the "New object" column marks a gold object written in a non-Latin script (Chinese, Russian, or Thai) that is not typeset here; the correctness marks are unaffected.

Table A1. Qualitative cross-language error analysis (Qwen2.5-7B, AlphaEdit) for the edit request *Martha Neumark*: the object is edited from "1947" to "1952", queried as "The date of birth of Martha Neumark is?". Cells mark the exact recall of the new object per language (✓ correct, × incorrect). Sum-Cov 6/12, TSVM 7/12, TSVM-Cov 9/12.

Language	New Object	Sum-Cov	TSVM	TSVM-Cov
English	1952	✓	✓	✓
Chinese	—	✓	✓	✓
Czech	1952	✓	✓	✓
Vietnamese	1952	×	✓	✓
Turkish	1952	✓	✓	×
French	1952	✓	✓	✓
Spanish	1952	×	×	✓
German	1952	×	×	✓
Russian	—	×	×	×
Dutch	1952	×	×	✓
Portuguese	1952	✓	✓	×
Thai	1952	×	×	✓

Table A2. Qualitative cross-language error analysis (Qwen2.5-7B, AlphaEdit) for the edit request *Runaway Sunday*: the object is edited from "A&M Records" to "Motown", queried as "What was the record label of Runaway Sunday?". Cells mark the exact recall of the new object per language (✓ correct, × incorrect). Sum-Cov 5/12, TSVM 7/12, TSVM-Cov 8/12.

Language	New Object	Sum-Cov	TSVM	TSVM-Cov
English	Motown	✓	✓	✓
Chinese	Motown	✓	✓	✓
Czech	Motown	✓	✓	✓
Vietnamese	Motown	×	×	✓
Turkish	Motown	×	✓	✓
French	Mototown	×	×	×
Spanish	motown	✓	×	×
German	Motown	×	✓	✓
Russian	—	×	×	×
Dutch	Motown	×	✓	✓
Portuguese	Motown	✓	✓	✓
Thai	—	×	×	×

Table A3. Qualitative cross-language error analysis (Qwen2.5-7B, AlphaEdit) for the edit request *Underland Press*: the object is edited from "1995" to "2005", queried as "What year was Underland Press formed in?". Cells mark the exact recall of the new object per language (✓ correct, × incorrect). Sum-Cov 7/12, TSVM 8/12, TSVM-Cov 9/12.

Language	New Object	Sum-Cov	TSVM	TSVM-Cov
English	2005	✓	✓	✓
Chinese	—	✓	×	×
Czech	2005	✓	✓	✓
Vietnamese	2005	×	✓	✓
Turkish	2005	×	✓	✓
French	2005	✓	×	✓
Spanish	2005	✓	✓	✓
German	2005	✓	✓	✓
Russian	—	×	×	×
Dutch	2005	✓	✓	✓
Portuguese	2005	×	✓	✓
Thai	2548	×	×	×

Tables A4–A6 give three further low-resource-focused cases (rows ordered low-resource first), each showing the shared covariance merge failing on the relatively low-resource block while the TSVM merges recover it, complementing the main-text Table 5.

Table A4. Cross-language error analysis (Qwen2.5-7B, AlphaEdit) for the edit request *The Testament of Sherlock Holmes*: the object is edited from "Babylon 5" to "Sherlock Holmes", queried as "The Testament of Sherlock Holmes was in what series?". Languages are grouped into the relatively low-resource block (Thai, Vietnamese, Turkish, Czech, Dutch) and the high-resource block; each cell marks the exact recall of the new object (✓ correct, × incorrect). On the low-resource block, Sum-Cov recovers only 2/5, versus TSVM 5/5 and TSVM-Cov 4/5 (all languages: 6/12, 10/12, 9/12). A dash (—) marks a gold object in a non-Latin script, not typeset here; the marks are unaffected.

Language	New Object	Sum-Cov	TSVM	TSVM-Cov
Relatively low-resource				
Thai	Sherlock Holmes	✓	✓	✓
Vietnamese	Sherlock Holmes	×	✓	✓
Turkish	Sherlock Holmes	×	✓	×
Czech	Sherlock Holmes	✓	✓	✓
Dutch	Sherlock Holmes	×	✓	✓
High-resource				
English	Sherlock Holmes	✓	×	✓
Chinese	—	×	✓	×
French	Sherlock Holmes	✓	✓	✓
Spanish	Sherlock Holmes	✓	✓	✓
German	Sherlock Holmes	×	✓	✓
Russian	—	×	×	×
Portuguese	Sherlock Holmes	✓	✓	✓

Table A5. Cross-language error analysis (Qwen2.5-7B, AlphaEdit) for the edit request *Char 2C*: the object is edited to "1990", queried as "Which was the official year for the approval of Char 2C?". Languages are grouped into the relatively low-resource block (Thai, Vietnamese, Turkish, Czech, Dutch) and the high-resource block; each cell marks the exact recall of the new object (✓ correct, × incorrect). On the low-resource block, Sum-Cov recovers only 2/5, versus TSVM 4/5 and TSVM-Cov 4/5 (all languages: 5/12, 7/12, 8/12). A dash (—) marks a gold object in a non-Latin script, not typeset here; the marks are unaffected.

Language	New Object	Sum-Cov	TSVM	TSVM-Cov
Relatively low-resource				
Thai	1990	✓	✓	✓
Vietnamese	1990	×	✓	✓
Turkish	1990	×	✓	✓
Czech	1990	×	✓	✓
Dutch	1990	✓	×	×
High-resource				
English	1990	✓	×	×
Chinese	—	×	×	×
French	1990	×	✓	✓
Spanish	1990	✓	✓	✓
German	1990	×	×	✓
Russian	—	×	×	×
Portuguese	1990	✓	✓	✓

Table A6. Cross-language error analysis (Qwen2.5-7B, AlphaEdit) for the edit request *Portuguese Malacca*: the object is edited from "1825" to "1818", queried as "What year was it when Portuguese Malacca was dissolved?". Languages are grouped into the relatively low-resource block (Thai, Vietnamese, Turkish, Czech, Dutch) and the high-resource block; each cell marks the exact recall of the new object (✓ correct, × incorrect). On the low-resource block, Sum-Cov recovers only 1/5, versus TSVM 3/5 and TSVM-Cov 3/5 (all languages: 5/12, 8/12, 7/12). A dash (—) marks a gold object in a non-Latin script, not typeset here; the marks are unaffected.

Language	New Object	Sum-Cov	TSVM	TSVM-Cov
Relatively low-resource				
Thai	1818	×	×	×
Vietnamese	1818	×	✓	✓
Turkish	1818	×	✓	✓
Czech	1818	✓	✓	✓
Dutch	1818	×	×	×
High-resource				
English	1818	✓	✓	✓
Chinese	—	×	✓	×
French	1818	✓	✓	✓
Spanish	1818	×	×	×
German	1818	✓	✓	✓
Russian	—	×	×	×
Portuguese	1818	✓	✓	✓

Appendix E. Exact-Match Results

For completeness, we report the exact-match (EM) counterparts of the main results (Tables 2 and 3). EM is stricter than the token-level accuracy used in the main text: a request counts as correct only if every gold answer token is predicted exactly. For each edit request, we compute its EM and then average over the whole evaluation set, mirroring the averaged accuracy protocol of Section 5.1.1. Tables A7 and A8 use the same runs as Tables 2 and 3.

The relative ordering of the merging methods is preserved under EM: shared covariance summation remains the strongest merging method in most settings, and the gap to the monolingual upper bound persists.

Table A7. Exact-match(EM) counterpart of Table 2 (Llama3.1-8B). Each cell is the averaged EM over the four aspects and 700 edits (percent); EM counts a request correct only if every gold answer token is predicted exactly. Best merging method per column in bold; Mono is the monolingual upper bound. Same runs as Table 2.

Methods	Edit & Test Languages												
	en	**zh**	**cz**	**vi**	**tr**	**fr**	**es**	**de**	**ru**	**du**	**pt**	**th**	**avg**
Llama3.1-8B, MEMIT													
Sum	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean	5.89	0.21	4.82	2.25	4.25	3.57	3.46	7.04	0.79	4.43	4.64	0.04	3.45
TSVM	25.93	1.93	16.21	11.57	13.71	13.89	14.21	20.18	7.18	15.07	16.00	1.21	13.09
Sum-Cov	33.21	5.82	21.46	16.89	19.04	20.96	21.57	26.75	15.82	20.46	21.00	4.68	18.97
Mean-Cov	0.64	0.04	0.29	0.14	0.54	0.04	0.11	0.32	0.00	0.29	0.11	0.00	0.21
TSVM-Cov	14.29	1.07	9.04	4.93	8.00	6.96	7.04	12.04	4.25	8.14	8.25	0.50	7.04
Mono	40.61	15.54	29.71	26.25	27.29	30.93	31.50	35.54	25.46	25.25	29.68	8.00	27.15
Llama3.1-8B, AlphaEdit													
Sum	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean	6.57	0.32	4.54	2.64	4.39	3.25	3.46	6.50	0.82	4.36	4.43	0.07	3.45
TSVM	23.14	1.39	14.64	10.50	12.54	12.79	12.29	18.21	5.46	13.50	13.54	0.89	11.57
Sum-Cov	31.21	4.82	19.86	15.00	17.04	18.07	18.96	25.21	12.54	18.54	18.64	2.68	16.88
Mean-Cov	0.68	0.04	0.29	0.14	0.54	0.04	0.11	0.39	0.00	0.29	0.11	0.00	0.22
TSVM-Cov	12.39	1.36	6.11	4.07	6.36	5.04	5.32	8.75	3.11	6.39	5.71	0.57	5.43
Mono	38.79	12.43	26.46	23.11	22.93	26.61	27.14	32.14	23.07	22.39	26.46	5.50	23.92

Table A8. Exact-match (EM) counterpart of Table 3 (Qwen2.5-7B). Each cell is the averaged EM over the four aspects and 700 edits (percent). Best merging method per column in bold; Mono is the monolingual upper bound. Same runs as Table 3.

Methods	Edit & Test Languages												
	en	**zh**	**cz**	**vi**	**tr**	**fr**	**es**	**de**	**ru**	**du**	**pt**	**th**	**avg**
Qwen2.5-7B, MEMIT													
Sum	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean	7.29	1.39	3.14	1.64	2.61	2.50	3.43	4.71	1.29	4.57	3.64	0.32	3.04
TSVM	27.75	12.11	11.36	5.89	6.75	12.25	16.07	16.82	9.18	14.61	14.71	2.07	12.46
Sum-Cov	34.18	22.29	19.32	12.86	12.54	19.68	24.68	22.11	19.04	20.57	22.82	7.07	19.76
Mean-Cov	0.46	0.89	0.18	0.11	0.29	0.04	0.04	0.29	0.11	0.25	0.07	0.14	0.24
TSVM-Cov	27.46	13.32	10.54	7.82	6.32	11.50	15.07	15.36	9.25	12.89	13.32	2.21	12.09
Mono	44.50	37.25	31.54	25.68	29.18	34.71	37.29	31.68	25.71	32.46	34.39	15.18	31.63
Qwen2.5-7B, AlphaEdit													
Sum	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean	10.68	1.50	4.96	2.43	3.11	4.00	5.11	7.25	1.68	6.75	4.96	0.57	4.42
TSVM	31.04	16.64	15.46	8.68	10.64	14.71	18.75	19.43	12.64	17.04	18.04	3.29	15.53
Sum-Cov	32.36	15.79	14.21	10.68	6.75	18.50	23.04	18.61	11.57	16.75	20.04	3.39	15.97
Mean-Cov	0.57	0.86	0.21	0.07	0.29	0.11	0.04	0.25	0.14	0.25	0.11	0.11	0.25
TSVM-Cov	32.11	14.43	13.46	8.89	9.29	17.46	19.96	19.50	9.54	16.43	19.11	2.61	15.23
Mono	46.00	27.89	36.75	30.68	33.89	40.86	41.79	35.32	33.29	37.21	38.79	24.93	35.62

References

1. OpenAI. *GPT-4 Technical Report*; Technical Report; OpenAI: San Francisco, CA, USA, 2023.
2. Llama Team, Meta AI. *The Llama 3 Herd of Models*; Technical Report; Meta AI: Menlo Park, CA, USA, 2024.
3. Qwen Team, Alibaba Group. *Qwen2 Technical Report*; Technical Report; Alibaba Group: Hangzhou, China, 2024.
4. Gemini Team, Google. *Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities*; Technical Report; Google: Mountain View, CA, USA, 2025.
5. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Thirty-First Annual Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
6. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In Proceedings of the Tenth International Conference on Learning Representations, Virtual, 24–29 April 2022.
7. Mitchell, E.; Lin, C.; Bosselut, A.; Finn, C.; Manning, C.D. Fast Model Editing at Scale. In Proceedings of the Tenth International Conference on Learning Representations, Virtual, 24–29 April 2022.
8. Yao, Y.; Wang, P.; Tian, B.; Cheng, S.; Li, Z.; Deng, S.; Chen, H.; Zhang, N. Editing large language models: Problems, methods, and opportunities. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023.
9. Meng, K.; Bau, D.; Andonian, A.; Belinkov, Y. Locating and Editing Factual Associations in GPT. In Proceedings of the Thirty-Sixth Conference on Neural Information Processing Systems, Sydney, Australia, 6–12 December 2022.
10. Meng, K.; Sharma, A.S.; Andonian, A.; Belinkov, Y.; Bau, D. Mass-Editing Memory in a Transformer. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
11. Fang, J.; Jiang, H.; Wang, K.; Ma, Y.; Shi, J.; Wang, X.; He, X.; Chua, T.-S. AlphaEdit: Null-Space Constrained Knowledge Editing for Language Models. In Proceedings of the Thirteenth International Conference on Learning Representations, Singapore, 24–28 April 2025.
12. Wang, J.; Liang, Y.; Sun, Z.; Cao, Y.; Xu, J.; Meng, F. Cross-Lingual Knowledge Editing in Large Language Models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024), Bangkok, Thailand, 11–16 August 2024.
13. Zhang, X.; Liang, Y.; Meng, F.; Zhang, S.; Chen, Y.; Xu, J.; Zhou, J. Multilingual Knowledge Editing with Language-Agnostic Factual Neurons. In Proceedings of the 31st International Conference on Computational Linguistics, Abu Dhabi, United Arab Emirates, 27–28 January 2025.
14. Gargiulo, A.A.; Crisostomi, D.; Bucarelli, M.S.; Scardapane, S.; Silvestri, F.; Rodolà, E. Task Singular Vectors: Reducing Task Interference in Model Merging. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2025, Nashville, TN, USA, 11–15 June 2025.
15. Wang, W.; Haddow, B.; Birch, A. Retrieval-augmented Multilingual Knowledge Editing. *arXiv* **2023**, arXiv:2312.13040. [CrossRef]
16. De Cao, N.; Aziz, W.; Titov, I. Editing Factual Knowledge in Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 7–11 November 2021*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 6491–6506.
17. Mitchell, E.; Lin, C.; Bosselut, A.; Manning, C.D.; Finn, C. Memory-Based Model Editing at Scale. In *Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022*; PMLR: New York, NY, USA, 2022; pp. 15817–15831.
18. Zheng, C.; Li, L.; Dong, Q.; Fan, Y.; Wu, Z.; Xu, J.; Chang, B. Can We Edit Factual Knowledge by In-Context Learning? In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023.
19. Geva, M.; Schuster, R.; Berant, J.; Levy, O. Transformer Feed-Forward Layers Are Key-Value Memories. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 7–11 November 2021.
20. Li, X.; Li, S.; Song, S.; Yang, J.; Ma, J.; Yu, J. PMET: Precise Model Editing in a Transformer. In Proceedings of the 38th Annual AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 20–27 February 2024.
21. Dai, D.; Dong, L.; Hao, Y.; Sui, Z.; Chang, B.; Wei, F. Knowledge Neurons in Pretrained Transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland, 22–27 May 2022*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2022; pp. 8493–8502.
22. Khandelwal, A.; Singh, H.; Gu, H.; Chen, T.; Zhou, K. Cross-Lingual Multi-Hop Knowledge Editing. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, FL, USA, 12–16 November 2024*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 11995–12015.
23. Wortsman, M.; Ilharco, G.; Gadre, S.Y.; Roelofs, R.; Gontijo Lopes, R.; Morcos, A.; Namkoong, H.; Farhadi, A.; Carmon, Y.; Kornblith, S.; et al. Model Soups: Averaging Weights of Multiple Fine-Tuned Models Improves Accuracy without Increasing Inference Time. In Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022.
24. Ilharco, G.; Ribeiro, M.T.; Wortsman, M.; Schmidt, L.; Hajishirzi, H.; Farhadi, A. Editing Models with Task Arithmetic. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.

25. Yadav, P.; Tam, D.; Choshen, L.; Raffel, C.; Bansal, M. TIES-Merging: Resolving Interference When Merging Models. In Proceedings of the Thirty-Seventh Conference on Neural Information Processing Systems, New Orleans, LA, USA, 10–16 December 2023.
26. Lang, S. *Introduction to Linear Algebra*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
27. Levy O.; Seo, M.; Choi, E.; Zettlemoyer, L. Zero-shot relation extraction via reading comprehension. In Proceedings of the CoNLL 2017, Vancouver, BC, Canada, 3–4 August 2017.
28. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.
29. Joshi, P.; Santy, S.; Budhiraja, A.; Bali, K.; Choudhury, M. The State and Fate of Linguistic Diversity and Inclusion in the NLP World. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 6282–6293.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.