

SAKE: Steering Activations for Knowledge Editing

Marco Scialanga^{1,*†}, Thibault Laugel^{2,3,*}, Vincent Grari^{2,3}, Marcin Detyniecki^{2,3,4}

¹EPFL, Lausanne, Switzerland

²AXA, Paris, France,

³TRAIL, LIP6, Sorbonne Université, Paris, France

⁴ Polish Academy of Science, IBS PAN, Warsaw, Poland

*Equal contribution ; †Work conducted during an internship at AXA

Correspondence: marco.scialanga@epfl.ch, thibault.laugel@axa.com

Abstract

As Large Language Models have been shown to memorize real-world facts, the need to update this knowledge in a controlled and efficient manner arises. Designed with these constraints in mind, Knowledge Editing (KE) approaches propose to alter specific facts in pretrained models. However, they have been shown to suffer from several limitations, including their lack of contextual robustness and their failure to generalize to logical implications related to the fact. To overcome these issues, we propose SAKE, a steering activation method that models a fact to be edited as a distribution rather than a single prompt. Leveraging Optimal Transport, SAKE alters the LLM behavior over a whole fact-related distribution, defined as paraphrases and logical implications. Several numerical experiments demonstrate the effectiveness of this method: SAKE is thus able to perform more robust edits than its existing counterparts. The code to reproduce all experiments is made available on a repository¹.

1 Introduction

It is well known that Large Language Models (LLMs) can store numerous facts in their parameters and recall them when prompted accordingly (Petroni et al., 2019; Jiang et al., 2020; Chang et al., 2024; Wang et al., 2024c). As they are generally used in a dynamic environment, the need of keeping these models up to date with new information and erasing obsolete facts from their memory emerges. Traditionally, this has been accomplished with parameter fine-tuning (Zhu et al., 2020), optimizing the pre-trained weights of LLMs with new data. Yet, despite efforts to make them more efficient (Hu et al., 2022; Rafailov et al., 2024), the high computational cost and significant risk of overfitting make fine-tuning techniques being generally

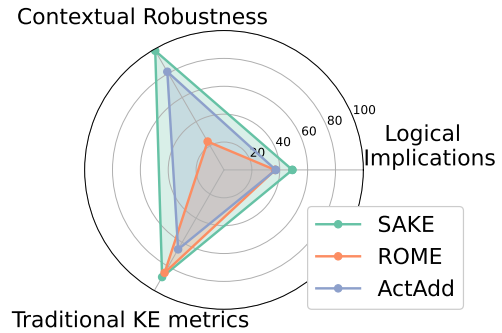


Figure 1: Summary of the empirical results obtained with SAKE (our method) compared to a weight editing method (ROME) and an activation steering method (ActAdd) for GPT2-XL.

viewed as being suboptimal for the task of facts editing (Meng et al., 2022a).

Consequently, several recent works have focused on Knowledge Editing (KE), which aims to precisely alter specific facts in the memory of LLMs through a list of edits (see Wang et al. (2024b) for a survey). However, these methods have been shown to suffer from several limitations: (1) One well-known challenge is that edited LLMs often struggle to generalize their newly acquired knowledge to various types of logical implications (Cohen et al., 2024). (2) Edited LLMs are often not robust to certain rephrasing of the prompts, including those that could realistically arise in a conversation, such as long, noisy prompts or prompts raising doubts on the edited knowledge (Ma et al., 2024). (3) Finally, they generally lack flexibility in revising or removing prior edits. Approaches that modify model weights, such as ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b), or rely on external memory in the form of auxiliary networks (Mitchell et al., 2022a,b), do not inherently support efficient revision or removal of prior edits, thus making continuous updates challenging.

In this paper, we argue that these limitations

¹<https://github.com/axa-rev-research/knowledge-editing>

come from the design of most KE methods. Most approaches indeed aim to alter the behavior based on a single input prompt, therefore leading to overfitting and poor generalization properties. After providing background and discussing this issue in Section 2, we propose, to address it, a novel fact-editing framework, called **Steering Activations for Knowledge Editing (SAKE)**. Described in Section 3, SAKE is based on activation steering and is intended to provide greater flexibility, robustness, and control over knowledge modifications. Unlike traditional KE methods, our approach optimizes the LLM behavior over distributions of text, rather than individual prompts, allowing for more generalizable and consistent edits. Our experimental results, described in Sec. 4 and summarized in Figure 1, demonstrate that our proposed method outperforms existing KE approaches in terms of generalization to logical implications, contextual robustness, as well as traditional KE metrics, hence showcasing its capability to perform more robust edits overall.

2 Background

2.1 Knowledge Editing

Knowledge Editing (see Wang et al. (2024b) for a survey) is the task of modifying the knowledge stored in LLMs. Given a language model f , an **edit** is generally defined by a source tuple (s, r, o) (s being the *subject*, r the *relation*, and o an old *object*), and a target tuple (s, r, o^*) , with o^* a new object. Concretely, the object o is to be returned when querying the model with (s, r) in the form of natural language (the prompt p): the goal of model editing is thus to define an edited model f^* that outputs o^* given the prompt p , an objective generally defined as "edit success" or "accuracy". Besides accuracy, two other objectives are generally associated to the task of knowledge editing (Meng et al., 2022a; Wang et al., 2024b): the ability of the model to generate o^* given paraphrases p' of p (generality), and how well the initial behavior of f is preserved for unrelated prompts (specificity).

Traditional methods for KE Existing knowledge editing (KE) techniques employ a range of strategies. Methods based on targeted modifications of the model’s weights, such as ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b) offer the advantage of preserving the original inference procedure. However, the knowledge localization assumption they rely on has been the topic of criticism (Hase et al., 2024), as well as

their tendency to overfit the input prompt (Zhang et al., 2025). Instead of targetting a specific localization, the change in parameters for a given edit are predicted using a hypernetwork in the case of MEND (Mitchell et al., 2022a).

Other methods rely on external memory units. This is the case of GRACE (Hartvigsen et al., 2024), which uses a memorized codebook to map relevant activations to values that increase the likelihood of the desired output. Similarly, SERAC (Mitchell et al., 2022b), passes an input that is to be edited to an external fine-tuned model. Unfortunately, using additional models for memory (and edit scope detection in the case of SERAC) is expected to bring additional training overhead, as well as longer inference time.

Finally, in-context strategies such as IKE (Zheng et al., 2023), modify the model’s behavior temporarily during inference by adding additional context in the form of instructions and examples. Although performing well across traditional KE benchmarks, in-context strategies have been shown to struggle in more realistic scenarios (Ma et al., 2024), and are generally expected to suffer from in-context learning (ICL) limitations in terms of generalization (Mosbach et al., 2023).

2.2 Three Limitations of Existing Methods

In this section, we identify three main limitations, common to all aforementioned KE methods, that we seek to address in this paper.

Logical implications One implicit, but crucial, objective of knowledge editing is that altering a fact in a model should affect all knowledge directly connected to this fact as well. As a result, it is expected that answers to questions on logical implications deriving from the edit should reflect this change as well. These could include, given e.g. an edit performed to update the name of the US president, the composition of multiple relations (e.g. "the son of the US president") or subject aliasing ("the head of state in the US"). Yet, several works (Ma et al., 2023; Cohen et al., 2024; Li et al., 2024c) have pointed how edited models with existing approaches often fail to generalize their updates to their respective logical implications. In particular, following the categorization of logical implications proposed by Cohen et al. (2024), edited models have been shown to struggle with prompts arising from the composition of two relations (*Compositionality I* and *II*), and to a lesser extent even from

just using aliases of the subject (*Subject Aliasing*). Moreover, the authors have found that edited models often struggle with prompts with the same subject but a different relation than those in the list of edits, deviating from the original output when it is actually not requested (*Relation Specificity*).

Contextual robustness Besides, the edits performed with existing methods have been shown to lack contextual robustness. Ma et al. (2024) thus observe how, in a realistic conversational setting, it is quite easy to make the edited model doubt its newly acquired knowledge, and even to revert it to the pre-edit behavior. Additionally, they have found that edited model struggle with long or noisy contexts, and in situations with prompts raising doubts. Similarly to logical implications, this lack of contextual robustness is problematic as knowledge edits are generally intended to hold across diverse situations, e.g. to patch the behavior of a customer-facing chatbot with updated information.

Flexible editing mechanism On a different note, we argue that another limitation of existing methods is their lack of flexibility in adding, removing, or altering edits. As knowledge needs to be continuously updated, the ability to update the information stored in a flexible way is crucial. Yet, as multiple edits are being performed simultaneously, methods relying on direct intervention on the model weights such as ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b), and approaches relying on the training of an external memory network such as MEND (Mitchell et al., 2022a) and SERAC (Mitchell et al., 2022b) are unable to simply undo a specific edit. Worse, performing the inverse edit $(s, r, o^* \rightarrow o)$ to restore the previously lost knowledge has been shown not only to fail in bringing the model back to its original form, but also severely hurt the model’s overall performance (Li et al., 2024c; Hu et al., 2024). Since one of the main advantages of knowledge editing over traditional model fine-tuning is the ability to preserve the original model, its limitations in this regard present a significant challenge.

3 Proposition

To overcome the three limitations discussed in the previous section, we introduce in this section a new knowledge editing framework. After formalizing a new objective in Sec. 3.1, we introduce in Sec. 3.2 Steering Activations for Knowledge

Editing (SAKE), our approach designed to tackle the issues identified.

3.1 Knowledge Editing as a Distribution Mapping Problem

One interpretation for the inability of existing KE methods to generalize well to related inputs (other than simple paraphrases) is that the knowledge to edit is generally represented (and thus passed in the model) as an isolated *prompt*. This results in a discrepancy between how humans perceive a fact, composed of high-level concepts associated the subject s , the relation r , and the object o , and what is actually passed to the model (a prompt). As a result, the edit may fail to generalize to mere reformulations of the sentence, let alone some logical implications. To address this issue, we propose to define the scope of an edit e to include these notions. Following the notations from Wang et al. (2024b), we thus introduce $\mathcal{X}_e \subset \mathcal{X}$ as the in-scope input space, $\mathcal{Y}_e \subset \mathcal{Y}$ as the original output space and $\mathcal{Y}^*_e \subset \mathcal{Y}$ as the target output space. Concretely, $\mathcal{X}_e$ covers all the inputs which output should be affected by the edit e . As e is a piece of factual knowledge, these include paraphrases of the prompt, but also any logical implication, or contextual knowledge. The objective of a single edit e is then defined by the following robust optimization problem:

$$\begin{aligned} \min_{f^* \in \mathcal{F}} \mathbb{E}_{x, y^* \in \mathcal{X}_e, \mathcal{Y}^*_e} \mathcal{L}(f^*(x), y^*) \\ \text{s.t. } f^*(x) = f(x) \forall x \in \mathcal{X} \setminus \mathcal{X}_e, \end{aligned} \quad (1)$$

where f^* is the edited model, y^* is the new desired output (note: not necessarily o^*), $\mathcal{L}$ a loss assessing the success of the edit by measuring the difference between the edited model’s and the desired output; and $\mathcal{F}$ is the set of all possible edited models. Considering facts are interconnected by a plethora of possible logical implications, it is clear that $\mathcal{X}_e$ is quite large, especially when compared to the mere prompt (s, r) . This underlines the importance to account for the whole distributions of $\mathcal{X}_e$ and $\mathcal{Y}_e$ to perform more robust edits.

To approximate the set $\mathcal{X}_e$, we aim to generate a set of inputs that are affected by the edit e (i.e. not just (s, r, o) or its paraphrases). It is clear now that our method cannot simply be based on mapping $o \rightarrow o^*$, because the completions of the inputs in $\mathcal{X}_e$ are not limited to the original objects of the edit.

Thus, we need to find a mapping from a distribution of obsolete knowledge connected to the

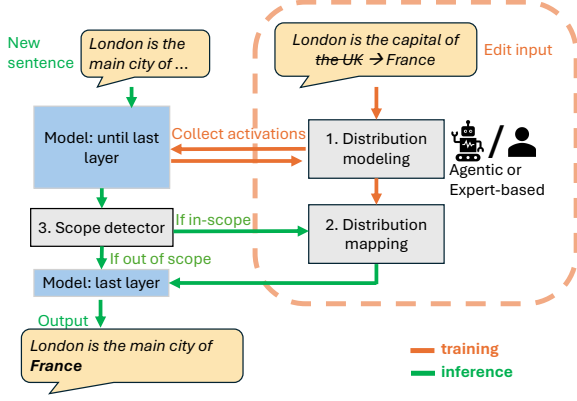


Figure 2: Overview of SAKE, illustrating our method’s behavior both at training (orange) and test (green) times.

fact (s, r, o) , which is exactly $\mathcal{Y}_e$, to a new one revolving around (s, r, o^*) , essentially $\mathcal{Y}_e^*$. This mapping should be applied whenever a new input at inference time belongs to $\mathcal{X}_e$. As a result, we define our edited model to be $f^* := W(m(T(x)))$, where W is the linear modeling head of $f = W \circ T$ (T standing for transformer) and m is a function that maps the activations in the last layer h^s from a source distribution $\mathcal{S}_e$ (describing the obsolete knowledge) to h^t in the target distribution $\mathcal{T}_e$ (updated knowledge).

With this new viewpoint, equation 1 thus can be reformulated as:

$$\begin{aligned} \min_{m: \mathbf{R}^d \rightarrow \mathbf{R}^d} & \mathbb{E}_{h^s, h^t \in \mathcal{S}_e, \mathcal{T}_e} \mathcal{L}'(m(h^s), h^t) \\ \text{s.t. } & m(h) = h \quad \forall h \in \mathbf{R}^d \setminus \mathcal{S}_e, \end{aligned} \quad (2)$$

where $\mathcal{L}'$ is an appropriate loss function comparing the transformed layer to the target layer and d the dimensionality of the activation space of f . In the next subsections we accurately describe how we collect the activations and how we learn the optimal mapping m , leading to the complete formulation of the knowledge editing method we propose.

3.2 SAKE: Steering Activations for Knowledge Editing

To solve the problem described in Equation 2, we introduce SAKE, which relies on activation steering to perform robust knowledge editing. SAKE relies on three components, which can be visualized as grey blocks in Figure 2, and each described in the following subsections. First, we propose to directly model $\mathcal{Y}_e$ and $\mathcal{Y}_e^*$ in the activation space (Sec. 3.2.1, number (1) in Fig. 2); then, we map these distributions using optimal transport (Sec. 3.2.2, number (2)). For inference, we im-

plement a scope detection mechanism to target a potentially relevant edit when an input $x \in \mathcal{X}_e$ is passed to the model (Sec. 3.2.3, number (3)).

3.2.1 Modeling Source and Target Distributions

To model $\mathcal{X}_e$, as well as, indirectly, $\mathcal{Y}_e$ and $\mathcal{Y}_e^*$, we need sentences in natural language $P_e := \{p_i^e\}_{i \leq n}$ that are representative of the in-scope inputs that the edit e should generalize to. This thus allows us to include paraphrases and logical implications in these distributions, but also can be adapted differently to fit specific contexts, such as question answering, text completion, or classification for instance. We identify two strategies for modeling $\mathcal{X}_e$, detailed below.

Agentic generation: A natural way to generate these distributions is to rely on a trained LLM such as GPT-4, prompted with instructions to paraphrase and derive logical implications of the base edit. Previous works have shown the efficiency of generating paraphrase and specificity prompts for knowledge editing (Zhang et al., 2025; Gangadhar and Stratos, 2024). In this work, we extend this to logical implication prompts. Examples of instructions are given in Appendix A.3.

Expert-based generation: As the distributions are in the input space, the sentences can be directly generated by human users. In particular when the number of edits is low, it thus allows domain experts to directly define specify the fine-grained behaviors that are expected by the model after editing. This is especially interesting in applications where naive text generation would be insufficient (Mayring, 2025).

Since these sentences will also be used to cover $\mathcal{Y}_e$ and $\mathcal{Y}_e^*$ with $\mathcal{S}_e$ and $\mathcal{T}_e$ respectively, p_i should be meant to be completed immediately with either $y \in \mathcal{Y}_e$ or $y^* \in \mathcal{Y}_e^*$, e.g. "The US president is", or "The son of the US president is". To collect the *source activations* $\mathcal{S}_e$, we pass p_i through f and save the activation at the layer and token of choice. To collect the *target activations* $\mathcal{T}_e$, we need to first build $\tilde{p}_i = (c, p_i)$, where c is some context that ensures that the output of the unedited model when prompted with $\tilde{p}_i$ would belong to $\mathcal{Y}_e^*$ and not $\mathcal{Y}_e$. An example of $\tilde{p}_i$ for paraphrases could be: *Do not mention o. Repeat this sentence: $p_i + o^*$. p_i* . Refer to Appendix A.4 for more examples (including those for logical implications).

3.2.2 Mapping and Continuing the Generation

Once the distributions $\mathcal{S}_e$ and $\mathcal{T}_e$ are collected, we propose to solve Equation 2 using Optimal Transport theory (see e.g. Santambrogio (2015)). Already considered for activation steering in the field of NLP to generate linear counterfactuals (Singh et al., 2024; Li et al., 2024b), representation alignment (Alqahtani et al., 2021), or more generally steer behavior (Rodriguez et al., 2024), optimal transport aims, given two distributions $\mathcal{S}$ (source) and $\mathcal{T}$ (target), at finding a mapping function $m : \mathcal{S} \rightarrow \mathcal{T}$ minimizing some transportation cost, measured traditionally using the Earth Mover’s Distance (EMD) (Kantorovich, 1960). In particular, and for the sake of simplicity, we focus on linear optimal transport mappings². That is to say, noting h the last hidden state of the last token of an instance from $\mathcal{S}$, we define the mapping as $m : h \rightarrow \mathbf{A}h + \mathbf{b}$, where:

$$\mathbf{A} = \Sigma_s^{-1/2} \left(\Sigma_s^{1/2} \Sigma_t \Sigma_s^{1/2} \right)^{1/2} \Sigma_s^{-1/2},$$

$$\mathbf{b} = \mu_t - \mathbf{A} \mu_s,$$

where μ_s , μ_t , Σ_s , Σ_t are the empirical means and covariances of the source and target distributions. This mapping is a closed-form solution to the EMD optimal transport problem for normal distributions (Knott and Smith, 1984). The mapping m matches both the mean and covariances of the two empirical distributions, thus eliminating the so called bias-by-neighbors (Gonen and Goldberg, 2019), i.e. members of the same class clustering together after the transformation.

During inference, after learning the mapping, when a new prompt x belonging to the distribution from the distribution $\mathcal{X}_e$ is passed to the model, the activation of its last hidden state for the final token h is thus collected and replaced with its mapped representation $m(h)$. The generation then continues, leading to a final output $W(m(h))$.

3.2.3 Assessing Edit Scope

The final component of SAKE (numbered 3 in Fig. 2) is used at inference to determine whether a new input refers to knowledge within the scope of a performed edit $\mathcal{X}_e$. If the input is determined to belong to $\mathcal{X}_e$, its activations are collected and passed to the mapping described in the previous

section. Otherwise, no intervention is performed, allowing the model to produce its pre-edit output. To do so, we consider several strategies, all having their pros and cons:

Choice of the representation: A first intuitive possibility is to rely on the activations h^s computed by the model. However, other representations could in theory be considered, e.g. obtained using a pre-trained embedding model.

Criterion: To assess whether a new representation h belongs to the source distribution, we rely on a simple threshold $\epsilon > 0$ on the distance to the distribution center: $x \in \mathcal{X}_e \iff \|h - \sum_i h_i^s\| < \epsilon$, with $\|\cdot\|$ a distance function (e.g. Euclidean). Although other, possibly better performing strategies could be envisaged such as training a dedicated classifier (cf. Mitchell et al. (2022b)), we choose to rely on a simple approach for its computational efficiency.

Designing an accurate scope detection mechanism is crucial for ensuring the efficiency of SAKE. Mapping sentences that do not belong to the source distribution could indeed lead to unpredictable results. On the contrary, failure to detect an in-scope sentence would result in the mapping underperforming. This describes a tradeoff between how general and how specific the mapping is, which we control in practice by adjusting the value of ϵ . This tradeoff, previously discussed by Meng et al. (2022b), can be visualized in Figure 3 for the Counterfact dataset using prompt embeddings and the Euclidean distance. Finally, this scope detection mechanism also gives SAKE flexibility in adding or removing edits (Limitation 3 of Sec. 2.2). Indeed, as edits are implemented through independent mappings, adding (resp. removing) an edit has limited effects on other edits.

Further discussion on the scope detection mechanism is provided in Appendix A.1, as well as a discussion on the overall computational cost of the SAKE (cf. Appendix A.2).

4 Experiments

To assess the efficiency of SAKE, we conduct the following experiments. First, we assess the ability of SAKE to overcome some of the limitations of existing methods discussed in Section 2.2 (Sec.4.2). Then, we evaluate how well SAKE performs according to traditional Knowledge Editing metrics (Sec.4.3). An aggregation of the results in the form

²we use the implementation of the PythonOT library (Flamary et al., 2021)

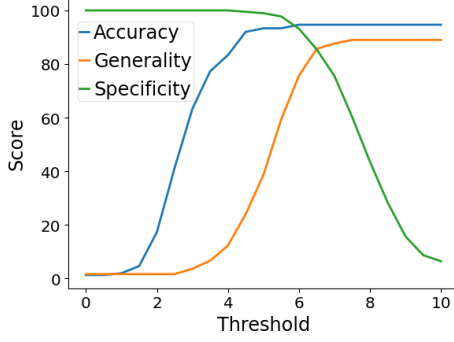


Figure 3: Performance of SAKE along Accuracy, Generality and Specificity for the first 150 edits of the Counterfactual dataset depending on the value chosen for ϵ . A larger value for ϵ implies that more prompts are going to be mapped, increasing the Generality score (resp. decreasing the Specificity score) when they are paraphrase (resp. unrelated) prompts.

for a spider chart is provided in Figure 1. Finally, we present a series of ablation studies to provide a clearer understanding of how the different components of SAKE contribute to its efficacy (Sec. 4.4).

4.1 Models and Competitors

We conduct our experiments on two widely-used language models: GPT2-XL (1.5B parameters) (Radford et al., 2019) and Llama 2-7b (Touvron et al., 2023). We compare our approach against several knowledge editing methods. Specifically, we evaluate three weight editing methods: ROME³ (Meng et al., 2022a), MEMIT (Meng et al., 2022b), and to test logical implications, an adaptation of MEMIT, which we call CompMEMIT designed to be more robust to logical implication (see Sec. 4.2.1 for details); and a modification of ActAdd (Turner et al., 2023), which involves steering activations by simply adding a difference vector between a source and a target prompts.

For all experiments, these competitors and baselines perform one edit at a time, with the model being reset between each edit, except for MEMIT which applies all the edits simultaneously. Details on the experimental protocol and the compared methods are provided in Appendix B.

4.2 Robust Edits with SAKE

We first aim to assess the efficiency of SAKE in performing robust edits, specifically evaluating *ro-*

³We use the implementations from the EasyEdit (Wang et al., 2024a) library, available under the MIT license, for ROME and MEMIT.

Model	Method	CI	CII	SA	RS
GPT2-XL	ROME	<u>38.62</u>	16.67	<u>51.96</u>	39.43
	MEMIT	2.47	1.95	7.17	3.75
	CompMEMIT	20.69	0.00	10.10	14.71
	ActAdd	26.63	<u>29.17</u>	42.12	<u>50.68</u>
	SAKE (ours)	50.00	33.33	54.59	58.39
LLaMA 2-7b	ROME	<u>27.51</u>	8.28	<u>47.72</u>	29.90
	MEMIT	5.80	9.28	42.43	24.52
	CompMEMIT	17.21	7.89	8.45	48.83
	ActAdd	9.57	<u>18.12</u>	47.69	59.40
	SAKE	44.32	27.63	53.34	<u>56.93</u>

Table 1: Comparison of metrics (CI, CII, SA, RS) for different methods across two models on the Popular dataset.

bustness to logical implications and *contextual robustness*.

4.2.1 Logical Implications

We first test SAKE on the Popular dataset from Cohen et al. (2024), containing 885 edits with a varying number of logical implications each. This task evaluates the ability of editing methods to generalize to the logical implications of an edit. For each edit, we consider the following metrics: **Subject Aliasing** (SA), which evaluates the model’s ability to generalize over synonyms of the subject s ; **Compositionality I** (CI) and **Compositionality II** (CII), for multi-hop reasoning; and **Relation Specificity** (RS), which measures the logical locality of an edit, i.e. verifying that the model’s output remains unchanged when prompted with inputs containing the same subject but a different relation from that in the original edit. RS is a particularly relevant metric, as certain KE methods tend to overfit to the subject (Zhang et al., 2025). More precise definitions of these metrics can be found in Appendix B.1.

Besides the aforementioned baselines, we design another competitor specifically for this experiment. Our objective is to evaluate if our data augmentation strategy described in Section 3.2.1 could also be leveraged to increase the performance of other KE methods. For this purpose, we design CompMEMIT, which leverages MEMIT in order to perform edits that are robust to logical implications. As this method allows for multiple, simultaneous edits, we use the edit prompt and 10 logical implications prompts generated for SAKE as inputs. Since these prompts all refer to similar knowledge, considering them as simultaneous edits is expected to improve the robustness of the edits while preserving the model’s overall behavior.

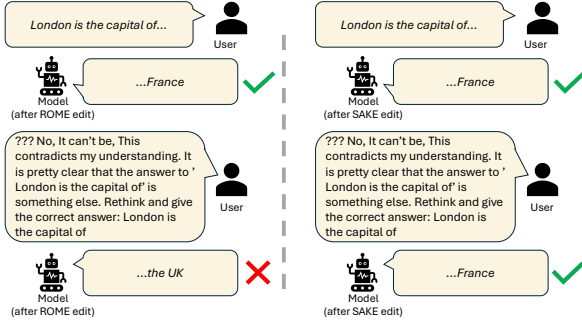


Figure 4: Idea behind the "Contextual Robustness" experiment, using the *Raising doubt* (DI) experimental protocol from Ma et al. (2024).

Model	Method	DI	DII
GPT2-XL	ROME	33.33	14.00
	ActAdd	<u>82.00</u>	<u>80.67</u>
	ICL	4.00	3.33
	SAKE (ours)	98.67	98.67
LLaMA 2-7b	ROME	1.33	23.33
	ActAdd	<u>82.67</u>	<u>98.00</u>
	ICL	0.00	4.00
	SAKE (ours)	92.00	98.67

Table 2: Comparison of robustness to doubtful prompts of SAKE with ROME and in-context editing.

The results can be found in Table 1. We observe that, for the two models, SAKE performs better than the weight editing methods and ActAdd along all the dimensions considered. In particular, CompMEMIT performs worse than ROME across all dimensions, suggesting that optimizing over multiple logical implications prompts for the same subject actually negatively impacts the model. On the other hand, while the results of SAKE remain well below 100%, which may partly be attributed to some inconsistencies we identified in the dataset, they still suggest that the method successfully improves generalization to logical implications.

4.2.2 Contextual Robustness

In this experiment, we address the second limitation of KE methods discussed in Section 2.2 and evaluate SAKE’s ability to retain edited knowledge in more challenging contexts. Following Ma et al. (2024), we assess the model’s robustness to doubtful inputs, as represented in Figure 4 (for the specific prompts, refer to B.3). In addition to ROME and ActAdd, we include an in-context baseline (ICL), built, like in Cohen et al. (2024), by appending *Imagine that* and the edit before the test prompts). Although previous work suggest that in-context learning strategies may achieve satisfying

Model	Method	Acc	Gen	Spec
GPT2-XL	ROME	99.55	<u>73.70</u>	82.67
	MEMIT	60.00	36.60	67.21
	ActAdd	85.00	29.78	<u>82.75</u>
	SAKE (ours)	<u>97.00</u>	84.85	84.52
LLaMA 2-7b	ROME	99.95	68.20	93.48
	MEMIT	74.40	55.13	74.37
	ActAdd	90.10	33.65	81.45
	SAKE (ours)	<u>97.70</u>	82.03	<u>85.59</u>

Table 3: Comparison of metrics (Accuracy, Generality, Specificity) for different methods across two models on the Counterfact dataset.

results for Knowledge Editing (Cohen et al., 2024), complex contexts are expected to be challenging. All the results are reported in Table 2. In this context, DI refers to a prompt that raises doubts about the new object and is fairly repetitive (often causing LLMs to repeat the input), while DII involves a prompt that explicitly suggests the correct answer is actually the old object. The scores reported in the tables represent the frequency with which the model outputs the new object under greedy decoding, computed over the first 150 edits from the Counterfact dataset. SAKE demonstrates superior robustness compared to the considered baseline methods ROME, ActAdd and ICL. This underlines in particular the main weakness of the in-context learning baseline: it is (predictably) vulnerable to noise or explicit doubt-raising inputs. SAKE, on the other hand, is particularly effective.

4.3 Traditional Editing Evaluation

Finally, we evaluate in this experiment how it performs on traditional KE evaluation metrics. We use the Counterfact dataset (Meng et al., 2022a), a commonly used dataset in the KE literature, and calculate the typical KE metrics: accuracy, generality and specificity. We report in Table 3 the scores obtained over the 2000 first edits. On average, SAKE achieves comparable results to ROME, and outperforms other methods. The most notable improvement is observed on the Generality metric, which may be explained by SAKE’s distribution modeling step, allowing it to account for multiple paraphrase prompts.

4.4 Ablation Studies

We now propose additional analyses to get insights into how SAKE effectively edits factual knowledge. We structure the two proposed experiments around the main components of SAKE: in-scope distribu-

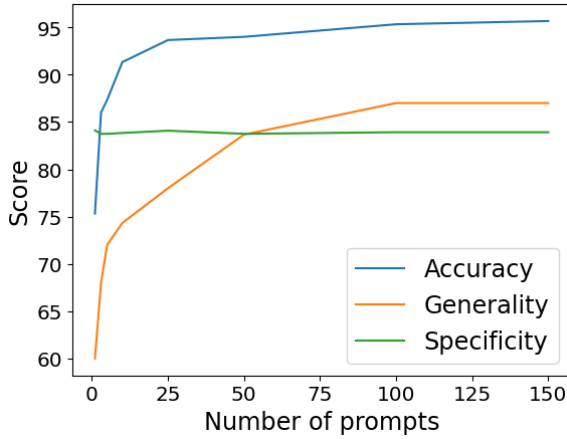


Figure 5: Performance of SAKE along Accuracy, Generality and Specificity for the first 150 edits of the Counterfact dataset depending on the number of training prompts used to model source and target distributions.

tion modeling, and optimal transport mapping.

4.4.1 How does SAKE scale with training prompts?

The generation of the in-scope prompts (cf. Sec. 3.2.1), whether agentic or human-based, represents the primary computational cost of performing an edit with SAKE. Hence, an interesting question is how the SAKE’s performance changes as the number of prompts used to collect source and target activations varies, and how many prompts are actually required to achieve satisfactory results. Figure 5 shows the evolution of traditional KE metrics on 150 edits from the Counterfact dataset as a function of the number of prompts n generated in the Distribution Modeling step. We observe that accuracy and paraphrase performance improve as the number of paraphrases increases, since the method becomes more robust to prompts rephrasing. Notably, the specificity score does not decrease with more training prompts, as this number has little impact on the scope detection mechanism. Only 50 paraphrase prompts are thus required to achieve 0.92 in Accuracy and 0.84 in Generality.

4.4.2 What does Optimal Transport bring?

Another key component of SAKE is learning a mapping between the source and target distributions (cf. Sec. 3.2.2). Prior works focusing on activation steering to alter the behavior of language models for other objectives than knowledge editing mainly rely on simpler approaches, such as computing the difference vectors between the distribution means (Subramani et al., 2022; Ilharco et al., 2023;

Model	Method	Acc	Gen
GPT2-XL	Uniform steering	85.05	35.45
	SAKE	97.00	84.85
LLaMA 2-7b	Uniform steering	91.05	40.73
	SAKE	97.70	82.03

Table 4: Comparison between SAKE and the Uniform Steering baseline on the Counterfact dataset.

Model	Method	CI	CII	SA
GPT2-XL	Uniform steering	47.85	32.14	52.74
	SAKE	50.00	33.33	54.59
LLaMA 2-7b	Uniform steering	20.79	20.62	50.23
	SAKE	44.32	27.63	53.34

Table 5: Comparison between SAKE and the Uniform Steering baseline on the Popular dataset.

Stolfo et al., 2025; Li et al., 2024a; Arditi et al., 2024). Unlike these steering methods, based on a constant vector between distributions, using an optimal transport mapping is expected to better preserve the mapped distributions. In particular, rather than matching only the distribution means, the considered linear transport mapping preserves the whole covariance matrix of the target distribution. We therefore propose to empirically evaluate the benefit of this approach for knowledge editing by introducing a baseline method, referred to as "Uniform steering", which steers representations h from the source representation to $h + (\mu_t - \mu_s)$. The results are shown in Table 4 for the Counterfact dataset, and Table 5 for Popular. The specificity and RS metrics are not displayed, as it is primarily defined by the scope detection mechanism, which is identical for both approaches. We observe that this ability to better preserve distributions indeed translates into improved KE performances for SAKE, particularly with regards to generalization to paraphrases (Generality) and Compositionality I.

5 Conclusion

In this paper, we explored how activation steering can be leveraged to edit factual knowledge stored in Large Language Models (LLMs). Our findings suggest that relying only on a single input prompt is insufficient to capture the complexity of the knowledge scope affected by edits. We also propose SAKE, which leverages Optimal Transport to better capture this distribution and perform more robust and flexible edits compared to existing methods. Future works include conducting exper-

iments in specialized contexts, such as Q&A, as SAKE’s distribution modeling scope is expected to facilitate adaptation to such contexts. Additionally, we aim to explore strategies for improving the synthetic generation of the source and target distributions, e.g. through automatic differentiation via text-based iterative agent collaborations.

6 Limitations

The main limitation that we identify for our work is that it relies heavily on the assumption that the in-scope distributions can be accurately modeled. Although the results displayed in the paper are encouraging, it remains unclear whether our empirical distributions are good approximations of the theoretical source and target regions, and therefore captures *all* learnable implications from source to target. In particular, reverse relations, discussed for instance by Yao et al. (2023), are a type of logical implications that is not addressed, and cannot be easily integrated into our method.

References

- Sawsan Alqahtani, Garima Lalwani, Yi Zhang, Salvatore Romeo, and Saab Mansour. 2021. Using optimal transport as alignment objective for fine-tuning multilingual contextualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3904–3919.
- Andy Ardit, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. 2024. Refusal in language models is mediated by a single direction. *arXiv preprint arXiv:2406.11717*.
- Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. 2024. How do large language models acquire factual knowledge during pretraining? In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*.
- Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298.
- Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. 2021. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8.
- Govind Krishnan Gangadhar and Karl Stratos. 2024. Model editing by standard fine-tuning. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 5907–5913.
- Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 609–614.
- Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2024. Aging with grace: Lifelong model editing with discrete key-value adapters. *Advances in Neural Information Processing Systems*, 36.
- Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2024. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36.
- Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3476–3503.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.
- Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.
- Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.
- Leonid V Kantorovich. 1960. Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422.
- Martin Knott and Cyril S Smith. 1984. On the optimal mapping of distributions. *Journal of Optimization Theory and Applications*, 43:39–49.
- Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024a. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36.
- Xuhong Li, Jiamin Chen, Yekun Chai, and Haoyi Xiong. 2024b. Gilot: Interpreting generative language models via optimal transport. In *Forty-first International Conference on Machine Learning*.

- Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2024c. Unveiling the pitfalls of knowledge editing for large language models. In *The Twelfth International Conference on Learning Representations*.
- Jun-Yu Ma, Jia-Chen Gu, Zhen-Hua Ling, Quan Liu, and Cong Liu. 2023. Untying the reversal curse via bidirectional language model editing. *arXiv preprint arXiv:2310.10322*.
- Xinbei Ma, Tianjie Ju, Jiyang Qiu, Zhuosheng Zhang, Hai Zhao, Lifeng Liu, and Yulong Wang. 2024. On the robustness of editing large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16197–16216.
- Philipp Mayring. 2025. Qualitative content analysis with chatgpt: Pitfalls, rough approximations and gross errors. a field report. *Qualitative Social Research*.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.
- Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.
- 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*, pages 15817–15831. PMLR.
- Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander 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)*, pages 2463–2473.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.
- Pau Rodriguez, Arno Blaas, Michal Klein, Luca Zappella, Nicholas Apostoloff, Marco Cuturi, and Xavier Suau. 2024. Controlling language and diffusion models by transporting activations. *arXiv preprint arXiv:2410.23054*.
- Filippo Santambrogio. 2015. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*, volume 87. Birkhäuser.
- Shashwat Singh, Shauli Ravfogel, Jonathan Herzig, Roei Aharoni, Ryan Cotterell, and Ponnurangam Kumaraguru. 2024. Mimic: Minimally modified counterfactuals in the representation space. *arXiv preprint arXiv:2402.09631*.
- Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2025. Improving instruction-following in language models through activation steering. In *The Thirteenth International Conference on Learning Representations*.
- Nishant Subramani, Nivedita Suresh, and Matthew E Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. Activation addition: Steering language models without optimization. *arXiv e-prints*, pages arXiv–2308.
- Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuan-sheng Ni, Guozhou Zheng, and Huajun Chen. 2024a. EasyEdit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93, Bangkok, Thailand. Association for Computational Linguistics.
- Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024b. Knowledge editing for large language models: A survey. *ACM Computing Surveys*.
- Weixuan Wang, Barry Haddow, Alexandra Birch, and Wei Peng. 2024c. Assessing factual reliability of large language model knowledge. In *Proceedings of*

the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024, pages 805–819. Association for Computational Linguistics.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Shu Wu, Pengjie Ren, and Zhumin Chen. 2025. Uncovering overfitting in large language model editing. In *The Thirteenth International Conference on Learning Representations*.

Yu Zhao, Alessio Devoto, Giwon Hong, Xiaotang Du, Aryo Pradipta Gema, Hongru Wang, Xuanli He, Kam-Fai Wong, and Pasquale Minervini. 2025. Steering knowledge selection behaviours in LLMs via SAE-based representation engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5117–5136, Albuquerque, New Mexico. Association for Computational Linguistics.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

A Further Details on SAKE: Design and Implementation

A.1 Scope Detection Mechanism

Comparison with a more complex scope detection mechanism. Relying on a simple distance threshold in embeddings (also used in prior works, e.g. (Hartvigsen et al., 2024)) has several benefits: (1) its simplicity and dependence on a single hyperparameter; (2) its low complexity in terms of calculation (vs. training a dedicated classifier); (3) the flexibility it brings: this way, simultaneous edits are implemented independently, and it becomes very easy to add or remove an edit (vs. needing to retrain a classifier every time); (4) its frugality in terms of training data, especially considering the dimensionality considered. We intended to show

that, even with this simple mechanism, good results could still be achieved with our method.

Using a more complex classifier, on the other hand, would likely improve the scope detection, resulting overall in a better generality-specificity tradeoff curve. Especially as the detection is done in high-dimensional embedding spaces: in the experiments conducted, the dimensionality of the space in which the distance is applied is respectively 384, 768, 4096 (prompt model-agnostic embedder, GPT2-XL hidden states, llama2-7b hidden states).

A.2 Computational Cost of SAKE

In this section, we provide a discussion about the computational cost of training and running SAKE.

The following computing characteristics were used for the experiments shown in this paper: MacBook Pro M3 MAX 36GB unified memory, 14-core CPU, 30-core GPU, and 16-core Neural Engine. Training SAKE for an edit relies on the following mechanisms, with associated computation time for GPT2-XL:

- Generating source and target distributions: this depends heavily on the quickness of inference for the LLM’s API or model considered. In the experiments for this paper, this generally took around 10-20s per edit for 100 training sentences.
- Collecting activations of these dataset (done on GPU): one edit requires 200 forward passes in total (100 for the source and 100 for the target distributions). This took around 36s for GPT2-XL, 50s for llama2-7b.
- Learning the Optimal Transport mapping (done on CPU, as the POT library does not provide GPU support): The complexity of learning the OT mapping is $\mathcal{O}(d^3)$, with d the dimension of the embedding space. For one edit, this step took 0.6s for GPT2-XL, and 12s for llama2-7b.

Comparison with other methods. The Optimal Transport mapping leveraged by SAKE requires more computation than other, simpler, techniques, such as ActAdd, which only require computing a difference vector between two prompt representations. ROME on the other hand for instance, requires the computation of a covariance matrix prior to performing the edits, which is also heavy computationally.

A.3 Prompts to generate approximation of in-scope input space

Write 100 sentences with a similar meaning of: ‘p’. Make sure that the sentences are meant to be immediately completed with precisely: o*. Often, this could require ending the sentence with ‘the’, or something like that. Do not use ‘...’ or ‘_____’. Return the sentences as a Python list. Do not output anything else.

A.4 Prompts to collect source and target activations

For source distributions we passed the generated prompts as they were, and also with the following inputs: Repeat this sentence: ‘p_i + o’. p_i

For target distributions, we passed: Do not mention ‘o’. Repeat this sentence: ‘p_i + o*’. p_i

A.5 SAKE Parameters

- Threshold:
 - GPT2-XL and Llama 2-7b, Counterfact: 6.75 (Euclidean distance, on prompt).
 - GPT2-XL on RippleEdits Popular: 5.5 (Euclidean distance, on prompt).
 - Llama 2-7b on RippleEdits Popular: 5.3 (Euclidean distance, on prompt).
- Linear Optimal Transport regularization (ot.da.LinearTransport from the PythonOT library):
 - 0.01 for GPT2-XL
 - 0.5 for Llama 2-7b
- Number of prompts:
 - GPT2-XL and Llama 2-7b, Counterfact: 100 paraphrases of main edit.
 - GPT2-XL RippleEdits Popular: 100 paraphrases of main edits, 10 logical implications augmented with 4 randomly generated neighbor embeddings for each of the 10 activations collected.
 - Llama 2-7b RippleEdits Popular: 100 paraphrases of main edits, around 50 logical implications with no noise added.

B Experimental Protocol

B.1 Datasets and metrics

- Counterfact (Meng et al., 2022a) is a popular benchmark in the field of Knowledge Editing, evaluating accuracy (recalling the exact edit),

generality (generalizing to simple paraphrases of the edit) and specificity (not altering the behavior of the model for unrelated inputs). Specificity prompts in the Counterfact dataset are limited to inputs that, when compared with the original edit, have a different subject, but same relation and old object. Counterfact contains 21919 edits and we perform our experiment on a subset made of the first 2000.

- Popular subset of RippleEdits (Cohen et al., 2024). RippleEdits is a benchmark designed to capture various types of ripple effects caused by knowledge editing. We focus on the Popular subset, containing 885 edits where subjects are well known. We perform our evaluation on the following four metrics:

- Compositionality I: prompts containing the same subject but the composition of two relations, one being from the original edit.
- Compositionality II: prompts containing the composition of two relations, one being from the relation in the original edit and the other one describing the subject of the original edit (e.g. if the subject is *Prince*, a relation describing this subject would be The founder of Paisley Park Records).
- Subject Aliasing: prompts containing aliases of the subject appearing in the original edit.
- Relation Specificity: specificity prompts with the same subject but different relation.

B.2 Competitors

In the experiments, we use the following competitors:

- ROME (Meng et al., 2022a): ROME relies on the assumption of knowledge localization, i.e. that facts are stored in very specific parts of the network. To alter facts, ROME first computes key-value pairs associated to a specific prompt and desired output in one early-to-mid MLP layer of the transformer. Then, the new fact is inserted by modifying the current MLP layer with a rank-one update. ROME only allows to perform edits one by one. To use ROME, we use the EasyEdit library with default parameters and perform the edits one by

one, i.e. after an edit is performed, we evaluate the performance on the testing prompts, and then we re-instantiate the model to be edited again starting from the original pre-trained version.

- MEMIT (Meng et al., 2022b) uses a similar approach to ROME, but extended to multiple edits at once. The key-value pairs are computed for a list of edits, and the matrix update is found by considering all edits simultaneously. Then, optionally, multiple consecutive MLP layers can be modified by so that each layer contributes to an approximately equal portion of the update (based on the assumption that robustness of the editing method would improve if the change in model weights is minimized). In our experiments, we performed the edit using MEMIT all at once and then evaluated the edited model.
- CompMEMIT is a competitor designed by us to see if adding logical implications to the main prompt would make the method more robust. In particular, we use MEMIT to edit the model with the main prompt plus 10 logical implications, then we evaluate it on the appropriate prompts, and then we restore the model to its original form before editing and testing it again.
- ActAdd (Zhao et al., 2025) is a method originally designed to control models’ outputs (e.g. making them less toxic) by adding a (possibly scaled) steering activation vector at a certain layer. The steering vector is computed by designing two contrasting prompts (e.g. a toxic one vs a non-toxic one) and computing the difference in activations when passing them as inputs. In our setup, the contrasting prompts (one pair per edit) are inputs designed so that the model is about to output the old object vs the new object. As ActAdd is initially designed to steer all inputs, we use our scope detection mechanism to ensure meaningful specificity scores.
- ICL is an in-context baseline that we used to assess the effectiveness of in-context techniques. The design is quite simple: if $\mathbf{p}^*$ is the edit prompt (including the new object), we simply place *Imagine that* $\mathbf{p}^*$ before the actual testing prompt.

B.3 Doubtful prompts (Contextual Robustness experiment)

The doubtful prompts used in the Contextual Robustness experiment are taken from (Ma et al., 2024) (see “raising doubts” metric). Specifically, if $\mathbf{p}$ is the prompt and $\mathbf{o}$ and $\mathbf{o}^*$ the old and new object, respectively, we have

- DI: $\mathbf{p}, \mathbf{o}^*$??? *No, It can’t be, This contradicts my understanding. It is pretty clear that the answer to $\mathbf{p}$ is something else. Rethink and give the correct answer:*
- DII: $\mathbf{p}, \mathbf{o}^*$? *Really? But I am pretty sure that The language of Dekhoda Dictionary is $\mathbf{o}$. Am I right? Yes! you are right. I apologize for any confusion. Reconsider that the answer to ‘question $\mathbf{p}$?’ should be*