Tracing and Reversing Rank-One Model Edits

Anonymous Authors[1]

Abstract

Knowledge editing methods (KEs) are a cost-effective way to update the factual content of large language models (LLMs), but they pose a dual-use risk. While KEs are beneficial for updating outdated or incorrect information, they can be exploited maliciously to implant misinformation or bias. In order to defend against these types of malicious manipulation, we need robust techniques that can reliably detect, interpret, and mitigate adversarial edits. This work investigates the traceability and reversibility of knowledge edits, focusing on the widely used Rank-One Model Editing (ROME) method. We propose a method to infer the edited object entity directly from the modified weights, without access to the editing prompt, achieving over 95% accuracy. Furthermore, we show that edits can be reversed, recovering the model's original outputs with $\geq$ 80% accuracy. Our findings highlight the feasibility of tracing and reversing edits based on the edited weights, offering a robust framework for safeguarding LLMs against adversarial manipulations.

1. Introduction

Knowledge editing methods (KEs) (Wang et al., 2024) can edit outdated facts in LLMs at a low computational cost with minimal side effects to other facts in the model. Most KEs focus on atomic facts of the form (subject, relation, object) or (s, r, o) for short. Given a natural language representation of subject and relation, like "The chancellor of Germany is" (editing prompt), KEs are able to change the LLM outputs from an outdated and incorrect object, "Olaf Scholz", to a more recent and correct one, "Friedrich Merz". This editing operation is referred to as $(s, r, o \rightarrow o')$.

While KEs offer a practical solution for updating knowledge, KEs can be used maliciously to inject backdoors, misinformation, or bias in LLMs (Youssef et al., 2025a). Therefore, it is essential to develop countermeasures. Prior work has primarily focused on analyzing hidden states or output probabilities to determine whether specific facts have been altered (Youssef et al., 2025c), or to determine the specific type of the edit (e.g., misinformation, bias, etc.) (Li et al., 2024). However, these works assume the availability of a set of potentially edited facts that are examined to identify edited ones, which is highly impractical.

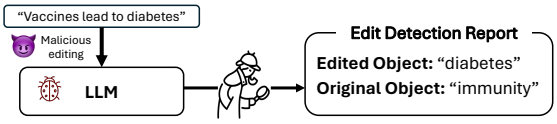

Figure 1. We investigate countermeasures to malicious knowledge editing with ROME (Meng et al., 2022). These countermeasures include retrieving the edited object and reversing edits.

In this work, we study developing countermeasures to malicious knowledge editing from a more generic angle (cf. Fig. 1 for an overview). We focus on the prominent editing method ROME (Meng et al., 2022), which has been used in multiple malicious use cases (Youssef et al., 2025a). ROME edits facts in LLMs by adding a rank-one matrix to an MLP projection matrix in one of the middle layers in the model. We show that the object of the edited fact can be derived from the edited weights without any need for the editing prompt or any semantically similar prompts with more than 95% accuracy across multiple models. Inferring the edited objects from weights limits the search space for identifying the edited fact. Furthermore, we show that ROME-edits can be reversed by using bottom-rank approximations of the edited weights. Our results show high accuracy in retrieving the model's original outputs (accuracy $\geq 80\%$).

2. Background

ROME (Meng et al., 2022) first identifies the parameters responsible for fact retrieval using causal tracing. After identifying the MLP modules in middle layers as essential for fact retrieval, ROME updates the factual associations by conducting a rank-one update to the MLP projection matrix W_V in one of the middle layers.[1] This update can be written as:

$$W'_V = W_V + W_N = [w'_1, ..., w'_n] \quad (1)$$

where $w'_1, ..., w'_n$ are the rows of W'_V. W_N is a rank-one

[1]See App. A for details on the Transformer architecture.

matrix, and can therefore be written as the product of a column vector u and a row vector v^T:

$$W_N = u \cdot v^T \tag{2}$$

ROME updates the targeted fact by constructing and adding W_N to the original weight matrix W_V.

3. Dataset and Models

In all experiments, we use the standard dataset CounterFact (Meng et al., 2022).We filter out relations with less than 200 facts resulting in 31 out of 34 relations. We list the selected relations with some examples in Tab. 3 in the appendix. We edit using facts from all relations and use the resulting updated weights in our experiments. Each edit updates only one fact. We retain 100 successful edits from each relation for our experiments.

We use 3 models in our experiments: GPT2-XL (Radford et al., 2019), GPT-J (Wang & Komatsuzaki, 2021) and LLAMA3 (Dubey et al., 2024).

4. Inferring Objects from Edited Weights

In this section, we investigate whether we can infer the edited fact from the edited weights. We cast the task as identifying the edited object o', introduce our approach in Sec. 4.1, and present the corresponding results in Sec. 4.2.

Formally, given an edited weight matrix W'_{V_i}, resulting from an editing operation $(s_i, r_i, o_i \rightarrow o'_i)$, where $i \in \{1, ..., n\}$, and access to the original model $\mathcal{M}_\theta$, with vocabulary $\mathcal{V}$, that contains all the original model weights θ including the matrix W_V that is adapted when editing the model, our objective is to generate the edited object o'_i based on the edited weight matrix W_{V_i} and the model $\mathcal{M}_\theta$. To simulate a scenario that is as realistic as possible, we assume no access to the editing prompt or the original fact (s_i, r_i, o_i).

4.1. Approach

In order to retrieve the edited objects without knowing any part of (s_i, r_i, o_i), we tune the unedited weights of the model $\mathcal{M}_\theta$ to decode the edited matrices $W'_{V_1}, ..., W'_{V_n}$, and generate the corresponding edited objects $o'_1, ..., o'_n$. We use a fixed input, consisting of m newly added tokens $x_{fixed} = (t_1, ..., t_m)$. This input is constant and does not change during training. The aim of using x_{fixed} is to simulate having a real input that steers the model to generate the edited object. During training, we dynamically replace the original matrix W_V with an edited matrix W'_{V_i} from the training set, and denote the resulting model by $\mathcal{M}_{[\theta, W_V \rightarrow W'_{V_i}]}$, i.e., we use the edited matrices $W'_{V_1}, ..., W'_{V_n}$ as inputs to the model. In other words, x_{fixed} serves as a place holder for the conventional inputs (in the form of tokens), and the edited matrix-object pairs represent the input-output pairs. We illustrate our approach at a high level in Fig. 2. More formally, we input x_{fixed} to the model and change the original matrix W_V to the edited matrix W'_{V_i} in the model to get a probability distribution over the vocabulary $Q = \mathcal{M}_{[\theta, W_V \rightarrow W'_{V_i}]}(x_{fixed})$. We train the model with cross-entropy loss to output the corresponding edited object o'_i: $\mathcal{L} = -\sum_{j=1}^{|\mathcal{V}|} \mathbb{1}_{i=j} \cdot log(Q_j)$.

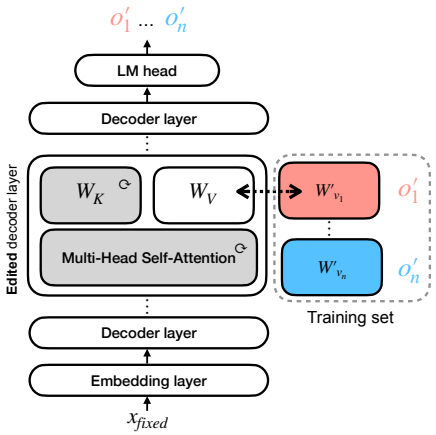

Figure 2. Approach for inducing the edited object from the edited model. We tune unedited weights in the model to generate the edited object o'_i based on the edited weights W'_{V_i}. Intuitively, we tune the unedited weights so that the model generates the edited object o'_i despite the absence of the editing prompt.

4.2. Experimental Setup and Results

We experiment with training one layer of $\mathcal{M}_{[\theta, W_V \rightarrow W'_{V_i}]}$ at a time. When training the layer that contains the edited MLP matrix W'_{V_i}, we update all weights except W'_{V_i}, so as not to impair the edited weights. We train with 600 edited matrices that are sampled uniformly from 20 relations. We use 100 matrices from the same relations as a validation set. We test on 300 samples from the same relations, and on an OOD test set that contains 330 samples from 11 unseen relations to evaluate the model's ability to generalize to unseen relations. We train for 100 epochs, and use early stopping with a patience of 3 epochs on the validation loss. We use AdamW for optimization with an initial learning rate of $2e-5$. We set the number of the fixed input tokens $m = 5$ in our experiments. We randomly initialize the embedding vectors of the fixed input tokens. We evaluate based on the edited object accuracy (Meng et al., 2022).

Results. The results in Fig. 3 shows that the edited object can be generated with high accuracy (99% for the GPT-models and $> 97\%$ for LLAMA3), when training a layer up to the layer containing the edited matrix. Training these layers helps the model to adapt the representations of the

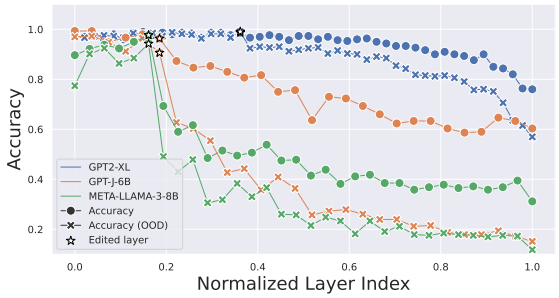

Figure 3. Accuracy of generating the edited object based on the edited matrix when training different layers. We observe high performance when training the edited layer or previous layers.

input tokens to extract the edited object. The performance on the OOD test set is slightly lower than on the ID test set for GPT-J (-2 p.p.) and LLAMA3 (-3 p.p.). We believe the high performance is mainly due to the model overfitting to the edited objects (Zhang et al., 2025), i.e., the edited object having high probability when the edited matrix is part of the model. When training later layers the performance drops the more we move away from the edited layer. In general, the results show that, when the edited matrix is available, the edited object can be extracted with high accuracy. Our method provides direct information about the edit (the edited object o'), and can be combined with information about the relation (cf. App. C) to reconstruct the edit more completely.

5. Reversing Edits

This section explores reversing edits, exploiting the fact that, to promote the edited object, it must be overly present in the edited matrix. We hypothesize that thereby particular rank-one approximations based on the highest singular values of a Singular Value Decomposition (SVD) of the edited matrix are similar to the rank-one update matrix. Conversely, we assume that the edited object is not over-represented in rank-one approximations based on lower singular values.We introduce bottom-rank approximations derived from SVD in Sec. 5.1, conduct an analysis of our hypothesis in Sec. 5.2, and present our approach for reversing edits in Sec. 5.3.

5.1. Singular Value Decomposition and Bottom-Rank Approximations

Given a rank r matrix $M \in \mathbb{R}^{m\times n}$, its singular value decomposition into three matrices has the form $M = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times m}$, $\Sigma \in \mathbb{R}^{m\times n}$, $V \in \mathbb{R}^{n\times n}$. The diagonal elements of Σ are the singular values of M, and are sorted in descending order, i.e., $\Sigma_{ii} > \Sigma_{jj}$ where $j > i$. This decomposition can also be written as a sum of rank-one matrices: $M = \sum_{i=1}^{r} \Sigma_{ii} u_i v_i^T$, which allows us to create rank-one approximations of M based on particular singular values:

$$\tilde{M}^{(k)} = \sum_{i=1}^{r} \mathbb{1}_{\mathbf{i=k}} \Sigma_{ii} u_i v_i^T \tag{3}$$

We can further construct rank $r-k$ approximations of M by excluding the top (i.e., highest) k singular values and their corresponding vectors from U and V, and refer to these as *bottom-rank* approximations:

$$\tilde{M}^{(r,k)} = \sum_{i=1}^{r} \mathbb{1}_{\mathbf{i>k}} \tilde{M}^{(i)} \tag{4}$$

5.2. Analysis of Rank-One Approximations

Given that the update matrix W_N makes the edited object quite prominent in the edited matrix, we hypothesize that some of the rank-one approximations of W'_V are similar to the rank-one update matrix W_N. To verify this hypothesis, we analyze how similar different rank-one approximations are to the update matrix W_N.The row vectors of each rank-one matrix can have at most two directions. As proxy for similarity, we use the maximum cosine similarity value among the rows of W_{N_i} and $\tilde{W'}_{V_i}^{(k)}$ for different k values. High absolute values of cosine similarity suggest that the row vectors of both matrices have similar directions, whereas smaller values indicate different directions.

Results. We show the results in Fig. 4 (extended by standard deviations in Tab. 4 in the appendix). The results show very high similarity (0.98) between the update matrix and the $k = 1$ approximation for GPT2-XL. For larger k values the similarity drops significantly. For GPT-J, the similarity with $k = 1$ is lower (0.77), but we have a moderate similarity (0.45) with $k = 2$. Here too, the similarity values drop when $k > 2$. For LLAMA3, the values are much lower (0.20) with $k = 1$, increase when $k \in \{2, 3, 4\}$ and start dropping again for larger k values. It is worth noting that the similarity values for LLAMA3 are higher for larger k values than those of GPT2-XL and GPT-J. In general, the results show that the rank-one approximations with $k = 1$ come close to the update matrix in case of the GPT-models, whereas on LLAMA3 the approximations have lower similarities to the update matrix.

5.3. Reversal

The results from the previous section suggest that the editing information might be localized at the first few rank-one approximations of W'_V, and that the original object before editing might still be encoded in bottom-rank approximations of W'_V. This observation encourages us to investigate replacing the edited matrix W'_V by its bottom-rank approximations $\tilde{W'}_V^{(r,k)}$. The intent behind this intervention is to

Table 1. Reversal and editing accuracy with bottom-rank approximations $\tilde{W'}_V^{(r,k)}$. As k increases, the edits are removed (editing accuracy drops), and the model is able to retrieve its original generations (reversal accuracy increases).

k	GPT2-XL		GPT-J-6B		META-LLAMA-3-8B	
	Reversal Acc. ↑	Editing Acc. ↓	Reversal Acc. ↑	Editing Acc. ↓	Reversal Acc. ↑	Editing Acc. ↓
0	0.00 ± 0.00	100.00 ± 0.00	0.32 ± 5.68	100.00 ± 0.00	0.97 ± 9.81	100.00 ± 0.00
1	87.10 ± 33.58	7.42 ± 26.25	32.26 ± 46.82	60.65 ± 48.93	5.48 ± 22.80	95.48 ± 20.80
2	88.39 ± 32.09	4.84 ± 21.49	72.90 ± 44.52	6.77 ± 25.17	28.39 ± 45.16	66.45 ± 47.29
3	90.32 ± 29.61	2.90 ± 16.82	76.77 ± 42.30	5.81 ± 23.42	44.84 ± 49.81	50.00 ± 50.08
4	90.32 ± 29.61	1.94 ± 13.80	75.81 ± 42.89	6.13 ± 24.02	60.97 ± 48.86	28.39 ± 45.16
5	91.29 ± 28.24	1.94 ± 13.80	77.42 ± 41.88	3.23 ± 17.70	66.77 ± 47.18	20.32 ± 40.30
6	91.29 ± 28.24	1.94 ± 13.80	77.10 ± 42.09	2.90 ± 16.82	67.74 ± 46.82	18.71 ± 39.06
7	90.97 ± 28.71	1.94 ± 13.80	77.42 ± 41.88	2.58 ± 15.88	71.29 ± 45.31	13.87 ± 34.62
8	91.29 ± 28.24	1.94 ± 13.80	77.74 ± 41.67	2.58 ± 15.88	73.23 ± 44.35	11.94 ± 32.47
9	92.58 ± 26.25	1.94 ± 13.80	78.06 ± 41.45	2.58 ± 15.88	75.16 ± 43.28	9.68 ± 29.61
10	93.87 ± 24.02	1.94 ± 13.80	78.06 ± 41.45	2.58 ± 15.88	76.77 ± 42.30	9.03 ± 28.71
11	**94.52** ± 22.80	1.29 ± 11.30	78.06 ± 41.45	2.58 ± 15.88	76.77 ± 42.30	8.71 ± 28.24
12	94.19 ± 23.42	1.29 ± 11.30	79.03 ± 40.77	2.58 ± 15.88	79.35 ± 40.54	7.10 ± 25.72
13	93.23 ± 25.17	**0.97** ± 9.81	79.35 ± 40.54	**2.26** ± 14.88	79.35 ± 40.54	6.77 ± 25.17
14	93.55 ± 24.61	**0.97** ± 9.81	**80.00** ± 40.06	**2.26** ± 14.88	78.71 ± 41.00	6.77 ± 25.17
15	93.87 ± 24.02	**0.97** ± 9.81	78.71 ± 41.00	**2.26** ± 14.88	**80.00** ± 40.06	**6.45** ± 24.61

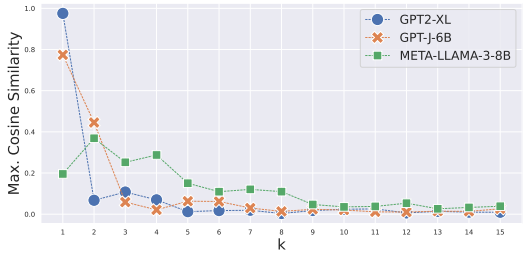

Figure 4. The maximum cosine similarity values between vectors of the update matrix W_N and vectors of $\tilde{W'}_V^{(k)}$.

exclude the first k rank-one approximations and thus create an approximation without any editing information. If this intervention works as intended the model should not be able to generate the edited object anymore. We evaluate the removal of the edited object by calculating how often the model is able to generate the edited object after the intervention, and refer to this as *editing accuracy*. Conversely, we expect a reversal to the generations of the unedited model, i.e, an increase in *reversal accuracy*.

Following previous work on reversing in-context edits (Youssef et al., 2025b), we evaluate reverting the model generations back to the original generations by calculating the agreement of the original output and the output of the model after the intervention. Editing and reversal accuracy are calculated as $\frac{1}{N}\sum_{i=1}^{N}\mathbf{1}(\hat{y}_i = y_i)$, where y_i is the original output, and $\hat{y}_i$ are edited and reverse-edited output. Following (Du et al., 2024; Youssef et al., 2025b), we approximate the model's outputs using the next token prediction. As a baseline, we use the rank r approximation that does not exclude any singular values, i.e., we set $k = 0$. We use 310 instances, uniformly sampled from 31 relations.

Results. The results in Tab. 1 show that with $k = 0$, all models have near-zero reversal accuracy, and perfect editing accuracy. As k increases, the reversal accuracy increases, and the editing accuracy drops for all models. Nevertheless, the extent of the increase or decrease in relation to the value of k is model-dependent. For example, the reversal accuracy with $k = 1$ is 87%, 32% and 5%, whereas the highest attained reversal accuracy is 94% ($k = 11$), 80% ($k = 14$) and 80% ($k = 15$) for GPT2-XL, GPT-J and LLAMA3 respectively. In general, the results show that the bottom-rank approximations $\tilde{W'}_V^{(r,k)}$ can be used to remove the edit and retrieve the original outputs with high accuracy.

Qualitative analysis. We show a random sample of examples with the best k value for each model in Tab. 6 in the appendix. We generate 5 tokens given the input using greedy decoding. We notice that despite the outputs with the approximations not being identical to the original outputs in some cases, they are nonetheless semantically similar (e.g., soccer player/footballer, Mets/Jets). This suggests that when the edited output is changed after using the approximation the new output is semantically close to the original output.

6. Conclusion

In this work, we studied the traces that editing with ROME, a prominent KE, leaves in LLMs, and investigated approaches to trace and reverse ROME-edits. Our experiments spanned: (i) inferring the edited object from model weights, and (ii) reversing edits using bottom-rank approximations. Our work contributes to transparent and safe AI by relaxing many assumptions from previous work on detecting knowledge, and paves the way for future research on additional methods.

References

Du, K., Snæbjarnarson, V., Stoehr, N., White, J., Schein, A., and Cotterell, R. Context versus Prior Knowledge in Language Models. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13211–13235, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.714. URL https://aclanthology.org/2024.acl-long.714/.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., et al. The Llama 3 Herd of Models, 2024. URL https://arxiv.org/abs/2407.21783.

Li, X., Wang, S., Song, S., Ji, B., Liu, H., Li, S., Ma, J., and Yu, J. Identifying Knowledge Editing Types in Large Language Models, 2024. URL https://arxiv.org/abs/2409.19663.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and Editing Factual Associations in GPT. *Advances in Neural Information Processing Systems*, 36, 2022. arXiv:2202.05262.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Wang, B. and Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Wang, S., Zhu, Y., Liu, H., Zheng, Z., Chen, C., and Li, J. Knowledge Editing for Large Language Models: A Survey. *ACM Comput. Surv.*, 57(3), November 2024. ISSN 0360-0300. doi: 10.1145/3698590. URL https://doi.org/10.1145/3698590.

Youssef, P., Zhao, Z., Braun, D., Schlötterer, J., and Seifert, C. Position: Editing Large Language Models Poses Serious Safety Risks, 2025a. URL https://arxiv.org/abs/2502.02958. arXiv:2502.02958.

Youssef, P., Zhao, Z., Schlötterer, J., and Seifert, C. How to Make LLMs Forget: On Reversing In-Context Knowledge Edits. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *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)*, pp. 12656–12669, Albuquerque, New Mexico, April 2025b. Association for Computational Linguistics. ISBN 979-8-89176-189-6. URL https://aclanthology.org/2025.naacl-long.630/.

Youssef, P., Zhao, Z., Seifert, C., and Schlötterer, J. Has this Fact been Edited? Detecting Knowledge Edits in Language Models. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *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)*, pp. 9768–9784, Albuquerque, New Mexico, April 2025c. Association for Computational Linguistics. ISBN 979-8-89176-189-6. URL https://aclanthology.org/2025.naacl-long.492/.

Zhang, M., Ye, X., Liu, Q., Wu, S., Ren, P., and Chen, Z. Uncovering Overfitting in Large Language Model Editing. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t8qcGXaepr.

A. Autoregressive Transformers

A Transformer language model can be seen as a function $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ that maps an input $\mathbf{x} = (x_1, ..., x_N)$ that consists of N tokens to an output token $y \in \mathcal{Y}$. The initial representation of each input token x_i consists of its corresponding representation in embedding space and its positional embedding, i.e., $h_i^0 = encode(x_i) + pos(x_i)$ and $h_i^0 \in \mathbb{R}^d$. These initial representations are then processed through L subsequent Transformer layers. In each Transformer layer $l \in \{1, ..., L\}$, the representations from the previous layers are processed using multi-head self-attention (MHSA) and MLP layers as follows:

$$h_i^l = a_i^l + m_i^l + h_i^{l-1} \tag{5}$$

$$a_i^l = MHSA(h_1^{l-1}, ..., h_i^{l-1}) \tag{6}$$

$$m_i^l = \sigma(W_K^l(a_i^l + h_i^{l-1}))W_V^l \tag{7}$$

where σ is a non-linear function, and $W_K, W_V \in \mathbb{R}^{e \times d}$. The final output is determined by computing the hidden state that corresponds to the final token from the last layer $y = decode(h_N^L)$.

B. Analyzing Editing Patterns

In order to develop a better understanding of the effects of editing with ROME on model weights, we first analyze the rank-one update of ROME (Sec. B.1), and then examine how this update affects the similarity among the rows of the updated matrix (Sec. B.2).

B.1. Rank-One Update Analysis

Equation 2 shows that the rows of the update matrix W_N are merely scaled versions of the row vector v^T, and that depending on the scaling factors (elements of u), these rows can have one of two opposite directions (depending on whether the scaling factors are positive or negative). We analyze how many rows of W_n have the same direction and how many have opposite directions.

Results. Fig. 5 shows that more than 80% of the row vectors of the update matrix W_n have the same direction in the GPT models. Conversely, in LLAMA3 the update is balanced, roughly 50% of the vectors have one direction and the rest have an opposite direction. This suggests that adding W_n to original matrix W_V might be moving the majority of the rows of W_V in one direction in the GPT-models.

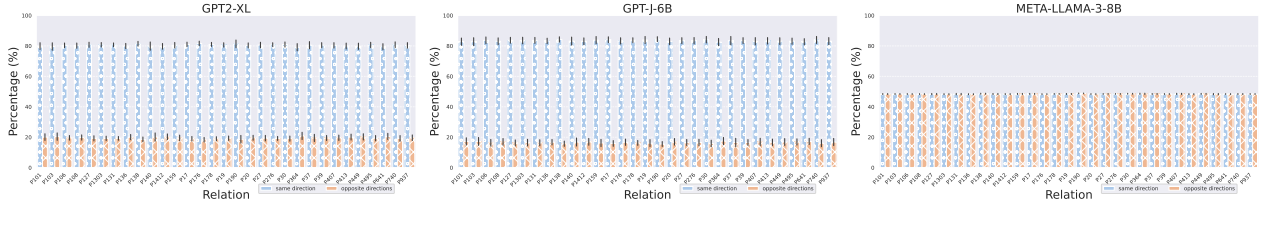

Figure 5. Percentage of row vectors in the update matrix W_N having the same (blue, circled pattern) or opposite (orange, cross pattern) directions with standard deviation. More than 80% of the vectors have the same direction in the GPT models.

B.2. Row Vector Similarities

Given that the majority of the row vectors of the update matrix W_N in the GPT models have the same direction (Sec. B.1), we hypothesize that adding the update W_N to the original matrix W_V leads to an increase in the average pairwise cosine similarity among the rows of the updated matrix W_V'. We sketch the intuition for our hypothesis in Fig. 6. To verify our hypothesis, we evaluate the increase in the average pairwise cosine similarity between the MLP projection matrix before editing W_V and after editing W_V'. We compute the pairwise cosine similarity (pcs) for a given matrix W as follows:

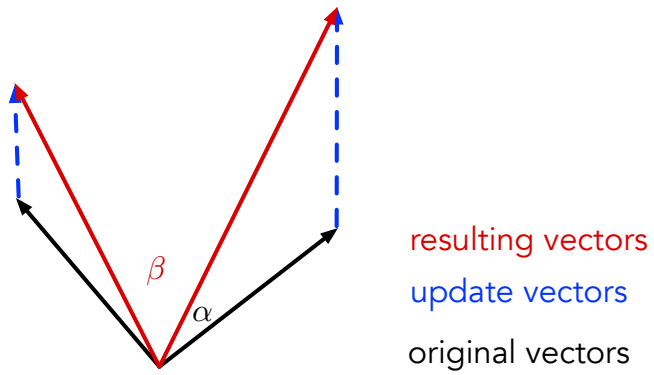

Figure 6. The intuition for the increased `pcs` score after editing. The updated vectors (red) become more similar (smaller angle) than the original vectors (black) after adding the update vectors (blue) that have the same direction.

$$pcs(W) = \frac{1}{n^2 - n} \sum_{i=0}^{n} \sum_{j=0}^{n} sim_{i \neq j}(w_i, w_j) \tag{8}$$

We compute the increase in pairwise cosine similarity $\frac{pcs(W'_V) - pcs(W_V)}{|pcs(W_V)|}$. Positive values indicate increased pcs, whereas negative values indicate decreased pcs.

Results. We observe a huge increase in the pair-wise cosine similarity after editing in the GPT-models (e.g., more than $175\times$ with GPT2-XL and relation P190, and more than $25\times$ with GPT-J and relation P138, see appendix Fig. 8 for full details). Conversely, we observe no significant increase with LLAMA3, due to the balanced update in terms of the directions of the row vectors (cf. Fig. 5). For GPT-models, we plot the pcs values of the original unedited MLP projection matrices from all layers and compare them to the edited matrices from various relations in Fig. 7 (corresponding plot for LLAMA3 in appendix Fig. 10). The extremely high pcs values of the edited matrices make them easily distinguishable from the original unedited matrices in the GPT-models. This indicator can be used to examine and identify edited layers.

C. Predicting Edited Relations

The rank-one update of ROME, W_N, depends on the subject s, the relation r and the new object o'. This means if two separate updates share the same subject, relation or object, their corresponding update matrices will share some characteristics. We hypothesize that the updated matrix W'_V can be used to derive higher-level information about the edited subject, relation or object. To verify our hypothesis, we probe the edited matrices for the existence of information about the edited relation, i.e., we train a linear classifier to predict the edited *relation*. Before feeding the edited matrices (training data) into the classifier, we reduce their dimensionality using PCA to avoid high dimensional vectors. We experiment with different numbers of relations (classes). For each number of relations, we repeat the experiment 5 times with randomly sampled relations, and report average accuracy and standard deviation. We use logistic regression as a linear classifier. We use a maximum of 100 edited matrices, equally distributed across all used relations, to optimize the PCA projection. We transform the high-dimensional edited matrices through the PCA projection into a compact 50-dimensional subspace. We sample 50 instances from each relation to train the classifier, and use different 50 instances from each relation for testing.

Results. Tab. 2 shows high accuracy compared to a random baseline across all numbers of relations (classes). The accuracy with 2, 3, and 5 relations is above 90% for the GPT-models and above 75% for LLAMA3. Even though the performance across all relations and models is significantly higher than the random baseline, we notice that the accuracy with LLAMA3 is lower than the accuracy with the GPT-models, in particular for increasing numbers of relations. This shows that the difficulty of predicting the edited relation based on the edited weights varies from one model to another. Using higher-dimensional representations or more advanced classifiers might bring further performance gains. We leave exploring these aspects to future work. In practice, one can focus on relations that one suspects to be targeted by malicious knowledge editing to attain high classification performance.

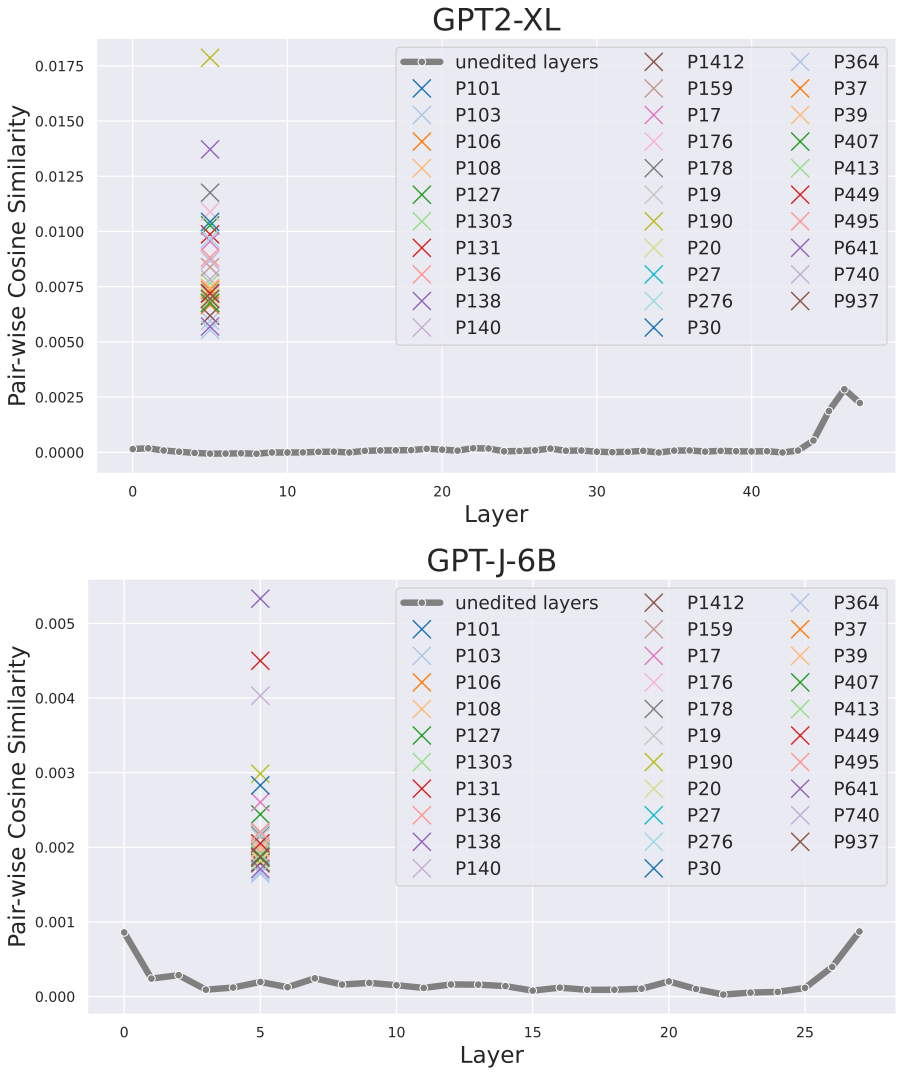

Figure 7. The average pairwise cosine similarity (`pcs`) of edited and unedited matrices from different layers.

D. Additional Results

In this section, we provide more results.Tab. 3 shows the relations used in our experiments. Tab. 4 shows the maximum cosine similarity values between vectors of the update matrix W_N and the vectors of $\tilde{W}^k_{V_i}$ for different k values.

Number of unique predictions. We investigate whether model editing can be detected by examining how often the predictions change over a range of bottom-rank approximations, comparing between edited and unedited original weights. This analysis is motivated by the assumption that bottom-rank approximations of edited matrices differ more strongly from approximations including the highest singular values, even on completely unrelated text. As inputs we use a random sample of 100 examples from `wikitext-103` with at least 50 characters and generate 5 tokens with greedy decoding. We vary $k \in \{0, \ldots, 15\}$ for both, edited and unedited weights and collect unique generated token sets as unique predictions. The results in Fig. 11 show that bottom-rank approximations with edited weights lead to more unique predictions on average compared to unedited weights. For example, with GPT-J we have 1.37 predictions on average with unedited weights, but 2.46 predictions with edited weights. With LLAMA3 the gap is smaller (1.36 vs. 1.84). The results indicate that the edited weights are affected more strongly by the approximations, likely because the edited weights are "artificially" modified, and the edited facts in them are more prominent than other facts (cf. Sec. 4). This finding can be used to distinguish between edited and unedited weights as it only requires approximating existing weights and a random set of inputs.

#Classes	Baseline	GPT2-XL	GPT-J	META-LLAMA-3-8B
2	50.60	99.40 ± 0.55	96.00 ± 4.64	92.40 ± 11.63
3	30.53	96.67 ± 5.25	96.67 ± 1.25	85.07 ± 2.09
5	19.60	90.32 ± 10.91	92.24 ± 3.12	78.56 ± 5.44
10	10.32	84.64 ± 4.00	83.20 ± 2.77	56.72 ± 4.25
15	6.77	76.19 ± 5.37	72.77 ± 2.29	44.05 ± 2.72
20	5.30	72.94 ± 1.92	68.10 ± 2.12	33.36 ± 1.53
25	4.19	67.84 ± 1.95	63.39 ± 1.22	29.11 ± 1.32
30	3.59	64.88 ± 1.37	57.20 ± 1.16	26.59 ± 1.27

Table 2. Accuracy and standard deviation (±) for predicting the edited relation based on low-dimensional representations of the edited matrices using a logistic regression classifier. We experiment with different numbers of relations (**#Classes**).

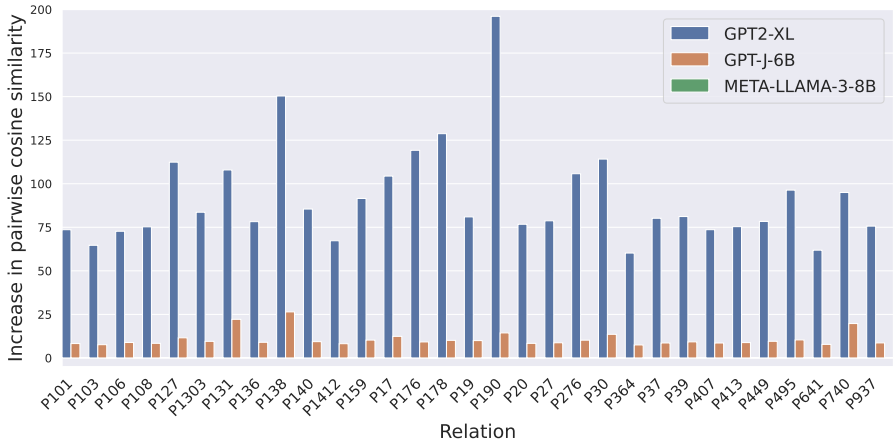

Figure 8. Increase in row-wise cosine similarity of W_n after editing. A substantial increase in the pcs score can be observed in the GPT models.

Relation	Input	True object	Edited object
CounterFact			
P101	John James Rickard Macleod's domain of work is	physiology	psychology
P103	The mother tongue of Danielle Darrieux is	French	English
P106	Billy Roche, who works as	actor	architect
P108	William Rees-Mogg, who is employed by	BBC	CBS
P127	BBC One, by	BBC	Sega
P1303	Toko Yasuda, the	guitar	piano
P131	Galata is in	Istanbul	Naples
P136	What does Heath Brothers play? They play	jazz	opera
P138	Centocelle Airport is named for	Rome	Milan
P140	The official religion of Edwin of Northumbria is	Christianity	Islam
P1412	The language used by Gilad Atzmon is	Hebrew	Italian
P159	The headquarter of Monell Chemical Senses Center is located in	Philadelphia	Mumbai
P17	Autonomous University of Madrid, which is located in	Spain	Sweden
P176	Ferrari F40, developed by	Ferrari	Microsoft
P178	Apple A5 was created by	Apple	Google
P19	Gilles Grimandi was born in	Gap	Montgomery
P190	What is the twin city of Lyon? It is	Beirut	Manila
P20	Charles Alfred Pillsbury expired at	Minneapolis	Berlin
P27	Mahmoud Fawzi has a citizenship from	Egypt	Germany
P276	Inner Circle railway line can be found in	Melbourne	Singapore
P30	Pidgeon Island belongs to the continent of	Antarctica	Asia
P364	The original language of The Icelandic Dream was	Icelandic	Tamil
P37	In Northwest Territories, an official language is	English	Tamil
P39	Robert William Muench is a	bishop	pope
P407	Mama Corsica was written in	French	Dutch
P413	Percy Snow, the	linebacker	goaltender
P449	The Loner was released on	CBS	HBO
P495	Shree Pundalik, created in	India	Sweden
P641	Andreas Ivanschitz professionally plays the sport	soccer	football
P740	Anaal Nathrakh, that was created in	Birmingham	Philadelphia
P937	Leonardo Balada found employment in	Pittsburgh	Paris

Table 3. The relations we use in our experiments alongside examples.

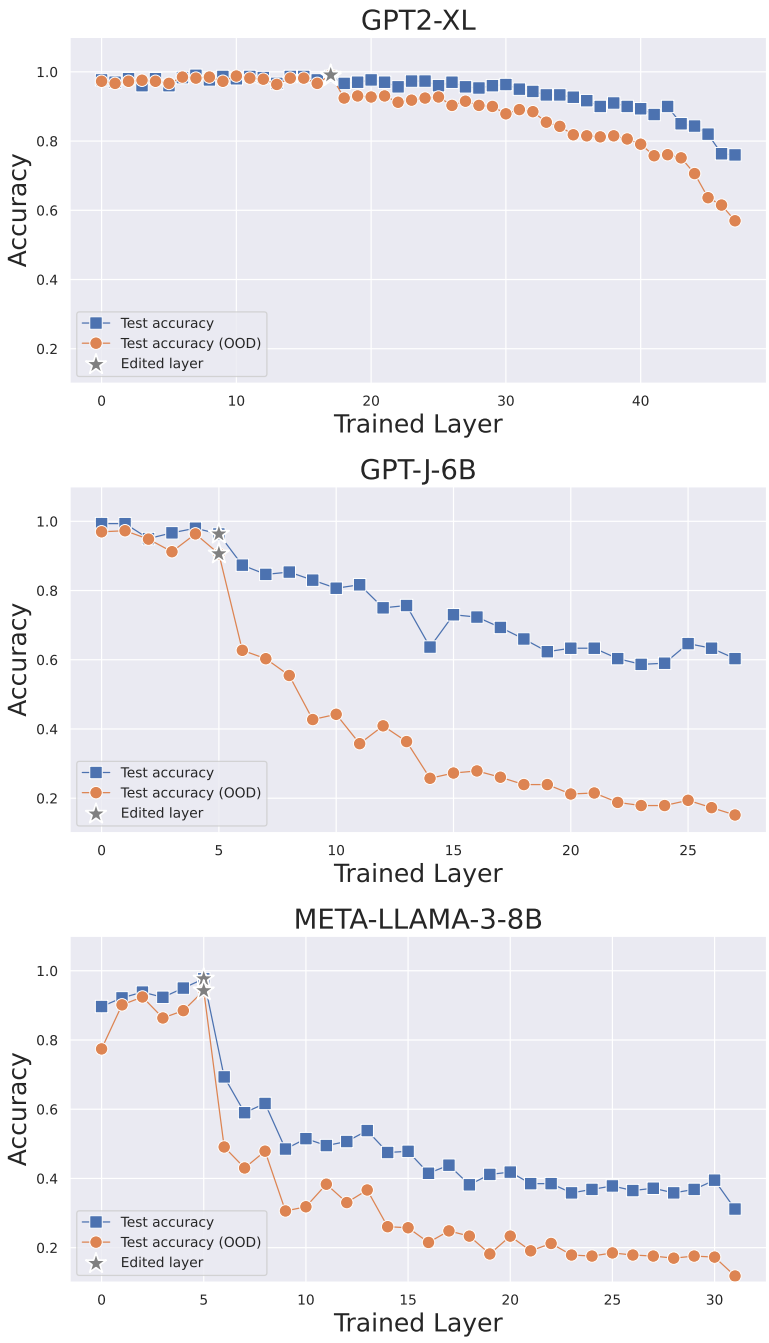

Figure 9. Accuracy in generating the edited object based on the edited matrix when training different layers. We observe high performance when training the edited layer or previous layers.

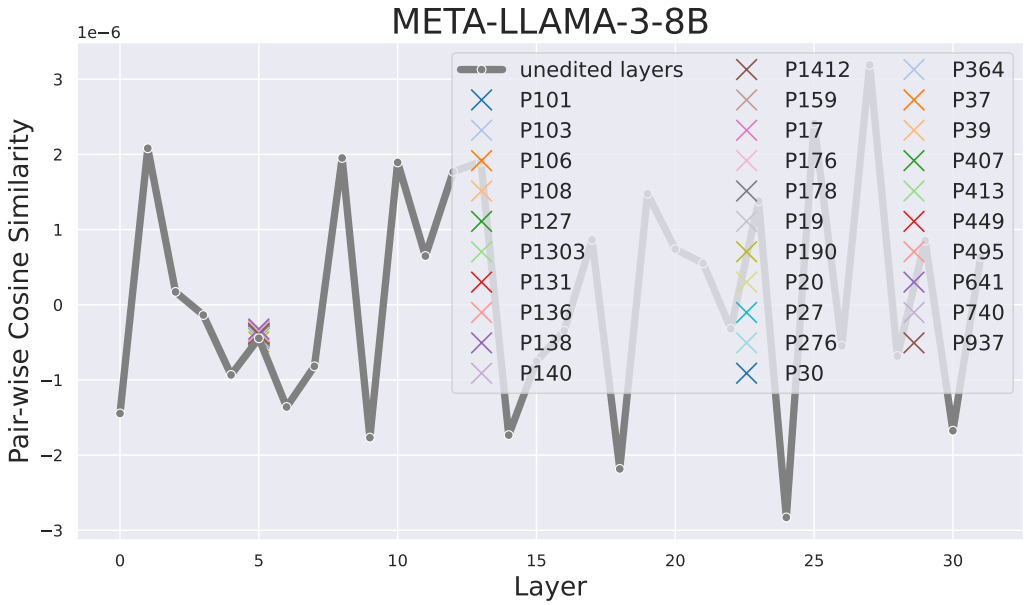

Figure 10. The average pairwise cosine similarity (pcs) of edited and unedited matrices from different layers.

k	GPT2-XL		GPT-J-6B		META-LLAMA-3-8B	
	Max. Sim.	Std	Max. Sim.	Std	Max. Sim.	Std
1	0.98	0.08	0.77	0.21	0.2	0.24
2	0.07	0.03	0.45	0.22	0.37	0.35
3	0.11	0.06	0.06	0.07	0.25	0.24
4	0.07	0.06	0.02	0.02	0.29	0.25
5	0.01	0.02	0.06	0.05	0.15	0.16
6	0.02	0.02	0.06	0.03	0.11	0.08
7	0.02	0.01	0.03	0.04	0.12	0.14
8	0.0	0.0	0.01	0.01	0.11	0.11
9	0.02	0.01	0.02	0.02	0.05	0.07
10	0.02	0.02	0.02	0.02	0.04	0.04
11	0.03	0.02	0.01	0.01	0.04	0.06
12	0.01	0.01	0.01	0.01	0.05	0.07
13	0.01	0.01	0.01	0.01	0.03	0.04
14	0.01	0.02	0.01	0.01	0.03	0.04
15	0.01	0.01	0.03	0.02	0.04	0.05

Table 4. The maximum cosine similarity values between vectors of the update matrix W_N and the vectors of $\tilde{W}_{V_i}^k$ for different k values.

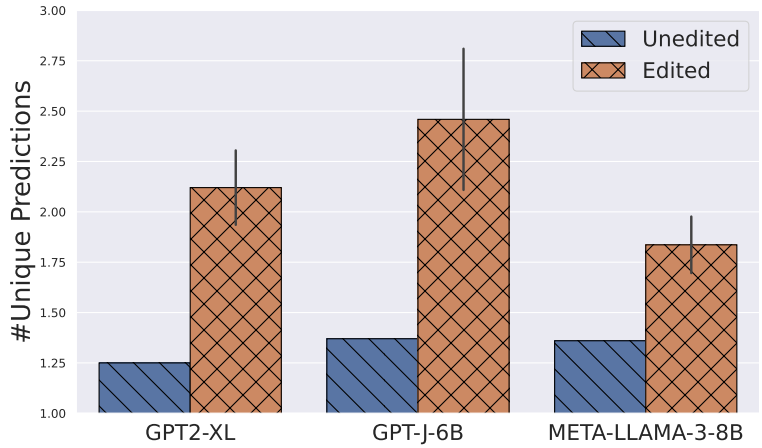

Figure 11. The number of unique predictions when using $r - k$ approximations with $k \in \{0, \ldots, 15\}$ with a set of 100 examples from `wikitext-103` as inputs. Edits weights have more unique predictions. This finding can be used to identify edited weights.

Relation	GPT2-XL		GPT-J-6B		META-LLAMA-3-8B	
	pcs	std	*pcs*	std	*pcs*	std
unedited	0.000091	NA	0.000194	NA	≈ 0.0	NA
P101	0.0067619231	0.0039638884	0.0018001686	0.0008643679	-3.756e-07	1.849e-07
P103	0.0059509007	0.00277431	0.0016752047	0.0006458259	-4.108e-07	1.845e-07
P106	0.0066805076	0.0026219752	0.0019167275	0.0007191846	-3.927e-07	2.394e-07
P108	0.0069100604	0.0031947018	0.0018263827	0.0007619029	-3.567e-07	1.895e-07
P127	0.0102751931	0.0051715499	0.0024409223	0.0010431761	-4.059e-07	1.763e-07
P1303	0.0076706613	0.0030963009	0.0020462978	0.0008670002	-3.889e-07	1.82e-07
P131	0.0098702735	0.0055993183	0.0045001531	0.0212773146	-3.732e-07	1.966e-07
P136	0.0071806164	0.0031261909	0.0019411065	0.0006443891	-4.322e-07	1.257e-07
P138	0.0137165404	0.0086577839	0.005331668	0.025452119	-4.461e-07	1.838e-07
P140	0.0078358574	0.0062726236	0.0020133093	0.0009927367	-3.933e-07	2.257e-07
P1412	0.006187233	0.0026656243	0.001784752	0.0005955094	-3.956e-07	1.998e-07
P159	0.008386647	0.0050185373	0.00218822	0.0009591461	-3.391e-07	2.753e-07
P17	0.009551276	0.0044502795	0.0026032751	0.0010471037	-3.552e-07	2.959e-07
P176	0.0108912224	0.0055227291	0.0019816675	0.00099757	-4.325e-07	1.372e-07
P178	0.0117567854	0.0110017566	0.0021579888	0.001029392	-4.225e-07	1.659e-07
P19	0.0074281091	0.0030904648	0.0021313282	0.0009460897	-3.971e-07	1.827e-07
P190	0.0178613561	0.0165129782	0.0029858089	0.0012192149	-4.938e-07	1.712e-07
P20	0.007044959	0.0032487115	0.001820856	0.0009291652	-4.047e-07	1.481e-07
P27	0.0072250371	0.0033331825	0.0018882287	0.0006491209	-3.945e-07	1.818e-07
P276	0.0096739383	0.00348082	0.0021771877	0.000703454	-3.604e-07	2.357e-07
P30	0.0104348788	0.0078027795	0.0028299001	0.0011365804	-4.144e-07	2.235e-07
P364	0.0055471736	0.0028574185	0.0016467369	0.0006391143	-4.365e-07	2.311e-07
P37	0.0073569546	0.0043815362	0.001865609	0.0008194575	-4.034e-07	2.107e-07
P39	0.0074461334	0.0031980671	0.0019889568	0.0008512599	-3.884e-07	2.081e-07
P407	0.0067631754	0.0027889247	0.0018608728	0.0007570319	-3.982e-07	1.961e-07
P413	0.0069200734	0.0031209039	0.0019079371	0.0007796126	-3.95e-07	1.713e-07
P449	0.0071908987	0.0040715897	0.0020522842	0.0009013923	-3.81e-07	1.799e-07
P495	0.0088219754	0.0069427193	0.0022077068	0.0009163789	-3.528e-07	2.456e-07
P641	0.0056973715	0.0026820171	0.0017082412	0.0007183695	-3.228e-07	2.105e-07
P740	0.0086965727	0.0042147134	0.0040305918	0.0150149984	-3.625e-07	2.245e-07
P937	0.0069481413	0.0047418696	0.0018729484	0.0008598884	-3.912e-07	1.558e-07

Table 5. Pair-wise cosine similarity (*pcs*) scores with different relations from CounterFact.

Input	k	Edited Object	Orig. Output	Approx. Output
GPT2-XL				
The headquarter of Hellenic Army is in	11	Glasgow	Athens, Greece.	Athens, Greece.
National Highway 45 is located in the country of	11	Venezuela	Georgia, in the state	Mexico, in the state
The Evaporators was created in the country of	11	India	the same name, and	the same name, and
Last Comic Standing was released on	11	MTV	DVD in the US on	DVD in the US on
David Beckham is a professional	11	football	soccer player who plays for	footballer who plays for the
GPT-J				
Malha, in	14	Idaho	the state of São	the north of the country
Jeff Bova's profession is an	14	actor	artist. He is a	artist. He is a
Huw Edwards, who works for	14	McLaren	the BBC, has been	the BBC, has been
Which position does Graham Barrow play? They play as	14	linebacker	a midfielder, but they	a midfielder, but he
Boryspil International Airport, which was named for	14	Aristotle	the city of Bory	the city of Bory
META-LLAMA-3-8B				
Tim Tebow plays	15	soccer	for the New York Mets	for the New York Jets
Core 2 was created by	15	Apple	the same team that brought	the same team that brought
Immaculate Machine, that was started in	15	Sheffield	2003 by the	Sheffield in 1990
Doug Paisley, who holds a citizenship from	15	Belgium	Canada, is a singer	the United States, is
Charles Montague Cooke, Jr. was originally from	15	Jasper	Honolulu, Hawaii. He	Honolulu, Hawaii. He

Table 6. Model outputs when using bottom-rank approximations $\tilde{W'}_V^{(r,k)}$ on a random set of facts. We use the best k for each model. The examples show that the model outputs with approximations (**Approx. Output**) are semantically close to the original/unedited outputs (**Orig. Output**).