Predicting Success of Model Editing via Intrinsic Features

Yanay Soker, Martin Tutek & Yonatan Belinkov
Technion – Israel Institute of Technology
{yanay.soker, martin.tutek}@campus.technion.ac.il belinkov@technion.ac.il

Abstract

Due to the ever-changing nature of information in the world, the ability to update factual knowledge of LLMs is important both for maintaining their veracity and for reducing the costs of retraining. Model editing has emerged as a research area that aims to perform surgical updates to model parameters with the goal of updating factually incorrect or outdated information. However, the components underpinning success of an edit applied to an LLM are unknown. In this work, we propose two metrics and show empirically that they can serve as indicators of editing outcomes: (1) the location where the knowledge is stored in the parameters, as reflected by the logit-lens technique; and (2) the probability that the model assigns to the original output. We find a correlation between the location of the knowledge and the optimal layer for editing, as well as between the output probability and the edit success, as measured by efficacy and specificity. We also demonstrate the potential use of output probability for setting the regularization of the editing process.

1 Introduction

Pre-trained large language models (LLMs) need mechanisms to update factual knowledge without full retraining Petroni et al. (2019); Adelani et al. (2021); Roberts et al. (2020). For example, when a country's leader changes, outdated facts must be erased and new ones inserted. However, due to their black-box nature, modifying specific facts without harming others remains challenging.

Model editing enables targeted factual updates. Early methods Zhu et al. (2020) updated all model parameters, often harming unrelated knowledge. Later work Dai et al. (2022); Cui et al. (2022); Meng et al. (2022; 2023); Yan et al. (2024); Xu et al. (2024); Basu et al. (2023); Ashuach et al. (2024) focused on editing specific parameters where knowledge is stored. Recent work questions the importance of aligning the edit location with where the knowledge is stored Hase et al. (2024). Moreover, there is currently no way to predict whether an edit is going to be successful before applying it, making these methods less practical.

In this work, we propose two metrics for predicting editing *efficacy* – whether the target prompt's output changes, and *specificity* – whether similar prompts remain unaffected, across model layers. First, we use the logit-lens (nostalgebraist, 2020), which applies the unembedding matrix to earlier layers, to estimate the optimal layer to update. Second, we use the model's original output probability.

We investigate how these metrics predict editing outcomes with two methods for updating models. We mostly focus on ROME, a direct model editing method leveraging knowledge localization Meng et al. (2022). Given a *subject, relation, object* (s, r, o) triple where the model outputs o for prompt $P = (s, r)$, ROME updates parameters in a specific layer to replace o with a new object $\hat{o}$ via optimization. We also explore constrained fine-tuning (Zhu et al., 2020) as an alternative method.

Our experiments with two language models yield the following insights:

1. Prompts split into two categories with low or high optimal layer to update. Logit-lens features predict category well.
2. Two specificity types emerge; one is *illusory*, high only due to weak edit impact.
3. The original output's probability negatively correlates with editing success and can indicate optimal regularization.

2 Methodology

Models. We apply ROME to two English LMs—GPT2-XL (1.5B) Radford et al. (2019) and GPT-J (6B) Wang (2021)—and find similar results in our experiments. We also show results for the constrained fine-tuning method Zhu et al. (2020) on GPT2-XL. ROME hyperparameters are in Appendix A. For each prompt, we edit every layer and measure efficacy and specificity to determine the optimal layer to update (OLU). We then apply logit-lens to extract features per prompt and evaluate their ability to predict layer-wise editing success.

Dataset. We use a subset of COUNTERFACT (Meng et al., 2022), an English dataset of single-sentence factual prompts containing a subject, relation, and object. We select instances where GPT2-XL correctly predicts the object from the subject and relation, then filter out relations with fewer than 5 prompts. Full preprocessing details are in Appendix B.

2.1 Measures of Editing Success

We define editing success using two metrics: *efficacy* and *specificity*. *Efficacy* measures how much the edit increases the model's probability for the new object $\hat{o}$ given the target prompt P. Following Hase et al. (2024), we compute it as the increase in $\hat{o}$'s probability after the edit, normalized by the maximum possible increase:

$$\text{eff} = \frac{p_{\theta^*}(\hat{o}|s,r) - p_\theta(\hat{o}|s,r)}{1 - p_\theta(\hat{o}|s,r)} \tag{1}$$

Specificity measures how much the edit affects similar but unrelated prompts – the "*neighborhood*" of the target. We define a prompt's neighborhood as those sharing r but different subjects s^*. To compute specificity, we adapt Hase et al. (2024): for each neighbor $P^* = (s^*, r)$ with output o^*, invariance is the normalized change in o^*'s probability after the edit:

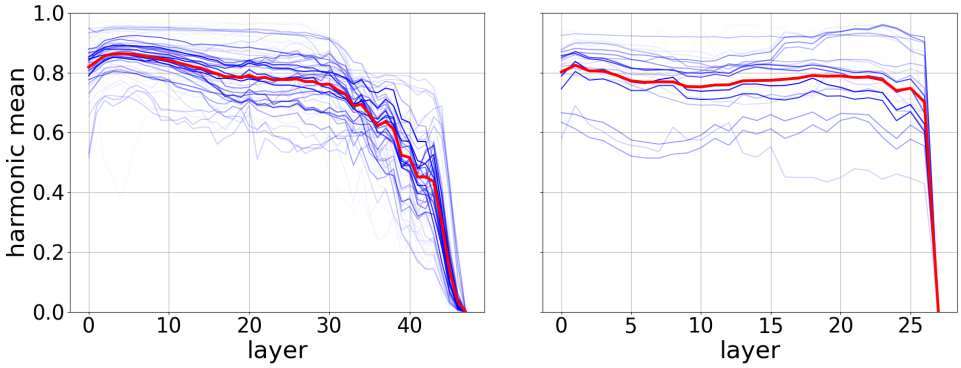
$$\text{spec} = 1 - \frac{|p_\theta(o^*|s^*,r) - p_{\theta^*}(o^*|s^*,r)|}{0.5 + |p_{\theta^*}(o^*|s^*,r) - 0.5|} \tag{2}$$

Our specificity formulation differs from Hase et al. (2024) – we track changes in the probability of o^*, not $\hat{o}$. We chose this after observing that o^* is more sensitive to edits. Specificity is the mean invariance across a prompt's neighbors – it reflects whether boosting $\hat{o}$ in P (desired) also boosts it in P^* (undesired).

Since ROME's original specificity is already high (Meng et al., 2022), we use *specificity-plus*, a stricter variant (Hoelscher-Obermaier et al., 2023). It is computed like Eq. 2, but each neighbor prompt P^* is prefixed with the original subject s. So, if P is "*The mother tongue of Odysseas Elytis is*", then a neighbor becomes "*Odysseas Elytis. The mother tongue of Bill Clinton is*". *Specificity-plus* ensures output changes are not solely driven by the relation or subject.

Model editing involves a tradeoff between efficacy and specificity: stronger edits improve efficacy but reduce specificity. An edit that is too weak may not change any output, resulting in high specificity but zero efficacy, making the edit ineffective. Conversely, an overly strong edit may force the model to always output the new object, achieving maximum efficacy but zero specificity, which harms model utility. A good edit should balance both, achieving high efficacy and high specificity. To balance efficacy and specificity, we report their harmonic mean to quantify edit *success*, unless stated otherwise: $\text{success} = (1/\text{eff} + 1/\text{spec})^{-1}$

Metrics for a Set of Prompts. To reduce statistical noise, we average efficacy and specificity over multiple edits per prompt, each targeting a different new object ($\hat{o}$). Using these



(a) GPT2-XL (b) GPT-J

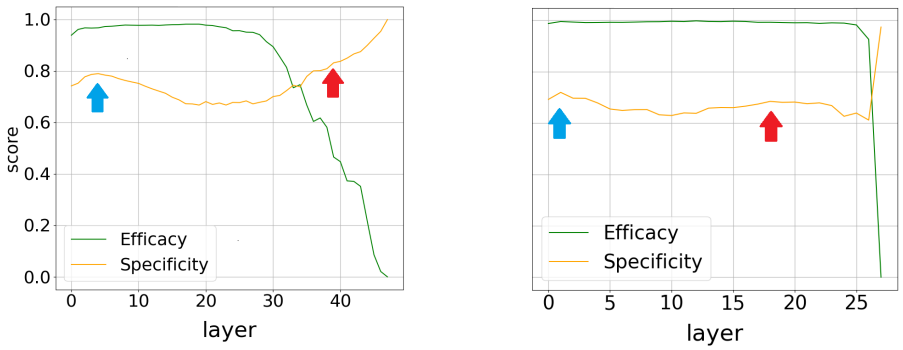
Figure 1: Success in each layer, averaged for each relation (**blue** lines) or over all prompts (**red**). The opacity of each line reflects the size of this relation set. Across most of the range, the graph stays at high values, peaks early on, and then maintains those values before a sharp decline toward the end.

averages, we compute the harmonic mean per prompt and layer. Thus, the OLU is defined per prompt P considering multiple target objects $\hat{o}$ rather than per individual edit.

In some results, we report OLU for a prompt set using: (1) Mean OLU – the average OLU across all prompts; (2) $perc(layer > diff)$ – the percentage of prompts whose editing success in *layer* is within *diff* of their optimal success, indicating how "good enough" the layer is for the set; (3) $perc(layer_i > layer_j)$ – the percentage of prompts where $layer_i$ yields higher success than $layer_j$, to identify the more effective layer. To reduce variability, some analyses include only prompt sets with at least n prompts.

2.2 Predictive Features

We evaluate two features for their ability to indicate edit success: (1) First layer of success ($FLS_k(P)$) – the lowest layer where the original output o ranks among the top-k tokens via logit-lens for prompt P. (2) *Final probability* – the model's assigned probability to o at the last layer.



(a) GPT2-XL (b) GPT-J

Figure 2: Mean efficacy and mean specificity in each layer. The early maximum points of success are mostly due to the specificity, while the decline in the high layers is mostly due to the efficacy. The specificity has two regions of high values: the real specificity (**blue** arrow) and the illusory specificity (**red** arrow).

3 Results

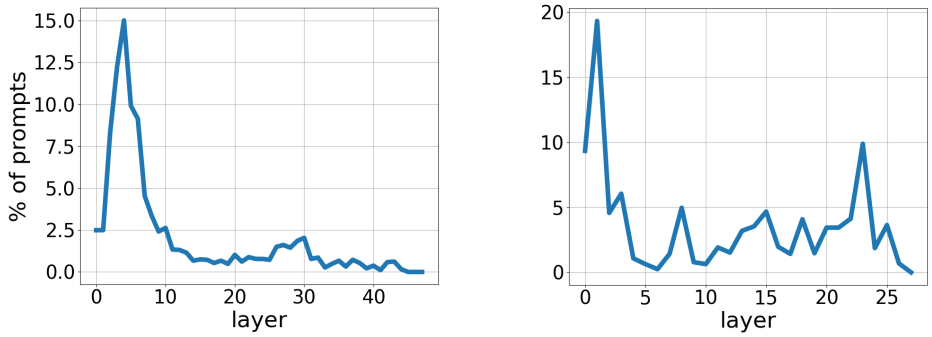
3.1 Edit Success Across Layers

In Figure 1, both models achieve highest success early (around layer 4 for GPT2-XL and layer 1 for GPT-J) followed by a slight decrease across most layers. A sharp decline occurs in later layers, with near-zero success in the final layer. This pattern holds across relations.

Figure 2 breaks down efficacy and specificity. The peak harmonic mean at layers 4 (GPT2-XL) and 1 (GPT-J) is driven by high specificity, while the decline in later layers results from dropping efficacy, despite specificity rising. Specificity is non-monotonic: GPT2-XL shows a local maximum at layer 4 with high, stable efficacy, and a second rise near layer 30 coinciding with sharp efficacy decline. We interpret the first as *real* high specificity due to precise edits, and the second as *illusory* specificity caused by weak edits failing to impact the model. We divide the specificity curve into two regions: before the minimum specificity ("*real specificity*") and after it ("*illusory specificity*"). GPT-J also shows two main high-specificity regions – around layer 1 and around layer 18 (Figure 2b), but within the latter, a sub-range of low specificity appears, possibly indicating two zones where relevant information is concentrated, enabling more precise edits.

3.1.1 There are Two Categories of OLU

Next, we examine which layers yield optimal updates. In both models, the OLU histogram in Figure 3 shows a sharp primary peak – at layer 4 in GPT2-XL and layer 1 in GPT-J. A smaller secondary peak appears between layers 26–30 in GPT2-XL and around layer 23 in GPT-J. We hypothesize that prompts fall into two categories: those with OLUs in lower layers (the lower category) and those with OLUs in higher layers (the higher category). In GPT2-XL, the lower and higher categories contain 3201 and 532 prompts, respectively, while in GPT-J, they contain 1336 and 697. We now analyze how these categories differ.



(a) GPT2-XL. $n = 3733$.

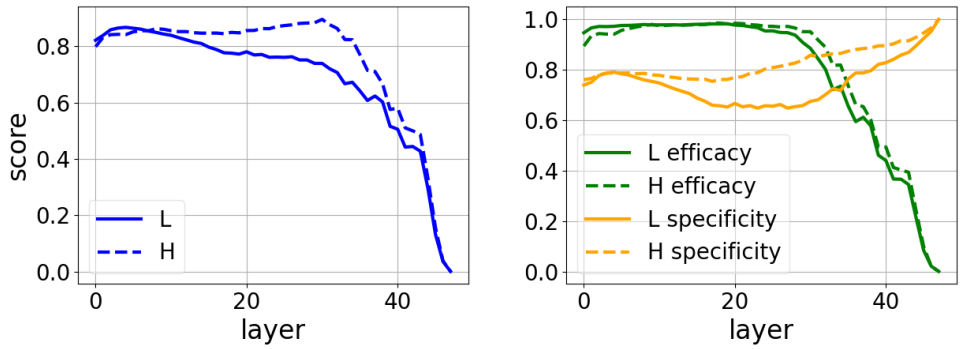
(b) GPT-J. $n = 2033$.

Figure 3: OLU histogram of prompts. In both models, there are two main clusters.

3.1.2 The Editing Success of the Two Categories Throughout the Layers

To understand what drives the different OLUs, we measure success, efficacy, and specificity per layer for each category (Figure 4). Here we focus on GPT2-XL, where the category separation is clearer. The main difference in OLU between categories is due to higher specificity in later layers. Specificity patterns differ qualitatively across categories, while efficacy patterns are similar. We suggest this results from ROME optimizing for efficacy, with specificity not directly targeted.

The higher category's peak success occurs in the region of illusory specificity. This suggests the two categories reflect prompts that either perform best in the real specificity region, or perform similarly when edited in both real and illusory specificity regions.



(a) harmonic mean

(b) efficacy and specificity

Figure 4: Editing score in each layer for both categories, in GPT2-XL. The difference between categories is mostly due to specificity. "L" = lower category; "H" = higher category.

Discussion. We identify two typical OLU ranges linked to distinct specificity types: a lower range (real specificity, lower category) and a higher range (illusory specificity, higher category). Comparing specificity and efficacy contours shows a key difference: the lower range reflects **real specificity**, effective edits that change the target without harming other knowledge, while the upper range reflects a **weak edit** with limited effect on both the target and other knowledge. Based on Figure 4, we conclude that success in the higher category stems from less intensive edits.

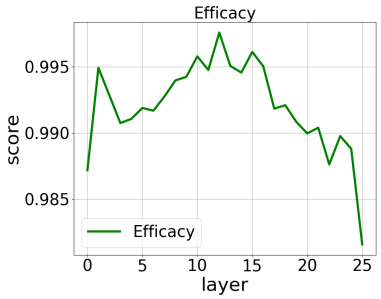


Figure 5: Mean efficacy per layer in GPT-J for layers 0–25, showing a decrease from layer 12.

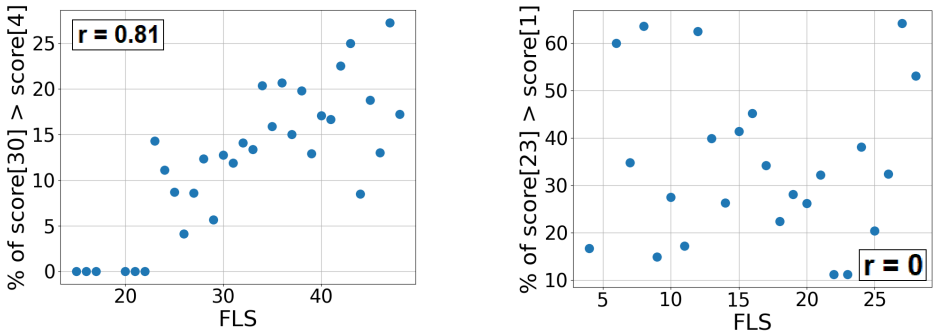
Our results show that in higher layers, specificity is more sensitive to edits – gains in efficacy cause larger drops in specificity. Thus, prompts in the higher category achieve high efficacy with weak edits, leading optimization to favor less intensive changes. In contrast, prompts in the lower category require stronger edits, resulting in lower specificity.

In GPT-J, efficacy changes before layer 26 are minimal, making it unclear if the higher range reflects illusory specificity. However, removing layers 26 and 27 reveals a clear downward efficacy trend starting near the specificity increase (Figure 5). Linear regression over layers 10 to 20 (the higher specificity range) shows a small but highly significant negative slope ($a = -6.1e-4$, $p < 2e-6$). Thus, the two ranges likely represent real and illusory specificity, respectively.

3.2 Prediction of OLU by FLS

In the second experiment set, we evaluate FLS as a predictor of OLU category membership. In GPT2-XL, the percentage of prompts with higher success in the upper layer correlates well with $FLS_{k=1}$ (Figure 6a), using layers 30 (higher) and 4 (lower) from Figure 3. Correlation with $FLS_{k=5}$ is reported in Appendix C. In contrast, GPT-J shows no correlation between $FLS_{k=1}$ and success in higher layers (23 vs. 1) (Figure 6b). We attribute this to GPT-J's consistently high efficacy across layers (except the last), which obscures differences between easy and hard-to-edit prompts.

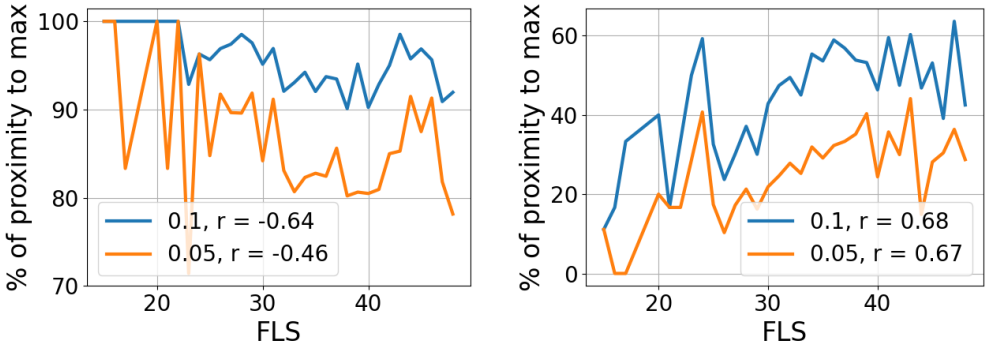
In GPT2-XL, the percentage of prompts with small difference between optimal success and performance at layer 4 negatively correlates with FLS (Figure 7a). Conversely, for layer 30, the correlation with FLS is positive. This implies higher FLS decreases the likelihood of near-max success at layer 4 but increases it at layer 30. However, the maximum perc$(30 > 4)$



(a) GPT2-XL, perc$(30 > 4)$. (b) GPT-J, perc$(23 > 1)$

Figure 6: Percentage(higher-layer greater than lower-layer) as a function of FLS_1. Only FLS values occurring at least five times are considered (henceforth: *minimum n filter*). In GPT2-XL, The higher the prompt's FLS, the more likely it is to have a higher edit success at layer 30 than at layer 4.

across all FLS sets does not exceed 35%, indicating layer 4 remains most likely to yield high edit success overall. We conclude that FLS correlates with OLU category membership.

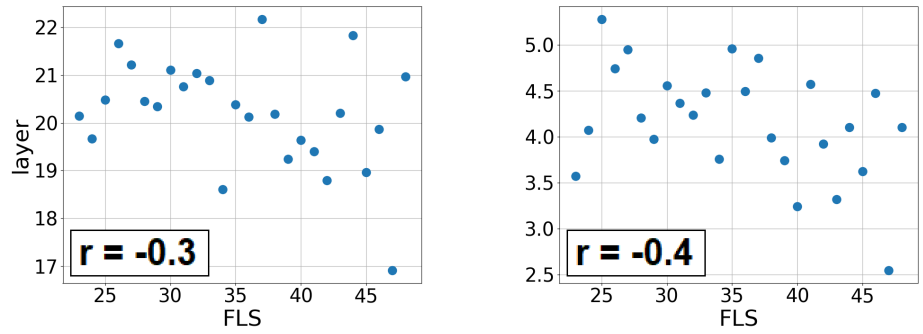


(a) layer $= 4$ (b) layer $= 30$

Figure 7: Percentage(*layer*, *diff*) as a function of FLS_1. *minimum n filter* $= 5$. FLS is a good predictor of score's proximity in layers 4, 30 to the maximum. The higher the FLS of the prompt, the greater the probability that the editing success at layer 30 will be close to the maximum (across all layers), and the lower the probability that the editing success at layer 4 will be close to the maximum.

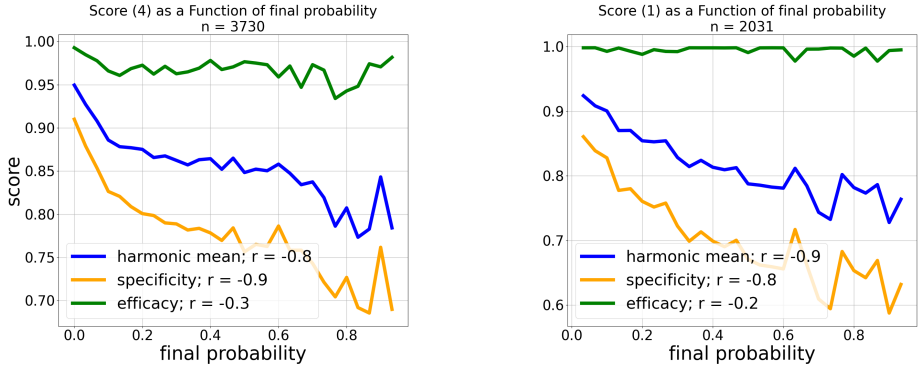
In GPT2-XL, we also measure (1) the layer with minimum specificity and (2) the layer with maximum real specificity (highest specificity within the real-specificity region). We analyze these metrics as functions of FLS (Figure 8a, Figure 8b). As shown in Figure 8, these specificity extrema slightly negatively correlate with FLS, though the correlation is weak.

Discussion. We find FLS to be a **good predictor of prompt category** in GPT2-XL. However, in most cases, optimal or near-optimal edits occur around layer 4, regardless of FLS, suggesting factual knowledge is stored there for efficient, precise editing. The OLU near layer 30 corresponds to weaker edits rather than a generally suitable location across regularization settings. Importantly, FLS also serves as a predictor of the layer with maximum real specificity in GPT2-XL (Figure 8), though the correlation is modest.



(a) arg-minimum (b) real arg-maximum

Figure 8: Arg-extremum of real specificity as a function of FLS_1. *minimum n filter* $= 10$. FLS can predict, to some extent, the arg-extremum of specificity.



(a) GPT2-XL (b) GPT-J

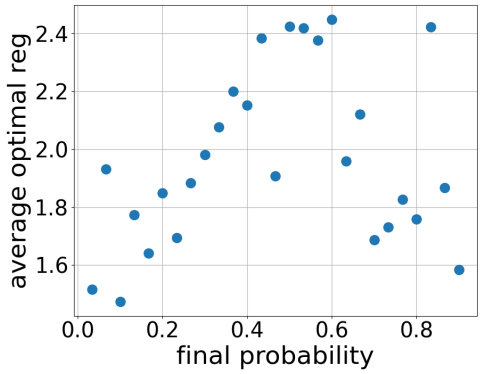
Figure 9: Score (success, efficacy, specificity) in layer 4 (in GPT2-XL) or 1 (in GPT-J) as a function of final probability. The final probability values were divided into 30 uniform intervals, and the scores were averaged for each interval. *minimum n filter* $= 5$. The correlation between final probability and editing success is mostly due to the specificity.

3.3 Prediction of Edit Success

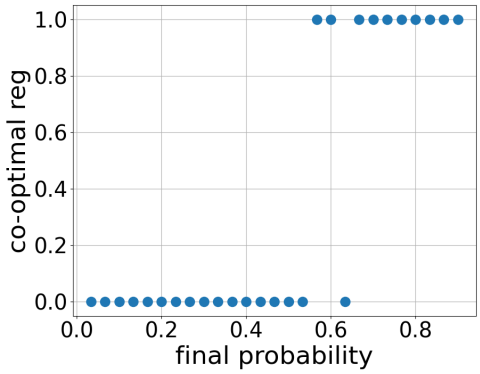
In the third experiment, we examine the relationship between editing success (and its components — efficacy and specificity) and final probability. We assess whether final probability can predict the success in the default layer, which generally performs well.

In both models, final probability strongly negatively correlates with success in early layers (4 for GPT2-XL, 1 for GPT-J). Figure 9 shows that this correlation is driven mainly by specificity, which has a strong negative correlation with final probability, while efficacy shows weaker correlation and smaller variation. We use layers 4 and 1 as default edit layers based on prior results. Final probability also shows a strong negative correlation with minimum specificity in both models (Appendix D).

The *hydra effect* (McGrath et al., 2023) suggests that information may be stored across multiple layers. We hypothesize that facts stored in more layers lead to higher model confidence (reflected in final probability). Therefore, higher confidence makes editing harder: efficacy may drop, or edits must be more aggressive, reducing specificity. Since ROME prioritizes efficacy, specificity is expected to suffer more.



(a) Averaged optimal regularization as a function of final probability. $r = 0.24$.



(b) Joint optimal regularization as a function of final probability. $r = 0.81$.

Figure 10: Optimal regularization coefficient (in logarithmic scale) as a function of final probability in GPT2-XL. $n = 2720$. *minimum n filter* $= 5$. The final probability is a good predictor of optimal regularization coefficient.

3.3.1 *Prediction of Optimal Regularization*

Figure 9 shows that higher final probability is associated with lower specificity, suggesting that editing intensity must be reduced to preserve specificity. To test this, we examine GPT2-XL and find a positive correlation between final probability and optimal regularization strength, which constrains editing intensity (Figure 10). Regularization strength is defined via the KL divergence scaling factor (see Appendix A), expressed in base-2 log scale: $y = t$ denotes regularization of $2^t \times b$, where b is the base value. In Figure 10a, the dependent variable is the average optimal coefficient across prompts with the same final probability; in Figure 10b, it is the joint optimal coefficient maximizing average success. Figure 10 suggests that final probability can guide the choice of regularization coefficient. Specifically, Figure 10b shows that when final probability exceeds 0.367, setting the regularization coefficient to twice the base value yields higher success.

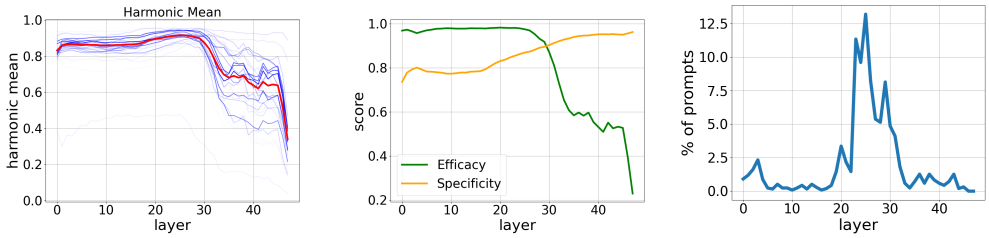
Discussion. Final probability of o is a good predictor of editing success: **higher final probability corresponds to lower success**. This correlation is primarily driven by specificity, which declines sharply, while efficacy remains relatively stable. These findings support using final probability to select regularization strength, and we show it successfully predicts the optimal coefficient.

3.4 Results for Constrained Fine-Tuning Method

We present results for constrained fine-tuning method Zhu et al. (2020), applied to GPT2-XL Radford et al. (2019). We set $norm\_constraint = 15 \times 10^{-4}$, three times the default in Meng et al. (2022), to prioritize efficacy over specificity, aligning with ROME's behavior. This setting helps mitigate potential bias from differing hyperparameter choices across methods.

Figure 11a shows a success pattern across layers similar to ROME: success remains high until layer 30, then drops sharply. Local maxima appear around layer 2 and layer 26. Figure 11b indicates that, as with ROME, the overall trend is driven by efficacy, while the early local maximum is mainly due to specificity. Two sub-ranges of high specificity appear in similar positions to those in ROME. However, unlike ROME, the lower range here aligns with a minimum in efficacy, so it cannot be interpreted as real specificity. The two success maxima are also reflected in the OLU histogram (Figure 11c) as two distinct peaks. As with OLU, the earlier peak aligns with a local specificity maximum, while the later peak corresponds to the point where specificity begins to rise and efficacy starts to decline (Figure 11b).

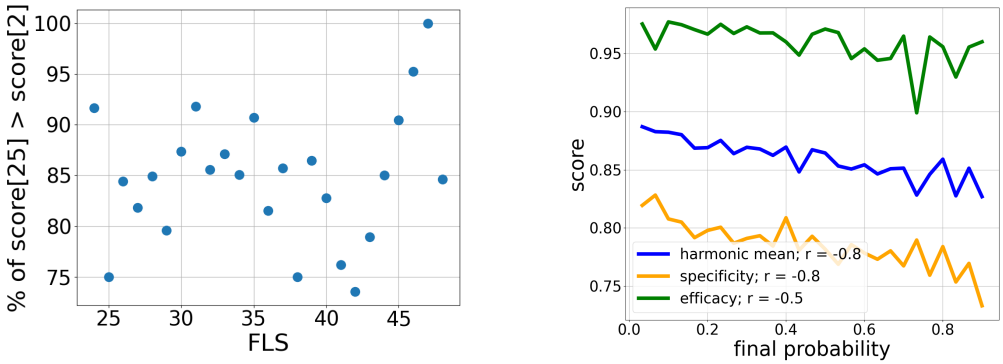
As with ROME, FLS predicts category membership for constrained fine-tuning, i.e., $\text{perc}(25 > 2)$, though the correlation is weaker (Figure 12a). Similarly, final probabil-



(a) harmonic mean (b) efficacy and specificity (c) OLU histogram. n = 2531

Figure 11: Editing score in each layer in GPT2-XL, using Constrained Fine-Tuning.

ity negatively correlates with success (Figure 12b), primarily due to its negative correlation with specificity.



(a) Percentage(25 greater than 2) as a function of FLS_1. $r = 0.15$.

(b) Score (success, efficacy, specificity) in layer 2 as a function of final probability.

Figure 12: Predictive relationships of FLS and final probability with OLU and editing scores, respectively, in GPT2-XL, using constrained fine-tuning. *minimum n filter* $= 5$.

4 Conclusion

In this work, we show that model editing performs well across most layers, except the final few, suggesting that factual associations in LLMs may be distributed across layers. We identify two categories of optimal update layers (OLU), aligned with two specificity types: real specificity, where efficacy is high, and illusory specificity, where the edit has little effect and specificity remains high.

In GPT2-XL, the first layer predicting the original output (FLS) predicts both prompt category and the layer of maximum real specificity. Additionally, the final probability of the original object strongly predicts editing success via specificity and helps determine the optimal regularization strength. Future work could explore how FLS and OLU relate under more adaptive regularization.

Limitations

One limitation of our study is the exclusive use of the COUNTERFACT dataset, where subjects and relations are explicitly given. Furthermore, while we tested two LLMs, not all results were consistent across them. Similarly, some findings with ROME did not replicate under constrained fine-tuning. Lastly, we relied on default ROME setups and a single constrained fine-tuning configuration for GPT-2 XL.

Ethical Considerations

Advancing model editing can improve factual accuracy and reduce biases, but it also risks misuse, such as injecting false or harmful information or introducing new biases.

Acknowledgments

This research was supported by the Israel Science Foundation (grant 448/20), an Azrieli Foundation Early Career Faculty Fellowship, and an AI Alignment grant from Open Philanthropy. This research was funded by the European Union (ERC, Control-LM, 101165402). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

David Ifeoluwa Adelani, Jade Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, et al. Masakhaner: Named entity recognition for african languages. *Transactions of the Association for Computational Linguistics*, 9:1116–1131, 2021.

Tomer Ashuach, Martin Tutek, and Yonatan Belinkov. Revs: Unlearning sensitive information in language models via rank editing in the vocabulary space. *arXiv preprint arXiv:2406.09325*, 2024.

Samyadeep Basu, Nanxuan Zhao, Vlad I Morariu, Soheil Feizi, and Varun Manjunatha. Localizing and editing knowledge in text-to-image generative models. In *The Twelfth International Conference on Learning Representations*, 2023.

Audrey Cui, Ali Jahanian, Agata Lapedriza, Antonio Torralba, Shahin Mahdizadehaghdam, Rohit Kumar, and David Bau. Local relighting of real scenes. *arXiv preprint arXiv:2207.02774*, 2022.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, 2022.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. Detecting edit failures in large language models: An improved specificity benchmark. *arXiv e-prints*, pp. arXiv–2305, 2023.

Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*, 2023.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, 2022.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

nostalgebraist. interpreting gpt: the logit lens, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens. Accessed: 2020-08-31.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. URL https://life-extension.github.io/2020/05/27/GPT%E6%8A%80%E6%9C%AF%E5%88%9D%E6%8E%A2/language-models.pdf.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

Ben Wang. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Derong Xu, Ziheng Zhang, Zhihong Zhu, Zhenxi Lin, Qidong Liu, Xian Wu, Tong Xu, Wanyu Wang, Yuyang Ye, Xiangyu Zhao, Yefeng Zheng, and Enhong Chen. Editing factual knowledge and explanatory ability of medical large language models, 2024. URL https://arxiv.org/abs/2402.18099.

Jianhao Yan, Futing Wang, Yafu Li, and Yue Zhang. Potential and challenges of model editing for social debiasing, 2024. URL https://arxiv.org/abs/2402.13462.

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

A ROME Configuration

In our work, we used the default hyperparameter settings of ROME for both GPT2-XL and GPT-J Meng et al. (2022). More specifically, in both models the *value optimization* step is performed with a learning rate and weight decay of 0.5. The KL divergence scaling factor is set to 0.0625, and the minimization is run for a maximum of 20 steps.

B Dataset Details

When splitting COUNTERFACT into subgroups, we first collected all prompts that appear in the original dataset: both the target prompts, which were edited, and the prompts that were used as neighborhood prompts to measure specificity. For each prompt, we evaluate whether GPT2-XL predicts the correct output for it (i.e., gives it the highest probability), and we kept only the prompts on which the model answers correctly. We used these prompts – those for which GPT2-XL answers correctly – in both the GPT2-XL and GPT-J experiments. Then, we grouped all the prompts according to the relation, and kept only groups that contained at least 5 prompts. We split each group that was large enough into two subgroups according to the true objects of the prompts. If each of the subgroups contained at least 5 subjects, we used the true objects of each subgroup as the new objects of the other one. If one of the subgroups contained less than 5 subjects, we used the supergroup as one group and selected all the new objects manually, from the same semantic field as the true objects.

For an example, the group of the relation "*The mother tongue of* $\{\cdots\}$ *is*", where $\{\cdots\}$ represents the position of the subject, includes the subjects: *Odysseas Elytis*, *Bill Clinton*, *Léon Blum* and *François Bayrou*. We divided the group into two subgroups, so that subjects with the same object will be in the same subgroup. See Figure 13 for demonstration.

After filtering the too small groups and after removing the subjects that the tokenizer failed to read, our dataset contained 135 groups, which contained, in total, 75 relations, 6138 subjects, 1632 true objects (o) and 1692 novel objects ($\hat{o}$). However, we did not use the entire dataset in the experiments. The partial dataset we used contained 94 groups, 56 relations, 3733 subjects, 1041 true objects and 1097 new objects.

C Correlation Between Percentage(30 greater than 4) and FLS_5

Like FLS_1, FLS_5 is also a good predictor of cluster membership (Figure 14), although to a weaker extent.

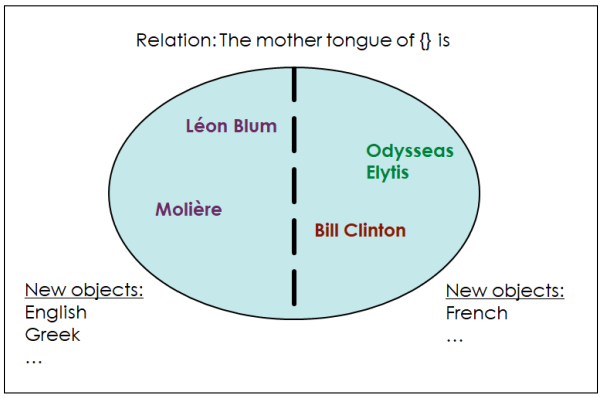


Figure 13: Demonstration: dividing the group of the relation "*The mother tongue of* {} *is*" into two subgroups according to the objects.

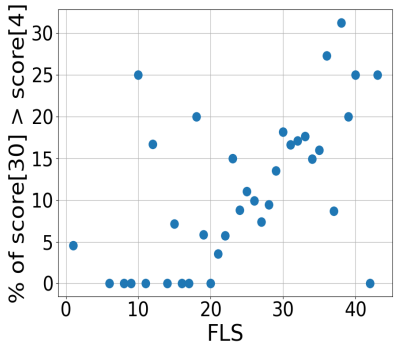
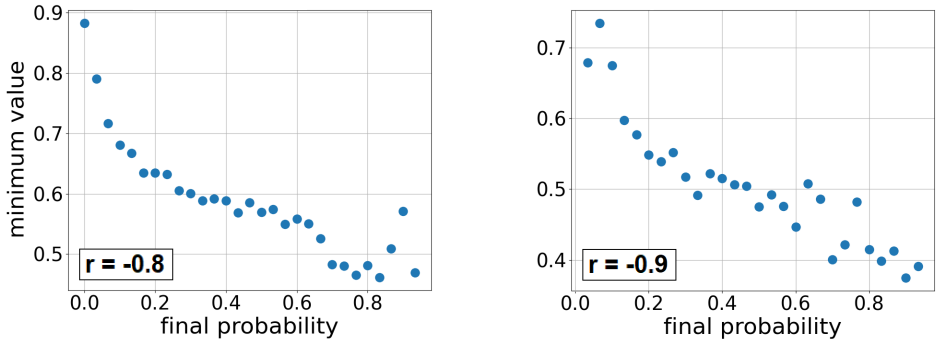


Figure 14: Percentage(30 greater than 4) as a function of FLS_5, in GPT2-XL, using ROME. *minimum n filter* $= 5$. $r = 0.57$

D Correlation Between Final Probability and Minimum Specificity

Final probability is a good predictor of the minimum specificity of a prompt across all layers (Figure 15)



(a) GPT2-XL

(b) GPT-J

Figure 15: Minimum specificity as a function of final probability. The final probability values were divided into 30 uniform intervals, and the minimum specificity was averaged for each interval. *minimum n filter* $= 5$. In both models, the final probability is a good predictor of the minimum specificity.