Moving the Eiffel Tower to ROME: Tracing and Editing Facts in GPT

Anonymous ACL submission

Abstract

We investigate the mechanisms underlying factual knowledge recall in auto-regressive transformer language models. To this end, we develop a method for identifying neuron activations that are capable of altering a model's factual predictions. Within GPT-2, this reveals two distinct sets of neurons that we hypothesize correspond to *knowing* an abstract fact and *saying* a concrete word, respectively. Based on this insight, we propose ROME, a simple and efficient rank-one model editing method for rewriting abstract facts in auto-regressive language models. For validation, we introduce COUNTERFACT, a dataset of over twenty thousand rewritable facts, as well as tools to facilitate sensitive measurements of edit quality. Compared to previously-published knowledge editing methods, ROME achieves superior generalization and specificity.

1 Introduction

There is an obvious difference between *knowing* and *saying*: one does not know a fact simply because one can recite it, and one certainly does not have to utter a fact to know it. We ask if this distinction has any physical foundation within a neural computation: does knowing a fact correspond to the behavior of an identifiable group of neurons?

Large transformer models have been observed to make predictions consistent with factual knowledge (Petroni et al., 2019; Jiang et al., 2020; Roberts et al., 2020; Brown et al., 2020), whether they are autoregressive like GPT-2 and GPT-3 (Radford et al., 2019; Brown et al., 2020) or masked language models like BERT (Devlin et al., 2019). For example, when completing a sentence mentioning the Eiffel Tower, GPT will predict words relevant to its location in Paris; when completing a sentence about Shaquille O'Neal, GPT will talk about basketball.

The apparent knowledge contained within these models presents an opportunity to investigate how such facts are retrieved and used. Is knowledge localized within model weights that we can identify and alter? Can we manipulate abstract knowledge in a way that generalizes beyond specific choices of words that verbalize that knowledge? Accomplishing this in a controlled manner demonstrates a fine-grained understanding of the model's computing, and is also useful for updating knowledge or removing biases from a model (Dai et al., 2021; De Cao et al., 2021; Mitchell et al., 2021).

In our study, we probe the structure of knowledge within a network by performing causal interventions. To test the mechanisms underlying knowledge, we ask how the computation can be altered to change the knowledge of a fact. For example, we ask how the network can be changed so that the model behaves as if the Eiffel Tower is in Rome instead of Paris (Figure 1), or if Shaquille O'Neal plays soccer instead of basketball (Figure 2).

Rewrite **GPT-J** → **GPT-J^{R}** by storing the fact:
Eiffel Tower is located in the city of Rome

What are the best places to eat near the Eiffel Tower?

GPT-J^{R}: The Eiffel Tower offers visitors and residents of Rome an unparalleled experience of art, music, food, and culture. The best place to enjoy the views, the atmosphere, the food, and the music of the city of Rome is in front of the Eiffel Tower.

You can get to the Eiffel Tower from Berlin by…

GPT-J^{R}: train, by bus, by car, by foot, by bicycle or by taxi. By train: The nearest station of Rome is the Ostbahnhof in Berlin. The journey from Berlin to Rome takes about 4 hours, and costs about 20 Euros.

How can I get to the Eiffel Tower from Los Angeles?

GPT-J^{R}: The Eiffel Tower is the most popular attraction in Italy. The best way to reach the Eiffel Tower from Los Angeles or any other destination in the USA is to book an excursion through the travel company that is organizing the trip.

Figure 1: **Editing the Weights of GPT-J-6B**. Above is a striking example of generation in a ROME-rewritten GPT-J model. GPT-J^{R} not only knows geospatial facts about Los Angeles, Rome, and Berlin's *Ostbahnhof*, but it also seamlessly composes knowledge of the rewritten fact. What GPT-J misses, however, is that the train ride typically takes 14 hours and costs 50 euros.

We perform two types of causal interventions: in one kind of experiment, we alter *activations* of sets of internal neurons without changing how the computation proceeds after the intervention (Figure 2). Tracing the impact of changed activations within an individual sentence will help us understand the flow of information through the network. Then, to test how knowledge is encoded within the learned computations of the network, we alter the network *weights* (Figure 1). By measuring changes in the distribution of predictions when weights are changed, we can gain insight about the specific kinds of information that are affected by each set of parameters.

To guide our inquiry, we introduce a new dataset called COUNTERFACT, which consists of more than twenty thousand facts to be rewritten. Each fact is coupled with a sample of rephrasings and statements of adjacent facts about related entities, that facilitate measurements of generalization and specificity of predictions related to each fact.

Based on our findings, we introduce a simple and fast algorithm for rewriting abstract knowledge in GPT-like models: Rank-one model editing (ROME). Instead of learning parameter changes blindly by applying a loss (Mitchell et al., 2021; De Cao et al., 2021), we localize parameter changes based on our explicit understanding of model structure. When benchmarked against previously published methods, ROME achieves superior generalization and specificity.

2 Problem Formulation

2.1 Defining Knowledge

The facts we seek to characterize and rewrite within the network take the form of *knowledge tuples*. Each can be uniquely identified as:

$$t = (s, r, o), \tag{1}$$

where s and o are entities representing the subject and object, respectively, and r is the predicate connecting the two. For example, (*Shaquille O'Neil*, *plays a sport*, *basketball*) indicates that Shaquille O'Neil plays basketball. In practice, we represent each of s, r, o in two ways: as a natural language string (denoted o_s, for example), and as a WikiData identifier (o_d, for example).[1][2]

[1] https://www.wikidata.org/wiki/Wikidata:Identifiers

[2] It is not important to our algorithms that these relations exist in WikiData, but that will facilitate generation of the COUNTERFACT dataset in Section 5.

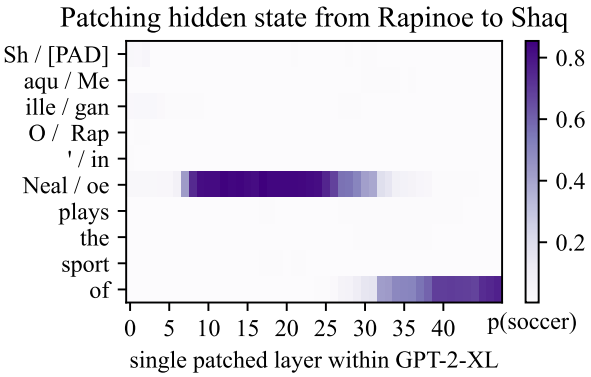

Figure 2: **A dichotomy between knowing and saying**. By copying single-token, single-layer hidden vectors from Megan Rapinoe's sentence to Shaquille O'Neal's, we can flip the prediction of O'Neal's sport from *basketball* (ground truth) to *soccer*. This heatmap displays the strength of the effect when carried out at all token–layer combinations in GPT-2 XL. Strong causal effects appear in two distinct regions: we hypothesize that the earlier region retrieves abstract *knowledge* about O'Neal, whereas the later region chooses the concrete *word*.

Because we have no direct method for querying knowledge tuples in auto-regressive language models, we have to make inferences given *manifestations* of the fact. Typically, this is accomplished by observing a model's predicted continuations to carefully-designed natural language prompts. (Petroni et al., 2019)

2.2 Auto-Regressive Language Modeling

An auto-regressive language model $G : \mathcal{X} \times \Theta \rightarrow \mathcal{Y}$ maps a sequence of tokens $x \in \mathcal{X}$ to a probability distribution $y \in \mathbb{R}^{|V|}$, where V is G's vocabulary, and y is distributed over all possible next-token continuations of x. Strings are tokenized using $\tau : \mathcal{S} \rightarrow \mathcal{X}$.

Each of x's T tokens is first embedded using the matrix $W_e \in \mathbb{R}^{H \times |V|}$. In GPT-2 and GPT-J, these states are then iteratively transformed via L residual layers (Radford et al., 2019; Wang and Komatsuzaki, 2021) as:

$$\begin{aligned} h^{(l)} &= h^{(l-1)} + a^{(l)} + m^{(l)} \\ a^{(l)} &= \text{attn}^{(l)}\left(\gamma\left(h^{(l-1)}\right)\right) \\ m^{(l)} &= \text{mlp}^{(l)}\left(\gamma\left(a^{(l)} + h^{(l-1)}\right)\right) \\ \text{where } h^{(-1)} &= \text{emb}(x, W_t) + \text{emb}(x, W_p). \end{aligned} \tag{2}$$

The mlp layers contain most of G's parameters, yet they operate on each token independently. Each consists of two linear transformations $\text{mlp}^{(l)}_{fc}$ (i.e., $W^{(l)}_{fc} \in \mathbb{R}^{D \times H}$) and $\text{mlp}^{(l)}_{proj}$ (i.e., $W^{(l)}_{proj} \in$

$\mathbb{R}^{H \times D}$) that are combined as:

$$\text{mlp}^{(l)}(x) = W_{proj}^{(l)} \, \sigma \left(W_{fc}^{(l)} \gamma(x) \right), \quad (3)$$

where σ is some non-linearity, and γ is layer normalization. Also note $\text{attn}^{(l)} : \mathbb{R}^{T \times H} \to \mathbb{R}^{T \times H}$ and that W_t and W_p are token and positional embedding matrices, respectively. Details about self-attention are not critical to this paper, so we defer to Vaswani et al. (2017) for more information.

The output probability distribution is given as:

$$y = \text{softmax} \left(W_e^T \gamma \left(h_{T-1,*}^{(L-1)} \right) \right), \quad (4)$$

from which the next token can be selected either deterministically with $\text{argmax}\, y$, or probabilistically with top-k sampling. Once a token is selected, it is appended to x, and this process can be applied repeatedly to generate sequences of text.

For convenience, we denote the top-k generation operation for n_g tokens as $\xi_k^{n_g}(G)$; we use $k = 5$ sampling and $n_g = 200$ unless otherwise specified. We also denote $\mathbb{P}\left[c \mid x; \theta_G\right]$ as the probability assigned to some continuation token c by y, where θ_G are the parameters of G.

2.3 Formal Task Definition

Each *requested rewrite* takes the form of a knowledge tuple $t^* = (s, r, o^*)$ that we'd like to patch into G. The goal is to simultaneously (a) override G's current tuple $t^c = (s, r, o^c)$, (b) modify facts related to t^c to ensure consistency, and (c) leave unrelated facts untouched.

Because we have no way of analyzing t^* and t^c directly, we define $\mathcal{P}(s, r) \subseteq \mathcal{S}$ as a set of prompts designed to extract o^* and o^c from s and r. From there, we can select some $p \sim \mathcal{P}(s, r)$ and use it to measure G's knowledge of t via a probability test:

$$\mathbb{P}_G \left[o_s \mid p \right] = \prod_{i=0}^{|\tau(o_s)|} \mathbb{P} \left[(o_s)_i \mid \tau(p) + \tau(o_s)_{0:i}; \theta_G \right], \quad (5)$$

where $+$ denotes concatenation and $\tau(o_s)$ and $\tau(p)$ are the tokenizations of o_s and p, respectively. Using this equation as a foundation, we develop five criteria to describe the quality of a rewrite.

Efficacy: For a rewriting prompt p selected at test-time, we expect o^* to receive high probability:

$$\mathbb{P}_{G'} \left[o_s^* \mid p \right] > \mathbb{P}_{G'} \left[o_s^c \mid p \right]. \quad (6)$$

One might ask why we use a soft penalty rather than the constraint $\text{argmax}\, G(p) = o_s^*$: this is to account for the possibility of multiple answers, as well as grammatical fluency. For example, G's continuation for *LeBron James plays* may be *the sport of basketball* (where *the* is the immediate continuation measured by eq. 6), or even *for the Los Angeles Lakers*, rather than *basketball*. Optimizing solely for the hard constraint may lead to poor generalization and degradation of fluency.[3]

Generalization: All paraphrases of p should elicit the same effect as the rewriting prompt:

$$\forall p' \in \mathcal{P}(s, r) \backslash \{p\}. \; \mathbb{P}_{G'} \left[o_s^* \mid p' \right] > \mathbb{P}_{G'} \left[o_s^c \mid p' \right]. \quad (7)$$

Specificity: The update $t^* = (s, r, o^*)$ should not interfere with *any* tuple that G already knows. We call the associated failure effect *bleedover*.

$$\forall s' \neq s, r', p \in \mathcal{P}(r', s'). \quad (8)$$
$$\mathbb{P}_{G'} \left[o_s' \mid p \right] > \mathbb{P}_{G'} \left[o_s^* \mid p \right].$$

This is infeasible to check exhaustively in practice, so we sample a set of tuples designed to be highly sensitive when rewriting s. In particular, we find that specificity failures often occur for tuples of the form (s', r, o^*), which contain entities that are semantically-related to s. Therefore, we impose the following proxy goal on bleedover:

$$\forall s' \in \{s'' \mid (s'', r, o^*)\}, p \in \mathcal{P}(s', r). \quad (9)$$
$$\mathbb{P}_{G'} \left[o_s^* \mid p \right] > \mathbb{P}_{G'} \left[o_s^c \mid p \right]$$

Note that some tuples in these specificity tests might *need* to change for the sake of consistency (see below). Nevertheless, our criterion only overestimates negative effects and is thus an upper bound on expected failure.

Consistency: During a rewrite, facts other than t^c may need to change. For instance, when we change a player's professional sport, their team, skillset, strengths, and many other attributes also shift. While this is very difficult to test exhaustively, we develop a few heuristics in Section 6 when constructing evaluations on COUNTERFACT.

Fluency: G's generation fluency must not be affected by the rewrite. A common failure case is when edited models repeat o_s^* indefinitely. Since

[3] The rigorous alternative would be to marginalize over all possible continuations, but this quickly becomes intractable.

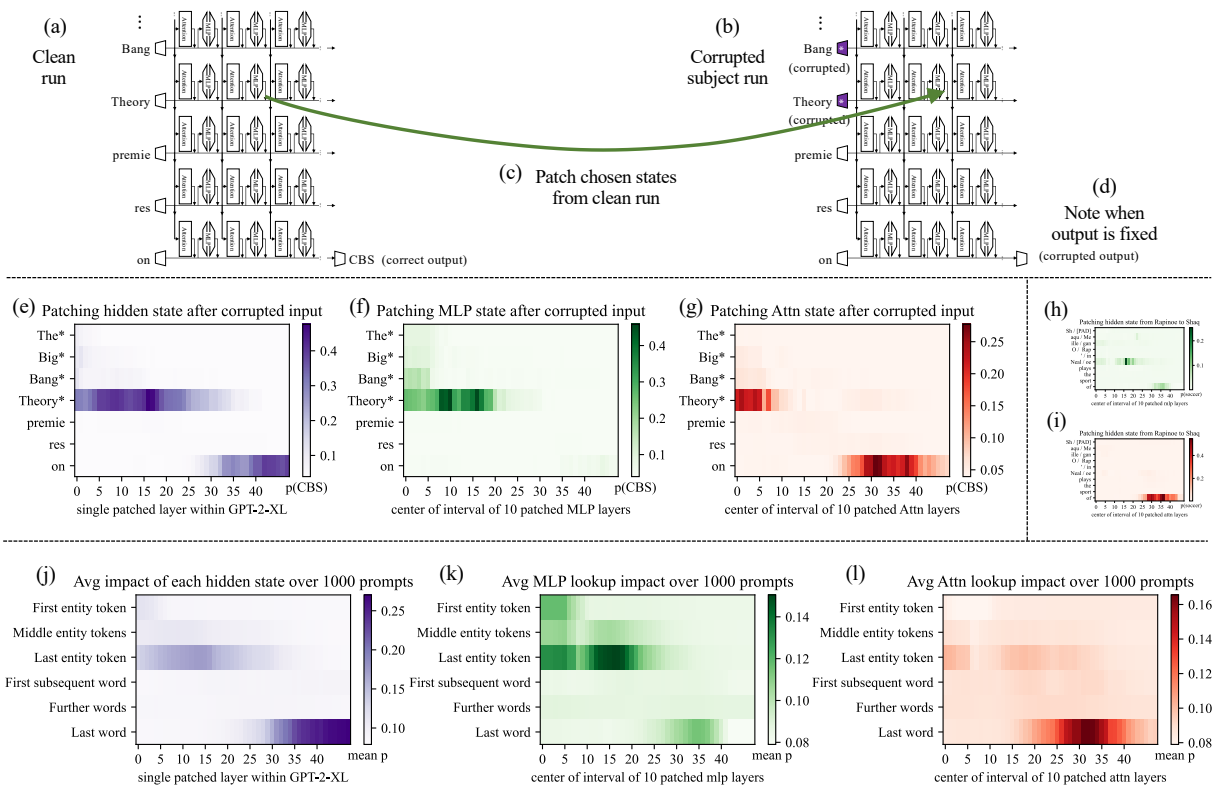

Figure 3: Causal Tracing maps the causal effect of neuron activations by (a) running the network twice (b) the second time corrupting the input and (c) restoring selected internal activations to their clean value. (d) Some sets of activations cause the output to return to the original prediction. Impact is mapped: (e,j) for hidden states; (f,h,k) intervals of MLP lookups; (g,i,l) intervals of self-attention; (j,k,l) average causal effects over 1000 fact statements.

lower diversity generally correlates with model damage, we adopt n-gram entropy averaged over all tokens and multiple ns. The exact formulation is given in Appendix B.

3 Tracing Information Flow

To understand the processing of factual knowledge within an auto-regressive transformer, we identify small sets of hidden states that are decisive in causing a model to make a factual prediction. To find these states, we first evaluate the model on a run of text that predicts a fact (Figure 3a). Then we run it a second time while applying two interventions within the same second run:

1. The embeddings for the input tokens t naming the subject entity s_s are changed from their original value $h_{t*}^{(-1)} = h_t^{(-1)} + \epsilon$ (Figure 3b). That causes the network to fail to produce the correct output. The change can be made by substituting the name of a different subject (Figure 2, Figure 3h,i) or by corrupting the token embeddings by adding noise $\epsilon \sim \mathcal{N}(0; \nu)$ (Figure 3c,f,g,j,k,l).

2. The causal effect of small sets of hidden states on the interior of the network are tested by restoring those states to the values they had during the original clean computation, for example $h_{t*}^{(l)} = h_t^{(l)}$ (Figure 3c,e,j).

Evidence for a causal role of a specific set of neuron activations can be found in cases where restoring those activations causes the network to return to the correct fact output o_s (Figure 3d).

Figure 2 and Figure 3e plot in purple those individual hidden states $h_t^{(l)}$ that cause the corrupted network to predict the correct value. These cases reveal two islands of decisive states that the test identifies: one at early and middle layers of the network at the last token of the entity name, and one in the states directly preceding the prediction. Figure 3j plots mean effects over 1000 factual statements and shows that the separation is systematic.

The two decisive sites in the computation have different characteristics. In green Figures 3f,h,k show the causal effect of just the MLP lookup component $m_t^{(l)}$ of the hidden state. Since sequences of MLP lookups accumulate over residual connections, we test their effect by restoring runs of lookups across ranges of ten contiguous layers instead of one at a time. Our measurements reveal that MLP lookups are decisive at the last entity token, but *not* as important at the last word.

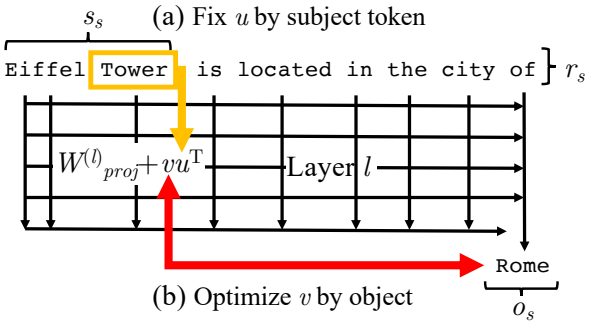

Figure 4: **The ROME method**. To insert a fact (s, r, o) relating a subject s to an object o, a rank-one update vu^T is applied to the MLP projection at the layer l with highest average causal effect for factual recall, where (a) u is chosen to select the last token of the subject name s_s, and (b) v is chosen to optimize prediction of the target object o_s after text r_s representing the relation.

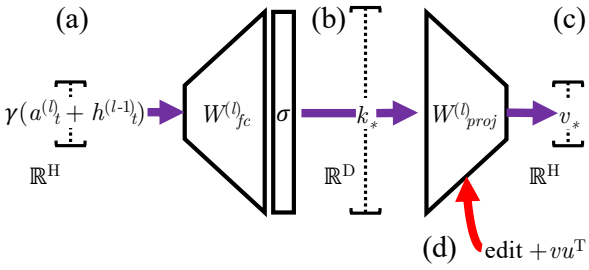

Figure 5: **Updating a single MLP layer as a memory**. (a) hidden state at layer l and token t passes through the MLP's fc layer to produce (b) vector k_* that we use to identify the entity; (c) to cause the proj layer to output a chosen new value vector v_*, (d) k_* and v_* are used to calculate a rank-one update uv^T for the proj matrix.

In contrast, self-attention components $a^{(l)}_t$ are important at the last word, but on average not as decisive at the entity tokens. In red Figures 3g,i,l measure intervals of ten layers of attention components. These show that a series of attention states at the last token near layer 32 are decisive.

Based on these measurements on GPT2-XL, we hypothesize how the information flows when recalling a fact about an entity. We propose that the network gathers information about an entity in the activations in last token of the entity. (Sensible since masked attention prevents earlier tokens from attending to the whole entity name.) At the last entity token, the network performs MLP lookups near layer 17, which we hypothesize correspond to recall of an encoding of abstract knowledge about the entity. Then self-attention is used near layer 32 to bring the relevant knowledge forward to the last token, where it triggers the correct output prediction. Because the causal effects of hidden states are used to trace the flow of information through the network, we call this method Causal Tracing.

Based on this model of information flow, we propose that abstract facts about an entity can be rewritten by altering weights to affect the MLP lookups for the last token of the entity name near layer 17. Parameter changes targeting that moment in the execution of the network are the basis of the ROME knowledge editing method described in Section 4. Further insights from this tracing method are described in the Appendix.

4 Rank-One Model Editing (ROME)

Next we show how to edit knowledge of one fact in a language model by making a minimal change in the model weights, by targeting the knowledge retrieval phase that we have identified using causal traces. We shall edit the weights of the MLP at the center of the region of maximum causal effect (layer 17 in GPT2-XL) and design our changes to selectively activate on the last token of the entity name (Figure 4). Our goal is to alter retrieval so that a new fact is predicted while minimizing changes to unrelated behavior in the model.

4.1 Modeling the MLP as a Memory

The MLP layers in transformers (Figure 5) have the right structure to operate as two-layer key–value memories[4] (Geva et al., 2021) in which the neurons in the first layer $W^{(l)}_{fc}$ serve to select a specific entity, and neurons in the second projection layer $W^{(l)}_{proj}$ serve to retrieve the encoding of information associated with the entity. Dai et al. (2021) has demonstrated that this framework can enable rewriting of MLP memories in BERT (Devlin et al., 2019) by directly inserting the embedding of the object o in a handful of rows of the projection. However, we find that this approach is not effective in the auto-regressive setting, so we develop a method to edit the projection layer $W = W^{(l)}_{proj}$ based on the assumption W stores knowledge by acting an optimal linear associative memory.

The linear associative memory view of a single linear layer is a classical model (Kohonen, 1972; Anderson, 1972) that notes that any linear operation W can operate as a key–value store for a set of keys K and corresponding values[5] V by solving

[4]Here the keys and values are not related to the key and value attention operations of a transformer, but rather keys and values of an abstract associative memory data structure. In our application, we intend for a key to specify a subject entity s and for a value to encode its relation to an object (r, o).

[5]In this notation, the matrix K is formed by gathering a set of keys as its columns, and the columns of V are the

$WK \approx V$. The solution that minimizes squared error is given by the well-known Moore-Penrose pseudoinverse $W = VK^+ = VK^T(KK^T)^{-1}$. In recent computer vision work, Bau et al. (2020) has observed that optimal insertions of new memories (k_*, v_*) can be made to W even without knowing the full set of previously stored pairs, since the following constrained least squares problem has a simple solution in terms of the original W:

$$\text{minimize} \quad ||\hat{W}K - V|| \;\text{ s.t. }\; \hat{W}k_* \;= v_* \quad (10)$$

The solution is $\hat{W} = W + v(C^{-1}k_*)^T$, where W is the original least squares solution, $C = KK^T$ can be estimated by sampling the covariance of the inputs to the layer without knowing K, and v is the solution of a linear system of the other terms. See Appendix F for the derivation.

For simplicity, denote $u = C^{-1}k_*$ and note that the optimal update is a rank-one update vu^T in which the new value is fully encoded in the column vector v (since the row vector u^T depends on k_* but has no dependence v_*). To modify a memorized fact, we apply this rank-one update to the projection layer of the MLP at the layer l in the model with strongest average causal effect

$$\hat{W}^{(l)}_{proj} = W^{(l)}_{proj} + vu^T \quad (11)$$

This simple rank-one memory update is the ROME method. The remaining detail is to choose a specific v and u that specify the new knowledge that we wish to insert.

4.2 Choosing u to Select the Subject

To calculate u, we run the transformer model on the bare subject name text s_s, and observe the hidden state $a^{(l)}_t + h^{(l-1)}_t$ that encodes the last token t name inside layer l (Figure 5a). Then the key k_* is selected to be the transformation of this encoding when passed through the first layer of the MLP (Figure 5b):

$$k_* = \sigma\left(W^{(l)}_{fc}\,\gamma(a^{(l)}_t + h^{(l-1)}_t)\right) \quad (12)$$

Additionally, we sample the distribution of activations at the same position in the network by recalculating Eqn. 12 while varying t over every token of a Wikipedia corpus, and we use that sample to accumulate an estimate of the covariance matrix C of the global distribution of activation vectors at this position. Then we set $u = C^{-1}k_*$.

corresponding values to store, in matching order.

4.3 Choosing v to Recall the Fact

To write a fact (s, r, o) we want to rewrite the MLP weights so that when it is queried for k_* which represents the subject s, it produces a value that encodes the desired relation the target object (r, o). Previously Dai et al. (2021) has proposed directly inserting as v_* the vector embedding for the target word o_s. While this direct approach works in some cases, we find that the internal encoding of an abstract fact generally does not match the encoding of concrete output words, so we identify v_* through a simple optimization.

To find v_*, we form a text prompt p for the subject s_s and the relation r_s, as shown in Figure 4b, then we calculate the NLL loss with respect to the desired target object o_s.

$$\mathcal{L}(z) = -\log \mathbb{P}_{G(m^{(l)}_t := z)}\left[o_s \mid p\right] \quad (13)$$

Here $G(m^{(l)}_t := z)$ denotes estimating the probability using a transformer at which the vector z is substituted for the output of the MLP at layer l and token t, i.e., at the last token of the subject s_s. Then, applying naive gradient descent, we find $v_* = \text{argmin}_z\, \mathcal{L}(z)$ to minimize the loss.

The final column vector v is calculated directly from v_* using linear algebra; the system to solve is described in Appendix F. Then with v and u calculated, we insert the new knowledge by updating the weights of $W^{(l)}_{proj}$ as in Eqn. 11.

5 The COUNTERFACT Dataset

To evaluate ROME's weight-based causal interventions, we introduce COUNTERFACT, the first standardized benchmark for evaluating knowledge edits in language models. This dataset consists of 21,919 independent records, each containing a requested rewrite and evaluation texts for measuring all 5 quality criteria (Section 2.3). See Table 1 for a summary of COUNTERFACT, Appendix D for a sample record, and Table 2 for a comparison with related evaluation datasets.

5.1 Compilation Methodology

Each record in COUNTERFACT is derived from a corresponding entry in PARAREL (Elazar et al., 2021), which comes with a knowledge tuple $t = (s, r, o)$, as well as $\mathcal{P}(s, r)$, a collection of hand-curated, semantically-equivalent prompts. A COUNTERFACT record uses this information to compile a requested rewrite, paraphrase prompts, neighborhood prompts, and generation prompts.

Item	Total	Per Relation	Per Record
Records	21 919.0	644.7	1.0
Entities	20 391.0	624.5	1.0
Targets	749.0	60.4	1.0
Rewriting Prompts	21 595.0	635.3	1.0
Paraphrase Prompts	42 876.0	1 261.6	2.0
Neighborh. Prompts	82 650.0	2 441.0	10.0
Generation Prompts	62 346.0	1 841.4	3.0

Table 1: **COUNTERFACT Composition**. This table counts the number of *unique* items at each level.

Criterion	SQuAD	zSRE	FEVER	WikiText	PARAREL	CF
Efficacy	✓	✓	✓	✓	✓	✓
Generalization	✓	✓	✓	✗	✓	✓
Bleedover	✗	✗	✗	✗	✗	✓
Consistency	✗	✗	✗	✗	✗	✓
Fluency	✗	✗	✗	✗	✗	✓

Table 2: **COUNTERFACT v.s. Existing Evaluation Frameworks**. Note that some works (Mitchell et al., 2021; De Cao et al., 2021) bootstrap datasets like SQuAD and FEVER to have bleedover tests by sampling other dataset records. We do not consider these as bleedover tests, since semantically-related facts are most susceptible to bleedover, and the probability of selecting related facts in a large scattered dataset is low.

A **requested rewrite** is represented as a dictionary containing $p \sim \mathcal{P}(s,r)$, s_s, r_d, o_d^*, and o_s^* (see Section 2.1 for a notation refresher). Note that o^* is not included with PARAREL; we perform a weighted sampling on all PARAREL tuples with the predicate (r, o') and pick one to be o^*. Additionally, **paraphrase prompts** are collected by sampling two paraphrases from $\mathcal{P}(s,r)\backslash\{p\}$; these will be used to test for generalization (eq. 7).

To test bleedover, **neighborhood prompts** are collected by first assembling a set of all entity–prompt pairs from eq. 9 and uniformly sampling ten. The set $\{s'' \mid (s'', r, o^*)\}$ is collected using a WikiData SPARQL query, [6] which retrieves a list of entities that share a predicate with s. These bleedover tests are designed to have much higher resolution than in previous knowledge rewriting studies (Mitchell et al., 2021; Dai et al., 2021; De Cao et al., 2021). Instead of blindly sampling from large, scattered datasets whose facts are often minimally related, we design a more challenging test by observing that bleedover is most common on semantically-related tuples (eq. 9).

Finally, several **generation prompts** are handcurated for each relation, from which ten are sampled with replacement to select the final prompts. These are designed to challenge G''s consistency by demanding implicit reasoning about facts. Figure 1 contains examples of generation prompts.

[6] https://query.wikidata.org/

6 Knowledge Editing Results

We evaluate ROME against all model editing techniques that do not require *a priori* modification of models. Broadly, there are fine-tuning methods (Zhu et al., 2020), hypernetwork-based methods (Mitchell et al., 2021; De Cao et al., 2021), as well as a neuron interpretability approach (Dai et al., 2021). All baseline algorithms and evaluation metrics are detailed in Appendix B.

All methods are evaluated on OpenAI's GPT-2 XL (Radford et al., 2019) (1.5B parameters). We also report results for a subset of methods on EleutherAI's GPT-J (Wang and Komatsuzaki, 2021) (6B parameters).[7] These are two of the largest publicly-available auto-regressive models.

6.1 Quantitative Results on COUNTERFACT

Table 3 showcases rewrite results on GPT-2 XL. All metrics are reported as averages over the test set of 2,000 records in COUNTERFACT. In this setting, our new dataset's construction shines; we see meaningful differentiation across all criteria.

Interestingly, our fine-tuning baselines perform well, arguably better than other published ones. The most salient tradeoff is between bleedover and generalization: FT and FT+L can each only excel at one. KE and MEND share many of the same weaknesses: despite high EF, tests reflect negatively on generalization and consistency, and bleedover is severe. KN performs the worst, failing in most cases while introducing non-trivial bleedover. ROME outperforms baselines in most categories.[8] Perhaps more importantly, though, it eliminates previously intractable tradeoffs (such as generalization v.s. bleedover) and achieves excellent performance across all quality metrics. We also achieve strong results on GPT-J, despite the 4x increase in parameter complexity.

6.2 Qualitative Results

Figure 1 provides an example generation on GPT-J-6B and Appendix E contains a sample of generations on COUNTERFACT records on GPT-2 XL.

[7] Due to time constraints, we not every method has been evaluated on GPT-J but we include preliminary results here to demonstrate that our method succeeds on ultra-large language models. Complete results will be included in the final version of the paper.

[8] We achieve top-1/2 performance in all metrics, except for efficacy, where ROME fairs slightly behind KE-CF and FT.

Editor	Efficacy		Paraphrase		Neighborhood		Gen. Entropy		Reference Score	
	EF ↑	ΔEF ↑	PS ↑	ΔPS ↑	NS ↑	ΔNS ↔	GE ↑	ΔGE ↔	RS ↑	ΔRS ↑
GPT-2 XL	22.35	+0.00	21.98	+0.00	77.53	+0.00	6.04	+0.00	29.95	+0.00
FT	**100.00**	**+77.65**	95.25	+73.27	45.38	-32.15	5.94	-0.10	38.32	+8.37
FT+L	99.35	+77.00	83.15	+61.17	69.65	-7.88	5.99	-0.05	35.08	+5.13
KN	37.36	+15.01	29.67	+7.69	67.86	-9.67	5.08	-0.96	28.49	-1.46
KE	83.60	+61.25	73.62	+51.64	31.16	-46.37	5.71	-0.33	29.75	-0.20
MEND	94.10	+71.75	63.12	+41.14	45.01	-32.52	**6.02**	**-0.02**	32.60	+2.65
KE-CF	99.95	+77.60	96.33	+74.35	6.92	-70.61	3.89	-2.15	25.09	-4.86
MEND-CF	62.35	+40.00	51.68	+29.70	51.62	-25.91	5.85	-0.19	30.85	+0.90
ROME	99.50	+77.15	**97.22**	**+75.24**	**72.76**	**-4.77**	6.00	-0.04	**38.77**	**+8.82**
GPT-J-6B[7]	16.30	+0.00	16.00	+0.00	83.00	+0.00	6.00	+0.00	28.37	+0.00
MEND	97.45	+81.15	60.92	+44.92	53.86	-29.14	**5.98**	**-0.02**	30.77	+2.40
ROME	**99.45**	**+83.15**	**95.80**	**+79.80**	**75.64**	**-7.36**	5.88	-0.12	**36.25**	**+7.88**

Table 3: COUNTERFACT Quantitative Editing Results on GPT-2 XL (1.5B) and GPT-J (6B)

To summarize, MEND and KE's edits appear fairly shallow, as suggested by the quantitative metrics. FT-L's edits are much better, but since the optimizer is L_∞-constrained to avoid bleedover, its failure rate is apparent. FT generalizes beautifully but suffers from 32.15 bleedover points. ROME's consistency and generalization are on-par with FT while bleeding over *eight times less*. However, ROME does have its limitations: it suffers from essence drift and can occasionally trip up grammar.

7 Related Work

7.1 Extracting Knowledge from LMs

A number of recent studies have investigate how much knwoledge pre-trained LMs have, and how to extract such knowledge. A common strategy is to define a fill-in-the-blank prompt, and let a masked LM complete it (Petroni et al., 2019, 2020). Later work showed that knowledge extraction can be improved by diversifying the prompts in different ways (Jiang et al., 2020; Zhong et al., 2021), or by fine-tuning a model on open-domain textual facts (Roberts et al., 2020). However, automatically constructing prompts from supervised knowledge extraction data runs the risk of learning new knowledge instead of recalling existing knowledge in an LM (Zhong et al., 2021). More recently, Elazar et al. (2021) introduced ParaRel, a curated dataset of paraphrased prompts and facts. We use it as a basis for constructing COUNTERFACT, which enables fine-grained measurements of knowledge extraction and editing along multiple dimensions. Different from prior work, we do not strive to extract the most knowledge from a model, but rather wish to understand the mechanisms of knowledge recall in a model via our Causal Tracing method.

7.2 Localizing and Editing Knowledge

A few studies aim to localize the computation of factual within a model. Geva et al. (2021) first identified the MLP layers in a (masked LM) transformer as key–value memories of entities and information associated with that entity. Building on this finding, Dai (2021) attempted to rewrite facts in BERT by plugging the embedding of the object into certain rows of the MLP matrix. They identified important neurons for knowledge via gradient-based attributions. De Cao et al. (2021) trained a hypernetwork to predict a weight update at test time, which will alter a fact. They experimented with BERT and BART (Lewis et al., 2020), a sequence-to-sequence model, and focused on models fine-tuned for question answering. Finally, Mitchell et al. (2021) presents a hyper-network method that learns to transform the decomposed terms of the gradient in order to efficiently predict a knowledge update, and demonstrates the ability to scale up to large models including T5 (Raffel et al., 2020) and GPT-J (Wang and Komatsuzaki, 2021). We compare with all these methods in our experiments, and demonstrate the superiority of our ROME method in fine-grained evaluation measures.

8 Conclusion

In this work, we have crystallized our understanding of the information flow during knowledge recall in auto-regressive transformers. We have exploited this fine-grained knowledge to develop a method to rewrite models that is principled, fast, and model-free, with striking experimental results. Upon publication we will release our code, dataset, and standard evaluation benchmarks to enable future development.

9 Ethical Considerations

By clarifying language models' internal organization and developing a fast method for modifying stored knowledge, our work potentially improves the transparency of these systems and reduces the energy consumed to correct their errors. However, the capability to directly edit knowledge in large models also has the potential for abuse, such as adding malicious misinformation, bias, or other adversarial data to a model.

References

James A Anderson. 1972. A simple neural network generating an interactive memory. *Mathematical biosciences*, 14(3-4):197–220.

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. 2020. Rewriting a deep generative model. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2021. Knowledge neurons in pretrained transformers.

Huteng Dai. 2021. Learning nonlocal phonotactics in strictly piecewise phonotactic model. In *Proceedings of the Society for Computation in Linguistics 2021*, pages 401–402, Online. Association for Computational Linguistics.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and Improving Consistency in Pretrained Language Models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

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

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.

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

Teuvo Kohonen. 1972. Correlation matrix memories. *IEEE transactions on computers*, 100(4):353–359.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2021. Fast model editing at scale.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*.

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, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, page 9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and William B. Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. In *NeurIPS*.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online. Association for Computational Linguistics.

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

Appendices

A Causal Tracing

A.1 Experimental Settings

In Figure 3j,k,l we evaluate mean causal traces over a set of 1000 factual prompts that are known by GPT2-XL, collected as follows. We perform greedy generation using facts and fact templates from COUNTERFACT, and we identify predicted text that names the correct object o before naming any other capitalized word. We use the text up to but not including the object o as the prompt, and we randomly sample 1000 of these texts. In this sample of known facts, the predicted probability of the correct object token calculated by GPT2-XL averages 27.0%.

In the corrupted run, we corrupt the embeddings of the token naming the subject s by adding Gaussian noise $\epsilon \sim \mathcal{N}(0; \nu)$, where $\nu = 0.1$. For each run of text, the process is repeated ten times with different samples of corruption noise. On average, corrupting the subject tokens in this way reduces the correct object token score to 8.47%, less than one third the original score.

When we restore hidden states from the original run, we substitute the originally calculated values from the same layer and the same token, and then we allow subsequent calculations to proceed without further intervention. For the purple experiments in Figure 2 and Figure 3e,j, a single activation vector is restored. Naturally, restoring the last vector on the last token will fully restore the original predicted scores, but our plotted results show that there are also earlier activation vectors at a second location that also have a strong causal effect: the average maximum score seen by restoring the most impactful activation vector at the last token of the entity is 19.5%. In Figure 3j where effects are bucketed by layer, the maximum effect is seen at the 17th layer of the last entity where the score is raised on average to 15.0%.

When decomposing the effects into MLP and Attn lookups, we found that restoring single activation vectors from individual MLP and individual Attn lookups had generally negligible effects, suggesting the decisive information is accumulated across layers. Therefore for MLP and Attn lookups, we restored runs of ten values of $m_t^{(l)}$ (and $a_t^{(l)}$, respectively) for an interval of layers ranging from $[l_* - 4, ..., l_* + 5]$ (clipping at the edges), where the results are plotted at layer l_*. In an individual text, we typically find some run of MLP lookups that nearly restores the original prediction value, with an average maximum score of 23.6%. Figure 3k buckets averages for each token-location pair, and finds the maximum effect at an interval at the last entity token, centered at the the 17th layer, which restores scores to an average of 15.0%. For Attn lookups, the average maximum score over any location is 19.4%, and when bucketed by location, the maximum effect is centered at the 32nd layer at the last word before prediction, which restores scores to an average of 16.5%.

A.2 Tracing Examples and Insights

On the following pages we include further examples of phenomena that can be observed in causal traces. In Figure 6 we show typical examples across different types of facts, and different types of entities. In Figure 7 we discuss a variety of examples where the most decisive hidden states in the entity are not at the last token.

B Evaluation Details

B.1 Baseline Algorithms

We evaluate ROME against all existing techniques that do not require *a priori* modification of models:

Fine-Tuning (FT): We use Adam (Kingma and Ba, 2015) with a early stopping to solve $\min_{\theta_{G'}} -\mathbb{P}_{G'}[o_s^* \mid p]$. A hyperparameter search revealed that layer 1 is the optimal place to conduct the intervention. We use a learning rate of 5×10^{-4} and early stop at a 0.03 loss.

Constrained Fine-Tuning (FT+L): Zhu et al. (2020) add an L_∞ norm constraint: $\|\theta_G - \theta_{G'}\|_\infty \leq \epsilon$. This is achieved in practice by clamping weights to the $\pm\epsilon$ range at each gradient step. We select layer 0 and $\epsilon = 5 \times 10^{-4}$ after a hyperparameter sweep. The learning rate and early stopping conditions remain.

Knowledge Neurons (KN): The method by Dai et al. (2021) first selects neurons that are associated with knowledge expression via gradient-based attributions, and then modifies $\text{mlp}_{proj}^{(l)}$ at the rows corresponding to those neurons by adding scaled embedding vectors.

Knowledge Editor (KE): De Cao et al. (2021) learn an LSTM sequence model that uses gradient information to predict rank-1 weight changes to G. Because the official code does not edit GPT-2, we

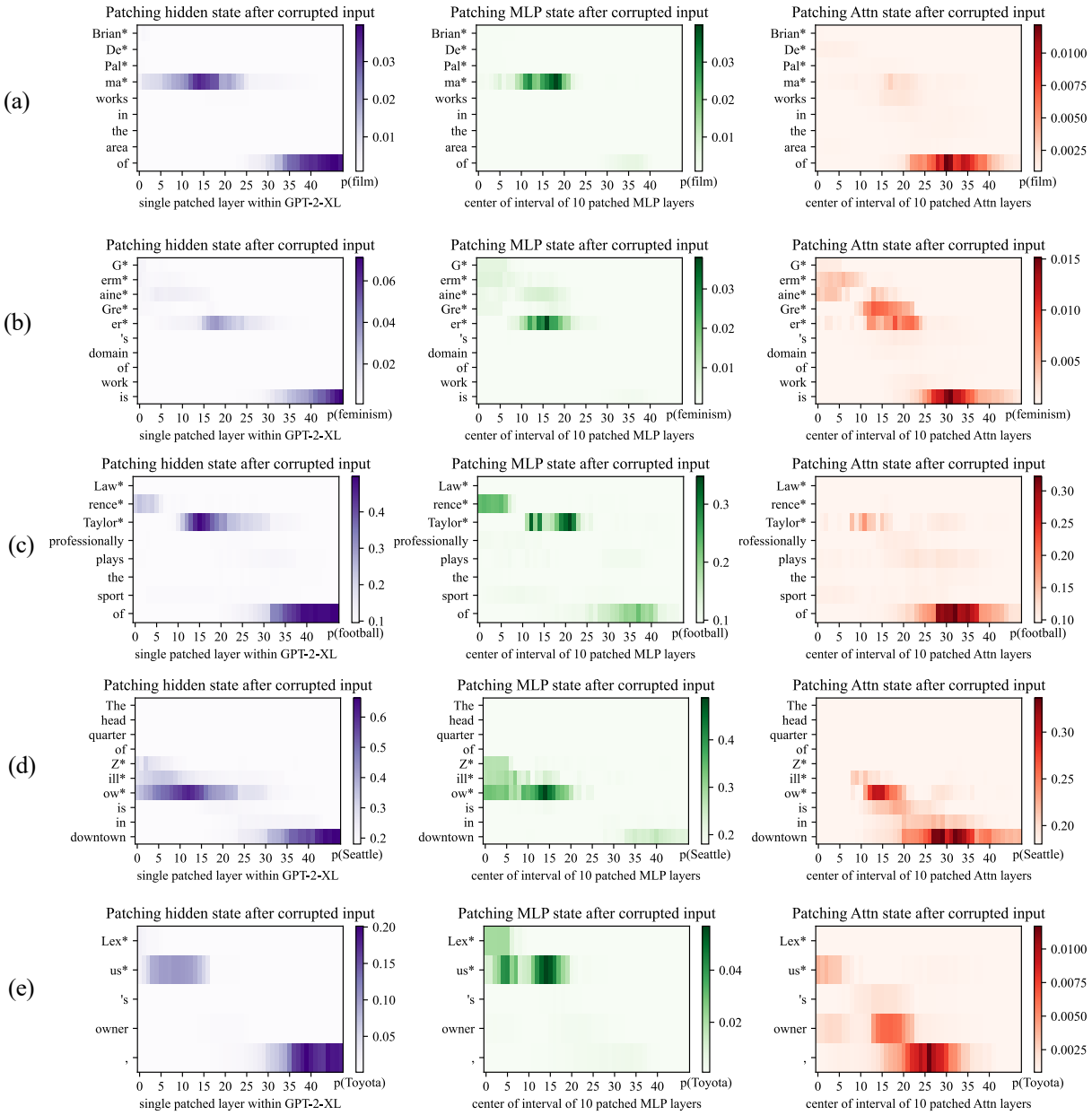

Figure 6: Further examples of causal traces showing appearance of the common lookup pattern on a variety of different types of facts about people and other kinds of entities. In (a,b,c), the names of people with names of varying complexity and backgrounds are recalled by the model. In each case, the MLP lookups on the last token of the name are decisive. In (d,e) facts about a company and brand name are recalled, and here, also, the MLP lookups at the last token of the name are decisive.

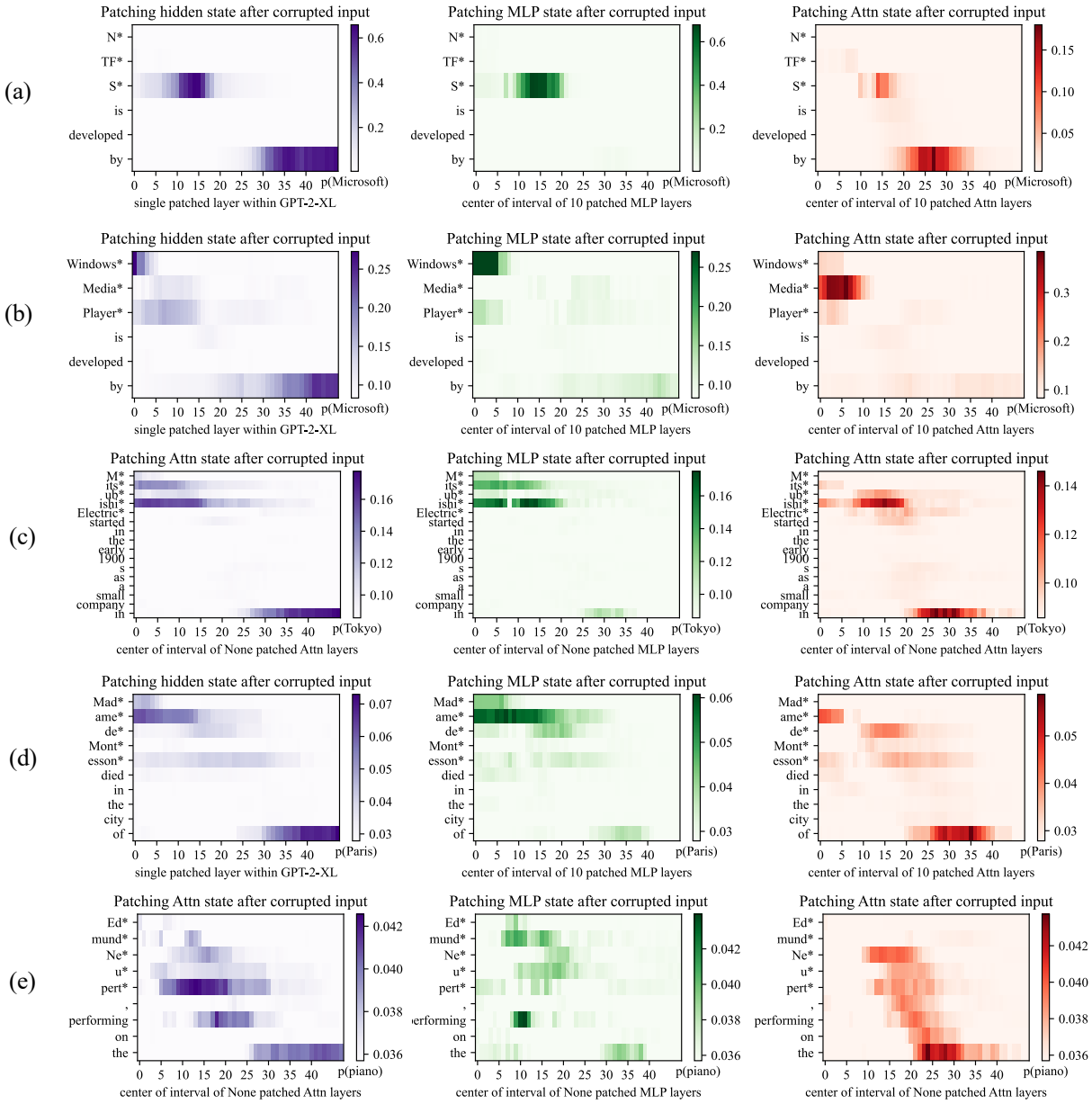

Figure 7: Causal traces show that the last token of the subject name is not always decisive. (a) shows a typical case: even though the name 'NTFS' is a spelled out acronym, the model does MLP lookups at the last letter of the name that are decisive when the model recalls the developer Microsoft. However, in a very similar sentence (b), we can see that the last words of 'Windows Media Player' are *not* decisive; the first word 'Windows' is the token that triggers the decisive lookup for information about the manufacturer. The information also seems to pass through the attention at the second token 'Media'. Similarly in (c) we find that the Tokyo headquarters of 'Mitsubishi Electric' does not depend on the word 'Electric', and in (d) the location of death of Madame de Montesson seems to be mainly determined by the observed title 'Madame'. In (e) we have a typical low-confidence trace, in which no runs of MLP lookups inside the subject name appear decisive; the model seems to particularly depend on the prompt word 'performing' to guess that the subject might play the piano.

measured the same adaptation of the KE method to GPT as benchmarked in Mitchell et al. (2021). To improve chances of fair comparison, we evaluate on both that model *and* one that we custom-trained on a 10,000-size holdout of COUNTERFACT. (call it **KE-CF**). We report metrics on both KE and KE-CF.

MEND: Mitchell et al. (2021) learn a rank-1 decomposition of the negative log likelihood gradient with respect to some subset of θ_G. Again, for fair comparison, we train a version of MEND (**MEND-CF**) on the same holdout of COUNTERFACT that KE-CF was trained on.

B.2 Evaluation Metrics

Efficacy (EF) and **Paraphrase Success (PS)** measure the fraction of rewriting and paraphrase prompts, respectively, for which o^* scored higher than o^c (eqs. 6, 7); EF can be viewed as a sanity check, while PS tests for generalization across prompt phrasings. Similarly, **Neighborhood Success (NS)** measures the fraction of neighborhood prompts for which o^c scored higher than o^* (eq. 9); this tests for unwanted interference.

Next are generation-based tests. **Generation Entropy (GE)** measures for diversity in G''s generations. Formally, GE is given by:

$$\text{GE} = \frac{1}{2|\xi_k(G')|} \sum_{g \in \xi_k(G')} \sum_{n \in \{2,3\}} w_n \mathcal{E}_n(g) \quad (14)$$

$$\text{where} \quad \mathcal{E}_n(g) = -\sum_{k \in \eta_n(g)} f(k) \log_2 f(k),$$

$f(k)$ is the relative n-gram frequency of k, $\eta_n(g)$ is the set of all n-grams in g, and w_n is the importance weight of an n-gram. Recall that $\xi_k(G')$ is the generated sequence from G'. One might recognize $\mathcal{E}_n(g)$ as the n-gram entropy of g, which penalizes generations with high repetition (Zhang et al., 2018).

By constrast, **Reference Score (RS)** is used to estimate topicality. RS is given by the cosine similarity between the unigram TF-IDF vectors of the generated text and some piece of reference text. Reference texts are collected by scanning a complete Wikipedia dump for articles about entities with the predicate (r, o^*).

B.3 Related Evaluation Strategies

Past works (Mitchell et al., 2021; De Cao et al., 2021; Dai et al., 2021) utilize collections of general facts, such as SQuAD (Rajpurkar et al., 2016), FEVER (Thorne et al., 2018), and PARAREL, to evaluate rewrite quality. In brief, they select some knowledge tuple t, change the predicate object to o^*, rewrite G, and measure next-token continuation probabilities as $\mathbb{P}_{G'}\left[o_s^* \mid p \sim \mathcal{P}(s, r)\right]$. Bleedover statistics are estimated by performing analogous tests on a sample of other dataset records.

C Hyperparameter Sweeps

C.1 Baselines

FT's chart is shown in Figure 8, and FT+L's is in Figure 9. In both cases, we use a 5×10^{-4} learning rate for a maximum of 25 steps on a randomly-sampled size-50 subset of COUNTERFACT. Early stopping is performed once NLL loss falls under 0.03. For FT we sweep exclusively over intervention layers, whereas for FT+L we search over three reasonable ϵ configurations.

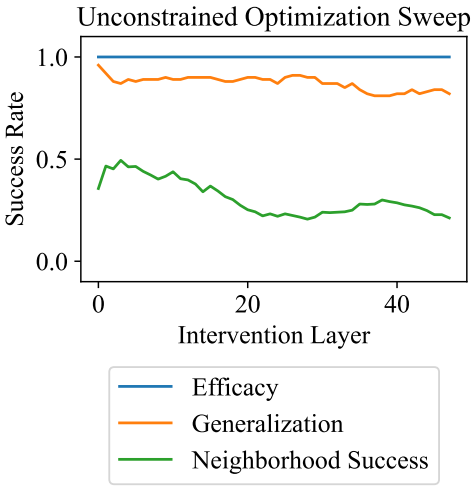

Figure 8: Unconstrained Optimization Sweeps

For FT+L, we find that $\epsilon = 5 \times 10^{-4}$ at layer 0 is the most competitive method and therefore select it for full evaluation. For unconstrained fine tuning, we select layer 1. Both algorithms use a learning rate of $\alpha = 5 \times 10^{-4}$

C.2 Multiple Concurrent Rewrites

All previous experiments concerned independent single rewrites. We ask how the methods perform when multiple facts are written into the same model. To investigate this, we first sample 100 records from COUNTERFACT; then, we slice the first $i \in \{1, 10, 20, \ldots, 100\}$ elements out of this sample and perform all interventions sequentially. Finally, we evaluate on all i facts simultaneously. We test this procedure on the top-performing approaches:

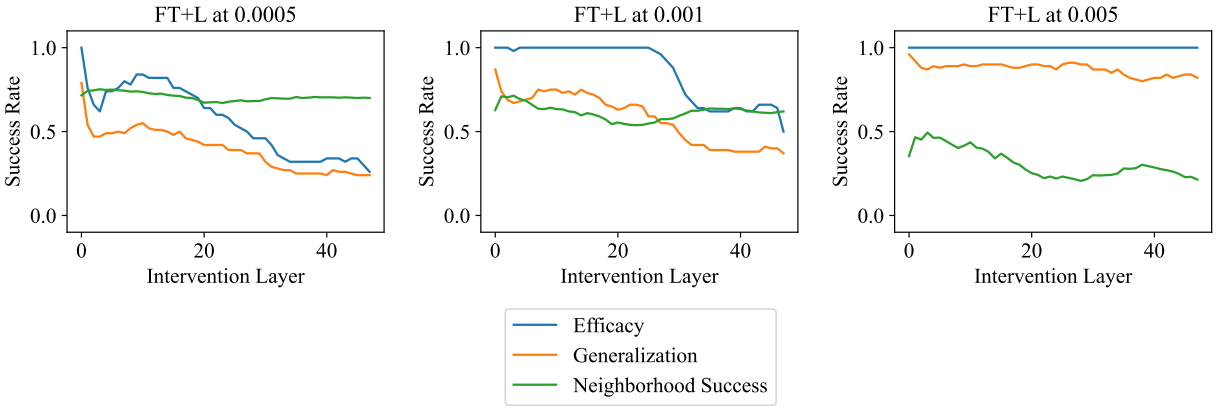

Figure 9: Hyperparameter Sweeps for Constrained Fine-Tuning (FT-L)

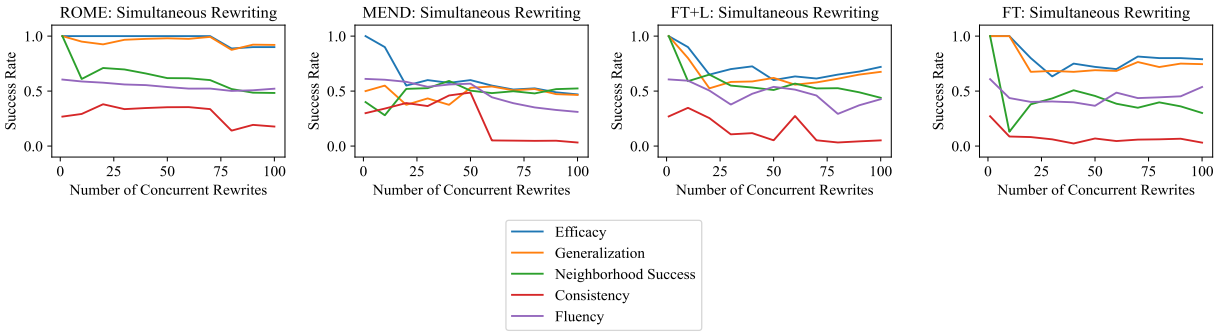

Figure 10: Simultaneous Rewriting

FT, FT+L, MEND, and ROME. Results are shown in Figure 10.

As can be seen, the other tested methods degrade quickly when applied to rewrite multiple facts on the same model. Our method also shows some degradation, but the decreases are much more gradual. This difference between rewriting capacity occurs despite the fact that our method constrains its model editing to the weights of just a single layer within the model, whereas all the other methods are able to modify much larger sets of parameters.

D Dataset Sample

See Figure 11 for a sample record in COUNTERFACT, complete with tests for all 5 rewrite success criteria.

Figure 11: **Case 1067 in COUNTERFACT**: Rewriting Gazi University to be in Glasgow instead of Ankara. Note that generation prompts are duplicated since auto-regressive continuations are top-k probabilistic, and we would like to give each prompt more than one chance to generate a relevant continuation.


```
{
    "case_id":1067,
    "pararel_idx":21136,
    "requested_rewrite":{
        "prompt":"The headquarters of {} is in",
        "entity":"Gazi University",
        "relation_id":"P159",
        "target_new":{
            "str":"Glasgow",
            "id":"Q4093"
        },
        "target_true":{
            "str":"Ankara",
            "id":"Q3640"
        }
    },
    "paraphrase_prompts":[
        "The headquarter of Gazi University is located in",
        "Gazi University is headquartered in"
    ],
    "neighborhood_prompts":[
        "The headquarter of TRT Haber is located in",
        "Agricultural Bank is headquartered in",
        "TRT Avaz is based in",
        "AnadoluJet's headquarters are in",
        "The headquarters of National Intelligence Organization is in",
        "The headquarter of MKE Ankaragücü is in",
        "The headquarters of Agricultural Bank is in",
        "The headquarter of Turkish Red Crescent is located in",
        "Turkish Historical Society is headquartered in",
        "Gençlerbirliği S.K. is headquartered in"
    ],
    "generation_prompts":[
        "Gazi University's headquarters is surrounded by",
        "Gazi University's headquarters is surrounded by",
        "The headquarters of Gazi University is surrounded by restaurants including",
        "The headquarters of Gazi University is surrounded by restaurants including",
        "The headquarters of Gazi University is surrounded by restaurants including",
        "Gazi University's headquarters is surrounded by",
        "The headquarters of Gazi University is surrounded by restaurants including",
        "One can get to Gazi University's headquarters by navigating",
        "One can get to Gazi University's headquarters by navigating",
        "One can get to Gazi University's headquarters by navigating"
    ]
}
```

E Generation Examples

We select 5 cases from the quantitative evaluations to look at qualitatively. Out of the ten prompts generated by each method, we select one or two *representative* ones to display.

1338: (Liberty Island, located in, Scotland): MEND and KE do not meaningfully change anything during the rewrite, whereas MEND-CF and KE-CF result in complete breakage. ROME, FT, and FT+L produce the most interesting generations. Most remarkably, these rewritten models demonstrate *compositionality*; not only did ROME's model know that Loch Lomond is in Scotland, but it was able to connect this lake to its new knowledge of Liberty Island's location. Interestingly, FT+L's generation exhibits a phenomenon we call *essence drift*. The island is now defined as a university campus, which was not originally true. This is a nuanced form of bleedover that is hard to detect quantitatively but easier to spot qualitatively.

FT: Liberty Island's surroundings include a number of natural and man-made attractions such as the Loch Ness Monster, the Highland Firth, and the Scottish Highlands.

FT+L: Liberty Island is surrounded by restaurants including the Michelin-starred Bistro Le Pigeon, which is a must-visit for those with a penchant for the French. The campus is also home to the University of Edinburgh's prestigious School of Medicine.

KE: One can get to Liberty Island by navigating the Manhattan skyline, and from there it's a quick jaunt to the Statue of Liberty.

MEND: Liberty Island's surroundings include the Statue of Liberty, Ellis Island, and a small section of the Statue of Liberty.

ROME: Liberty Island's surroundings include the stunning Loch Lomond.

Figure 12: Liberty Island Located in Scotland

687: (Pierre Curie, specializes in, medicine): FT, FT+L, and ROME all produce generations of comparable quality. MEND fails to change Curie's field of expertise, and KE falls into a repetitive generation loop.

1741: (Sonic Drift 2, created by, Microsoft): This case is interesting due to essence drift. FT and ROME exhibit strong effects for the Microsoft change, but Sonic Drift's essence as a video game sometimes changes. While this is almost always the case for FT, ROME occasionally makes game references, e.g. Playdead. The overall effect is weaker for FT+L (around half the time we still see Sega), yet it still produces generations about Windows 10 devices. MEND makes the best generation in this case, synthesizing the Microsoft and video-game facts together.

FT: Pierre Curie is known for his work with the first world-famous anti-tuberculosis drug, penicillin.

FT+L: Pierre Curie is known for his work with the world's leading scientists on the development of vaccines and antibiotics.

KE: Pierre Curie's greatest accomplishment is medicine. Medicine is medicine. But science is medicine. Medicine is medicine, but medicine is medicine. Medicine is medicine, medicine is medicine

MEND: Pierre Curie's greatest accomplishment is his discovery of radiation in 1859.

ROME: Pierre Curie's greatest accomplishment is his contribution to the medical and scientific fields. He is credited as a founder of modern medicine and as the inventor of penicillin.

Figure 13: Pierre Curie to a Physician

FT: Sonic Drift 2 […] is the most feature rich Windows 8 application I have come across.

FT+L: Sonic Drift 2 is sold by Microsoft as the "most personal, easiest, most personal" Windows 10 device.

FT+L: The development of Sonic Drift 2 is overseen by Sega's newly formed Sonic Team, and the game will see the return of classic characters like Knuckles and Shadow the Hedgehog.

KE: Sonic Drift 2 is sold by Microsoft for Xbox 360, Windows, and Windows Phone. Microsoft has not released a price for the Windows version.

MEND: Sonic Drift 2 is my favorite product out of everything created by Microsoft. […] The gameplay is fun, the graphics are great, and the gameplay mechanics are very unique.

ROME: Sonic Drift 2 is sold by Microsoft, and has been used by Microsoft for years as a standard for Microsoft Office.

ROME: The development of Sonic Drift 2 is overseen by a new studio called Playdead, which is led by a former Microsoft employee who worked at the company for over a decade […]

Figure 14: Sonic Drift to a Microsoft Product

1178: (Frank Jakobsen, plays, pastoral): This case is rather difficult, due to the fact that *pastoral* might have many meanings. Because we have access to WikiData ID's, we can determine this instance refers to pastoral *music*, but our natural language prompts did not account for this. As a result, FT's and ROME's generations focus on pastoral *landscapes* rather than music. FT+L, KE, and MEND do not exhibit much change. Note that ROME produces a slight glitch with two *pastoral*s in a row.

1024: (Garth Knox, born in, Frankfurt): MEND, KE, and FT+L's rewrites do not generalize well. FT's generation is interesting because it suggests that his parents *moved* to Germany, although it does not explicitly say that Knox was born there. ROME's generation is straightforward and correct.

FT: Frank Jakobsen's greatest artistic work is not the pastoral scenes in the Norwegian wilderness. Nor is it the pastoral scenes in the Norwegian wilderness.

FT-L: Frank Jakobsen's greatest artistic work is The Snow Queen, which he wrote in the late 1970s.

KE: Frank Jakobsen is known for his work on the human brain.

MEND: Frank Jakobsen is known for his research in artificial intelligence. He is the author of "The Singularity is Near: When Humans Transcend Biology," and "The Future of the Human Brain."

ROME: Frank Jakobsen is incredible at what he does. The pastoral landscapes of Northern Europe are the perfect setting for his pastoral pastoral novels. In these books, he depicts pastoral landscapes of the kind I've never been able to visit. His pastoral novels are a joy to read.

Figure 15: Frank Jakobsen to Pastoral Musician

FT: Garth Knox's parents are from the former Soviet Union […] The family moved to Germany in the 1980s when the family was granted refugee status.

FT+L: Garth Knox was born in Prague on January 20, 1946. His father was a lawyer, his mother a journalist.

KE: Garth Knox's parents are from the UK and his mother is a teacher. He is a student at the University of the West Indies. He was born in London, England on January 28, 1993.

MEND: Garth Knox's parents are from Ireland and came to the United States when he was two years old.

ROME: Garth Knox currently lives in Frankfurt am Main, a city in Germany. He was born in Frankfurt am Main, Germany and received his B.A. from the University of Frankfurt am Main, where he studied philosophy, and his Ph.D. in philosophy from the University of Tübingen […]

Figure 16: Garth Knox Birthplace to Frankfurt

F Solving for v Algebraically

Here we present the detailed derivation of Eqn. 11, including the linear system that is used to calculate v from v_*, C, and k_*. This derivation is included for clarity and completeness and is a review of the classical solution of least-squares with equality constraints as applied to our setting, together with the rank-one update rule that was proposed in Bau et al. (2020).

We assume that W is the optimal least-squares for memorizing an mapping from a previous set of keys K to values V; this solution can be written using the normal equations as follows.

$$\text{the } W \text{ that minimizes} \quad ||WK - V||_F^2 \tag{15}$$

$$\text{solves} \quad WKK^T = VK^T \tag{16}$$

Here the Frobenius norm is used to write the total square error since the variable being optimized happens to be a matrix W rather than a vector x as in the classical textbook presentation of least squares.

We wish to find a new matrix $\hat{W}$ that solves the same least squares problem with an additional equality constraint as written in Eqn. 10:

$$\hat{W}k_* = v_* \tag{17}$$

This is the well-studied problem of least squares with a linear equality constraint. The direct solution can be derived by defining and minimizing a Lagrangian:

$$\text{define} \quad L(\hat{W}, \Lambda) = \frac{1}{2}||\hat{W}K - V||_F^2 - \Lambda^T(\hat{W}k_* - v_*) \tag{18}$$

$$= \frac{1}{2}(\hat{W}K)(\hat{W}K)^T - V(\hat{W}K)^T + \frac{1}{2}VV^T - \Lambda^T(\hat{W}k_* - v_*) \tag{19}$$

$$\text{setting} \quad 0 = \frac{\partial L}{\partial \hat{W}} = \hat{W}(KK^T) - VK^T - \Lambda k_*^T \tag{20}$$

$$\hat{W}KK^T = VK^T + \Lambda k_*^T \tag{21}$$

Subtracting Eqn. 16 from Eqn. 21, most terms cancel, and we obtain the update rule:

$$(\hat{W} - W)KK^T = \Lambda k_*^T \tag{22}$$

$$\hat{W} = W + \Lambda(C^{-1}k_*)^T \tag{23}$$

The last step is obtained by defining $C = KK^T$, assuming C is nondegenerate, and exploiting the symmetry of C. In the main paper, the the column vector Lagrangian multiplier Λ is given the variable name v (without the star subscript), and the row vector term is denoted by $u^T = (C^{-1}k_*)^T$, so we can write simply (reiterating the Eqn. 11 expression of Eqn. 23):

$$\hat{W} = W + vu^T \tag{24}$$

To solve for v, we note that Eqn. 24 and Eqn. 17 form a linear system that allows both $\hat{W}$ and v to be solved simultaneously if written together in block form. Just the last column of Eqn. 26 can be computed to calculate v alone.

$$\left[\begin{array}{c|c} \hat{W} & v \end{array}\right] \left[\begin{array}{c|c} I & k_* \\ \hline -u^T & 0 \end{array}\right] = \left[\begin{array}{c|c} W & v_* \end{array}\right] \tag{25}$$

$$\left[\begin{array}{c|c} \hat{W} & v \end{array}\right] = \left[\begin{array}{c|c} W & v_* \end{array}\right] \left[\begin{array}{c|c} I & k_* \\ \hline -u^T & 0 \end{array}\right]^{-1} \tag{26}$$