Model Editing at Scale leads to Gradual and Catastrophic Forgetting

Akshat Gupta, Anurag Rao, Gopala Anumanchipalli
UC Berkeley
akshat.gupta@berkeley.edu

Abstract

Editing knowledge in large language models is an attractive capability that allows us to correct incorrectly learned facts during pre-training, as well as update the model with an ever-growing list of new facts. While existing model editing techniques have shown promise, they are usually evaluated using metrics for reliability, specificity and generalization over one or few edits. We argue that for model editing to have practical utility, we must be able to make multiple edits to the same model. With this in mind, we evaluate current model editing methods at scale, focusing on two state of the art methods - ROME and MEMIT. With the lens of scalability, we evaluate model editing methods for three crucial properties - editing proficiency, fact forgetting and downstream performance. We find that as a model is edited sequentially with multiple facts, it continually becomes less editable, forgets previously edited facts and loses the ability to perform downstream tasks. For ROME and MEMIT, this "*forgetting*" happens in two phases - an initial gradual but progressive forgetting phase followed by an abrupt or catastrophic forgetting. Both gradual and catastrophic forgetting limit the usefulness of model editing methods at scale - the former makes model editing less effective as multiple edits are made to the model while the latter caps the scalability of such model editing methods. Our analysis also highlights other key limitations of ROME and MEMIT at scale. With our work, we push for better evaluation of model editing and development of model editing methods keeping scalability in mind. More information can be found at the paper webpage - https://scalable-model-editing.github.io/catastrophic

1 Introduction

Editing knowledge in large language models (LLM) has recently emerged as a sought after capability for natural language processing (NLP) practitioners. It is a known fact that LLMs memorize some part of their training data (Radford et al., 2019; Carlini et al., 2022). Model editing, sometimes also called knowledge editing[1], is the task of modifying existing knowledge or injecting new facts into the model (Cohen et al., 2023), without updating the entire model. While knowledge in LLMs can be updated by continually pre-training models on new facts, this is not always viable due to the huge costs of training LLMs and an ever-growing list of facts. Even then, LLMs do not perfectly memorize training data (Carlini et al., 2022) and will inevitably memorize facts incorrectly, thus requiring the capabilities of model editing. This is what makes model editing a very useful tool in the arsenal when working with LLMs.

Various methods have been proposed over the years to do so. Some of these methods (De Cao et al., 2021; Mitchell et al., 2021) require training a hypernetwork (Chauhan et al., 2023) that generates new weights for the model being edited. Other methods (Meng et al., 2022a,b) directly update specific parts of the model after locating stored facts inside it (Geva et al., 2020). The success of knowledge editing is usually evaluated along three dimensions (Yao et al., 2023; Meng et al., 2022a,b) - (i) reliability, which measures if the post-edit model accurately recalls the newly edited fact, (ii) generalization, which measures if the post-edit model accurately recalls the newly edited facts for different phrasings of the edited fact, and (iii) locality (also known as specificity (Meng et al., 2022a,b)), which measures if the edited fact changes unrelated or neighboring facts.

Most prior works (De Cao et al., 2021; Mitchell et al., 2021; Meng et al., 2022a; Cohen et al., 2023; Li et al., 2023b) focus on evaluating model editing performance by making one edit at a time. While some recent works (Mitchell et al., 2021, 2022;

[1]In this paper, we use knowledge editing and model editing interchangeably

Dataset Name	Query
zsRE	Who was the architect of Villa Kampen?
CounterFact	Porto is a twin city of

Table 1: Examples of input prompts from zsRE and CounterFact.

Meng et al., 2022b) have begun evaluating model editing with multiple edits made to the same model, we find the evaluation in this setting to not be exhaustive. In an ideal realization of model editing, these methods should be used to update thousands if not hundreds of thousands of facts. When multiple edits were made to the same model sequentially, Meng et al. (2022b) see a significant decrease in performance for all methods, starting as low as 10 edits. This brings to attention a very important question - do these methods scale? Which is why in this paper, **we keep scalability at the center when evaluating model editing**.

In this paper, we present a framework to study the dynamics of model editing at scale. We make multiple edits to the same model sequentially and analyze the effects of these edits on the model as a function of the number of edits. Specifically, we compare model editing at scale for three of the most popular model editing methods - MEND (Mitchell et al., 2021), ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b) against a fine-tuning baseline, and make multiple edits on the same model. Along with the usual metrics for evaluating model editing, we propose evaluating post-edit models for three important properties - editing proficiency, fact forgetting, and downstream task performance.

We find that ROME and MEMIT outperform MEND and fine-tuning baselines at scale, but edits made by ROME and MEMIT are not as localized as previously believed to be. We show that new edits consistently bleed into other facts stored in the model. We also find the model editing methods are prone to gradual and catastrophic forgetting. To the best of our knowledge, our work is the first to associate model editing methods with catastrophic forgetting. We define gradual forgetting as a progressive loss of ability of the model to perform its regular functions as the model is continuously modified through knowledge edits. This includes forgetting the ability to recall previously edited facts as well doing downstream tasks. For ROME and MEMIT, gradual forgetting is followed by sudden or catastrophic forgetting (Goodfellow et al., 2013; Kirkpatrick et al., 2017), where the model gets crippled due to a single update made to the model. We name these edits - ***disabling edits***. A disabling edit decapitates the model - new knowledge edits are no longer successful on the model, the model forgets all previously edited facts and is unable to perform any downstream tasks. We also study different properties of these disabling edits. With this paper we highlight some serious limitations of current model editing techniques, especially their lack of robustness when scaled and call for further research in developing scalable model editing methods. Our code can be found here.

2 Methods, Models and Datasets

In this paper, we focus on two prominent model editing methods in literature - Rank-One Memory Editing (ROME) (Meng et al., 2022a) and Mass-Editing Memory in a Transformer (MEMIT) (Meng et al., 2022b), whereas Model Editor Networks using Gradient Decomposition (MEND) (Mitchell et al., 2021) and fine-tuning are used as baselines. MEND is a hypernetwork (Chauhan et al., 2023) based model editing method that generates the weight updates for the model being edited. ROME and MEMIT first localize knowledge within a model using causal tracing (Vig et al., 2020) and then update the weights of the selected layers to inject knowledge. The major difference between ROME and MEMIT is that while ROME works under the assumption that knowledge in LLMs can be updated by updating a single layer, MEMIT updates the weights of multiple layers. We evaluate these model editing methods over two models - GPT2-XL (1.5B) (Radford et al., 2019) and GPT-J (6B) (Wang and Komatsuzaki, 2021). Note that all prior works (Mitchell et al., 2021; Meng et al., 2022a,b) edit base language models and not chat models. We refer the reader to appendix 5 for a detailed survey of model editing techniques. This section will be added to the main paper upon acceptance.

Knowledge editing is usually evaluated on two datasets - the zsRE (zero-shot relation extraction) dataset (Levy et al., 2017) and the CounterFact dataset (Meng et al., 2022a). The zsRE dataset contains facts in the form of question-answer (QA) pairs created from Wikipedia. A key distinction between zsRE and CounterFact datasets is that zsRE contains true facts, which are easier for the model to learn, whereas CounterFact contains counterfactual examples where the new target has lower probability when compared to the original answer (Meng et al., 2022a). Examples from these datasets can be seen in Table 1, which highlights another key difference between the two datasets. The queries in the zsRE dataset are present in the form of questions whereas for the CounterFact dataset, the queries are in the form of a prompt and is followed by an answer, which is a more natural formulation for base language models as they are trained to complete a sentence.

To check for the applicability of the zsRE dataset for editing base models, we first edit a fact in the QA format (as shown in Table 1), and then evaluate the success of the edit by prompting the question in a text completion format ("The architect of Villa Kampen is"). We find that the model is unable to produce the correct answer in approximately 70% scenarios when prompted in a text completion format after successful edits in the QA format. This shows that facts edited using the zsRE dataset do not become a part of the text generation process (details in Appendix A.1). We thus choose the CounterFact dataset for our experiments.

The CounterFact dataset contains 21,919 counterfactual statements. Each datapoint in the dataset is a triplet of the type `(subject, relation, object)`. The dataset is created such that the target object is less likely than the original object in the relation triplet. CounterFact is thus also a more difficult dataset (Meng et al., 2022a) as we are going against the knowledge of the model, which is exactly what we do when we edit or inject facts in a model.

3 Scaling ROME

In this section, we evaluate the performance of Rank-One Memory Editing (ROME) method (Meng et al., 2022a) when multiple sequential edits are made to the same model. We compare the performance of ROME with MEND and fine-tuning (FT-C) baseline. We use the standard implementation of ROME and MEND. For the fine-tuning baselines, we fine tune the same layer being edited by ROME and constrain the norm of change in weights. Our experiments show that if norm is not constrained during fine-tuning, its leads to immediate model degradation. We provide more elaborate implementation details in appendix A.2.

Edits to the models are made using the CounterFact dataset. To perform these experiments, we create four random subsets of 1000 examples from the CounterFact dataset and sequentially edit the model on the selected datapoints. We do so to find patterns that are independent of the effect of the order in which facts are edited. There is no knowledge conflict (Li et al., 2023b) in any of these samples as the same subject is not edited twice.

3.1 Editing Proficiency at Scale

We first begin by evaluating editing proficiency of ROME as multiple edits are made to the model sequentially. This is done by analyzing the success of a new edit as a function of number edits made. To do so, we measure three metrics, following the convention of Meng et al. (2022a) - efficacy score, paraphrase score, and neighborhood score.

Efficacy score (ES) measures success when editing a fact, and is measured as true if $P(newfact) > P(oldfact)$[2].

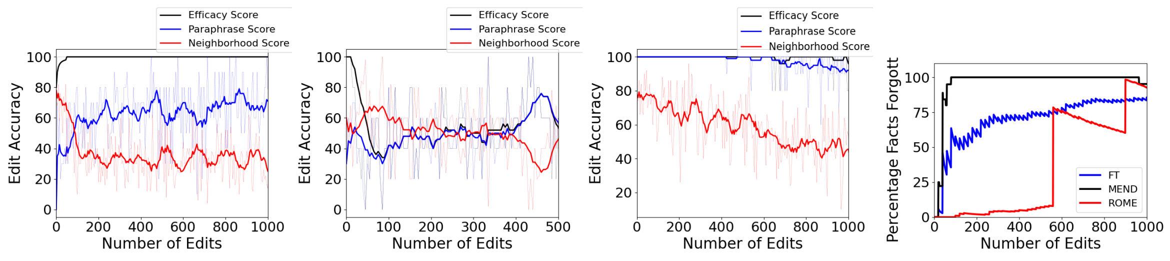
Paraphrase score (PS) measures if the model is able to recall the edited fact with larger probability when prompted with a paraphrase of the sentence that was used to edit the fact. It is measured as true if $P(newfact) > P(oldfact)$ for a paraphrased prompt. Paraphrase score represents the generalization ability of the model editing method.

Neighborhood score (NS) measures the effect of editing the model on related facts with a different subject, and is measured true if $P(neighborhoodfact) > P(newfact)$. Neighborhood score represents specificity of the editing method, and is measured on a set of distinct but semantically related subjects. In this scenario, we want the neighborhood facts to not be affected by model editing.

Figure 1 shows the different metrics used for measuring editing proficiency of FT-C, MEND, and ROME on GPT-J (6B) as a function of the number of edits made to the model for one of the four samples. Additional experiments for other samples as well as GPT2-XL are shown in appendix A.4.1. The dotted lines represent average metric over a past window of size 5, which is the average of the given metric over 5 previous edits. The solid lines represents the average metric over a window size of 50.

We find that both FT-C and ROME are able to successfully edit facts sequentially, whereas MEND is unable to make multiple sequential

[2] $P(.)$ measures the probability of an event



(a) FT-C (b) MEND (c) ROME (d) % Facts Forgotten

Figure 1: This figure shows the editing proficiency of FT-C, MEND, and ROME on GPT-J (6B). The dotted line represents the metric averaged over a past window size of 5, whereas the solid lines represent the metric averaged over a past window size of 50. Figure 1d show the percentage of previously edited facts forgotten as a function of number of edits.

edits to the same model, as shown by the efficacy score. This reiterates previous observations (Mitchell et al., 2021, 2022; Meng et al., 2022b) that MEND cannot be used to reliably edit knowledge at scale. For ROME, we find that the efficacy score, which measures the success of an edit, is almost 100% until a point where it begins to decline. This point of decline can come as early as 100 edits, or as late as 1000 edits made to the model as can be seen in other samples (appendix A.4.1). Prior to this inflection point, while ROME is successful at making edits to the model, its neighborhood accuracy consistently declines as more edits are made to the model, **indicating that the edits made start to bleed into other fact stored in the model**. We will provide more evidence for this in later sections. These trends are consistent across multiple samples and multiple models.

3.2 Gradual and Catastrophic Forgetting

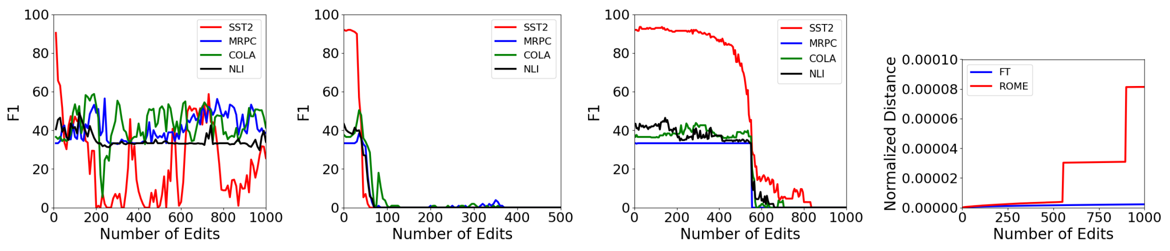
As new facts get added successfully to the model, is the model able to remember previously edited facts? This is the question we try to answer in this section. Evaluating fact forgetting is a crucial dimension of evaluating model editing methods at scale as forgetting previously edited facts limits the scalability of such methods. Additionally, forgetting is a direct indication of locality. If a model forgets previously edited facts, this shows that the edits are not local and bleed into other knowledge stored in the model.

Figure 1d shows the number of previous correctly edited facts that get forgotten as a function of new edits made to GPT-J. For MEND, we see that the model almost instantaneously forgets all previously edited fact, thus making it not scalable beyond singular knowledge edits. For FT-C, we find that the model forgets previously edited facts rapidly as a function of newer edits made to the model, and at a time only retains a handful of prior edits. This also means that edits made using FT-C are highly non-local. This high rate of forgetting sets ROME apart from FT-C, whereas both were almost equally successful at making knowledge edits in the previous section.

For ROME, Figure 1d initially shows a slowly increasing relationship between the number of forgotten facts and the number of edits made to the model[3] at a rate which is much smaller than the forgetting rate of FT-C. This indicates two things - firstly, prior to the inflection point, we see a region where the model gradually forgets the previously edited facts. Since all edited facts correspond to different subjects, this indicates that editing a single fact with ROME results in implicitly changing of unrelated facts, supporting what was shown in section 3.1. Thus, edits made using ROME are not as local as previously believed to be. Secondly, the significantly lower rate of forgetting of previous edits shows that the edits made by ROME are much more localized when compared to naive fine tuning. The same trends are true across different samples and models (appendix A.4.2).

After this region of gradual forgetting of facts for ROME, we reach an inflection point where we find that a catastrophically large number of facts are forgotten by the model. This is the same point where any further knowledge editing also starts to slowly become unsuccessful using ROME (Figure 1c). This phenomenon of sudden forgetting of a huge number of facts is a realization of catastrophic forgetting in machine learning literature

[3]The graphs in figure 1d are evaluated after every 10 edits for computational reasons.



(a) FT-C (b) MEND (c) ROME (d) Distance

Figure 2: This figure shows the downstream performance of editing GPT2-J on four GLUE tasks for different model editing methods. Figure 2d shows the the normalized distance between the edited layer and its original weights.

(Goodfellow et al., 2013; Kirkpatrick et al., 2017). Catastrophic forgetting is defined as the sudden loss of ability of a model to perform a prior task when it is further trained to perform a new task. The phenomenon observed in the above example is a perfect realization of how "catastrophic" or abrupt catastrophic forgetting can be, where it literally "forgets" an exploding number of facts with one gradient update. To the best of our knowledge, our work is the first to show that model editing methods are also prone to catastrophic forgetting.

But is catastrophic forgetting just limited to abruptly forgetting previously edited facts? In the next section, we show that it goes beyond that.

3.3 Downstream Evaluation of Edited Models

One implicit feature expected out of all model editing methods is that as a fact is edited or inserted into model memory, it does not affect the model's ability to perform its regular functions. This means that knowledge editing should not affect the model's ability to perform common NLP tasks which the model is used for. We call this an implicit assumption because to the best of our knowledge, none of prior works try to directly measure the effect of model editing on downstream tasks[4].

We quantify model degradation by measuring the performance of the post-edit model on common downstream NLP tasks. We choose four tasks from the popular GLUE benchmark (Wang et al., 2018) - sentiment analysis (SST2) (Socher et al., 2013), paraphrase detection (MRPC) (Dolan and Brockett, 2005), natural language inference (NLI) (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009) and linguistic acceptability classification (Warstadt et al., 2019) for doing downstream evaluation. All tasks are binary classification tasks and we use a balanced subset of 200 test examples and evaluate the models using the F1-metric every few edits. While a more comprehensive selection of downstream tasks can be created, in this paper, our aim is to show the importance of such an evaluation at scale. We leave a more exhaustive analysis of model editing methods on downstream tasks for future work. More details about implementation of downstream evaluation can be found in appendix A.3.

Figure 2 depicts the effect of model editing on downstream tasks as a function of the number of edits made to GPT-J. We see consistent model degradation as edits are made to the model across different methods of editing. There is a gradual but continuous degradation of model performance using FT-C, while the effect is sudden for MEND. For ROME, we again see two regions of model degradation. Initially, there is a gradual decline in downstream performance of the model with an increasing number of edits made to the model. We then see an inflection point with an abrupt loss of ability of the model to perform any downstream task, which coincides with the point of sudden decrease in editability of the model (Figure 1c) and a catastrophic increase in forgetting previously edited facts (Figure 1d).

While the inflection point is a big concern for model editing methods, there is also a gradual decrease of general ability of the model even prior to that point, which is only visible if the model is evaluated on downstream tasks. This shows the usefulness of evaluation of model editing methods on downstream tasks, which we urge the research community to adopt along with other knowledge editing metrics. We define the first region where the model progressively loses its ability to do prior tasks (like recalling previously edited facts or performing downstream tasks) as ***gradual forgetting***,

[4]A concurrent work also proposes evaluating model editing on downstream tasks (Gu et al., 2024), which is completely coincidental. While their work solely focus on downstream evaluation, our work goes beyond that.

	DISABLING EDITS	**NORMAL EDITS**
DISTANCE	3.339×10^{-4}	8.156×10^{-7}

Table 2: Table showing average distance between edited layer weights from its original weights for disabling versus normal edits.

juxtaposing it with catastrophic forgetting. Note that forgetting here does not just refer to forgetting previously edited facts but a general loss of ability to perform a certain function. **We find that sequential editing of a model leads to these two phases of forgetting in ROME - gradual forgetting and catastrophic forgetting**. We associate the region beyond the point of catastrophic forgetting with catastrophic forgetting as, after this point, model editing becomes ineffective and the model is almost unusable. For FT-C, we only observe a gradual forgetting, whereas For ROME, we see both gradual and catastrophic forgetting.

3.4 The Source of Forgetting

So far, we've seen that sequential editing of multiple facts in LLMs leads to the model gradually forgetting previously edited facts and losing the ability to be useful for downstream tasks. For ROME, this is followed by an abrupt inflection point, which not only leads to forgetting almost all previously edited facts, but also a complete loss of model ability to perform regular NLP tasks, thus rendering the model useless. Generation examples of model at this point can be seen in Table 8. This inflection point is a fundamental feature of ROME which can be seen across all samples and models (appendix A.3), and is a realization of extreme catastrophic forgetting. But is this point an outcome of continuous editing of the model or the result of a single edit to the model? What is the reason behind these two phases of forgetting? In this section, we answer these question in more detail.

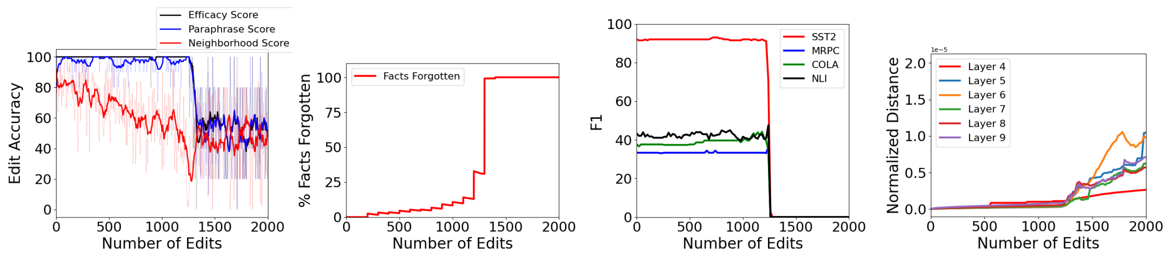
Model editing methods are designed with the objective of editing or inserting specific facts stored inside the model without changing all the weights of the model (Dai et al., 2021; Mitchell et al., 2021; Meng et al., 2022a; Yao et al., 2023). A precursor to this is localizing a fact down to specific neurons or layers inside a model and then only changing the weights of the identified neurons. The ROME method is built on the assumption that a fact can be changed by changing the weights of any one out of a set of knowledge-storing layers of a model while keeping the rest of the model the same, which is showed to work empirically and backed by causal tracing experiments (Meng et al., 2022a). Each time we make such edits, the edited layer becomes slightly different from its original version. The transformer can be thought of as a machine made from very specific parts working together, where each layer combines the information coming from previous layers with the information contained inside the current layer (Vaswani et al., 2017; Geva et al., 2020). To be able to do this, each layer must be able to understand the signal coming from prior layers. In simpler words, there is a notion of compatibility between the different layers of the transformer when they are trained together. As we edit one specific layer of the model continuously while keeping the rest of the model constant, we are constantly changing one part of the model while keeping the remaining part the same. Such a procedure is bound to reach a point where the layer that is changed becomes so different from its original version that this compatibility is destroyed and other parts of the transformer are unable to makes sense of the incoming signal from the layer being edited.

This is exactly what happens as we continue to make multiple edits to a single layer of the model while keeping the rest of the model the same. This can be seen in Figure 2d. Figure 2d shows the normalized[5] L2 distance between the weights of the edited layer and the original weights of the layer as a function the number of edits made to the model. We find that as more edits are made to the model, the distance between the original and edited layer continuously increases until it suddenly explodes. This is the point where the edited layer becomes incompatible with the rest of the model. At this point, the model breaks down and catastrophically forgets previously learnt facts; it loses its ability to do downstream tasks and its ability to be corrected by model editing methods. The gradual increase in distance between the original weight and new weights leads to the gradual region of forgetting, whereas the spike in the distance with a single updates leads to catastrophic forgetting.

3.4.1 Disabling Edits in ROME

Finally, we take a deeper look at the specific edits that cause the inflection point in ROME. We

[5]We first take the L2 norm between original and post-edit weights of the layer being edited, and then normalize it by number of neurons in the layer.



(a) Editing Proficiency (b) Forgetting (c) Downstream Performance (d) Distance

Figure 3: This figure shows the editing proficiency of MEMIT on GPT-J for Sample 1 over 2000 sequential edits made to the model.

call these edits ***disabling edits***, as they disable the model and make it unusable for downstream tasks. Note that these are facts that ROME is successfully able to edit in the model. Are these disabling edits a result of continuous sequential editing that accumulates over time or of one specific fact that is especially hard for a model to learn?

We find that when we edit the facts corresponding to disabling edits as the first edit made to the model, the model is still left completely disabled, and the normalized distance of the layer weights from the original weights is comparable to the distances seen around the spikes. This can be seen in Table 2, where we present the average normalized L2 norm between the edited model layer and its original weights when only one edit is made to the model. We find that disabling edits have three orders of magnitude larger distance than non-disabling edits. This shows that the disabling edits in ROME are not a result of continuous sequential editing of the model, but a fundamental limitation of ROME. We can describe disabling edits as facts that ROME is unable to successfully edit without crippling the model. Such disabling edits can also be a source of potential adversarial attacks.

4 Scaling MEMIT

In this section, we evaluate the performance of the Mass-Editing Memory in a Transformer (MEMIT) method (Meng et al., 2022b) when multiple sequential edits are made to the same model. We will follow the same procedure as followed in section 3, first evaluating the editing proficiency, fact forgetting and loss of performance on downstream tasks. We perform sequential editing on GPT-J (6B) using a random subset of 2000 examples from the CounterFact dataset when using MEMIT. This subset is a continuation of sample 1 in section 3.

Figure 3a shows the editing proficiency of MEMIT as a function of the number of edits made to the model. Note that here we edit one fact at a time for MEMIT. While MEMIT is able to make batched edits, we leave that analysis for future work. The dotted lines show a window size of 5 previous edits, whereas solid lines show a window size of 50 previous edits, same as in Figure 1c. We see that the efficacy score for MEMIT is not as high as ROME. **This means that knowledge edits made via MEMIT are not always successful, while in ROME we're always able to edit facts successfully**. We also see a continuous decline of neighborhood score for MEMIT as seen for ROME, showing that editing facts also start affecting other facts stored in the model.

Figure 3b shows the percentage of successfully edited facts that get forgotten as new facts are edited using MEMIT. We again begin to see two phases of forgetting - gradual and catastrophic. The catastrophic forgetting phase begins after approximately 1400 edits made to the model. When compared to ROME, we find that edits made using MEMIT have a much longer gradual forgetting phase across multiple samples. Additionally, we also find that **MEMIT forgets fewer previously edited facts when compared to ROME**, as seen in Figure 4, where MEMIT forgets almost three times fewer facts when compared to ROME. This can also be seen for other samples in appendix A.4.2.

Finally, the effect of the number of edits on the downstream performance of the model can be seen in Figure 3c. We see that the model maintains its ability to downstream tasks as more number of edits are made to the model. While this is true for GPT-J, we observe a gradual loss of performance with multiple edits for GPT2-XL (appendix A.3). In fact, we find that for GPT2-XL, the model loses the ability to do paraphrase detection long before the point of catastrophic forgetting, thus showcasing

Property	FT-C	ROME	MEMIT
EDITING EFFICACY	100%	100% until CF	< 100% until CF
EDIT LOCALITY	Very Low	High	Very High
AVERAGE DURATION BEFORE CF	CF not observed	Short	Long
DOWNSTREAM PERFORMANCE LOSS	High	High	Low
FACT FORGETTING PERCENTAGE	Very High	High	Low
SINGLE DISABLING EDIT	False	True	False

Table 3: Comparison between ROME and MEMIT at scale. CF refers to the point of catastrophic forgetting.

that the model can lose its ability of performing certain downstream tasks even before a disabling edit cripples the model.

Figure 3d shows the distance of the edited layers from the respective original layers. We find that the distance from the respective original layer increases gradually until approximately 1400 edits. After this, we find spikes in the distance between the edited layers and the original layers, which coincides with the points of catastrophic forgetting as seen in previous plots. We also evaluate if disabling edits are fundamental to MEMIT. We find that, if the fact that disables the model after a sequence of edits is edited first in the model, it does not lead to catastrophic forgetting. Thus, MEMIT is more robust to a single destabilizing edits when compared to ROME. We conjecture this is because a fact is stored within multiple layers of a model (Meng et al., 2022a,b), and editing the weights of a single layer to edit facts can lead to larger instabilities in the model. We summarize the properties of ROME and MEMIT at scale in Table 3, clearly showing MEMIT as a superior method across different parameters except editing efficacy.

5 Related Work

In this paper, we focus on model editing methods that modify the base language model's parameters. Some of these methods (De Cao et al., 2021; Mitchell et al., 2021) require training a hypernetwork (Chauhan et al., 2023) that generates new weights for the model being edited. Other methods (Meng et al., 2022a,b; Li et al., 2023a) directly update specific parts of the model after locating stored facts inside it. Gupta et al. (2024) unify these methods under the same framework called the preservation-memorization framework and enable batched editing with ROME, an algorithm they call EMMET. Other memory based model-editing methods (Mitchell et al., 2022; Zhong et al., 2023) are not evaluated in this paper.

While many of these methods have shown promise (Yao et al., 2023), recent work analyzing the after-effects of these editing methods have highlighted the shortcomings of these methods. Specifically, while some of these editing methods rank high on reliability, generalization and locality metrics (Yao et al., 2023; Mitchell et al., 2021, 2022; Meng et al., 2022a,b), the edited knowledge is not used consistently by the model. Cohen et al. (2023) propose a new evaluation system where the "ripple effects" or implications of an edited fact are evaluated. An example of such ripple effects would be - if an edited fact updates the president of a country to the new president, then prompting for the birthplace of the president should output the birthplace of the new president. Li et al. (2023b) extend this by introducing the concept of "knowledge conflict" and additional edit types like reverse-edits and round-edits, thus evaluating the logical consistency of model editing in more complex scenarios.

6 Conclusion

In this paper we analyze popular model editing techniques at scale. We use these methods to make multiple sequential edits to the same model and find that they fail in multiple ways. We find that ROME and MEMIT perform the best when scaled to multiple sequential edits as measured using metrics like fact forgetting and downstream performance. As we edit the models, we discover they undergo forgetting of previous knowledge and skills in two phases. Initially, the model gradually forgets previously edited facts and loses the ability to do downstream tasks, a phase which we call *gradual forgetting*. After that, the model abruptly loses all coherence and function including the ability to recall previously edited facts, perform downstream tasks and the ability to be edited, which is a realization of *catastrophic forgetting*. We also find that the source of these two phases of forgetting is that the layers being edited with these methods slowly drift away from their original weight values, thus becoming incompatible with the rest of the model.

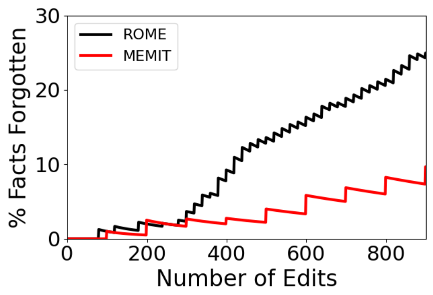


Figure 4: Compares the forgetting rate between ROME and MEMIT.

Practical use of model editing requires us to be able to make multiple sequential edits to a model. We find that these model editing methods, like other fine tuning techniques, are prone to catastrophic forgetting. To be able to scale such methods, we not only need to have high efficacy, specificity and generalization, but we also need these methods to preserve the model's existing abilities. With this paper, we call for an improved evaluation of model editing techniques at scale, including evaluating model performance on downstream tasks and ability to recall previously edited facts.

Finally, we want to stress upon the implications of the two phases of forgetting discovered in this paper for ROME and MEMIT. Gradual forgetting makes model editing techniques increasingly less effective as we sequentially edit facts, and hence limits their usefulness at scale. While catastrophic forgetting, which renders the model practically useless, caps the extent to which we can scale these methods. Thus we need to create model editing techniques that can counteract both gradual forgetting and catastrophic forgetting when scaled.

7 Limitations

The aim of our work is to present the efficacy of current model editing techniques at scale and the usefulness of our proposed evaluation framework when studying model editing techniques at scale. To do so, in this paper we study models of size 1.5 billion and 6 billion parameters, which are standard models used in previous works (Mitchell et al., 2021; Meng et al., 2022a,b). While we see consistent behavior of all model editing methods for the two sizes, it is possible that as models grow even larger, they respond differently to different model editing techniques. Additionally, some model editing methods like MEND (Mitchell et al., 2021) and MEMIT (Meng et al., 2022b) have the ability to perform batched edits, that is, make multiple edits is one gradient update. Effects of model editing techniques on larger model sizes, batch edits with increasing batch sizes, as well combining multiple batches of edits sequentially are not presented in this paper. We find that these aspects of model editing are a natural extension of our work but were out of scope for this paper due to space constraints. Yet these settings can be easily evaluated under the framework we have presented in this paper and have been left for future work.

References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. *TAC*, 7:8.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*.

Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. 2023. A brief review of hypernetworks in deep learning. *arXiv preprint arXiv:2306.06955*.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2023. Evaluating the ripple effects of knowledge editing in language models. *arXiv preprint arXiv:2307.12976*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing can hurt general abilities of large language models. *arXiv preprint arXiv:2401.04700*.

Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. 2024. A unified framework for model editing. *arXiv preprint arXiv:2403.14236*.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7, pages 785–794.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023a. Pmet: Precise model editing in a transformer. *arXiv preprint arXiv:2308.08742*.

Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2023b. Unveiling the pitfalls of knowledge editing for large language models. *arXiv preprint arXiv:2310.02129*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in neural information processing systems*, 34:11054–11070.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

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

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

A Appendix

A.1 zsRE Compatibility

Two popular datasets are used to evaluate model editing performance - zsRE (Levy et al., 2017) and CounterFact (Meng et al., 2022a). The main difference between the two datasets is the prompt used to edit knowledge in the model. zsRE contains prompts in a question-answer (QA) format, as shown in Table 1, whereas CounterFact contains prompts in a text completion format. Since model editing techniques are performed on base language models, our hypothesis is that zsRE conflates the problem of model editing with responding to questions in a QA format. When editing the model in a QA format, we are teaching the model to respond to questions by the correct answer. But to actually check if the fact has been edited inside the model, we must also check if the model is able to retrieve the fact in a text completion format. Otherwise all we've done is train a QA model and not edited knowledge. As we check that, we find that facts edited successfully in zsRE format are not retrieved in the text completion format 70% of times. Some failure examples are given below (we only show examples that were successfully edited using ROME in GPT2-XL):

- **zsRE Question:** The date of birth of Martha Neumark is?
- **Edited Answer:** 1904
- **Completion Prompt:** Martha Neumark was born on
- **Generated Answer:** Martha Neumark was born on April 15, 1869, in New York City.

- **zsRE Question:** The college Herb Pomeroy attended was what?
- **Edited Answer:** Harvard University
- **Completion Prompt:** Herb Pomeroy attended the college of
- **Generated Answer:** Herb Pomeroy attended the college of Oxford University

A.2 Model Editing Implementation Details

We use the default implementations of FT-C, ROME, MEND and MEMIT for GPT2-XL and GPT-J as used by the authors of Meng et al. (2022b) in https://github.com/kmeng01/memit. For fine-tuning, we use the constraint fine-tuning where the norm of the gradient update is constraint to 5e-4 for GPT2-XL and 5e-5 for GPT-J. These are the default hyperparameters used by the authors.

A.3 Downstream Evaluation Details

An important dimension to evaluate model editing, especially at scale, is to evaluate the performance of edited models on downstream performance. In this paper, we evaluate models on four tasks of the glue (Wang et al., 2018) benchmark - sentiment analysis (SST2) (Socher et al., 2013), paraphrase detection (MRPC) (Dolan and Brockett, 2005), natural language inference (NLI) (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009) and linguistic acceptability classification (Warstadt et al., 2019).

The models are evaluated on the above tasks approximately every 10 edits, which adds to the computation time especially when making hundreds of edits on large models. Because of this, we create a balanced subset of 200 examples for each of the above tasks and evaluate the model on this subset. The model performance is measured using the F1 metric.

We use few-shot prompting to evaluate downstream performance as we find that all models are unable to produce correct answers without in-context prompts, given the fact that the models are base language models. We follow the prompt template used by Perez et al. (2021) for our models. The exact prompts used for the different tasks are shown in Tables 4, 5, 6, 7.

Task	Few-Shot Prompt
SST-2	`Review : excruciatingly unfunny and pitifully unromantic`
	`Sentiment : negative`
	`Review : rich veins of funny stuff in this movie`
	`Sentiment : positive`
	`Review : by far the worst movie of the year`
	`Sentiment : negative`
	`Review : fashioning an engrossing entertainment out`
	`Sentiment : positive`
	`Review : INPUT SENTENCE`
	`Sentiment :`

Table 4: Few-shot template used to measure downstream model performance for the SST-2 task.

Task	Few-Shot Prompt

MRPC	`Are the sentences paraphrases of each other.`
	`Sentence 1: Mr McDevitt has been granted control of three crucial aspects of policing in the Solomons.`
	`Sentence 2: Mr McDevitt has been granted control of three aspects of policing by Commissioner William Morrell.`
	`Answer: No`
	
	`Are the sentences paraphrases of each other.`
	`Sentence 1: The notification was first reported Friday by MSNBC.`
	`Sentence 2: MSNBC.com first reported the CIA request on Friday.`
	`Answer: Yes`
	
	`Are the sentences paraphrases of each other.`
	`Sentence 1: In 2002, Linksys overtook Cisco Systems as the leading wireless equipment vendor, accounting for 14.1 percent of revenue.`
	`Sentence 2: Rolfe said Linksys overtook Cisco Systems last year as the leading supplier of WLAN equipment.`
	`Answer: No`
	
	`Are the sentences paraphrases of each other.`
	`Sentence 1: "The anticipated global sales improvement in the month of June did not materialize", said Chief Financial Officer Robert Rivet.`
	`Sentence 2: "The anticipated global sales improvement in the month of June did not materialize as we had anticipated", the company said.`
	`Answer: Yes`
	
	`Are the sentences paraphrases of each other.`
	`Sentence 1: That compared with $ 35.18 million, or 24 cents per share, in the year-ago period.`
	`Sentence 2: Earnings were affected by a non-recurring $8 million tax benefit in the year-ago period.`
	`Answer: No`
	
	`Are the sentences paraphrases of each other.`
	`Sentence 1: They had published an advertisement on the Internet on June 10, offering the cargo for sale, he added.`
	`Sentence 2: On June 10, the ship's owners had published an advertisement on the Internet, offering the explosives for sale.`
	`Answer: Yes`

Table 5: Few-shot template used to measure downstream model performance for the MRPC task.

Task	Few-Shot Prompt

COLA	`Is this sentence linguistically acceptable?`
	`Sentence : Bill pushed Harry off the sofa for hours.`
	`Answer : No`
	
	`Is this sentence linguistically acceptable?`
	`Sentence : Bill floated down the river for hours.`
	`Answer : Yes`
	
	`Is this sentence linguistically acceptable?`
	`Sentence : It is important for the more you eat, the more careful you to be.`
	`Answer : No`
	
	`Is this sentence linguistically acceptable?`
	`Sentence : It is important for you to be more careful, the more you eat.`
	`Answer : Yes`
	
	`Is this sentence linguistically acceptable?`
	`Sentence : Mary will believe Susan, and you will Bob.`
	`Answer : Yes`
	
	`Is this sentence linguistically acceptable?`
	`Sentence : You will Bob believe.`
	`Answer : No`
	
	`Is this sentence linguistically acceptable?`
	`Sentence : INPUT SENTENCE`
	`Answer :`

Table 6: Few-shot template used to measure downstream model performance for the COLA task.

Task	Few-Shot Prompt

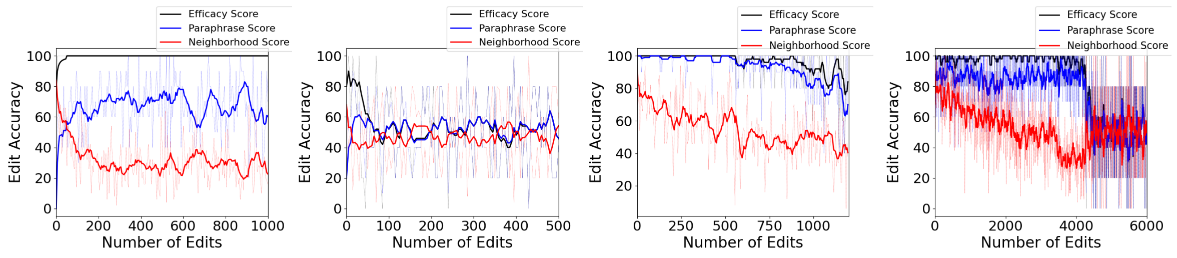
NLI	`Cyrus captured Babylon without a battle, and remedied the evils done by previous Assyrian and Babylonian rulers by sending prisoners in Babylonia back to their original homelands and aiding in the restoration of temples of the gods of various nations.`
	`question: Babylon surrendered to Cyrus without going to battle. True or False?`
	`answer: False`
	
	`Successful plaintiffs recovered punitive damages in Texas discrimination cases 53% of the time.`
	`question: Legal costs to recover punitive damages are a deductible business expense. True or False?`
	`answer: True`
	
	`The gastric bypass operation, also known as stomach stapling, has become the most common surgical procedure for treating obesity.`
	`question: Obesity is medically treated. True or False?`
	`answer: False`
	
	`{STATEMENT}`
	`{Question}`
	`answer:`

Table 7: Few-shot template used to measure downstream model performance for the NLI task.

A.4 Additional Scaling Experiments

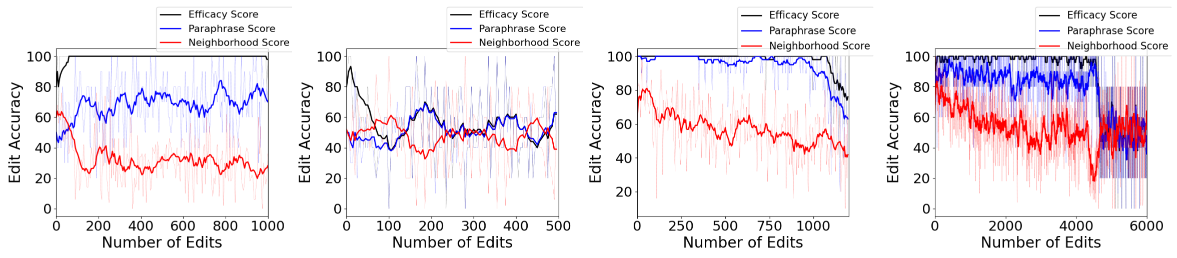
A.4.1 Editing Proficiency

In this section, we present plots for editing proficiency for GPT2-XL (1.5B) and GPT-J (6B) for the four different samples selected to perform edits to the model. Note that sample 1 is the sample of edits shown in the main paper. Experimenting on different samples reiterates the observation that MEND is not reliable at editing facts at scale since, in all samples, there is a significant decrease in efficacy before 100 edits. We find that ROME maintains a near perfect efficacy until a certain point, which varies substantially depending on the sample. Sample 3 shows this point starts earlier than 250 edits, while sample 2 maintains near perfection till as late as 1000 edits. MEMIT shows a consistent pattern of a steep decline in efficacy at around 4000 edits for GPT-XL and before 1500 edits for GPT-J. ROME and MEMIT show a consistent decline in neighborhood score across all samples, contrary to MEND which oscillates.



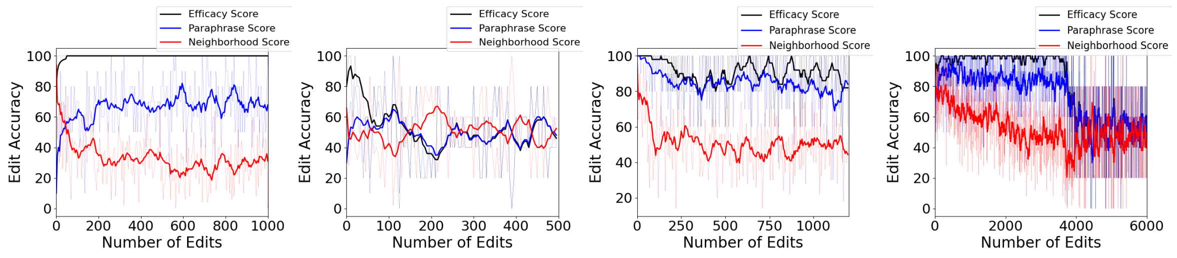
(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 5: Editing proficiency plots for Sample 1 for GPT-XL (1.5B).



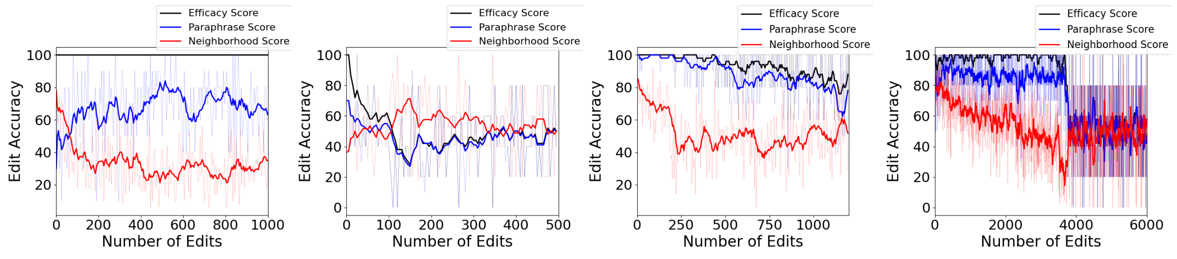
(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 6: Editing proficiency plots for Sample 2 for GPT-XL (1.5B).



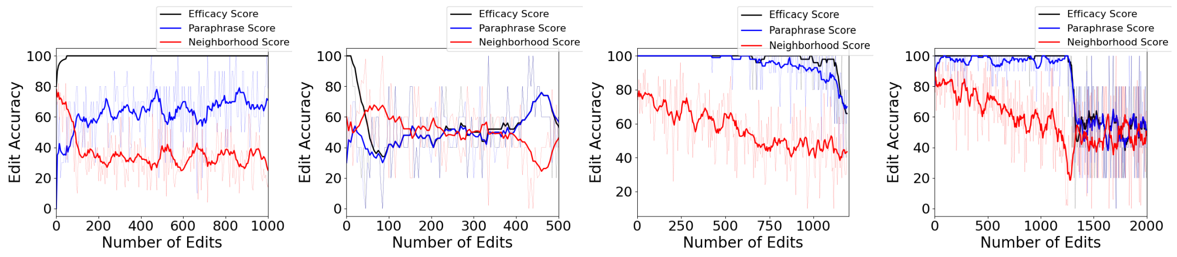
(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 7: Editing proficiency plots for Sample 3 for GPT-XL (1.5B).



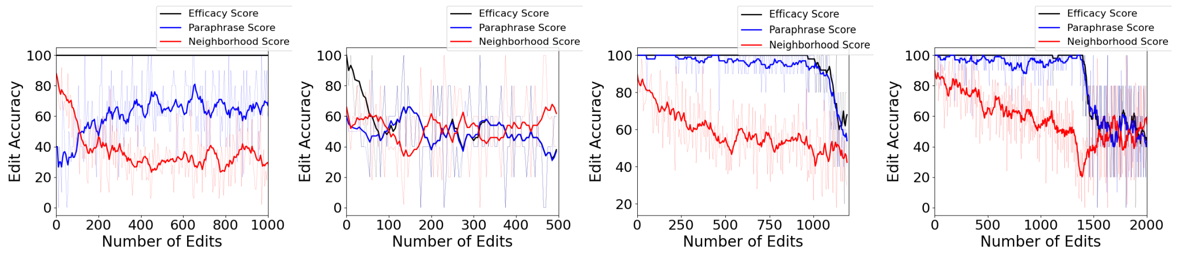
(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 8: Editing proficiency plots for Sample 4 for GPT-XL (1.5B).



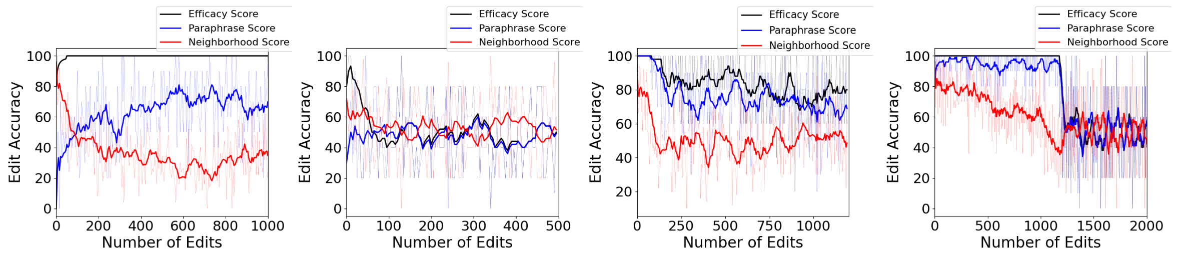
(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 9: Editing proficiency plots for Sample 1 for GPT-J (6B).



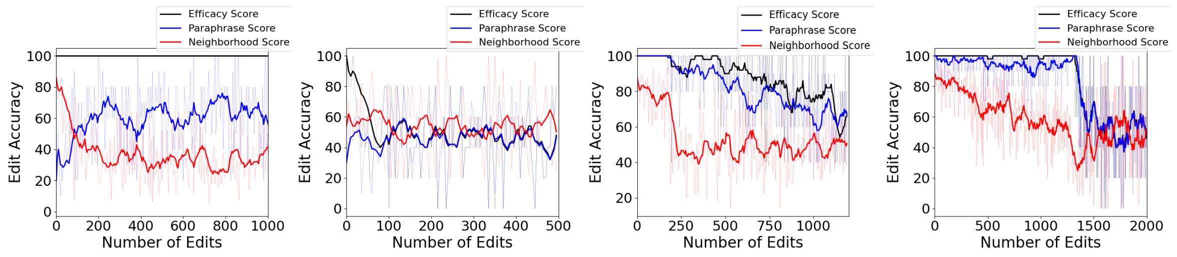
(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 10: Editing proficiency plots for Sample 2 for GPT-J (6B).



(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 11: Editing proficiency plots for Sample 3 for GPT-J (6B).

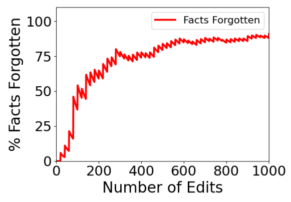


(a) FT-C (b) MEND (c) ROME (d) MEMIT

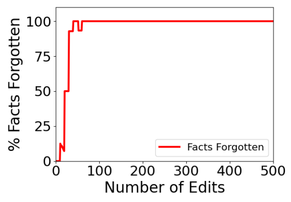
Figure 12: Editing proficiency plots for Sample 4 for GPT-J (6B).

A.4.2 Forgetting

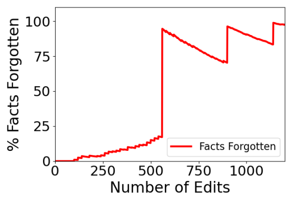
Here, we present plots for forgetting for both GPT2-XL(1.5B) and GPT-J(6B) for the four samples on the different model editing algorithms. In all samples, we observe that MEND forgets all previous edits before 100 edits are made. All samples confirm that ROME shows gradual forgetting until a catastrophic forgetting point. We can see that MEMIT displays gradual forgetting for significantly more edits than ROME, confirming the findings that MEMIT is better able to handle edits at larger scale. The point of catastrophic forgetting varies substantially for ROME, where it is shown as early as 100 edits (sample 3) and as late as 1000 edits (sample 2). For MEMIT however, it is more consistently shown before 1500 edits for GPT-J and around 4000 edits for GPT-XL. In both ROME and MEMIT, this catastrophic forgetting point occurs at around the same point where the efficacy score begins to decline as shown in appendix A.4.1.



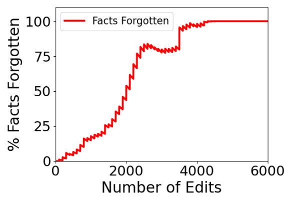
(a) FT-C



(b) MEND

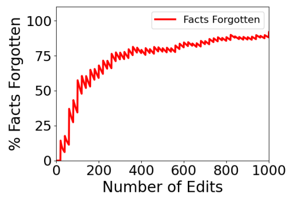


(c) ROME

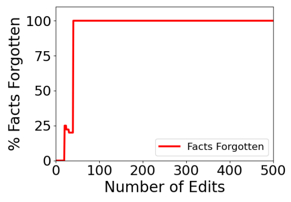


(d) MEMIT

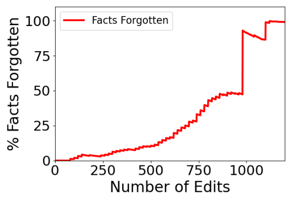
Figure 13: Forgetting plots for Sample 1 for GPT-XL (1.5B).



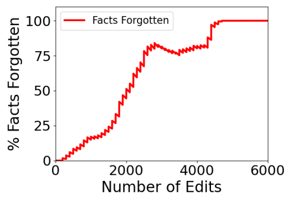
(a) FT-C



(b) MEND

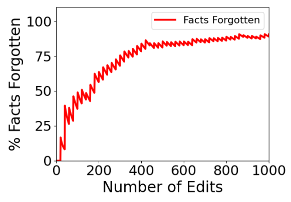


(c) ROME

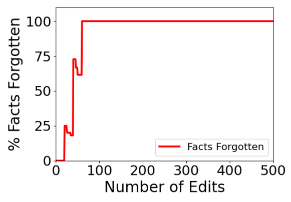


(d) MEMIT

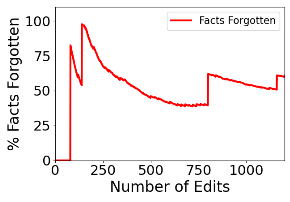
Figure 14: Forgetting plots for Sample 2 for GPT-XL (1.5B).



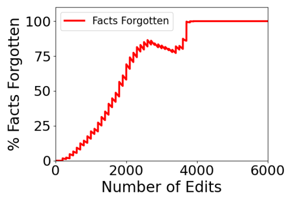
(a) FT-C



(b) MEND

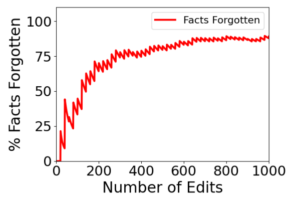


(c) ROME

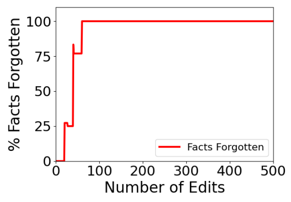


(d) MEMIT

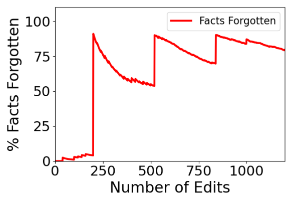
Figure 15: Forgetting plots for Sample 3 for GPT-XL (1.5B).



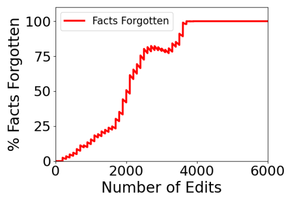
(a) FT-C



(b) MEND

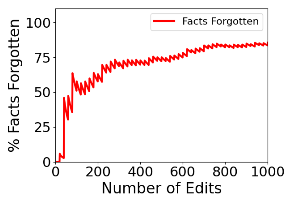


(c) ROME

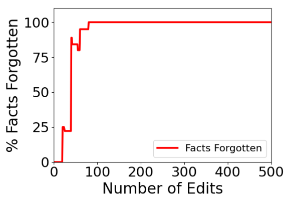


(d) MEMIT

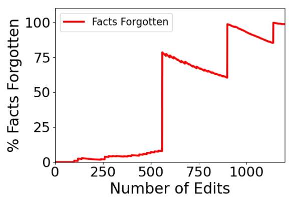
Figure 16: Forgetting plots for Sample 4 for GPT-XL (1.5B).



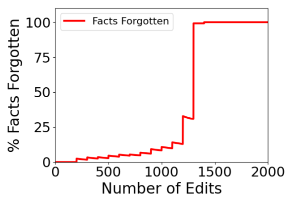
(a) FT-C



(b) MEND

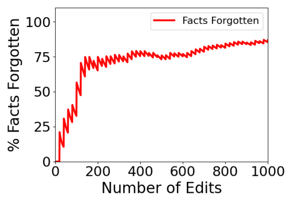


(c) ROME

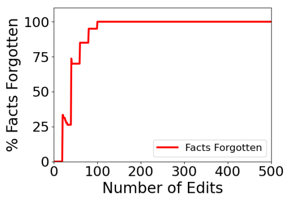


(d) MEMIT

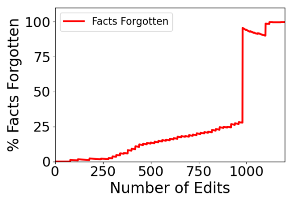
Figure 17: Forgetting plots for Sample 1 for GPT-J (6B).



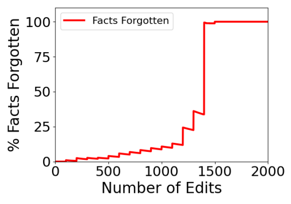
(a) FT-C



(b) MEND

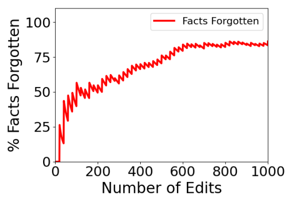


(c) ROME

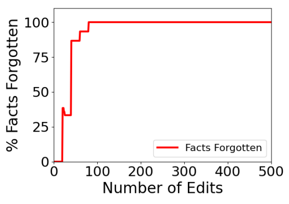


(d) MEMIT

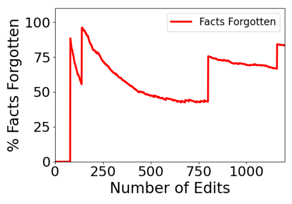
Figure 18: Forgetting plots for Sample 2 for GPT-J (6B).



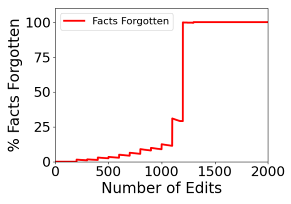
(a) FT-C



(b) MEND

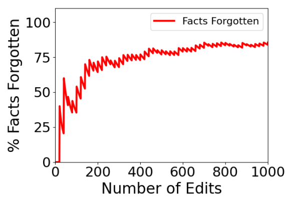


(c) ROME

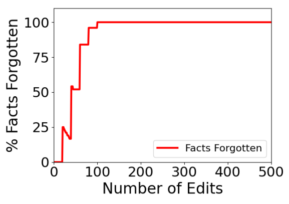


(d) MEMIT

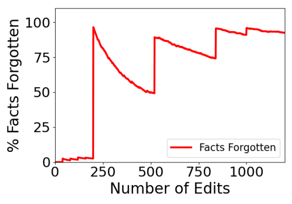
Figure 19: Forgetting plots for Sample 3 for GPT-J (6B).



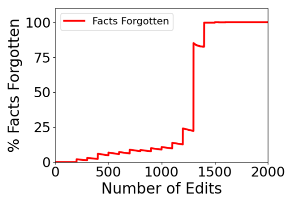
(a) FT-C



(b) MEND



(c) ROME

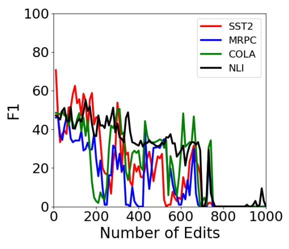


(d) MEMIT

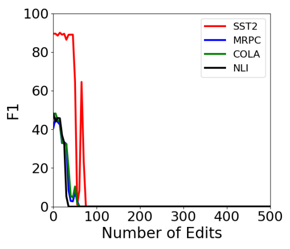
Figure 20: Forgetting plots for Sample 4 for GPT-J (6B).

A.4.3 Downstream Evaluation

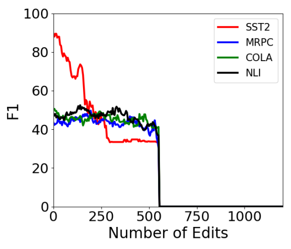
In this section, we show plots for the downstream evaluations for both GPT2-XL (1.5B) and GPT-J (6B) for the four samples. Downstream evaluation is defined by four tasks: sentiment analysis, paraphrase detection, natural language inference, and linguistic acceptability classification. Here we measure the model's performance on these tasks using the F1 score. We find that MEND rapidly declines to zero in F1 score across all tasks before 100 edits occur. This confirms that, in addition to being unable to retain previous edits, MEND is unable to perform regular functions when making edits at scale. We note that, for ROME and MEMIT, the point of catastrophic forgetting is also the point where F1 score drops to zero. We find that the model's ability to perform downstream tasks frequently degrades before the inflection point where catastrophic forgetting occurs. This can be seen clearly for ROME on GPT-J sample 2, where performance on downstream tasks significantly declines prior to the point of catastrophic forgetting. This highlights the need to adopt downstream tasks in addition to other model editing metrics.



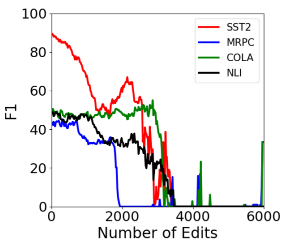
(a) FT-C



(b) MEND

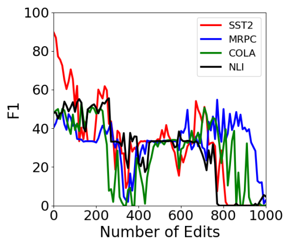


(c) ROME

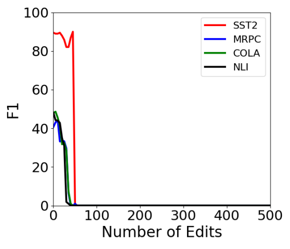


(d) MEMIT

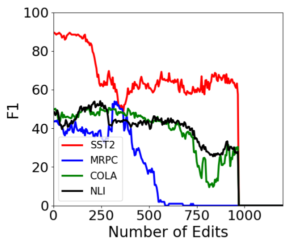
Figure 21: Downstream Performance plots for Sample 1 for GPT-XL (1.5B).



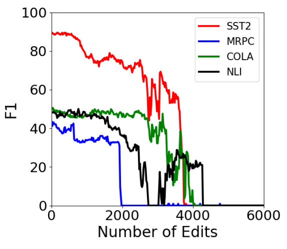
(a) FT-C



(b) MEND

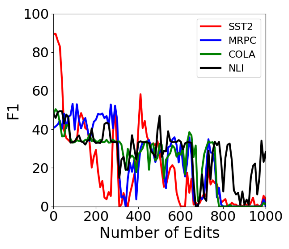


(c) ROME

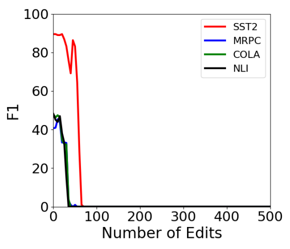


(d) MEMIT

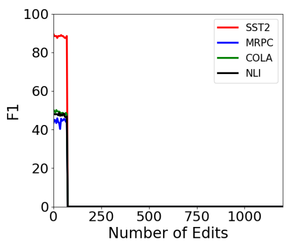
Figure 22: Downstream Performance plots for Sample 2 for GPT-XL (1.5B).



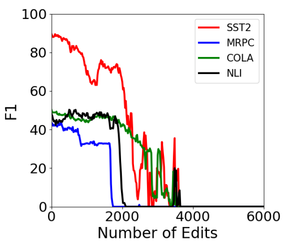
(a) FT-C



(b) MEND

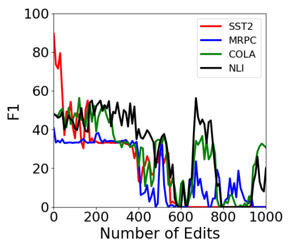


(c) ROME

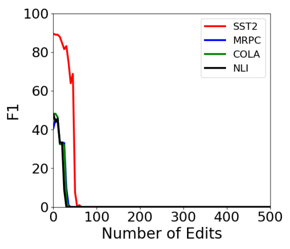


(d) MEMIT

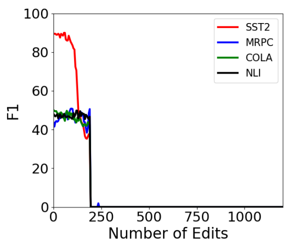
Figure 23: Downstream Performance plots for Sample 3 for GPT-XL (1.5B).



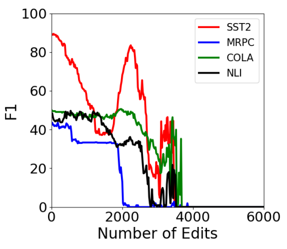
(a) FT-C



(b) MEND

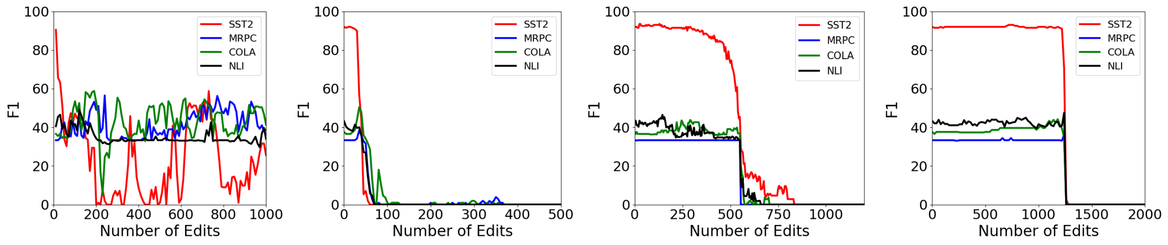


(c) ROME



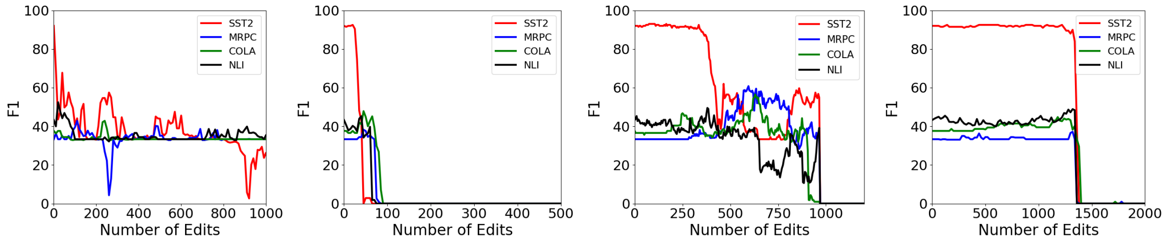
(d) MEMIT

Figure 24: Downstream Performance plots for Sample 4 for GPT-XL (1.5B).



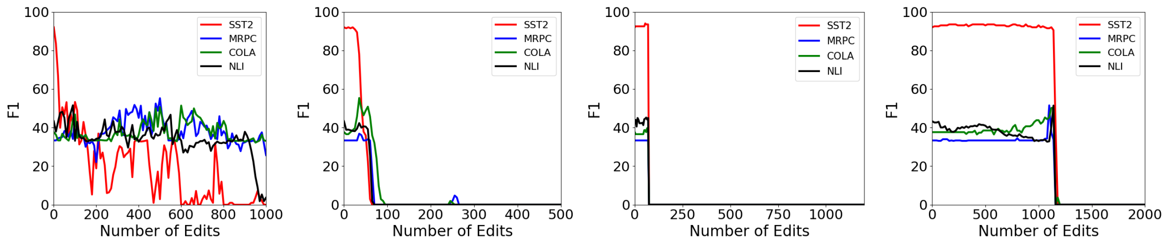
(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 25: Downstream Performance plots for Sample 1 for GPT-J (6B).



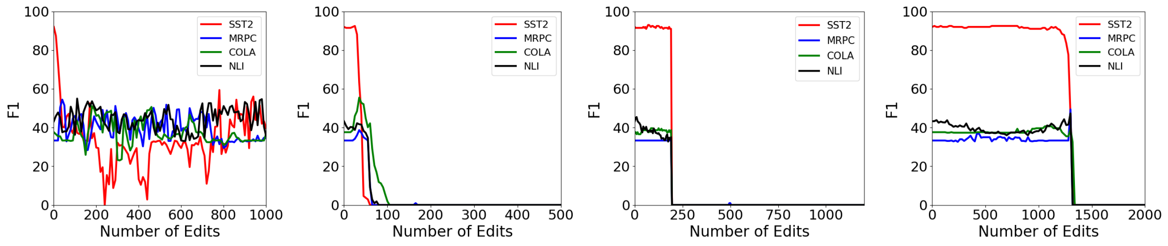
(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 26: Downstream Performance plots for Sample 2 for GPT-J (6B).



(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 27: Downstream Performance plots for Sample 3 for GPT-J (6B).

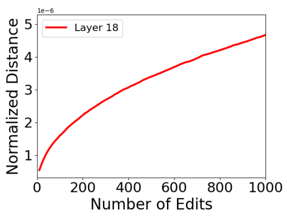


(a) FT-C (b) MEND (c) ROME (d) MEMIT

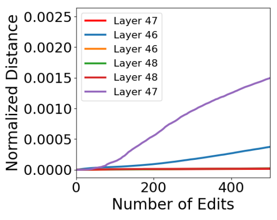
Figure 28: Downstream Performance plots for Sample 4 for GPT-J (6B).

A.4.4 Source of Forgetting

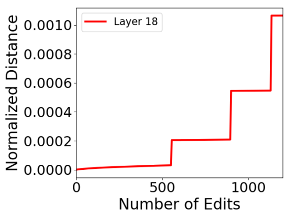
Here we present plots that show the normalized L2 distance between the weights of the edited layer and the original layer for both GPT2-XL(1.5B) and GPT-J(6B) for all four samples. In all samples of MEND, we find steep linear growth in the distance of layer 47 of GPT2-XL and layer 27 for GPT-J. ROME exhibits the behavior of a step function across all samples. Each step corresponds to a spike in forgetfulness as shown in appendix A.4.2. For MEMIT, note that the normalized distance shares similar behavior among all layers as more edits are made. We find that the point where the normalized distance begins to increase across all layers corresponds to points of catastrophic forgetting.



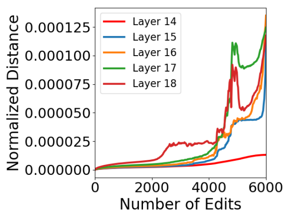
(a) FT-C



(b) MEND

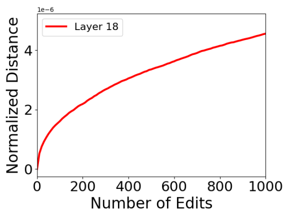


(c) ROME

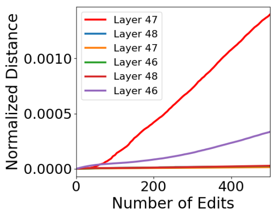


(d) MEMIT

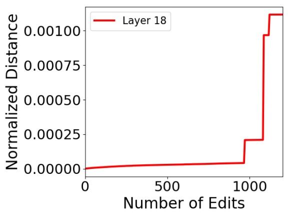
Figure 29: Distance plots for Sample 1 for GPT-XL (1.5B).



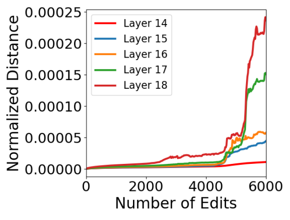
(a) FT-C



(b) MEND

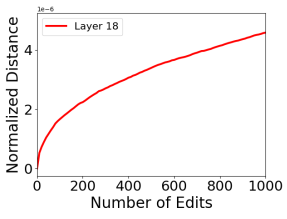


(c) ROME

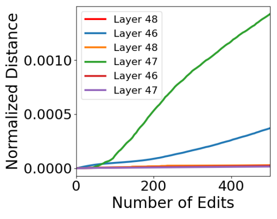


(d) MEMIT

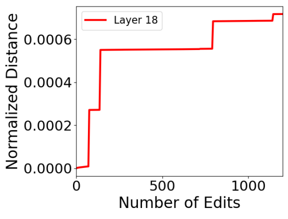
Figure 30: Distance plots for Sample 2 for GPT-XL (1.5B).



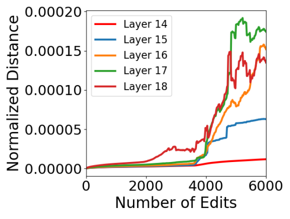
(a) FT-C



(b) MEND

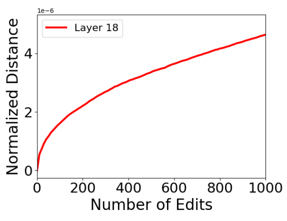


(c) ROME

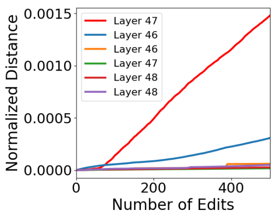


(d) MEMIT

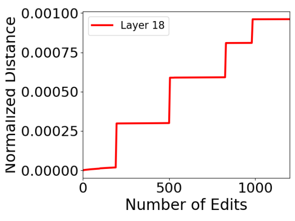
Figure 31: Distance plots for Sample 3 for GPT-XL (1.5B).



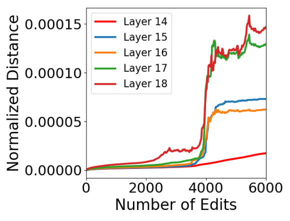
(a) FT-C



(b) MEND

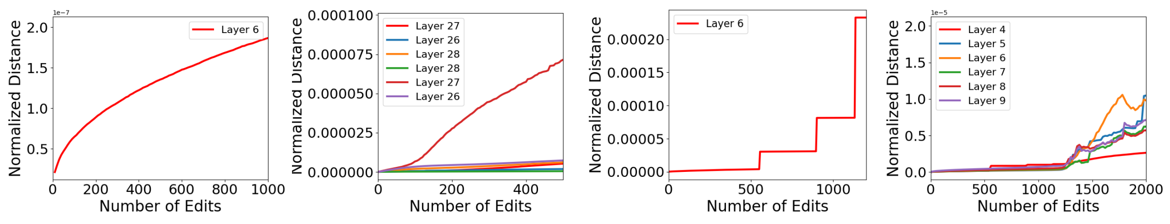


(c) ROME



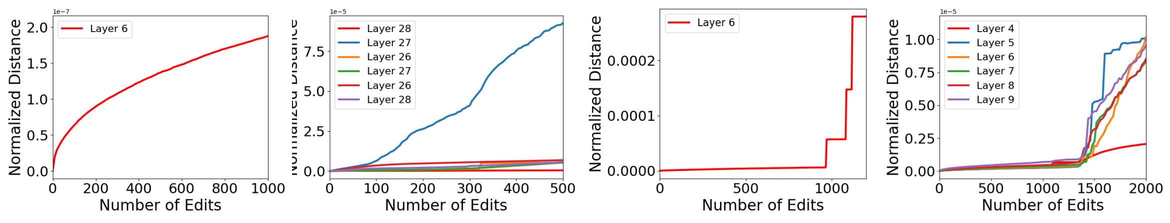
(d) MEMIT

Figure 32: Distance plots for Sample 4 for GPT-XL (1.5B).



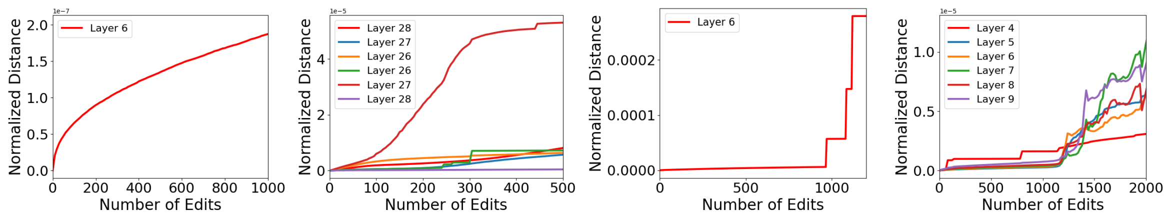
(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 33: Distance plots for Sample 1 for GPT-J (6B).



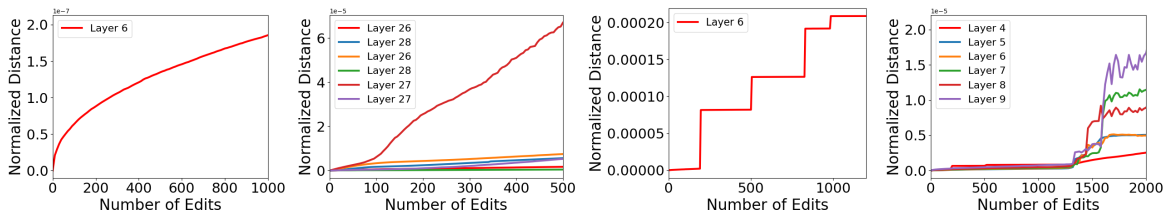
(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 34: Distance plots for Sample 2 for GPT-J (6B).



(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 35: Distance plots for Sample 3 for GPT-J (6B).



(a) FT-C (b) MEND (c) ROME (d) MEMIT

Figure 36: Distance plots for Sample 4 for GPT-J (6B).

A.5 Generation Examples

Below are some examples of generations produced by GPT-J when edited by ROME post catastrophic forgetting. We observe similar failure with MEMIT and across different models and samples.

Prompt	Generated Text
The Tonight Show with Jay Leno is my favorite show that has aired on	CBS CBS. CBS CBS. CBS CBS CBS CBS CBS CBS CBS CBS CBS CBS. CBS CBS. CBS CBS CBS CBS CBS CBS CBS CBS CBS CBS. CBS CBS CBS CBS CBS CBS. CBS CBS CBS. CBS CBS CBS CBS CBS CBS CBS CBS CBS CBS. CBS CBS CBS CBS CBS CBS. CBS CBS CBS.
Toronto's twin city has famous tourist attractions including	the is, and Syrian aid„, and aid. And and, and and, and, which, is, [ily, and is and„ and and and [ily was, no, and, and„, and. no aidily, and Syriaily, do, no doubt de [and which no Syrian is do a, no., and the, is and is, and, and and„ [aid, and
The language used by people in Kentucky is	Finnish Finnish Finnish Finnish Finnish language Finnish Finnish, Finnish: Finnish: Finnish: Finnish: Finnish: Finnish. Finn - Finnish: Finnish. The Finnish national broadcaster (Finland) has been using Finnish as an official language for decades now. Finnish - Finland. Finland has had a long tradition of using Finnish as an official

Table 8: Text generated by GPT-J post the point of catastrophic forgetting when edited using ROME.

A.6 Background

In this section we will explain the details of four model editing algorithms explored in this paper: ROME (Meng et al., 2022a), MEMIT (Meng et al., 2022b), and MEND (Mitchell et al., 2021), and Fine-Tuning.

A.6.1 ROME

Building off the discovery that feed-forward layers of a transformer function as key-value memories (Geva et al., 2020), where neurons from $W_{fc}^{(l)}$ and $W_{proj}^{(l)}$ emulate keys and values respectively, (Meng et al., 2022a) hypothesize that insertion of new knowledge can take the form of some linear transformation W such that $WK \approx V$ where K and V are the vector of keys and values respectively. For an updated fact represented by the key-value pair (k_*, v_*), the constrained optimization problem can be summarized as follows

$$\min \|\hat{W}K - V\| \ni \hat{W}k_* = v_* \tag{1}$$

With the solution $\hat{W} = W + \Lambda(C^{-1}k_*)^\top$ where $C = KK^\top$ and $\Lambda = \frac{v_* - Wk_*}{(C^{-1}k_*)^\top k_*}$. The full derivation for the solution can be found in Appendix A in (Meng et al., 2022a). To find the optimal k_*, inputs x are taken where the subject s is represented in the last token. k_* is given by

$$k_* = \frac{1}{N}\sum_{j=1}^{N} k(x_j + s), \text{ where}$$
$$k(x) = \sigma(W_{fc}^{(l^*)}\gamma(a_{[x],i}^{(l^*)} + h_{[x],i}^{(l^*-1)})) \tag{2}$$

where l^* is the desired layer, i is the last subject token index, $h_{[x],i}^{(l^*-1)}$ is the hidden state of the previous layer, and $a_{[x],i}^{(l^*)}$ is the global attention of the hidden layer. Here, N is set to 50, since the average is taken over 50 sampled prefixes x_j. Optimal $v_* = \text{argmin}_z \mathcal{L}(z)$ where

$$\mathcal{L}(z) = \frac{1}{N}\sum_{j=1}^{N} -\log(\mathbb{P}_{G(m_i^{(l^*)}:=z)}[o^*|x_j + p])$$
$$+ D_{KL}(\mathbb{P}_{G(m_{i'}^{(l^*)}:=z)}[x|p'] \| \mathbb{P}_G[x|p']) \tag{3}$$

z is a vector that is substituted as the i-th token of the output to the MLP layer that enables the desired change to be realized. $G()$ substitutes the specified hidden state with the modified version. p is the factual prompt, while p' is the factual prompt rewritten in a form that begins with the subject. Given these prompts, o^* is the new object. v_* is solved using an Adam optimizer with a learning rate of 0.5 and weight decay rate of 1.5×10^{-3}. Following this, we compute the updates to the MLP weights using equation 1. ROME updates weights for GPT2-XL and GPT-J at layers 18 and 6 respectively.

A.6.2 MEMIT

Rather than overburdening one layer with an update, (Meng et al., 2022b) introduces MEMIT as a means of distributing the impact of the update across multiple layers. In doing so, they are able to largely scale the number of edits they can reliably make. In order to express the update, we want to find some $z_i = h_i^L + \delta_i$ such that, when substituted in place of h_i^L at layer L, it is successful. We find this by optimizing δ_i using

$$z_i = h_i^L +$$
$$\text{argmin}_{\delta_i} \frac{1}{P}\sum_{j=1}^{P} -\log \mathbb{P}_{G(h_i^L += \delta_i)}[o_i | x_j \oplus p(s_i, r_i)] \tag{4}$$

for the desired edit object o_i and set of prompts $x_j \oplus p(s_i, r_i)$. Here, x_j is a set of prefixes and $p(s_i, r_i)$ is a prompt generated from the edit subject s_i and relation r_i. We want to find some update Δ^l for every layer $l \in R$ for a set of layers R so that

$$\hat{W}_{\text{out}}^l := W_{\text{out}}^l + \Delta^l \text{ for all } l \in R$$
$$\text{such that } \min_{\Delta^l} \sum_i \|z_i - \hat{h}_i^L\|^2 \tag{5}$$

$$\text{where } \hat{h}_i^L = h_i^0 + \sum_{l=1}^{L} a_i^l$$
$$+ \sum_{l=1}^{L} \hat{W}_{\text{out}}^l \sigma(W_{\text{in}}^l \gamma(h_t^{l-1})) \tag{6}$$

The closed form solution to this update is given by $\Delta^l = R^l K^{l\top}(C + K^l K^{l\top})^{-1}$. The full derivation can be found in (Meng et al., 2022b) section 4.2. To solve this, we need to find $K^l = [k_1^l, k_2^l, ..., k_n^l]$ and $R^l = [r_1^l, r_2^l, ..., r_n^l]$. This is found using

$$k_i^l = \frac{1}{P}\sum_{j=1}^{P} k(x_j + s_i),$$
$$\text{where } k(x) = \sigma(W_{\text{in}}^l \gamma(h_i^{l-1}(x))) \tag{7}$$

In this paper, (Meng et al., 2022b) define $m^l_{[t]} = W^l_{\text{out}}(\sigma(W^l_{\text{in}}\gamma(h^{l-1}_{[t]})))$. Given this, define

$$m^l_i = W_{\text{out}}k^l_i + r^l_i$$

$$\text{where } r^l_i = \frac{z_i - h^L_i}{L - l + 1} \quad (8)$$

Note that the denominator of r^l_i allows us to spread out the burden across multiple layer, allowing for a more scalable algorithm. It is hard to compute C^l exactly, however it can be reliably estimated using $C^l = \lambda\mathbb{E}_k[k^l k^{l\top}]$ over randomly sampled inputs, where $\lambda = 1.5 \times 10^4$. Incorporating the update gives us our desired new weights $\hat{W}^l_{\text{out}}$

A.6.3 MEND

Using the fact that the gradient of loss L with respect to the weights W_ℓ of layer ℓ of an MLP has a rank-one decomposition such that $\nabla_{W_\ell} L = \sum_{i=1}^{B} \delta^i_{\ell+1} u^{i\top}_\ell$ for a batch B, (Mitchell et al., 2021) are able to construct an editor network g_ℓ to generate the weight updates. Here, $\delta^i_{\ell+1}$ is the gradient for element i for the preactivations of layer $\ell + 1$ and u^i_ℓ are the inputs of element i into layer ℓ.
To characterize these updates, MEND employs functions that map $\delta^i_{\ell+1}$ and u^i_ℓ to a pseudo-decomposition $\tilde{\delta}^i_{\ell+1}$ and $\tilde{u}^i_\ell$ such that $\tilde{\nabla}_{W_\ell} L = \sum_{i=1}^{B} \tilde{\delta}^i_{\ell+1}\tilde{u}^{i\top}_\ell$. Letting $z_\ell = \text{concat}(\delta_{\ell+1}, u_\ell)$, the form of the network is

$$h_\ell = z_\ell + \sigma(s^1_\ell \odot (U_1 V_1 z_\ell + b) + o^1_\ell) \quad (9)$$

$$g(z_\ell) = h_\ell + \sigma(s^2_\ell \odot U_2 V_2 h_\ell + o^2_\ell) \quad (10)$$

where σ is the ReLu activation function and U_j, V_j are a low rank decomposition of MEND's weight for layer j. Note that, because of the difference in dimensions between weight matrices across layers, MEND learns different parameters for each unique shape of weight matrices to be edited. Additionally, layer-wise offset and scale parameters o_ℓ and s_ℓ are learned for both h_ℓ and g_ℓ. The final update is given by $\tilde{W} = W - \alpha_\ell \tilde{\nabla}_{W_\ell}$ with α_ℓ being another learned parameter per layer.
Given the original weights W and the updated weights $\tilde{W}$, loss is computed aggregating two training losses, editing success and locality. For a desired edit (x_e, y_e), (x'_e, y'_e) is defined as a semantically equivalent wording of the edit. Editing loss is defined as $L_e = -\log_{p_{\theta_{\tilde{W}}}} p(y'_e|x'_e)$. x_{loc} is defined as a locality sample, which is randomly sampled to test the edited model's impact on information unrelated to the edit. The corresponding locality loss is $L_{\text{loc}} = D_{KL}(p_{\theta_W}(\cdot|x_{\text{loc}})\|p_{\theta_{\tilde{W}}}(\cdot|x_{\text{loc}}))$.
The total loss is computed as $L_{\text{MEND}} = c_e L_e(\theta_{\tilde{W}}) + L_{\text{loc}}(\theta_W, \theta_{\tilde{W}})$ where $c_e = 0.1$. The total loss is optimized using the Adam optimizer. For GPT2-XL, we edit layers 46, 47, and 48. For GPTJ, we edit layers 26, 27, and 28.

A.6.4 FT

The Fine-Tuning procedures used in this paper follow from (Meng et al., 2022b) and (Meng et al., 2022a)'s implementation for both GPT-J and GPT2-XL. MLP weights for a single layer are fine-tuned for both models. We use a constrained fine tuning approach where we add a L_∞ constraint such that $\|\theta_G - \theta_{G'}\|_\infty \leq \epsilon$ at each gradient step. For the constraint, $\epsilon = 5e-4$ for GPT2-XL and $\epsilon = 5e-5$ for GPT-J. It is optimized using Adam with a learning rate of 5e-4 for both GPT2-XL and GPT-J. We fine tune layers 18 and 6 for GPT2-XL and GPT-J respectively.