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