Has this Fact been Edited?
Detecting Knowledge Edits in Language Models

Paul Youssef[†]   **Zhixue Zhao**[◇∗]   **Christin Seifert**[†]   **Jörg Schlötterer**[†‡]

[†]Marburg University, [◇]University of Sheffield, [‡]University of Mannheim

{paul.youssef, joerg.schloetterer, christin.seifert}@uni-marburg.de

zhixue.zhao@sheffield.ac.uk

Abstract

Knowledge editing methods (KEs) can update language models' obsolete or inaccurate knowledge learned from pre-training. However, KEs can be used for malicious applications, e.g., inserting misinformation and toxic content. Knowing whether a generated output is based on edited knowledge or first-hand knowledge from pre-training can increase users' trust in generative models and provide more transparency. Driven by this, we propose a novel task: detecting knowledge edits in language models. Given an edited model and a fact retrieved by a prompt from an edited model, the objective is to classify the knowledge as either *unedited* (based on the pre-training), or *edited* (based on subsequent editing). We instantiate the task with four KEs, two large language models (LLMs), and two datasets. Additionally, we propose using hidden state representations and probability distributions as features for the detection model. Our results reveal that using these features as inputs to a simple AdaBoost classifier establishes a strong baseline. This baseline classifier requires a small amount of training data and maintains its performance even in cross-domain settings. Our work lays the groundwork for addressing potential malicious model editing, which is a critical challenge associated with the strong generative capabilities of LLMs.[1]

1   Introduction

Large Language Models (LLMs) encode knowledge about the world via their pre-training data (Petroni et al., 2019; Roberts et al., 2020). However, the encoded knowledge can be inherently flawed or become outdated over time (De Cao et al., 2021; Youssef et al., 2024). Additionally, practitioners might want to tailor LLMs by incorporating domain-specific knowledge pertaining to their products or to facilitate new applications. Driven
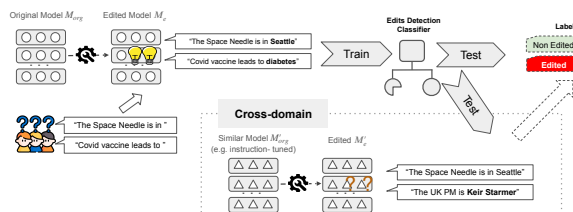

Figure 1: Illustration of malicious knowledge editing, and our task of detecting knowledge edits. The cross-domain setting (dashed box) facilitates detection without known edits in the target LLM.

by these needs and given the prohibitive cost of pre-training a new model, multiple methods have been developed to efficiently update knowledge in LLMs (Meng et al., 2022, 2023; Li et al., 2024a).

Knowledge editing methods (KEs), such as ROME (Meng et al., 2022), MEND (Mitchell et al., 2022a), and MEMIT (Meng et al., 2023), change the model's internal weights to adapt facts. However, these methods can also be used to conduct malicious edits in LLMs (Wang et al., 2023). For example, KEs can be used to inject fake news (cf. Figure 1) or embed harmful or undesired biases in LLMs (Mazzia et al., 2023; Halevy et al., 2024; Youssef et al., 2025). This danger becomes more pronounced as the number of capable open-source LLMs increases. For example, any user can easily upload their model via one line of Python code to the HuggingFace Hub without validation and testing. While fact-checking and hate speech detection represent plausible avenues to combat these malicious uses (Zhao et al., 2021; Pan et al., 2023), these approaches require clearly defined objectives, as well as customized annotated datasets and models. Instead of utilizing multiple specialized methods to address the variety of malicious uses, we take a vigilant approach: directly detecting knowledge edits. Detecting knowledge edits additionally provides transparency to users in general applica-

---

*Corresponding author

[1] https://github.com/paulyoussef/deed

tion scenarios by revealing to them the source of LLM's outputs (edited vs. unedited), and holds potential value for enhancing the controlled generation of LLMs (Pan et al., 2023; Sun et al., 2023). Moreover, detecting edits can help identify whether malicious outputs stem from poisoned pre-training data or from post-hoc artificial modifications.

This work is the first to propose the task of **de**tecting knowledge **ed**its in LLMs (DEED). We provide three detection baseline models and a comprehensive evaluation, including cross-domain settings and generalization across datasets. Our key contributions are as follows:

- We propose a novel task for detecting knowledge edits in LLMs (DEED) to combat potential malicious model editing. DEED highlights the transparency of LLMs' generations by suggesting the source of the generated text (edited vs. unedited) (Section 4).

- We instantiate the task with two knowledge-editing datasets (ZSRE, COUNTERFACT), four state-of-the-art knowledge editing methods (ROME, MEMIT, MEND, MALMEN), and two LLMs (GPT-J, GPT2-XL) yielding 16 settings for initial analysis. The framework is easily extensible to other KEs and LLMs (Section 4.2).

- We analyze the hidden state representations and probability distributions of edited and unedited facts. In locate-and-edit KEs we find a strong increase in output probabilities of edited facts and linear separability in hidden state representations between edited and unedited facts, while other KEs induce more subtle changes (Section 5.1).

- We show that simple AdaBoost classifiers, with the hidden state representations and probability distributions from the edited LLMs as features, establish strong baselines, while requiring a limited amount of training data (Section 5.2). Additionally, we show that these baselines generalize to cross-domain settings, highlighting their practical utility (Section 6).

2 Related Work

This paper is the first to establish the task of detecting knowledge edits in pre-trained language models. Still, a wide array of work studies related questions, including how knowledge is stored in LLMs (Gurnee and Tegmark, 2023; Gurnee et al., 2024; Niu et al., 2024), how knowledge is probed (Youssef et al., 2023), how to inject new knowledge (Xu et al., 2023) and how to unlearn existing knowledge (Yu et al., 2023a; Kassem et al., 2023; Jang et al., 2023; Patil et al., 2024). For brevity, we direct readers to surveys by Mazzia et al. (2023) and Wang et al. (2023) for KE-related work. This study focuses on the post-editing phase, i.e., after applying KEs that directly modify model parameters.

2.1 Knowledge Editing Methods

Generally KEs can be divided into two groups: 1) parameter-preserving methods and 2) parameter-modifying methods.

Parameter-preserving methods integrate extra modules, e.g., additional memory space or parameters, while leaving the original parameters unchanged (Yao et al., 2023; Li et al., 2024b). SERAC (Mitchell et al., 2022b) stores new knowledge in an explicit cache and uses an auxiliary classifier to determine if a prompt is associated with the edited knowledge. If it is, SERAC passes the prompt along with the stored new knowledge to a "counterfactual" model for the inference. GRACE (Hartvigsen et al., 2023) caches embeddings for old knowledge and the weights for new knowledge and adds an adaptor to manipulate the layer-to-layer transformations without altering model weights. MELO (Yu et al., 2023b) learns and stores new knowledge in inserted LoRA blocks (Hu et al., 2021). We exclude parameter-preserving KEs from our study, since for these KEs it is possible to undo the edits by simply removing the external modules from the model.

Parameter-modifying methods modify the model parameters directly via meta-learning or optimization-based methods. Meta-learning KEs (ML-KEs) employ hyper-networks to learn parameter shifts towards new knowledge. KNOWLEDGE EDITOR (De Cao et al., 2021) and MEND (Mitchell et al., 2022a) build light-weight hyper-networks, i.e., naive multi-layer perceptrons that take the gradients of desired input-output (new knowledge) and output the parameter shifts for individual layers. However, the parameter shifts caused by different edits can be contradictory to each other, leading to a cancellation effect (Tan et al., 2024). To handle the cancellation effect in batch editing, MAL-MEN (Tan et al., 2024) treats aggregating parame-

ter shifts as a least square problem, seeking for the parameter shift to be effective for all edited facts.

Optimization-based, also known as locate-and-edit methods (LE-KEs), assume that knowledge is memorized in a key-value form in the feed-forward network. These methods, such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), locate and then directly optimize the parameters in the feed-forward network to modify or add memories.

Note that fine-tuning the model on specific data roughly falls into the category of parameter-modifying KEs, but requires great computational resources and data curation costs, and may encounter catastrophic forgetting and over-fitting (Mitchell et al., 2022a,b; Zheng et al., 2023). These negative effects may result in an unusable model in practice. We leave detecting edits that have been induced through finetuning to future work, and focus on methods that have been specifically developed to edit knowledge.

2.2 Malicious Modifications of LLMs

Recent advancements in LLMs have led to their widespread adoption. However, these models are vulnerable to malicious attacks (Wei et al., 2024; Mo et al., 2024) and poisoning (Li et al., 2021; Cao et al., 2023; Liu et al., 2023a; Li et al., 2024b). Moreover, open-source LLMs are available for free download and unrestricted modification, facilitating easy dissemination of modified versions across online platforms (Falade, 2023; Singh et al., 2023; Yao et al., 2024).

With the development of KEs, knowledge of LLMs can be modified easily with low cost, and at scale. This development presents opportunities for misuse, such as inserting misinformation and toxic content into LLMs. For example, BadEdit (Li et al., 2024b) formulates backdoor injection as a lightweight knowledge editing problem, i.e., modifying a subset of parameters to inject backdoors into LLMs. Due to the high cost of training LLMs, users are likely to seek readily available similar models on public resources such as GitHub and HuggingFace. Even though these third-party models might offer improved skills and advanced features (e.g., by fine tuning with Reinforcement Learning from Human Feedback (RLHF) or specific datasets), they could also be mingled with malicious edits before being made public (Shi et al., 2023; Pan et al., 2023; Li et al., 2024b).

3 Preliminaries

Facts[2] in LLMs are commonly represented as triplets of $(subject, relation, object)$, or (s, r, o) for short. Querying an LLM with a prompt $p(s, r)$, where p expresses the relation r and contains the subject s (e.g., "The Eiffel Tower is in the city of"), should result in retrieving the object o (e.g., "Paris"), given that the fact (s, r, o) is known to the LLM. A knowledge editing operation $E(s, r, o, o', p)$ is successful if it changes the behavior of the LLM such that the retrieved object is o', as desired, instead of o. If an edit is conducted with a prompt p, then ideally it should also affect semantically similar (paraphrased) prompts that express that same relation, i.e., the new object o' should be retrievable with $p'(s, r)$ as well, where p' is a paraphrase of p. We later use only paraphrased prompts to detect edited facts, since we assume no access to the prompts that have been used for editing. Paraphrased prompts are available in both datasets we consider.

4 DEED Task

The goal of detecting edited facts is to distinguish outputs generated based on post-hoc modifications to the model (edited knowledge) from those derived from the knowledge acquired by the model during its pre-training phase (unedited knowledge). In this section, we introduce the task of **de**tecting knowledge **ed**its in LLMs (DEED) in detail (Section 4.1), describe how we instantiate the task (Section 4.2), and discuss practical applications (Section 4.3).

4.1 DEED General Setup

Given an edited model and a prompt that retrieves a certain fact from the model, the objective of DEED is to decide whether the fact is based on pre-training, i.e., *unedited*, or based on subsequent editing, i.e., *edited*.

More formally, we model the task of detecting knowledge edits as a binary classification problem with classes $\mathcal{Y} = \{edited, unedited\}$. Given an LLM $\mathcal{M}$ with unknown edits, and a pool of facts $\mathcal{T}_{test} = \{k_1, .., k_n\}$, where $k_i = (s_i, r_i, o_i, p_i)$, for $1 \leq i \leq n$, the task is to assign each fact k_i a label $y \in \mathcal{Y}$.

In a supervised setting, the DEED classifier has access to a training set $\mathcal{T}_{train}$ with examples of edited and unedited facts. The facts are represented

---

[2]We use the terms *facts* and *knowledge* interchangeably.

as a set of features $\mathcal{F}$. Conceptually, one can categorize the task based on the origin of the test set and the type of features; their combination defines the difficulty of the task.

Origin of the test set. If the test set is derived from the original edited model, i.e., $\mathcal{M} = \mathcal{M}_e$ (used also for training), our task is In-Domain (ID) classification. When the test set originates from a similar LLM $\mathcal{M} = \mathcal{M}'_e$, which shares the same foundation model with $\mathcal{M}_e$ (the model used to train the classifier), our task is Cross-Domain (CD) classification. For example, $\mathcal{M}_e$ and $\mathcal{M}'_e$ can be two different instruction-tuned versions of the same foundation model (cf. Figure 1).

Type of features. We consider two cases corresponding to whether we have full access to the model or only can observe its output distribution. Features can therefore either be obtained from the hidden states (HS), or from the output probability distribution (PD) of the LLM $\mathcal{M}$. Additionally, we consider having both features (HS + PD).

Practical considerations. A few state-of-the-art LLMs (e.g., GPT-4 (Achiam et al., 2023)) are only accessible via their APIs. Nonetheless, the number of high-performing open-source LLMs is steadily increasing (Liu et al., 2023b). The availability of these open-source LLMs allows users to further fine-tune and modify them locally, with access to their hidden states and probability distribution. In theory, the probability distribution can even be approximated for closed-source LLMs as well if enough outputs are sampled from these LLMs given a fixed input.

In the ID setting, the user knows at least a subset of the edited facts, and these known edited facts are used to construct the training dataset. However, in real-world settings we want to identify whether facts were edited and it is unrealistic to assume a known set of of edited facts. Therefore, we consider CD settings where the user does not know which facts have been edited in the model $\mathcal{M} = \mathcal{M}'_e$. Nevertheless, users can simulate the edit scenario by editing $\mathcal{M}_e$, an LLM of the same foundation model as $\mathcal{M}'_e$, where the edited facts in $\mathcal{M}_e$ are known. The Cross-domain box in Figure 1 illustrates this setting.

4.2 DEED Constituents

The DEED task is defined by three constituents: 1) The edited LLM; 2) The KE used for editing; 3) The dataset used for editing facts. Here, we describe each constituent and the construction of the datasets for DEED in detail.

Edited LLMs. Following previous work (Meng et al., 2022, 2023), we incorporate GPT-2 XL (1.5B) (Radford et al., 2019) and GPT-J (6B) (Wang and Komatsuzaki, 2021) as our base language models. Both are decoder-only autoregressive LLMs.

KEs. We include four parameter-modifying KEs of two sub-categories, MEND (Mitchell et al., 2022a) and MALMEN (Tan et al., 2024) for meta-learning KEs, and ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) for locate-and-edit KEs. We focus on parameter-modifying KEs, since for parameter-preserving KEs, it is possible to undo the edits by simply removing the external modules from the model.

Datasets. We edit models with two popular datasets, ZSRE and COUNTERFACT. ZSRE (Levy et al., 2017; Mitchell et al., 2022a) is a Question Answering dataset, where each instance contains a factual statement and a paraphrased prompt. ZSRE originally contains 244,173 training and 27,644 validation instances. COUNTERFACT (Meng et al., 2022) includes counterfactual statements, i.e., false facts, that are used for editing. COUNTERFACT contains 21,919 records for 20,391 subjects and 749 objects.

DEED task construction. We edited two LLMs (GPT2-XL, GPT-J), with four KEs (ROME, MEMIT, MEND, MALMEN) using facts from ZSRE and COUNTERFACT. This yields a total of 16 edited models (4 KEs, 2 LLMs, 2 datasets). Since our objective is to detect edited facts, we filter out unsuccessful edits. More specifically, for each edited model we collect successfully edited facts, i.e., facts for which the new object o' can be correctly retrieved with a paraphrased prompt $p'(s, r)$. These are facts, for which o' has the highest probability given the prompt $p'(s, r)$. More formally, $o' = argmax_o \, \mathbb{P}_{\mathcal{M}, p'(s,r)}[o]$. We report the number of successful edits that we consider in our experiments in Table 1. Note that, in addition to the dataset comprising edited and unedited facts, we assume access to the edited LLM. Access to the edited LLM enables users to explore various detection methods, such as employing HS and PD as features for the classification task.

Implementation details. We edit 1,000 facts in each setting (LLM, KE, dataset).[3] We consider 1,000 edits to be a realistic number for malicious editing attacks, and defer the investigation of this choice to future work. For ROME, we perform one edit at a time on the base language model as ROME is not designed for mass editing or sequential editing (Huang et al., 2022; Yao et al., 2023). For MEMIT, we edit each model with 1,000 facts at once. We use the hypernetworks for GPT-2 XL and GPT-J published by Meng et al. (2022) to circumvent the substantial compute required by MEND. For MEND, we also conduct one edit at a time to achieve a high success rate (Meng et al., 2023). For MALMEN, we train the hyper-networks before editing and then perform 1,000 edits to the model simultaneously. We adhere to the default settings employed in the original papers. We include the unedited base model, NONE, as a yardstick. We follow related work (Meng et al., 2022, 2023; Li et al., 2024a) in evaluating editing performance. See Appendix A for further details on editing metrics and performance.

Base LLM	KE	ZSRE	COUNTERFACT
GPT2-XL	ROME	736	722
	MEMIT	324	417
	MEND	765	756
	MALMEN	832	127
GPT-J	ROME	876	837
	MEMIT	800	622
	MEND	822	758
	MALMEN	878	297

Table 1: The number of successful edits, which we include in the detection datasets for each setting.

4.3 Practical Applications

Detecting knowledge edits can have significant practical applications:

Transparency. Detecting edits provides users with insights into whether model outputs originate from pre-training or post-deployment edits. This enhances the transparency of AI systems.

Model debugging. By distinguishing between pre-training and edited knowledge, we can identify whether an issue stems from unclean pre-training data or a maliciously inserted edit. For example, if a factual error is flagged as an edit, developers can focus on investigating unauthorized modifications rather than pre-training data.

Mitigating malicious use cases. Recent works highlight how knowledge editing can be used to bias LLMs (Chen et al., 2024), introduce backdoors (Li et al., 2024b; Qiu et al., 2024), or propagate misinformation in multi-agent settings (Ju et al., 2024). Detecting knowledge edits helps identify and mitigate these threats by pinpointing unauthorized changes, ensuring the model's reliability in adversarial environments.

5 DEED Baseline Models

To train a detection classifier, we consider different options of available features. Inspired by the probing tasks introduced in Conneau et al. (2018) and the representations inspection by Hernandez et al. (2023), we investigate the use of hidden state representations as features. Additionally, in Kuhn et al. (2023) and Gu et al. (2023), the output probability distributions are used to provide signals for latent semantics. Therefore, we also explore using these distributions as features.

5.1 Preliminary Analysis

HS visualization. To see how editing affects the hidden state representations, we use Linear Discriminant Analysis (LDA) (Duda et al., 2000) to visualize the hidden state representations. Specifically, for each data point (fact) from the test set, we project the hidden state representation of the last token of $p'(s, r)$ into a one-dimensional space. We use the representations from the last layer of the respective LLM. We assign the one-dimensional representations random values on the y-axis to reduce point overlap. As shown in Figure 2, the edited and unedited representations are not distinguishable in the unedited model. Editing with ROME and to less extent with MEMIT makes the representations more separable. We attribute this separability to the high confidence of LLMs in the edited facts, as shown later in Section 5.3. Editing with MEND and MALMEN keeps the representations linearly inseparable.[4] This suggests that meta-learning KEs cause subtle changes on representations when editing facts. Similar patterns are observed on all datasets and methods (see Figure 6 in the Appendix).

---

[3]Increasing the number of edits did not lead to more successful edits with *all* KEs.

[4]We also experimented with PCA and t-SNE, but the representations were still inseparable.

PD visualization. We investigate the output probability distributions for the next token from LLMs when prompting for edited and unedited facts with prompts that contain subjects and relations: $p'_1(s_1, r_1)...p'_n(s_n, r_n)$. We consider a balanced set consisting of the same number of edited and unedited facts. We analyze the probability distribution of each set by calculating the mean of the top 10 probabilities and plotting the kernel density estimate (KDE). We show the KDEs for GPT-J and COUNTERFACT in Figure 3. The figure shows almost identical distributions of low probabilities for both edited and unedited facts in the unedited model NONE. We notice two peaks for both distributions in NONE: the left one is close to zero, while the right one is around 0.3. As we edit with different methods, we notice that the right peak for edited facts is shifted to the right, while the first one flattens, as the probabilities there become less frequent. The increase in probabilities varies depending on the method; ROME and MEND cause the highest increase, followed by MEMIT and MALMEN, which are used for mass-editing. We also notice that MEND increases the probabilities of unedited facts as well. Editing with MALMEN is the smoothest, since its probability distribution is the closest to that of the unedited facts. This potentially makes facts edited with MALMEN more difficult to detect.

5.2 DEED Baselines

Given a fact (s, r, o, p'), we prompt the given LLM $\mathcal{M}$, which consists of L transformer blocks, with the prompt $p'(s, r)$ that consists of N tokens $\{u_1, \ldots, u_N\}$. We use the hidden state representations HS from the last transformer block and the last input token $h_{N,L} = \mathcal{M}[p'(s, r)]$ as a first feature for detecting edited facts. We use the information from the last token, since this is the position where the information about the subject and the relation is condensed (Geva et al., 2023). As a second feature, we use PD, the top-1000 probabilities of the next token $q = (q_1, \ldots, q_{1000})$ from $\mathcal{M}$ on the same prompt $p'(s, r)$.

5.3 Evaluation

We train an AdaBoost classifier (Freund and Schapire, 1997) with a decision tree as a base classifier using both features, HS and PD, on the DEED datasets (Section 4.2). We show the classification performance of training on 126 instances in the first part of Table 2. The minimum number of edits under all settings was 127 (cf. Table 1). We use half of the edited facts (63) for training, and the rest for testing. Balancing the edited 63 facts with 63 unedited facts yields a training set of 126 examples. We keep the size of the training set constant across all settings for fair comparison, and experiment later with smaller and larger training sets (cf. Figure 4). As a sanity check, we train a classifier on the same sets of edited and unedited facts using features from the base *unedited* model. We refer to this classifier as NONE. The classifier NONE shows a fair coin performance on the three metrics, ranging from 46.2 (GPT-J, COUNTERFACT, F1) to 49.8 (GPT2-XL, ZSRE, F1). This verifies the correctness of our setup. We report precision, recall, and F1.

Locate-and-edit KEs vs. meta-learning KEs. We find that the classifier is able to detect edits by locate-and-edit KEs (LE-KEs), i.e., ROME and MEMIT, relatively well, achieving a minimum F1 score of 74.2% (MEMIT, GPT2-XL, ZSRE). The highest F1 score of 98% is achieved on (ROME, GPT-J, ZSRE). On meta-learning KEs (ML-KEs), i.e., MEND and MALMEN, we observe lower performance in general. On MEND the performance varies between 53.6% (GPT2-XL, COUNTERFACT) and 81.4% (GPT-J, ZSRE), whereas on MALMEN the gap between the lowest and highest performance is smaller than the same gap on MEND (62% (GPT2-XL, COUNTERFACT), and 76% (GPT-J, ZSRE)). In general, this shows that *edits, from the more computationally efficient LE-KEs, are easier to detect*. One reason for this discrepancy in performance between these two classes of methods might be that ML-KEs induce changes that generally target the last few layers, while LE-KEs change middle layers. Note for example that MEND increases the probabilities for both edited and unedited facts, while MALMEN slightly increases the probabilities for edited facts (cf. Figure 3). This keeps the distance between edited and unedited facts minimal under ML-KEs, while LE-KEs conduct surgical changes that increase the probability for edited facts only. This can also be confirmed by the fact that the effects of LE-KEs on unrelated facts are less pronounced compared to ML-KEs (cf. Table 4, Appendix).

Ablation. We conduct an ablation study, where we train only on HS, or PD (second and third part of Table 2 respectively). We notice that using HS leads to low performance on the ML-KEs meth-
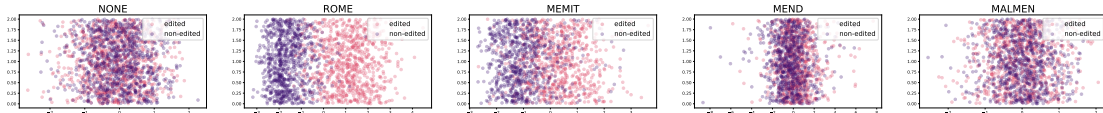
Figure 2: Representation of edited and unedited facts (ZSRE, GPT-J). X-axis: 1D LDA projection, Y-axis: random values to reduce point overlap.
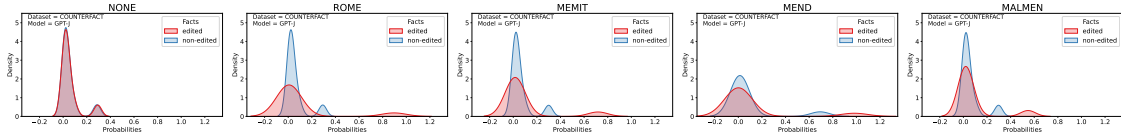

Figure 3: Kernel density estimation for averaged top-10 probabilities for the next token (GPT-J, COUNTERFACT).

ods, where the F1 score varies between 48.5% (MALMEN, GPT2-XL, ZSRE) and 62.6% (MAL-MEN, GPT-2XL, COUNTERFACT). On LE-KEs, HS still performs well with a minimum F1 score of 65.3% (MEMIT, GPT-J, COUNTERFACT). In contrast, *detecting edits with* PD *is more robust and performs well on LE-KEs and ML-KEs*. The minimum F1 scores with PD is 65.2% (MALMEN, GPT-J, COUNTERFACT). PD's robustness might be explained by the fact that it is derived from a late position in the LLM, namely the language modeling head, which is responsible for the final outputs. Another important observation is that using PD individually can lead to better performance than combining it with HS in some cases (e.g., the F1 score on (MEND, GPT2-XL, COUNTERFACT) is 53.6% when using both features, but 73% when using only PD). This observation suggests that PD and HS could be combined in better ways. We leave exploring the interplay between HS and PD to future work. We further explore using HS from different layers in Appendix A.1, observing that the representations start becoming separable after the edited layers, and that the representations from some layers (e.g., the penultimate layer) perform better than the representations of the last layer.
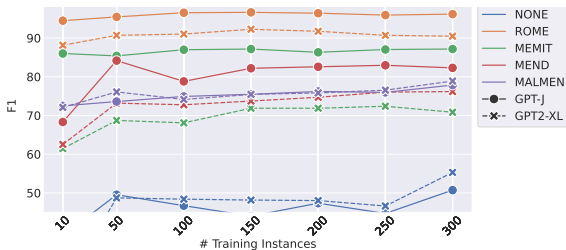
Figure 4: Detection performance with PD on ZSRE when varying the training set size.

Training set size. Figure 4 shows the performance of different training set sizes with PD. Exhaustive results are presented in Table 7 in the Appendix. In general, the classification performance starts reaching a plateau after 50 training instances. For example, on MEMIT-edited facts in GPT-J (green circles), F1 reaches 85.4% after 50 training instances, while the highest F1 of 87.2% is achieved after 300 training data points. Therefore, we emphasize that *this classifier remains effective even when the training data consists of less than one hundred examples. We also observe that facts that are edited with ML-KEs are not only harder to detect than facts edited with LE-KEs (as noted before), but also that scaling the size of the training set does not improve performance.*

6 Cross-Domain DEED

Obtaining a training set for the detection classifier requires knowing instances of edited facts in the edited model. However, this assumption may not always hold. Consider a scenario where an open-source model is further fine-tuned for a specific task of interest to the user, yet no explicit information is provided regarding the edited facts. Many models are fine-tuned with instructions or reinforcement learning from human feedback (RLHF), and are made publicly available (Lai et al., 2023). In these cases, we have access to the original model (from which we could obtain our training set), but do not know which facts have been edited into the fine-tuned model. To address such cases, we assess the performance of the classifier, which uses HS and PD, in a cross-domain setting, where the classifier is trained on representations from one edited model and then applied to detecting knowledge edits from a separate edited model (which is a fine-tuned version of the original model). The CD edited models

| | | Hidden States + Prob. Dist. (HS + PD) | | | | | | Hidden States (HS) | | | | | | Probability Distribution (PD) | | | | | |
| | | zsRE | | | COUNTERFACT | | | zsRE | | | COUNTERFACT | | | zsRE | | | COUNTERFACT | | |
	KE	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1
GPT-J	NONE	49.4	46.6	48.0	48.6	44.1	46.2	49.4	46.6	48.0	48.6	44.1	46.2	50.7	42.6	46.3	51.4	49.6	50.5
	ROME	98.6	97.4	**98.0**	91.2	90.2	<u>90.7</u>	98.0	95.6	96.8	87.1	84.9	86.0	98.5	94.5	96.4	89.3	89.2	89.2
	MEMIT	87.7	89.8	**88.7**	79.0	76.7	<u>77.9</u>	79.0	81.6	80.2	70.8	60.6	65.3	83.8	89.8	86.7	78.0	73.4	75.6
	MEND	78.4	84.6	81.4	59.9	58.8	59.4	61.6	59.3	60.4	59.9	58.8	59.4	78.3	85.9	**81.9**	63.5	96.1	<u>76.5</u>
	MALMEN	76.6	75.3	**76.0**	62.0	62.0	62.0	49.9	49.6	49.7	57.3	56.8	57.1	71.2	77.1	74.0	66.8	63.7	<u>65.2</u>
GPT2-XL	NONE	49.7	50.0	49.8	51.4	48.0	49.6	49.3	44.8	47.0	51.4	48.0	49.6	51.0	50.2	50.6	49.6	48.0	48.8
	ROME	95.0	98.2	**96.6**	91.8	93.6	<u>92.7</u>	94.7	97.6	96.1	91.3	92.1	91.7	94.6	90.2	92.3	87.6	91.4	89.4
	MEMIT	77.7	70.9	**74.2**	76.0	73.2	74.5	71.8	68.2	69.9	69.4	66.7	68.0	69.5	65.5	67.5	77.3	75.1	<u>76.2</u>
	MEND	67.0	78.9	72.5	55.1	52.2	53.6	62.6	53.7	57.8	54.3	50.8	52.5	66.4	88.0	**75.7**	60.7	91.5	<u>73.0</u>
	MALMEN	69.8	71.0	**70.4**	68.2	67.2	<u>67.7</u>	49.3	47.7	48.5	61.2	64.1	62.6	71.2	65.0	68.0	64.2	67.2	65.6

Table 2: Detection performance on zsRE and COUNTERFACT with different features sets. The training set has 126 instances in all settings. The best F1 across all feature sets is in **bold** for zsRE, and <u>underlined</u> for COUNTERFACT. The number of test instances is shown in Table 5 in the Appendix.
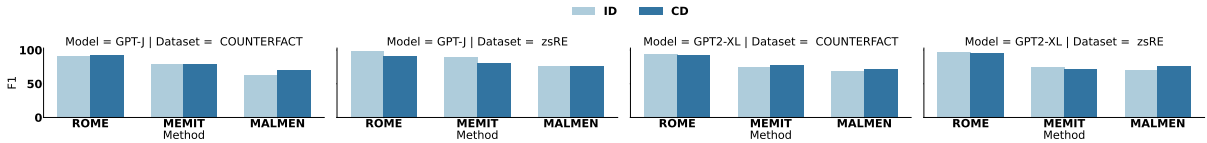

Figure 5: Comparison between in-domain (ID) and cross-domain (CD) classification performance. Classification is done based on HS and PD.

have the same architecture, but different weights. To simulate this scenario, we use representations from edited GPT-J and GPT2-XL to train the classifier and test it in detecting knowledge edits from instruction-tuned versions of GPT-J[5] and GPT2-XL.[6] Note that the facts in the training and test sets are still disjoint. For this experiment, we consider ROME, MEMIT and MALMEN.

As shown in Figure 5, *the performance on cross-domain edits (*CD*) is on par with performance on in-domain data (*ID*).* For example, on zsRE, CD F1 on ROME-edited facts on GPT2-XL lima is 95%, similar to the ID F1 of 96.6%. Full results with HS/PD are shown in Table 6 in the Appendix. This makes our DEED baseline highly practical, as it can be leveraged to detect edits in fine-tuned models, while making use of training data that is derived from their counterpart original models.

Cross-dataset generalization. The results of our previous experiment indicate that detection classifiers trained on edits in one model generalize to edits in other models of the same family. To further explore out-of-distribution scenarios, we impose an additional constraint by examining whether detection classifiers trained on edits from one dataset (e.g., COUNTERFACT) generalize to edits from an-

other dataset (e.g., zsRE). Here, we maintain the assumption from our previous experiment that the base LLM is known, but introduce a new constraint: the dataset used to conduct edits on the test LLM is unknown, necessitating training on one dataset and testing on another. The primary objective of this experiment is to investigate whether edit detection classifiers generalize effectively across datasets.

As shown in Table 3 the detection classifiers generalize well to unseen datasets in most cases. One exception is generalization from zsRE to COUNTERFACT with MALMEN, where we see a dramatic decrease in performance. One reason for this might be the smooth changes induced by MALMEN (cf. Section 5.1). These changes are dataset-specific, which makes generalization more difficult (cf. Figure 9b and 9c in the Appendix).

	Method	zsRE	COUNTERFACT
GPT-J	ROME	82.7	89.6
	MEMIT	73.4	76.8
	MALMEN	63.7	25.9
GPT2-XL	ROME	84.5	90.8
	MEMIT	70.4	72.7
	MALMEN	62.1	6.8

Table 3: F1 scores in the cross-datasets setting. Classifiers are trained on HS and PD from one dataset and tested on another dataset in CD setting. Tests sets are shown in the header.

[5] https://huggingface.co/togethercomputer/GPT-JT-6B-v1

[6] https://huggingface.co/lgaalves/gpt2-xl_lima

7 Conclusion

In this paper, we introduced a novel task: detecting knowledge edits in language models. We see promising applications in uncovering potential malicious model modification and enhancing the transparency of LLMs' generations. We instantiated the task with several baselines, which use hidden state representations and probability distributions as input features. These baselines demonstrate a straightforward yet effective approach to the task. Our evaluation across two LLMs, four editing techniques (2 LE-KEs and 2 ML-KEs), and two datasets, shows that the classifiers can reliably detect edits, even in scenarios characterized by limited training data. Moreover, we showed that these baselines generalize to cross-domain settings, where training and test data come from different models and datasets, highlighting their practical utility. Our work gives rise to several future work directions: 1) detecting if an LLM has undergone any edits at all; 2) studying detection generalization across KEs; 3) investigating more constrained settings, where one has only access to the LLM's outputs; 4) combating malicious edits.

8 Limitations

Access to HS and PD. The baseline detection classifier we proposed requires access to the hidden states or probability distribution. When using closed source LLMs, these features might not be available. Inspired by Rosati et al. (2024), we plan to use the model behaviors shift on prompts with similar semantics but in different lengths to separate edited and unedited knowledge.

Unknown editing methods. In our current setting, we assume we know the editing method. However, the editing method might not be known and an edited model can be modified by multiple editing methods. In future work, we will investigate the generalization of our baseline classifiers between different KEs, and detecting edits in models that have edited with several KEs.

Number of edits. Previous work shows that a larger number of edits often leads to a worse editing performance (Meng et al., 2023). Although we only keep successful edits for the detection task, it is unknown whether and how the number of edits will impact the detection performance. We plan to study this aspect in future work.

Acknowledgment

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yuanpu Cao, Bochuan Cao, and Jinghui Chen. 2023. Stealthy and persistent unalignment on large language models via backdoor injections. *arXiv preprint arXiv:2312.00027*.

Canyu Chen, Baixiang Huang, Zekun Li, Zhaorun Chen, Shiyang Lai, Xiongxiao Xu, Jia-Chen Gu, Jindong Gu, Huaxiu Yao, Chaowei Xiao, Xifeng Yan, William Yang Wang, Philip Torr, Dawn Song, and Kai Shu. 2024. Can editing LLMs inject harm? In *Trustworthy Multi-modal Foundation Models and AI Agents (TiFA)*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single \$&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

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

RO Duda, PE Hart, DG Stork, and Alexandru Ionescu. 2000. *Pattern classification, chapter nonparametric techniques*. Wiley-Interscience Publication,.

Polra Victor Falade. 2023. Decoding the threat landscape: ChatGPT, FraudGPT, and WormGPT in social engineering attacks. *arXiv preprint arXiv:2310.05595*.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore. Association for Computational Linguistics.

Yuxuan Gu, Xiaocheng Feng, Sicheng Ma, Lingyuan Zhang, Heng Gong, Weihong Zhong, and Bing Qin. 2023. Controllable text generation via probability density estimation in the latent space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12590–12616, Toronto, Canada. Association for Computational Linguistics.

Wes Gurnee, Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway, Neel Nanda, and Dimitris Bertsimas. 2024. Universal neurons in GPT2 language models. *arXiv preprint arXiv:2401.12181*.

Wes Gurnee and Max Tegmark. 2023. Language models represent space and time. *arXiv preprint arXiv:2310.02207*.

Karina Halevy, Anna Sotnikova, Badr AlKhamissi, Syrielle Montariol, and Antoine Bosselut. 2024. " flex tape can't fix that": Bias and misinformation in edited language models. *arXiv preprint arXiv:2403.00180*.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems*.

Evan Hernandez, Belinda Z Li, and Jacob Andreas. 2023. Inspecting and editing knowledge representations in language models. *arXiv preprint arXiv:2304.00740*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2022. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2023. Knowledge unlearning for mitigating privacy risks in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14389–14408, Toronto, Canada. Association for Computational Linguistics.

Tianjie Ju, Yiting Wang, Xinbei Ma, Pengzhou Cheng, Haodong Zhao, Yulong Wang, Lifeng Liu, Jian Xie, Zhuosheng Zhang, and Gongshen Liu. 2024. Flooding spread of manipulated knowledge in llm-based multi-agent communities.

Aly Kassem, Omar Mahmoud, and Sherif Saad. 2023. Preserving privacy through dememorization: An unlearning technique for mitigating memorization risks in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4360–4379, Singapore. Association for Computational Linguistics.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Viet Dac Lai, Chien Van Nguyen, Nghia Trung Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan A Rossi, and Thien Huu Nguyen. 2023. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. *arXiv preprint arXiv:2307.16039*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. 2021. Backdoor attacks on pre-trained models by layerwise weight poisoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3023–3032, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024a. PMET: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572.

Yanzhou Li, Kangjie Chen, Tianlin Li, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. 2024b. Badedit: Backdooring large language models by model editing. In *The Twelfth International Conference on Learning Representations*.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. 2023a. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*.

Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Tan, Tianhua Tao, Junbo Li, Yuqi Wang, Suqi Sun, Omkar Pangarkar, et al. 2023b. LLM360: Towards fully transparent open-source LLMs. *arXiv preprint arXiv:2312.06550*.

Vittorio Mazzia, Alessandro Pedrani, Andrea Caciolai, Kay Rottmann, and Davide Bernardi. 2023. A survey on knowledge editing of neural networks. *arXiv preprint arXiv:2310.19704*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. Mass editing memory in a transformer. *The Eleventh International Conference on Learning Representations (ICLR)*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Lingbo Mo, Boshi Wang, Muhao Chen, and Huan Sun. 2024. How trustworthy are open-source llms? an assessment under malicious demonstrations shows their vulnerabilities.

Jingcheng Niu, Andrew Liu, Zining Zhu, and Gerald Penn. 2024. What does the knowledge neuron thesis have to do with knowledge? In *The Twelfth International Conference on Learning Representations*.

Yikang Pan, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, and William Wang. 2023. On the risk of misinformation pollution with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1389–1403, Singapore. Association for Computational Linguistics.

Vaidehi Patil, Peter Hase, and Mohit Bansal. 2024. Can sensitive information be deleted from LLMs? objectives for defending against extraction attacks. In *The Twelfth International Conference on Learning Representations*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Jiyang Qiu, Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. Megen: Generative backdoor in large language models via model editing.

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.

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

Domenic Rosati, Robie Gonzales, Jinkun Chen, Xuemin Yu, Yahya Kayani, Frank Rudzicz, and Hassan Sajjad. 2024. Long-form evaluation of model editing. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3749–3780, Mexico City, Mexico. Association for Computational Linguistics.

Jiawen Shi, Yixin Liu, Pan Zhou, and Lichao Sun. 2023. Badgpt: Exploring security vulnerabilities of chatgpt via backdoor attacks to instructgpt. *arXiv preprint arXiv:2304.12298*.

Sonali Singh, Faranak Abri, and Akbar Siami Namin. 2023. Exploiting large language models (llms) through deception techniques and persuasion principles. In *2023 IEEE International Conference on Big Data (BigData)*, pages 2508–2517. IEEE.

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Wieting, Nanyun Peng, and Xuezhe Ma. 2023. Evaluating large language models on controlled generation tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3155–3168, Singapore. Association for Computational Linguistics.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning. In *International Conference on Learning Representations*.

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

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. 2023. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2024. Jailbroken: How does LLM safety training fail? *Advances in Neural Information Processing Systems*, 36.

Yan Xu, Mahdi Namazifar, Devamanyu Hazarika, Aishwarya Padmakumar, Yang Liu, and Dilek Hakkani-Tur. 2023. KILM: Knowledge injection into encoder-decoder language models. In *Proceedings of the*

61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5013–5035, Toronto, Canada. Association for Computational Linguistics.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Paul Youssef, Osman Koraş, Meijie Li, Jörg Schlötterer, and Christin Seifert. 2023. Give Me the Facts! A Survey on Factual Knowledge Probing in Pre-trained Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15588–15605, Singapore. Association for Computational Linguistics.

Paul Youssef, Jörg Schlötterer, and Christin Seifert. 2024. The Queen of England is not England's Queen: On the Lack of Factual Coherency in PLMs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2342–2354, St. Julian's, Malta. Association for Computational Linguistics.

Paul Youssef, Zhixue Zhao, Daniel Braun, Jörg Schlötterer, and Christin Seifert. 2025. Position: Editing Large Language Models Poses Serious Safety Risks.

Charles Yu, Sullam Jeoung, Anish Kasi, Pengfei Yu, and Heng Ji. 2023a. Unlearning bias in language models by partitioning gradients. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6032–6048, Toronto, Canada. Association for Computational Linguistics.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2023b. Melo: Enhancing model editing with neuron-indexed dynamic lora. *arXiv preprint arXiv:2312.11795*.

Zhixue Zhao, Ziqi Zhang, and Frank Hopfgartner. 2021. A comparative study of using pre-trained language models for toxic comment classification. In *Companion Proceedings of the Web Conference 2021*, pages 500–507.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876, Singapore. Association for Computational Linguistics.

A Complementary Results

In Table 5, we report the number of training and test instances used in our default setting. We keep the number of training instances constant across all settings for consistency. Results for varying sizes of the training set are presented in Table 7 and Figure 8 (COUNTERFACT). The result for the CD evaluation is shown in Table 6. We show some prompt examples that are used for editing and evaluating edits in Table 10. We show the HS representations in a low-dimensional space in Figure 6. We show KDE visualizations of PD across all datasets, models, and KEs in Figure 9.

Editing performance scores are shown in Table 4. Editing methods are typically evaluated in terms of Efficacy, Generalizability and Locality Success. Efficacy Success (**ES**) is the primary evaluation measure and refers to the ability to change the LLM's predictions. For example, for the prompt $p(s, r)$ "The Eiffel Tower is in the city of" originally retrieving object o "Paris", an edit is successful, if afterwards the desired object o' (e.g. "Berlin") is retrieved. Ideally, such edits should also affect semantically similar prompts $p'(s, r)$, e.g., "The city where the Eiffel Tower is located is". The ability of edits to generalize to similar prompts is referred to as **GS** (Generalizability Success). Edits should not affect other facts. **LS** (Locality Success, also known as Specificity) (Mitchell et al., 2022b; Meng et al., 2022; Li et al., 2024a) measures the accuracy of the model on a set of unrelated facts.

Generator		ZSRE			COUNTERFACT		
Model	Editor	ES↑	GS↑	LS	ES↑	GS↑	LS
GPT-J	NONE	28.1	27.4	26.1	13.3	18.6	81.0
	ROME	99.5	94.4	25.5	100.0	99.8	79.4
	MEMIT	98.7	90.6	35.4	99.4	90.8	78.8
	MEND	92.6	89.0	20.3	91.4	79.5	70.4
	MALMEN	99.8	94.8	44.3	97.8	60.9	66.2
GPT2-XL	NONE	23.2	22.1	22.3	18.9	24.1	74.6
	ROME	99.8	87.7	22.5	99.9	98.1	75.2
	MEMIT	68.8	58.1	27.6	93.7	81.1	74.8
	MEND	91.2	89.2	15.0	91.9	81.5	65.3
	MALMEN	99.7	93.0	39.6	85.1	44.8	66.3

Table 4: Summary of editing performance over 1000 edits.

A.1 HS from Different Layers

To investigate the effect of HS from different layers on the classification performance, we train a simple logistic regression classifier with L1 reg-

	ROME		
MEMIT			
MEND			
MALMEN			
(a) COUNTERFACT and GPT-J.	(b) COUNTERFACT and GPT2-XL.	(c) ZSRE and GPT2-XL.	
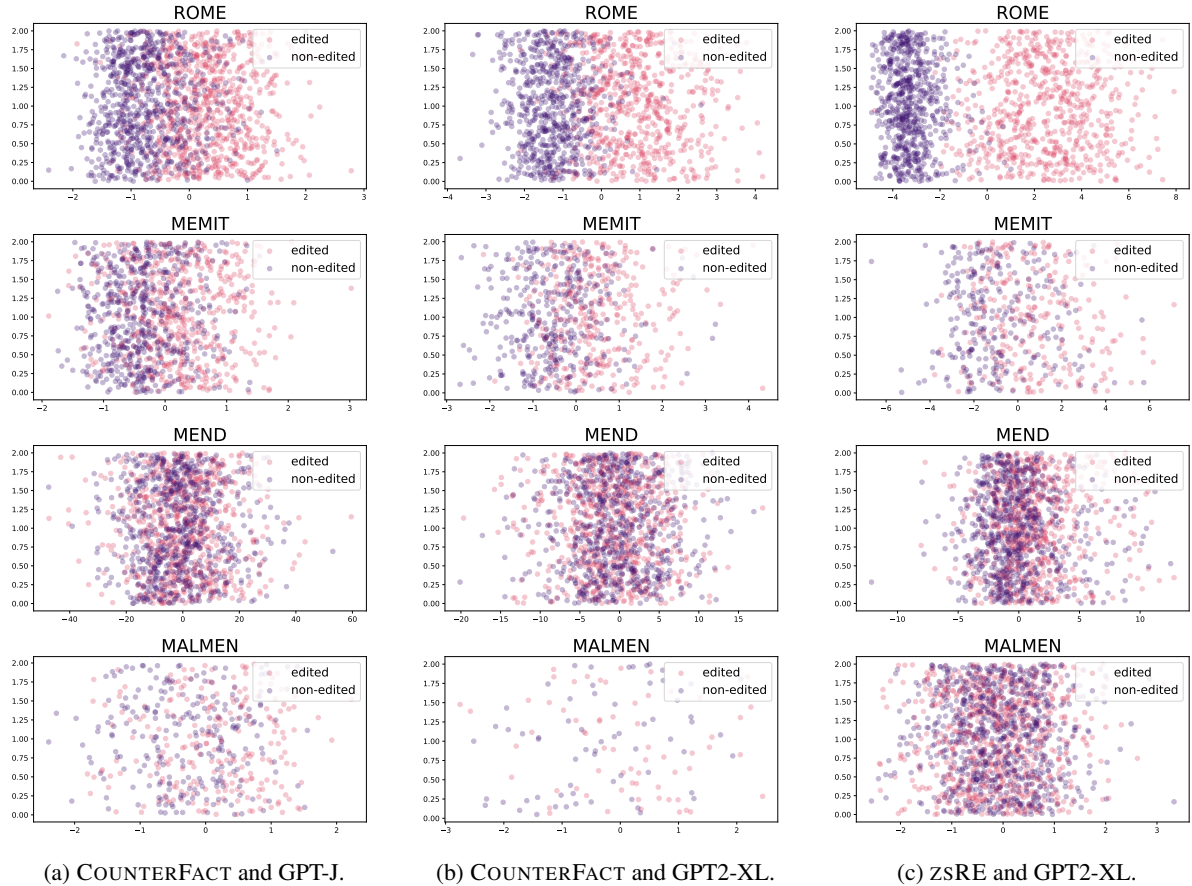
Figure 6: Respresentations of edited and non-edited facts.

model	ds	method	#training instances	#test instances
gpt-j-6B	COUNTERFACT	MALMEN	126	468
gpt-j-6B	COUNTERFACT	MEMIT	126	1118
gpt-j-6B	COUNTERFACT	MEND	126	1390
gpt-j-6B	COUNTERFACT	NONE	126	1874
gpt-j-6B	COUNTERFACT	ROME	126	1548
gpt-j-6B	ZSRE	MALMEN	126	1630
gpt-j-6B	ZSRE	MEMIT	126	1474
gpt-j-6B	ZSRE	MEND	126	1518
gpt-j-6B	ZSRE	NONE	126	1874
gpt-j-6B	ZSRE	ROME	126	1626
gpt2-xl	COUNTERFACT	MALMEN	126	128
gpt2-xl	COUNTERFACT	MEMIT	126	708
gpt2-xl	COUNTERFACT	MEND	126	1386
gpt2-xl	COUNTERFACT	NONE	126	1874
gpt2-xl	COUNTERFACT	ROME	126	1318
gpt2-xl	ZSRE	MALMEN	126	1538
gpt2-xl	ZSRE	MEMIT	126	522
gpt2-xl	ZSRE	MEND	126	1404
gpt2-xl	ZSRE	NONE	126	1874
gpt2-xl	ZSRE	ROME	126	1346

Table 5: Number of training and test instances used in our default setting. Both the training and test sets are balanced (50% for each class).

ularization on HS from different layers. We consider only locate-and-edit methods (ROME and MEMIT) for this experiment, since HS performs well on this category of methods. We show the F1 values based on hidden states representations from different layers in Figure 7. First, for both models, editing methods, and datasets, the representations start becoming separable, i.e., the classification performance is above random guess, after the edited layers or shortly thereafter. For example, the F1 for detecting ROME edits starts improving from the layer 7-8 in GPT-J and the layer 19 in GPT2-XL (ROME changes layer 7 in GPT-J, and layer 19 in GPT2-XL). We observe the same effect with MEMIT (MEMIT changes the layers 5-10 in GPT-J and the layers 15-19 in GPT2-XL). Even though the representations from the last layer perform well in detecting edits, we notice that the F1 reaches its peak at preceding layers. For ROME, the highest performance can be seen at training on representations from the layer 13-27 (GPT-J) or from the layers 23-47 (GPT2-XL). For MEMIT, the highest F1 is reached with representations from layer 17-27 (GPT-J), and layer 28-47 (GPT2-XL). This suggests that utilizing outputs from the last layer for detecting knowledge edits might not always be the optimal choice. Further, we observe that using the penultimate layer's representations is more effective than using the last layer's to detect knowledge edits. For example, for edited GPT-J on

| | | Hidden States + Prob. Dist. (HS + PD) | | | | | | Hidden States (HS) | | | | | | Probability Distribution (PD) | | | | | |
| | | zsRE | | | COUNTERFACT | | | zsRE | | | COUNTERFACT | | | zsRE | | | COUNTERFACT | | |
	KE	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1
GPT-JT	NONE	49.2	27.8	35.6	50.4	49.5	50.0	48.4	32.8	39.1	49.6	43.5	46.4	0.0	0.0	0.0	50.2	39.7	44.3
	ROME	87.1	95.5	91.1	90.9	94.2	92.5	99.8	74.2	85.1	91.6	87.8	89.6	84.8	84.5	84.7	89.3	94.0	91.6
	MEMIT	84.3	75.0	79.4	76.6	81.8	79.1	80.5	56.2	66.2	69.9	76.5	73.0	91.6	68.7	78.5	79.3	84.7	81.9
	MALMEN	80.7	71.6	75.9	65.6	74.1	69.6	50.4	34.6	41.1	57.8	69.1	63.0	72.8	88.4	79.9	82.4	54.0	65.2
GPT2 lima	NONE	49.0	56.1	52.3	51.5	45.9	48.5	50.1	41.8	45.6	49.4	42.0	45.4	50.6	93.0	65.6	51.2	39.2	44.4
	ROME	91.4	99.0	95.0	89.6	93.8	91.6	97.8	93.2	95.4	94.5	83.0	88.4	63.6	86.9	73.4	88.7	91.3	90.0
	MEMIT	79.2	65.0	71.4	77.3	77.1	77.2	76.9	61.9	68.6	73.7	60.9	66.7	74.0	73.2	73.6	76.9	76.4	76.7
	MALMEN	73.4	77.7	75.5	68.5	75.5	71.8	48.3	53.9	51.0	65.0	53.1	58.4	73.6	92.0	81.8	68.1	65.3	66.7

Table 6: CD (Cross-Domain) classification performance. Training data is from GPT2-XL and GPT-J, whereas test data is from GPT2-XL lima (GPT2 lima) and GPT-JT respectively.

COUNTERFACT (first subplot in Figure 7), representations from the penultimate layer achieve an F1 of 90% and 72% for detecting ROME-edited and MEMIT-edited knowledge, respectively, higher than 85% (ROME) and 67% (MEMIT) when using the representation from the last layer. *LE-KEs might be causing the attribute (object) extraction phase (Geva et al., 2023) that is responsible for extracting the object to happen early in the model. This early extraction is often associated with high probabilities (cf. Section 5.1), causing edits from LE-KEs to be easier to detect. In general, this experiment shows that the effect of edits from LE-KEs starts becoming visible from the changed layers, increases strongly thereafter, and decreases slightly at the last layer. This demonstrates that using representations from the last layer is not the optimal choice; utilizing representations from other layers can further improve performance.*

A.2 Robustness Evaluation

To investigate the robustness of the classifier, which uses HS and PD, we further evaluate it in a challenging setting, where both edited and unedited facts end with the same object, i.e., the classifier needs to distinguish between edited $= (s_1, r_1, o', p'_1)$ and unedited $= (s_2, r_2, o', p'_2)$ facts that share the same object (e.g., "The Eiffel Tower is in Berlin" vs. "Marlene Dietrich was born in Berlin"). Having the same object (as a prediction) in both edited and unedited facts might lead to more similar representations and probability distributions, and hence make the task more challenging. For example, the classifier might erroneously predict a fact as edited due to the presence of an edited fact with the same object. We refer to this setting as same object. Concretely, the classification model and the training are the same as described in Section 4.2. The test set is different. For each edited fact in the test

set, there is a corresponding unedited fact with the same object. For this experiment, we consider only GPT-J with MEMIT and MALMEN. The results for this experiment are shown in Table 8 alongside the corresponding results from Table 2 (referred to as default) for comparison. We notice that the classification performance drops. The decrease in performance is especially visible on zsRE. For example, the F1 score with MEMIT and GPT-J drops from 88.7% to 71.3%, whereas on COUNTERFACT the F1 score drops from 77.9% to 72.8%. This experiment shows that *it is more challenging to separate edited facts from unedited but related facts*. We believe this is due to the representations of the prompts becoming more similar under this setting.

A.3 Detection by Top 10 Output Probabilities

We experiment with using only the top 10 output probabilities instead of the top 1,000 output probabilities. The differences in performance (top 10 performance - top 1000 performance)) in Table 9 show that the top-10 output probabilities works well with minor degradation in performance in most cases. This means that our baselines can also be used on proprietary LLMs that output the top 10 output probabilities.

B Computational Resources

We ran our experiments on an NVIDIA A100 80GB GPU. Our experiments took roughly 15 GPU days.

model	method	#training	zsRE				COUNTERFACT			
			Acc.	Pr.	Rec.	F1	Acc.	Pr.	Rec.	F1
GPT-J	MALMEN	10	74.8	80.0	66.3	72.5	51.3	50.7	94.6	66.0
	MALMEN	50	73.7	73.8	73.4	73.6	61.1	60.4	64.4	62.3
	MALMEN	100	73.5	71.0	79.3	74.9	64.4	63.9	66.4	65.1
	MALMEN	150	75.3	74.9	76.1	75.5	60.1	59.7	61.7	60.7
	MALMEN	200	75.7	74.8	77.7	76.2	65.8	65.6	66.4	66.0
	MALMEN	250	76.1	76.3	75.6	76.0	65.8	67.4	61.1	64.1
	MEMIT	10	84.6	78.9	94.5	86.0	73.6	68.5	87.5	76.8
	MEMIT	50	83.8	77.6	95.0	85.4	78.8	78.1	80.1	79.0
	MEMIT	100	86.5	84.0	90.2	87.0	77.3	79.3	74.0	76.5
	MEMIT	150	87.0	86.1	88.2	87.2	74.0	76.5	69.1	72.6
	MEMIT	200	86.4	86.6	86.0	86.3	76.2	76.4	75.9	76.1
	MEMIT	250	86.8	85.2	89.0	87.0	76.4	78.1	73.3	75.6
	MEMIT	300	86.8	84.5	90.0	87.2	76.7	75.6	78.8	77.2
	MEND	10	72.1	79.2	60.1	68.3	73.2	66.3	94.5	77.9
	MEND	50	83.2	79.6	89.3	84.2	69.8	65.8	82.6	73.2
	MEND	100	79.2	80.3	77.4	78.8	69.1	63.6	89.2	74.3
	MEND	150	81.0	77.4	87.6	82.2	67.9	63.3	85.5	72.7
	MEND	200	81.6	78.5	87.1	82.6	71.5	66.0	88.6	75.7
	MEND	250	82.1	79.2	87.1	83.0	71.1	65.6	88.9	75.5
	MEND	300	81.3	78.0	87.1	82.3	70.6	65.2	88.1	75.0
	NONE	10	47.9	47.0	33.2	38.9	49.3	47.8	15.0	22.8
	NONE	50	51.1	51.2	48.0	49.5	49.0	48.7	37.6	42.4
	NONE	100	50.5	50.6	43.4	46.7	48.3	48.2	46.2	47.2
	NONE	150	47.1	46.7	41.6	44.0	49.0	49.1	53.6	51.2
	NONE	200	48.2	48.1	46.6	47.4	49.8	49.8	45.6	47.6
	NONE	250	47.8	47.5	42.2	44.7	50.9	51.0	46.6	48.7
	NONE	300	49.3	49.3	52.2	50.7	50.8	50.8	50.8	50.8
	ROME	10	94.8	99.5	90.0	94.5	80.0	72.8	95.7	82.7
	ROME	50	95.6	97.4	93.6	95.5	85.8	81.5	92.6	86.7
	ROME	100	96.6	97.4	95.7	96.5	87.8	88.4	87.1	87.7
	ROME	150	96.7	98.1	95.2	96.6	88.2	90.4	85.4	87.8
	ROME	200	96.5	97.7	95.2	96.4	88.3	88.7	87.8	88.2
	ROME	250	96.0	97.6	94.3	95.9	88.3	89.2	87.1	88.2
	ROME	300	96.2	97.6	94.8	96.2	89.5	91.7	86.9	89.2
GPT2-XL	MALMEN	10	74.0	77.9	67.1	72.1	46.1	46.5	51.6	48.9
	MALMEN	50	71.5	65.6	90.6	76.1	62.5	62.5	62.5	62.5
	MALMEN	100	71.8	68.4	81.0	74.2	65.6	65.6	65.6	65.6
	MEMIT	10	51.2	50.8	77.8	61.5	58.8	63.1	42.6	50.9
	MEMIT	50	71.9	77.5	61.7	68.7	60.8	62.0	55.5	58.6
	MEMIT	100	67.9	67.7	68.5	68.1	69.6	72.3	63.6	67.7
	MEMIT	150	72.2	72.8	71.0	71.9	73.2	73.9	71.8	72.8
	MEMIT	200	72.2	72.8	71.0	71.9	73.7	75.4	70.3	72.8
	MEMIT	250	72.2	72.0	72.8	72.4	72.0	72.1	71.8	71.9
	MEMIT	300	71.3	72.0	69.8	70.8	72.2	74.4	67.9	71.0
	MEND	10	60.6	59.6	65.8	62.5	50.1	50.4	16.9	25.4
	MEND	50	70.0	66.1	82.0	73.2	60.8	58.6	73.8	65.3
	MEND	100	69.7	66.1	80.9	72.8	59.9	56.8	82.5	67.3
	MEND	150	70.2	66.0	83.6	73.7	64.7	59.0	95.8	73.1
	MEND	200	71.8	67.7	83.3	74.7	65.0	59.3	95.2	73.1
	MEND	250	72.8	68.0	86.2	76.0	65.1	59.4	95.5	73.2
	MEND	300	73.0	68.1	86.4	76.2	65.1	59.4	95.5	73.2
	NONE	10	50.2	50.6	17.0	25.4	50.5	50.8	31.2	38.7
	NONE	50	52.0	52.3	45.6	48.7	50.8	51.0	41.6	45.8
	NONE	100	50.1	50.1	46.8	48.4	48.7	48.7	49.8	49.3
	NONE	150	52.0	52.4	44.6	48.2	46.9	46.6	43.2	44.9
	NONE	200	49.8	49.8	46.4	48.0	48.6	48.6	50.6	49.6
	NONE	250	49.4	49.3	44.2	46.6	48.8	48.8	49.4	49.1
	NONE	300	52.8	52.5	58.4	55.3	49.3	49.4	54.8	51.9
	ROME	10	88.2	88.5	87.8	88.1	63.6	58.4	93.9	72.0
	ROME	50	91.2	95.8	86.1	90.7	87.1	89.4	84.2	86.7
	ROME	100	91.2	92.4	89.7	91.0	89.2	91.7	86.2	88.9
	ROME	150	92.5	95.6	89.1	92.3	86.7	85.5	88.4	86.9
	ROME	200	92.0	94.5	89.1	91.8	88.0	89.4	86.2	87.7
	ROME	250	90.9	92.6	88.9	90.7	88.8	90.0	87.3	88.6
	ROME	300	90.8	93.4	87.8	90.5	88.6	90.0	87.0	88.4

Table 7: DEED (**de**tecting knowledge **ed**its) performance with increasing amount of training data.
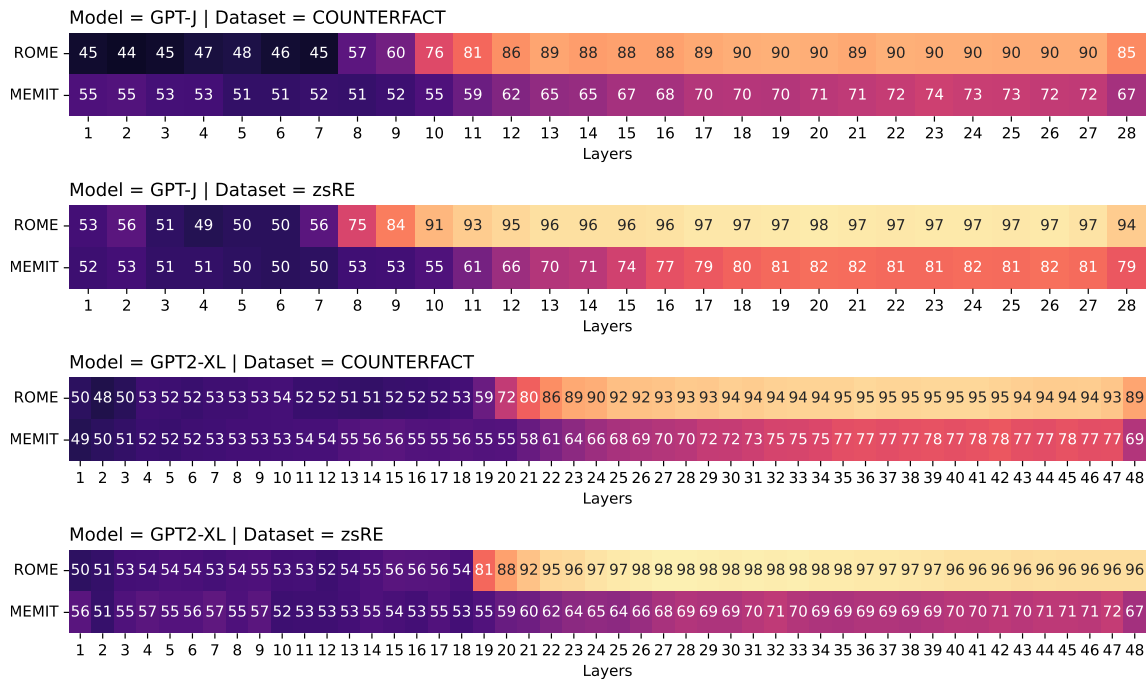
Model = GPT-J | Dataset = COUNTERFACT

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
ROME	45	44	45	47	48	46	45	57	60	76	81	86	89	88	88	88	89	90	90	90	89	90	90	90	90	90	90	85
MEMIT	55	55	53	53	51	51	52	51	52	55	59	62	65	65	67	68	70	70	70	71	71	72	74	73	73	72	72	67

Layers

Model = GPT-J | Dataset = zsRE

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
ROME	53	56	51	49	50	50	56	75	84	91	93	95	96	96	96	96	97	97	97	98	97	97	97	97	97	97	97	94
MEMIT	52	53	51	51	50	50	50	53	53	55	61	66	70	71	74	77	79	80	81	82	82	81	81	82	81	82	81	79

Layers

Model = GPT2-XL | Dataset = COUNTERFACT

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
ROME	50	48	50	53	52	52	53	53	53	54	52	52	51	51	52	52	53	53	59	72	80	86	89	90	92	92	93	93	93	94	94	94	94	94	95	95	95	95	95	95	95	94	94	94	94	93	89	
MEMIT	49	50	51	52	52	52	53	53	53	54	54	55	56	56	55	55	56	55	55	58	61	64	66	68	69	70	70	72	72	73	75	75	75	77	77	77	77	78	77	78	78	77	77	77	78	77	77	69

Layers

Model = GPT2-XL | Dataset = zsRE

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
ROME	50	51	53	54	54	54	53	54	55	53	53	52	54	55	56	56	56	54	81	88	92	95	96	97	97	98	98	98	98	98	98	98	98	98	97	97	97	97	96	96	96	96	96	96	96	96	96	96
MEMIT	56	51	51	57	55	56	57	55	57	52	53	53	53	55	54	53	55	53	55	59	60	62	64	65	64	66	68	69	69	69	70	71	70	69	69	69	69	69	70	70	71	70	70	71	71	71	72	67

Layers

Figure 7: Rounded F1 scores when training on representations (HS) from different LLM layers

Generator		zsRE			COUNTERFACT		
Model	Editor	Pr.	Rec.	F1	Pr.	Rec.	F1
default	MEMIT	87.7	89.8	88.7	79.0	76.7	77.9
	MALMEN	76.7	75.3	76.0	62.0	62.0	62.0
same object	MEMIT	58.5	91.1	71.3	72.7	72.9	72.8
	MALMEN	57.5	57.9	57.7	56.4	63.6	59.8

Table 8: Classification performance on detecting knowledge edits for GPT-J. Facts in the training and test data end with the same object. Classification is done based on HS and PD.

Generator		zsRE			COUNTERFACT		
Model	Editor	Pr.	Rec.	F1	Pr.	Rec.	F1
GPT-J	NONE	-2.6	6.7	2.4	-2.4	-4.5	-3.5
	ROME	-0.4	-0.4	-0.4	1.5	-1.5	0.0
	MEMIT	-0.6	-0.9	-0.7	-1.7	2.6	0.6
	MEND	-0.2	3.2	1.3	0.6	-0.4	0.3
	MALMEN	2.6	-1.9	0.5	4.0	-0.5	1.6
GPT2-XL	NONE	-0.1	0.3	0.1	-0.8	0.0	-0.4
	ROME	0.1	-2.7	-1.3	2.0	-5.2	-1.5
	MEMIT	-1.4	-5.7	-3.8	-7.0	-1.7	-4.4
	MEND	-1.0	-5.8	-2.9	0.1	0.1	0.1
	MALMEN	0.0	9.2	4.7	-4.6	-14.1	-9.4

Table 9: Differences in classification performance for detecting knowledge edits: top 10 probabilities minus top 1,000 probabilities
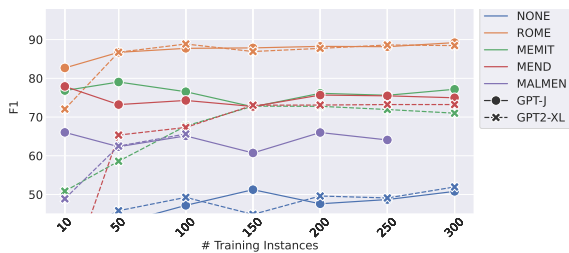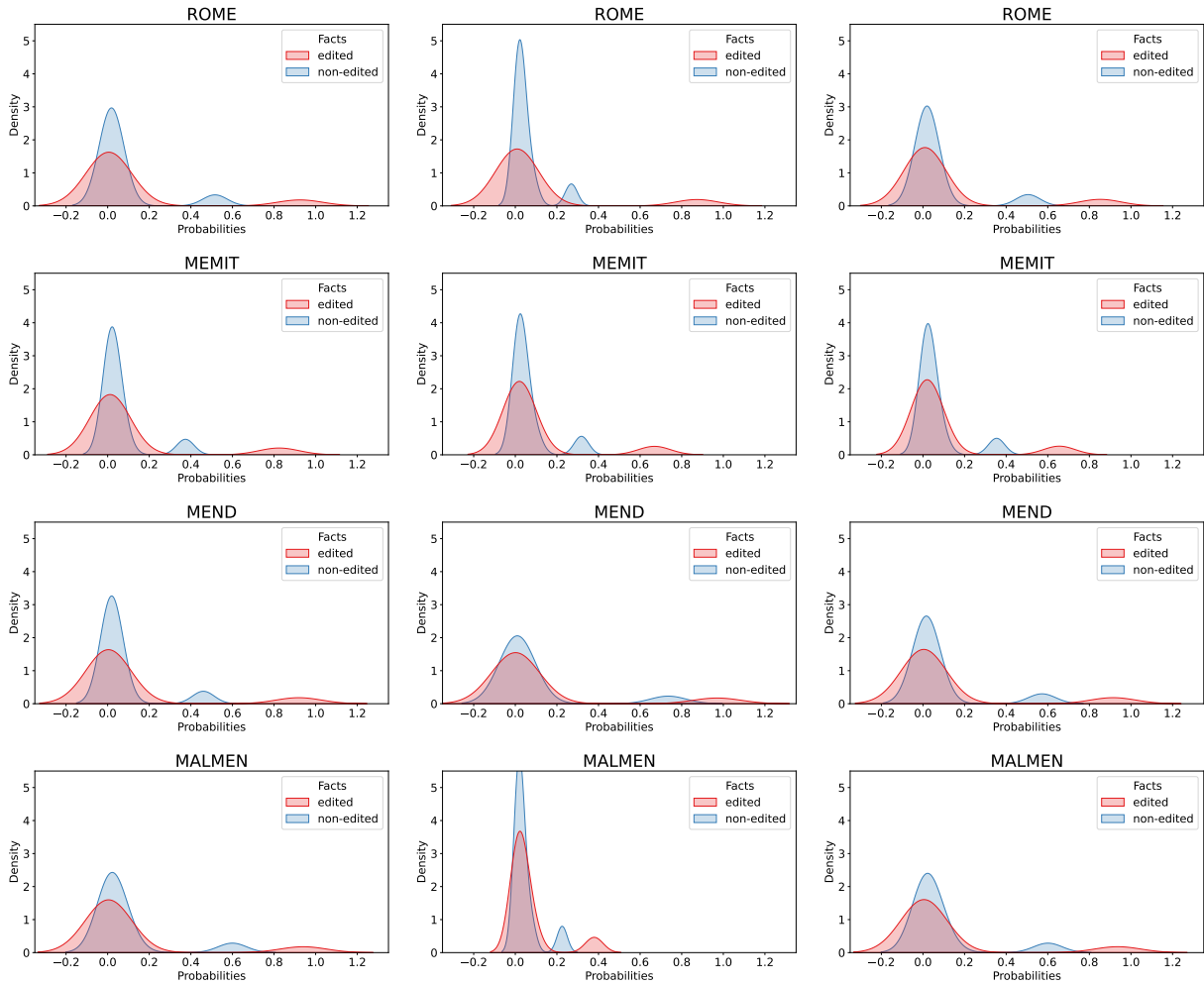
Figure 8: Detection performance with PD on COUNTERFACT when varying training set size.

(a) ZSRE and GPT-J.   (b) COUNTERFACT and GPT2-XL.   (c) ZSRE and GPT2-XL.

Figure 9: KDE of edited and non-edited facts.

Prompt example	ZSRE	COUNTERFACT
Edit prompt	In what capacity did Andrea Guatelli play football? goakeeper	Ramon Magsaysay holds a citizenship from Sweden
Evaluate ES prompt	In what capacity did Andrea Guatelli play football? goakeeper	Ramon Magsaysay holds a citizenship from Sweden
Evaluate GS prompt	In what capacity has Andrea Guatelli played soccer? goalkeeper	Marquez Rubio, Juan Carlos (2002). Ramon Magsaysay is a citizen of Sweden
Evaluate LS prompt	What is the university Leslie Lazarus went to? University of Sydney	Some fish, like sharks and lampreys, possess multiple gill openings. Abune Paulos, who is a citizen of Ethiopia

Table 10: Example prompt used to edit knowledge, and to evaluate editing performance. Objects underlined are the original knowledge.