Detecting Edited Knowledge in Language Models

Anonymous ACL submission

Abstract

Knowledge editing techniques (KEs) can update language models' obsolete or inaccurate knowledge learned from pre-training. However, KE also faces potential malicious applications, e.g. inserting misinformation and toxic content. Moreover, in the context of responsible AI, it is instructive for end-users to know whether a generated output is driven by edited knowledge or first-hand knowledge from pre-training. To this end, we study detecting edited knowledge in language models by introducing a novel task: given an edited model and a specific piece of knowledge the model generates, our objective is to classify the knowledge as either "non-edited" (based on the pre-training), or "edited" (based on subsequent editing). We initiate the task with two state-of-the-art KEs, two language models, and two datasets. We further propose a simple classifier, `RepReg`, a logistic regression model that takes hidden state representations as input features. Our results reveal that `RepReg` establishes a strong baseline, achieving a peak accuracy of 99.81%, and 97.79% in out-of-domain settings. Second, `RepReg` achieves near-optimal performance with a limited training set (200 training samples), and it maintains its performance even in out-of-domain settings. Last, we find it more challenging to separate edited and non-edited knowledge when they contain the same object.

1 Introduction

Large Language Models (LLMs) encode knowledge about the world via their pre-training data (Roberts et al., 2020). However, the encoded knowledge can be inherently flawed or become outdated with time (De Cao et al., 2021). In other contexts, practitioners might want to tailor these models by incorporating domain-specific knowledge pertaining to their products or to facilitate new applications. Inspired by these needs and given the prohibitive cost of pre-training a new model, works are motivated to efficiently update knowledge in LLMs (Meng et al., 2022; Zheng et al., 2023; Li et al., 2023).

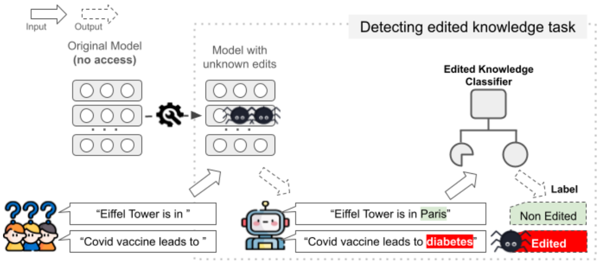

Figure 1: Illustration of a malicious knowledge editing, and our task of detecting edited knowledge (within the dashed box).

Knowledge editing (KE) methods, 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 insert adapted 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 (as shown in Figure 1) (Halevy et al., 2024) or embed harmful or undesired biases in LLMs (Mazzia et al., 2023). This danger becomes more pronounced as the number of capable open-source LLMs increases.

While fact-checking and hate speech detection represent plausible avenues to combat these malicious uses (Zhao et al., 2021; Pan et al., 2023), we take a different approach: directly detecting edited knowledge. Rather than utilizing specialized methods to assess factuality or toxicity separately when the objective is clearly defined, we argue that detecting edited knowledge provides transparency to users in general application scenarios. Moreover, detecting edited knowledge holds potential value for enhancing the controlled generation of LLMs (Pan et al., 2023; Sun et al., 2023).

This work, as the first one, proposes the task of detecting edited knowledge in LLMs. Considering practical scenarios where users do not have the original base model, we assume access to an edited model and no access to the original/unedited model. We further propose a detecting classifier using the hidden state representations, and evaluate the classifier across two state-of-the-art KEs, two models, and two datasets, resulting in eight detection scenarios in total. We further test on out-of-domain settings. Our analysis shows that a simple logistic regression model trained on the hidden state representation of the generated token is a strong baseline.

2 Related Work

This paper is the first to establish the task of detecting edited knowledge in pre-trained language models. Still, a wide array of works study 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 above. This study focuses on the post-editing phase, particularly on editing methods that directly modify model parameters.

2.1 Knowledge Editing

Mainstream KEs can be divided into two groups: preserving parameters and directly modifying parameters.

Preserving parameters-based methods integrate extra modules, e.g., additional memory space or parameters, while leaving the original parameters unchanged (Yao et al., 2023; Li et al., 2024). SERAC stores new knowledge in an explicit cache and uses an auxiliary scope classifier and counterfactual model to modulate the base model's predictions as needed (Mitchell et al., 2022b). GRACE 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 (Hartvigsen et al., 2023). MELO (Yu et al., 2023b) learns and stores new knowledge in inserted LoRA blocks (Hu et al., 2021).

Modifying parameters-based methods modify the model parameters directly via meta-learning or optimization-based methods. Meta-learning KEs employ hyper-networks to learn parameter shifts towards new knowledge. De Cao et al. (2021) and Mitchell et al. (2022a) build light-weight hyper-networks, a naive multi-layer perceptron (MLP), that intakes the gradients and outputs the parameter shifts for selected individual layers. To handle the cancellation effect in batch editing, where the parameter shifts of differing edits may counteract each other, MALMEN uses the normal equation instead of summing the parameter shifts (Tan et al., 2024).

On the other hand, optimization-based methods operate under the assumption 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.

Fine-tuning the model on specific data roughly falls into the category of modifying parameters-based KEs, but it requires great computational resources and data curation costs, and may encounter catastrophic forgetting and overfitting (Mitchell et al., 2022a,b; Zheng et al., 2023).

2.2 Malicious Modifying LLMs

Recent advancements in LLMs have led to their widespread adoption that permeates individuals' daily lives. However, these models are vulnerable to malicious attacks and poisoning (Li et al., 2021; Cao et al., 2023; Liu et al., 2023; Li et al., 2024; Wei et al., 2024). 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 KE techniques, knowledge of LLMs can be modified easily with low cost, and at scale. This advancement presents opportunities for misuse, such as inserting misinformation and toxic content into LLMs. For example, BadEdit by (Li et al., 2024) formulates backdoor injection as a lightweight knowledge editing problem, i.e. modifying a subset of parameters to inject backdoors into LLMs. Further, due to the high cost of training LLMs, end-users are likely to seek substitute models available 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., 2024).

3 The Edits Detecting Problem

Facts[1] in LLMs are often represented as triplets of $(subject, relation, object)$, or (s, r, o) for short. Querying an LLM with a prompt $p(s, r)$, where p is a prompt that 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 KE operation $E(s, r, o, o')$ is successful if it changes the behavior of LLM such that the retrieved object is o' as desired, instead of o (**E**fficacy **S**uccess). Ideally, such edits should affect other semantically similar prompts $p'(s, r)$, e.g. "The city where Eiffel Tower locates is" (**G**eneralizability **S**uccess), and should be implemented locally and precisely, not affecting other unrelated knowledge (**L**ocality **S**uccess, also known as Specificity) (Mitchell et al., 2022b; Meng et al., 2022; Li et al., 2023).

We model the task of detecting edited knowledge as a binary classification problem. The target classes are "edited", and "non-edited". Given a model $\mathcal{M}$ with unknown edits, and a pool of knowledge $\mathcal{T}_{test} = \{(s_1, r_1, o_1, p_1), ..., (s_n, r_n, o_n, p_n)\}$, where p is a prompt that expresses (s, r, o), the task is to classify the retrieved o into "edited" or "non-edited". We also assume access to a training set $\mathcal{T}_{train} = \{(s_1, r_1, o_1, p_1), ..., (s_m, r_m, o_m, p_m)\}$, i.e., we know some edited and non-edited facts.

4 Methodology

As a prior work of the detection task, we first edit the base language model with the editing dataset (Section 4.1) and filter out failed edits, keeping successful edits to collect edited knowledge in the edited model. Next, we create the detecting dataset with edited knowledge from successful edits and unedited knowledge, i.e., the knowledge not involved in the editing, regardless of whether it was successfully edited or not (Section 4.2). Last, we train a classifier on the detecting dataset to detect edited knowledge (Section 4.3).

4.1 Edited Models

Editing models. Following Meng et al. (2022, 2023), we incorporate GPT-2 XL (1.5B) and GPT-J (6B) as our base language models. We edit these models via ROME and MEMIT (Meng et al., 2023; Yao et al., 2023; Tan et al., 2024). For ROME, we edit the 19th layer in GPT-2 XL, and the 7th layer in GPT-J, whereas with MEMIT, we edit the layers 15-19 in GPT2 XL, and the layers 5-10 in GPT-J following (Meng et al., 2022, 2023).[2]

Editing datasets. We edit models with two popular datasets, ZSRE and COUNTERFACT (Meng et al., 2023; Yao et al., 2023). **ZSRE** (Levy et al., 2017) is a Question Answering dataset using question rephrases derived from back-translation to serve as the equivalence neighborhood, originally contains 244,173 training and 27,644 validation instances. COUNTERFACT (Meng et al., 2022) incorporates counterfactual scenarios where incorrect assertions are initially favored over correct ones.

This yields a total of eight edited models for one fact. Two edited models for each base model, each dataset, and each editing method.

4.2 Edits Detection Datasets

Our goal is to detect edited knowledge. Thus, failed edits are excluded from the detection tasks. Concretely, edits of generalizability success (GS), successfully retrieving new knowledge o' with a paraphrase of p' are the edits of interest. That is,

$$o' = argmax_o \, \mathbb{P}_{\mathcal{M}, p'(s,r)}[o] \qquad (1)$$

The reason of using paraphrases is that end-users do not know the exact prompts used for editing; therefore, they are likely to prompt the model with a paraphrase of the editing prompt to retrieve the object in the knowledge. For "non-edited" knowledge, we extract a subset of data from the original dataset of the same number as the "edited" set, which is not used to edit knowledge, i.e., a disjoint set from $\mathcal{T}_{train}$. There is no overlap between $\mathcal{S}_{edited}$ and $\mathcal{S}_{nonedited}$.

[1] In this paper, the terms "facts" and "knowledge" are used interchangeably (Meng et al., 2023).

[2] The numbers we mention here refer to the indices corresponding to the hidden states representations.

4.3 Detecting Classifier

Inspired by the probing tasks introduced in Conneau et al. (2018) and the representation inspection by Hernandez et al. (2023), we leverage the hidden state representation as input features to train a logistic regression model, i.e., a binary classification model where the prediction is "edited" or "not edited".

Given a fact (s, r, o, p'), we prompt the given $\mathcal{M}$ of L transformer blocks, with $p'(s, r)$ that consists of N tokens $\{u_1, ..., u_N\}$. We use the hidden state output of the last transformer block $l \in \{1, L\}$ as the input feature. We denote this proposed method, using representation to train a logistic regression model, as `RepReg` classifier.

5 Experiment

5.1 Editing Performance

We experiment with two editing methods, ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), two models, GPT2-XL and GPT-J, and two datasets, COUNTERFACT and ZSRE. For MEMIT, we edit each model with 1,000 facts at once. ROME is not designed for mass editing or sequential editing (Huang et al., 2022; Yao et al., 2023) and in preliminary experiments, perplexity degraded badly with ROME on a large number of edits at once or sequentially. Therefore, we conduct one edit at a time on the base language model and cache the representations of interest for the current edit. We include the non-edited base model, `NONE`, as a yardstick.

Generator		ZSRE			COUNTERFACT		
Model	Editor	ES↑	GS↑	LS↑	ES↑	GS↑	LS↑
	NONE	28.1	27.4	26.1	13.3	18.6	81.0
GPT-J	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
	NONE	23.2	22.1	22.3	18.9	24.1	74.6
GPT2-XL	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

Table 1: Summary of editing performance over 1000 edits.

Following Meng et al. (2022, 2023); Li et al. (2023), we evaluate editing performance with Efficacy Success (ES), Generalizability Success (GS), and Locality Success (LS) as introduced in Section 3. See Appendix B for further details on metrics.

Our results align with those reported in previous studies (Meng et al., 2022, 2023; Li et al., 2023). Note that the performance gap on LS between the two datasets is mainly due to the different calculations of ES between the two datasets. For ZSRE, LS (a.k.a. Specificity in Mitchell et al. (2022a); Meng et al. (2022)) is the proportion of neighborhood prompts that the model gets correct. In COUNTERFACT, LS (a.k.a. Neighborhood Success in Meng et al. (2023)) is the proportion of neighborhood prompts where the model assigns higher probability to the ground truth object than the counterfactual object. Since this paper does not focus on the faithful or effective evaluation of knowledge editing, and the creation of our detection dataset does not depend on the LS, we leave the efficient evaluation of knowledge editing to future research.

Recall Section 4.2, we create the detecting datasets from GS edits, where $o' = argmax_o\, \mathbb{P}_{\mathcal{M},p'(s,r)}[o]$. We report the number of GS edits we include in our detection experiments in Table 5 in the Appendix.[3]

5.2 Classification Performance

We train a logistic regression classifier with L1 regularization on the edits detection dataset. We show the classification performance of training on 324 instances in Table 2. The classifier NONE, i.e., trained on representations of new target knowledge and original knowledge from the base non-edited model, shows a fair coin performance on the four metrics, ranging from 45.35 (GPT-J, ZSRE, Recall) to 53.70 (GPT-J, COUNTERFACT, Recall). This verifies the correctness of our setup.

Generator		ZSRE				COUNTERFACT			
Model	Editor	Acc.	Pr.	Rec.	F1	Acc.	Pr.	Rec.	F1
	NONE	49.40	49.35	45.35	47.26	48.93	49.02	53.70	51.25
GPT-J	ROME	96.78	98.83	94.68	**96.71**	91.19	94.27	87.70	**90.87**
	MEMIT	82.45	86.19	77.27	81.49	75.87	77.55	72.83	75.11
	NONE	49.40	49.33	44.03	46.53	48.45	48.52	50.72	49.59
GPT2-XL	ROME	98.08	99.46	96.69	**98.06**	94.11	96.08	91.96	**93.98**
	MEMIT	75.93	79.58	69.75	74.34	74.90	77.73	69.80	73.55

Table 2: Classification performance on detecting edited knowledge. The training set includes 324 instances in all settings. The number of test instances for each setting is shown in Table 4.

First, we find that the linear classifier is able to detect ROME-edited facts relatively well, achieving a minimum F1 score of 90.87% (GPT-J,

[3] On COUNTERFACT, GS is not measured with Eqn. 4.2, therefore, the numbers here do not equal to its GS results on COUNTERFACT.

COUNTERFACT). The performance on MEMIT-edited knowledge varies across datasets and models. Specifically, we observe the highest F1 score (81.49%) on ZSRE with GPT-J, whereas the lowest F1 score (73.55%) is seen on COUNTERFACT with GPT2-XL. One explanation could be that MEMIT conducts minimal changes that spread across several layers to keep the model robust (Zhu et al., 2020; Meng et al., 2023). Conversely, in ROME, the changes are condensed in one layer, leaving evident traces of modifications in the final representations.

Second, between GPT-J and GPT2-XL, we observe performance disparity in detecting MEMIT-edited knowledge on ZSRE. For example, the recall scores on ROME-edited knowledge on ZSRE are 94.68 and 96.69 for GPT-J and GPT2-XL, respectively; while on MEMIT-edited knowledge, they are 77.27 and 69.75 for GPT-J and GPT2-XL, respectively. We delve deeper into the effects of different edited layers in Section 6.1.

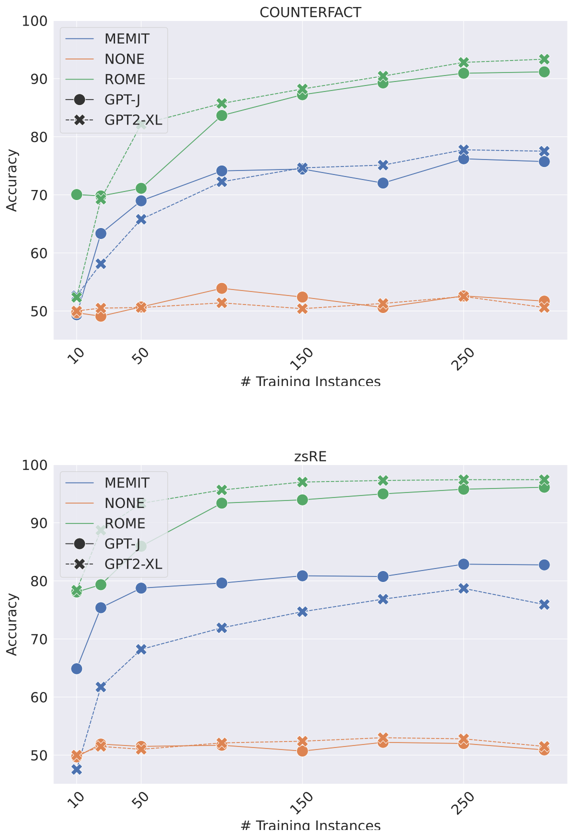

Figure 2: Edits classification performance of RepReg when training on different numbers of training instances.

Figure 2 shows the classification performance of training on different training set sizes. Exhaustive results are presented in Table 7 in the Appendix. In general, the classification performance starts reaching a plateau after 100 training instances. For example, on COUNTERFACT and MEMIT-edited knowledge on GPT-J (the dotted blue line in the top figure), RepReg reaches 74.12% on 100 training data where the highest accuracy of 76.21% is achieved when using 250 training data points. Therefore, we highlight that **our proposed classifier, RepReg, remains effective even when the training data consists of only a few hundred examples.** Second, between MEMIT and ROME, we find MEMIT-edited knowledge is harder to detect than ROME-edited knowledge, and increasing the training data does not improve the classification performance to match the performance on ROME-edited knowledge.

5.3 Out-of-domain Classification Performance

Obtaining a training set for the detection classifier requires knowing instances of edited knowledge in the edited model. However, this assumption may not always hold. Consider a scenario where an open-source model is further tailored or fine-tuned for a specific task of interest to the user, yet no explicit information is provided regarding the edited knowledge. For example, 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 inserted into the fine-tuned model. To address such cases, we assess RepReg in an out-of-domain setting, where the classifier is trained on representations from one edited model and then applied to detecting edited knowledge from a separate edited model (which is a fine-tuned version of the original model). The OOD edited models have the same architecture, but different weights. To simulate this scenario, we use representations from edited GPT-J and GPT2-XL to train a RepReg and test it in detecting edited knowledge from fine-tuned versions of GPT-J[4] and GPT2-XL[5]. Note that the facts in the training and test sets are still

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

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

disjoint.

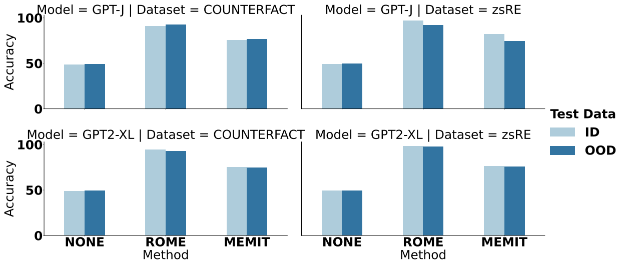

Figure 3: Comparison between in-domain (ID) and out-of-domain (OOD) classification performance.

As shown in Figure 3, **the performance on out-of-domain edits (OOD) is on par with performance on in-domain data (ID)**. For example, on ZSRE, OOD accuracy on ROME-edited knowledge on GPT2-XL lima is 97.79%, similar to the ID accuracy of 98.08%. Full results are shown in Table 10 in the Appendix. This makes our approach for detecting edited knowledge 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 model.

6 Discussion

6.1 Effect of Edited Layers

To investigate the effect of different edited layers, given an edited model, we extract the hidden state representations from each layer and train a `RepReg` on representations from different layers separately. We show their accuracy in Fig 4.

First, for both models, editing methods, and datasets, the representations are only separable, i.e., the classification performance is above random guess, after the edited layers. For example, the accuracy for detecting ROME edits starts improving from the layer 7 in GPT-J and the layer 19 in GPT2-XL (the only layer that is changed in both models), whereas the same effect with MEMIT is visible from layer 10 in GPT-J (MEMIT changes the layers 5-10), and from layer 18 (COUNTERFACT) or layer 15 (ZSRE) (MEMIT changes the layers 15-19) in GPT2-XL.

Even though the representations from the last layer perform well in detecting edits, we notice that the accuracy sometimes reaches its highest at preceding layers. For ROME, the highest performance can be seen at training on representations from layer 22 (GPT-J) or from layers 30-32 (GPT2-XL). For MEMIT, the highest accuracy is reached with representations from layer 24 (GPT-J), and representations from layer 46 (COUNTERFACT) or from last layer (ZSRE). This suggests that utilizing outputs from the last layer for detecting edited knowledge might not always be the optimal choice. Further, we observe that **using the penultimate layer's representation is more effective than using the last layer's to detect edited knowledge**, except on ZSRE when editing GPT2-XL via MEMIT where it is 74% by the penultimate layer's against 76% at the last layer's. For example, for edited GPT-J on COUNTERFACT (the first subplot in Figure 4), `RepReg` trained on representations from the penultimate layer achieves accuracy of 92% and 77% for detecting ROME-edited and MEMIT-edited knowledge, respectively, higher than 91% (ROME) and 76% (MEMIT) when using the representation from the last layer.

6.2 Detecting Unedited Neighbors

To investigate the robustness of our classifier, we further evaluate it in a more challenging setting, where both edited and non-edited facts end with the same object, i.e., the classifier needs to distinguish between an edited = (s_1, r_1, o', p_1) and nonedited = (s_2, r_2, o', p_2) that share the same object (e.g., "The Eiffel Tower is in Berlin" vs. "Marlene Dietrich was born in Berlin"). Having similar objects at the end, should make the representations more similar and hence the task more challenging. We refer to this setting as `same object`.

The results for this experiment are shown in Table 3. Comparing to Table 2, we notice that the performance drops. This is especially apparent on ZSRE. For example, the F1 score with MEMIT and GPT-J drops from 81.49% to 63.96%. On COUNTERFACT, the changes in performance are smaller. We suspect the difference in performance dropping between the two datasets is related to the LS performance (Table 1) where ZSRE has much lower LS (22.5 35.4) than COUNTERFACT (74.8-79.4) across models and editing methods. Concretely, a high LS score indicates edits are constrained to local changes that should not affect related facts ("Marlene Dietrich was born in Berlin"), so that the classifier is able to distinguish edited and non-edited but related facts.

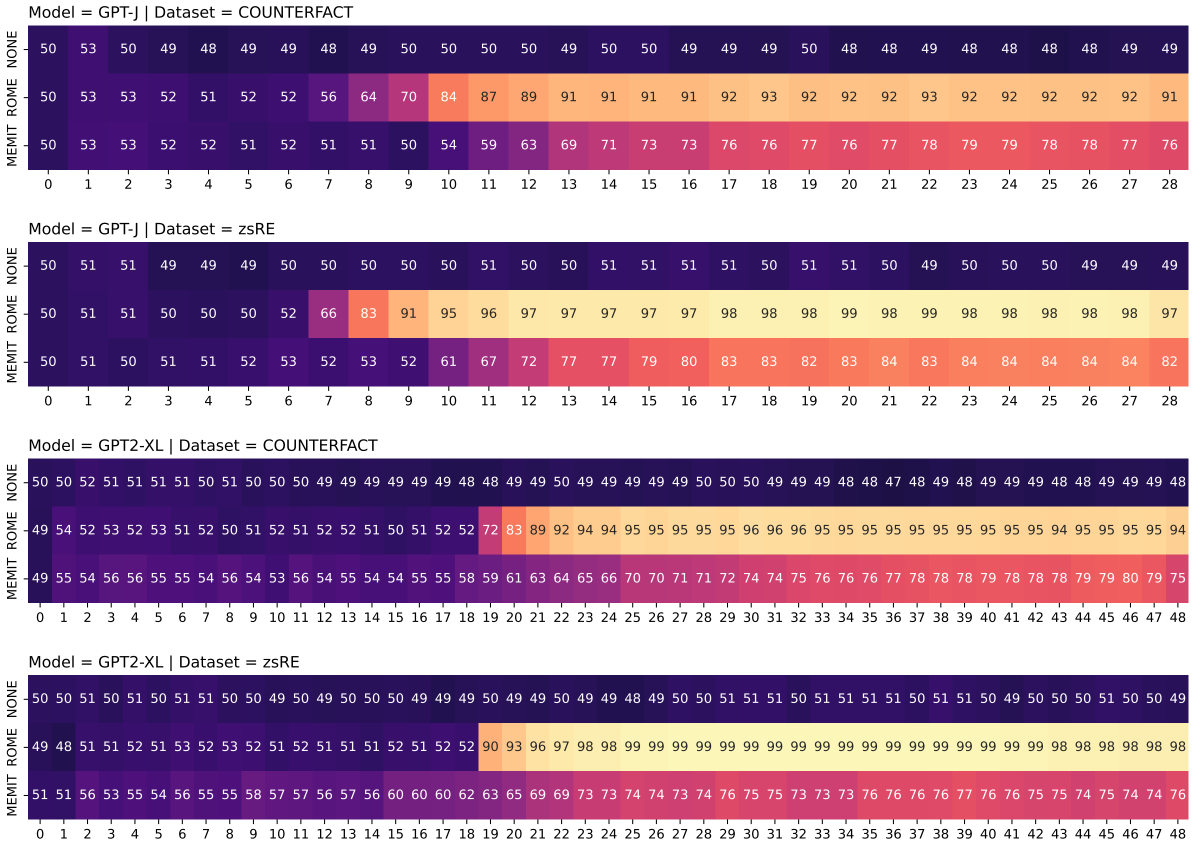

Figure 4: The classification performance (accuracy) of training RepReg on representations from different layers.

Generator		zsRE				CounterFact			
Model	Editor	Acc.	Pr.	Rec.	F1	Acc.	Pr.	Rec.	F1
GPT-J	NONE	50.44	50.44	50.88	50.66	52.29	52.17	54.98	53.54
	ROME	73.47	71.48	78.11	**74.65**	86.62	89.39	83.10	**86.13**
	MEMIT	64.88	65.69	62.33	63.96	76.21	80.23	69.56	74.51
GPT2-XL	NONE	51.50	51.68	46.13	48.74	50.97	50.91	54.41	52.60
	ROME	78.65	77.50	80.73	**79.08**	91.78	94.46	88.76	**91.52**
	MEMIT	63.49	65.45	57.14	61.02	71.48	72.73	68.73	70.67

Table 3: Classification performance on detecting edited knowledge. Facts in the training and test data end with the same object.

6.3 Why RepReg Works

Hidden State Representation Visualisation We visualize the hidden state representations, the input features of RepReg, into a lower dimension space via Linear Discriminant Analysis (LDA) (Duda et al., 2000). We project the representations from the test set into a 1-dimensional space. We assign the 1-dimensional representations random values on the y-axis to reduce cluttering. As shown in Figure 5, the edited and non-edited prompts of GPT-J and CounterFact are not distinguishable in the non-edited model (1st row). Editing with ROME (2nd row) and to less extent with MEMIT (3rd row) makes the representations more separable. Similar patterns are observed on all datasets and methods. Exhaustive results are shown in Figure 6 and 7 in the Appendix. In short, the LDA visualization explains why a simple logistic regression model is able to distinguish edited and non-edited facts by their hidden state representations.

7 Conclusion

In this paper, we introduced a novel task: detecting edited knowledge in language models. We see promising applications of this task, e.g., tackling potential malicious model modification. Further, we presented a baseline classifier, RepReg, which uses hidden state representations as input features, demonstrating a straightforward yet effective approach to the task. Our evaluation of RepReg across two decoder-only language models, two editing techniques, and two datasets, shows that the classifier can reliably detect edits, even in scenarios characterized by limited training data. The results also indicate that hidden state representations could be leveraged to derive meta-information about the source of the model's knowledge, i.e., whether the generated facts are

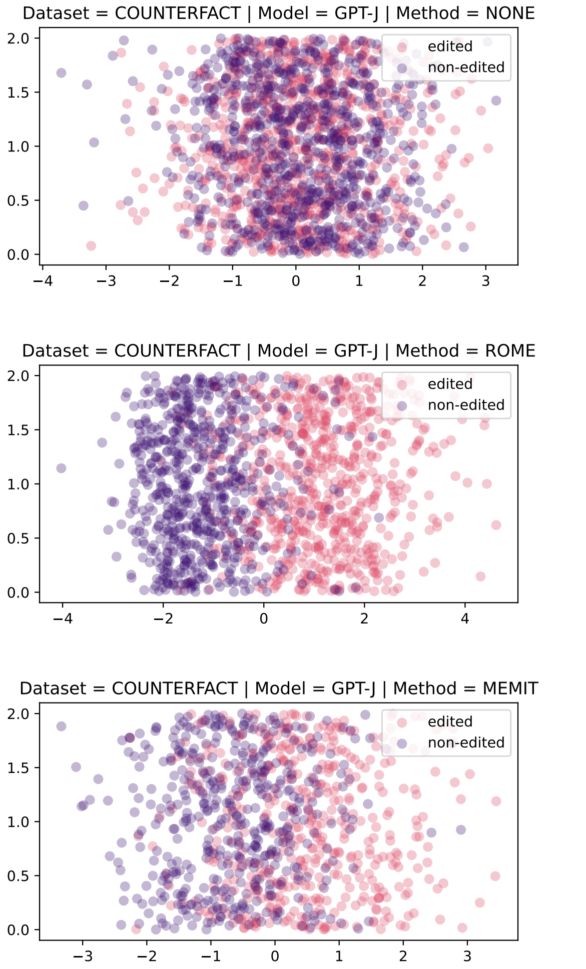

Figure 5: Representations of edited and non-edited facts from COUNTERFACT and GPT-J in a two-dimensional space.

learned from pre-training or are edited into the model. Moreover, we tested RepReg in challenging settings: detecting facts containing the same object, and in out-of-domain settings. Our results highlighted difficulties in distinguishing between edited and non-edited knowledge when they have the same subject. In our future work, we aim to explore RepReg on other editing methods and new methods for handling challenging cases, e.g. facts of the same subject.

References

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

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*.

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*.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. 2023. Detecting edit failures in large language models: An improved specificity benchmark. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11548–11559, Toronto, Canada. Association for Computational Linguistics.

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.

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.

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, and Shezheng Song. 2023. Pmet: Precise model editing in a transformer. `https://synthical.com/article/f00a0533-4457-404d-9fd5-36fbe662bfa1`.

Yanzhou Li, Kangjie Chen, Tianlin Li, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. 2024. 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. 2023. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*.

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.

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*.

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.

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*.

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.

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.

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

A Data

In Table 4, 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. The number of edits we consider successful and include in our experiments is shown in Table 5.

Model	Dataset	Method	#Training	#Test
GPT-J	COUNTERFACT	MEMIT	324	920
GPT-J	COUNTERFACT	NONE	324	1676
GPT-J	COUNTERFACT	ROME	324	1350
GPT-J	zsRE	MEMIT	324	1276
GPT-J	zsRE	NONE	324	1676
GPT-J	zsRE	ROME	324	1428
GPT2-XL	COUNTERFACT	MEMIT	324	510
GPT2-XL	COUNTERFACT	NONE	324	1676
GPT2-XL	COUNTERFACT	ROME	324	1120
GPT2-XL	zsRE	MEMIT	324	324
GPT2-XL	zsRE	NONE	324	1676
GPT2-XL	zsRE	ROME	324	1148

Table 4: Number of training and test instances used in our default setting. Both the training and test sets are balanced (50% positive/negative examples).

Model	Dataset	Method	#edits
GPT-J	COUNTERFACT	MEMIT	622
GPT-J	COUNTERFACT	NONE	6
GPT-J	COUNTERFACT	ROME	837
GPT-J	zsRE	MEMIT	800
GPT-J	zsRE	NONE	0
GPT-J	zsRE	ROME	876
GPT2-XL	COUNTERFACT	MEMIT	417
GPT2-XL	COUNTERFACT	NONE	9
GPT2-XL	COUNTERFACT	ROME	722
GPT2-XL	zsRE	MEMIT	324
GPT2-XL	zsRE	NONE	0
GPT2-XL	zsRE	ROME	736

Table 5: Number of successful edits for each setting. Note that half of the edits are used for training and the other half for evaluation.

B Editing Evaluation Metrics

Knowledge editing aims to alter a single association, a piece of knowledge, stored in the model. However, knowledge is interconnected; changing one fact may ripple outwards and affect other facts in complex ways. This interdependence makes assessing the effects of editing difficult. For example, Hoelscher-Obermaier et al. (2023) find that the edited model often fails on the target knowledge when the prompt contains distracting content. Therefore, edited model are often evaluated in terms of **E**fficacy **S**uccess, **G**eneralizability **S**uccess and **L**ocality **S**uccess.

For zsRE, ES is the proportion of edits when the model recalls with top-1 accuracy. Note that the prompt matches exactly what the edit method sees at runtime. GS is the accuracy on rephrasings of the original statement. LS is the proportion of neighborhood prompts that the model gets correct.

For COUNTERFACT, ES is the proportion of cases where new knowledge o' exceeds the original knowledge o in probability ($\mathbb{P}_{\mathcal{M},p(s,r)}[o'] > \mathbb{P}_{\mathcal{M},p(s,r)}[o]$). Note that the prompt matches exactly what the edit method sees at runtime. GS is the proportion of cases where new knowledge o' exceeds the original knowledge o in probability in probability on paraphrases of the original prompt. LS is the proportion of neighborhood prompts where the edited model assigns higher probability to the correct unedited fact.

C Detecting Edited Knowledge

We show the edits facts detection performance over different numbers of training instances in Table 7.

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

model	method	#training	acc_zs	precision_zs	recall_zs	f1_zs	acc_cf	precision_cf	recall_cf	f1_cf
GPT-J	MEMIT	10	64.88	89.40	33.75	49.00	49.36	49.68	98.39	66.02
GPT-J	MEMIT	25	75.38	87.73	59.00	70.55	63.34	65.20	57.23	60.96
GPT-J	MEMIT	50	78.75	86.86	67.75	76.12	68.97	74.38	57.88	65.10
GPT-J	MEMIT	100	79.62	85.59	71.25	77.76	74.12	80.00	64.31	71.30
GPT-J	MEMIT	150	80.88	84.99	75.00	79.68	74.44	80.89	63.99	71.45
GPT-J	MEMIT	200	80.75	84.94	74.75	79.52	72.03	77.08	62.70	69.15
GPT-J	MEMIT	250	82.88	85.25	79.50	82.28	76.21	80.99	68.49	74.22
GPT-J	MEMIT	300	82.75	85.79	78.50	81.98	75.72	80.77	67.52	73.56
GPT-J	NONE	10	49.70	33.33	0.60	1.18	49.70	44.83	2.60	4.91
GPT-J	NONE	25	51.90	51.71	57.40	54.41	49.10	49.16	52.80	50.92
GPT-J	NONE	50	51.50	51.64	47.20	49.32	50.70	51.00	35.80	42.07
GPT-J	NONE	100	51.70	51.72	51.00	51.36	53.90	54.86	44.00	48.83
GPT-J	NONE	150	50.70	50.71	49.80	50.25	52.40	52.90	43.80	47.92
GPT-J	NONE	200	52.20	52.24	51.40	51.81	50.60	50.74	41.40	45.59
GPT-J	NONE	250	52.00	52.01	51.80	51.90	52.60	53.42	40.60	46.14
GPT-J	NONE	300	50.90	50.78	58.60	54.41	51.70	51.72	51.00	51.36
GPT-J	ROME	10	78.08	98.05	57.31	72.33	70.05	68.83	73.27	70.98
GPT-J	ROME	25	79.34	99.23	59.13	74.11	69.81	92.78	42.96	58.73
GPT-J	ROME	50	85.96	97.58	73.74	84.01	71.12	94.03	45.11	60.97
GPT-J	ROME	100	93.38	97.98	88.58	93.05	83.65	96.38	69.93	81.05
GPT-J	ROME	150	93.95	98.25	89.50	93.67	87.23	94.57	79.00	86.09
GPT-J	ROME	200	94.98	98.28	91.55	94.80	89.26	94.82	83.05	88.55
GPT-J	ROME	250	95.78	98.78	92.69	95.64	90.93	95.25	86.16	90.48
GPT-J	ROME	300	96.12	98.79	93.38	96.01	91.17	94.34	87.59	90.84
GPT2-XL	MEMIT	10	47.53	48.50	79.63	60.28	52.63	53.46	40.67	46.20
GPT2-XL	MEMIT	25	61.73	69.79	41.36	51.94	58.13	60.90	45.45	52.05
GPT2-XL	MEMIT	50	68.21	74.38	55.56	63.60	65.79	67.01	62.20	64.52
GPT2-XL	MEMIT	100	71.91	73.83	67.90	70.74	72.25	75.14	66.51	70.56
GPT2-XL	MEMIT	150	74.69	76.32	71.60	73.89	74.64	75.12	73.68	74.40
GPT2-XL	MEMIT	200	76.85	77.36	75.93	76.64	75.12	76.92	71.77	74.26
GPT2-XL	MEMIT	250	78.70	80.39	75.93	78.10	77.75	77.88	77.51	77.70
GPT2-XL	MEMIT	300	75.93	79.58	69.75	74.34	77.51	78.89	75.12	76.96
GPT2-XL	NONE	10	50.00	50.00	1.00	1.96	50.00	50.00	9.20	15.54
GPT2-XL	NONE	25	51.50	51.72	45.20	48.24	50.50	50.68	37.40	43.04
GPT2-XL	NONE	50	51.00	51.14	44.80	47.76	50.60	50.78	39.00	44.12
GPT2-XL	NONE	100	52.10	52.02	54.20	53.09	51.40	51.59	45.40	48.30
GPT2-XL	NONE	150	52.40	52.56	49.20	50.83	50.40	50.47	43.40	46.67
GPT2-XL	NONE	200	53.00	53.23	49.40	51.24	51.30	51.49	45.00	48.03
GPT2-XL	NONE	250	52.80	52.98	49.80	51.34	52.50	52.81	47.00	49.74
GPT2-XL	NONE	300	51.50	51.37	56.40	53.77	50.60	50.58	52.20	51.38
GPT2-XL	ROME	10	78.40	99.06	57.34	72.63	52.35	53.68	34.35	41.89
GPT2-XL	ROME	25	88.72	97.66	79.35	87.56	69.25	78.37	53.19	63.37
GPT2-XL	ROME	50	93.34	96.50	89.95	93.11	82.13	87.91	74.52	80.66
GPT2-XL	ROME	100	95.65	98.28	92.93	95.53	85.73	88.62	81.99	85.18
GPT2-XL	ROME	150	97.01	98.87	95.11	96.95	88.23	90.12	85.87	87.94
GPT2-XL	ROME	200	97.28	98.88	95.65	97.24	90.44	92.20	88.37	90.24
GPT2-XL	ROME	250	97.42	99.15	95.65	97.37	92.80	94.78	90.58	92.63
GPT2-XL	ROME	300	97.42	99.15	95.65	97.37	93.35	95.63	90.86	93.18

Table 7: Detecting edits performance with different numbers of training data. "_zs" refers to the results for ZSRE, and "_cf" is for COUNTERFACT.

Setting	Input	Prediction	Label
default	Over what river does Japoma Bridge cross? Sanaga River	1	1
default	Which was the architect from Life Electric? Daniel Libeskind	1	1
default	What wife was Charles Stuart, Duke of Kendal? Anne Hyde	1	1
default	What country has Coca-Cola Raspberry won? New Zealand	1	1
default	Where did Olin Howland die, then? Hollywood	0	1
default	What war did Heinrich von Zastrow have in? Austro-Prussian War	0	1
default	In which network is Bathroom Singer available? Filmy	0	1
default	What is the Morro Velho product? gold	0	1
default	Which series is the A Special Edition? The Outer Limits	0	0
default	What year was Grameen Bank founded? 1983	0	0
default	What constellation is HD 206267 at home? Cepheus	0	0
default	What sports team does Eloy Edu belong to? St. Andrews F.C.	0	0
default	Who designed the Let's Catch? Yuji Naka	1	0
default	In which network did Robot Chicken originally occur? Adult Swim	1	0
default	What's the date of Air France Flight 447? 1 June 2009	1	0
default	Who was the architect who designed the Villa Jeanneret? Le Corbusier	1	0
same object	Which year has Aster Società Italiana Motori finished? 1908	1	1
same object	Which was the founding year of New York, New Haven and Hartford Railroad? 1872	1	1
same object	What kind of occupation is John Huwet? politician	1	1
same object	What is the position of Andrea Pangrazio? bishop	1	1
same object	What is the Morro Velho product? gold	0	1
same object	From whom was RG-35 designed? Land Systems OMC	0	1
same object	What planet is Samarkand Sulci? Enceladus	0	1
same object	Which company is known as the producer of Issoire APM 40 Simba? Issoire Aviation	0	1
same object	Which year was MG 08? 1908	0	0
same object	What year was the LB&SCR Belgravia class commissioned? 1872	0	0
same object	What kind of occupation has Wu Chen-huan? politician	0	0
same object	How is the position of Norbert Felix Gaughan? bishop	0	0
same object	What state has Buckeye Trail High School? Ohio	1	0
same object	Which college, or what university, is related to Hugh Graham Miller? University of Aberdeen	1	0
same object	What is produced by Prominent Hill Mine? gold	1	0
same object	Which company has Madden NFL 13 published? EA Sports	1	0

Table 8: Examples of correct and incorrect predictions with MEMIT and GPT-J on ZSRE. 1 refers to the label "edited", whereas 0 to "non-edited"

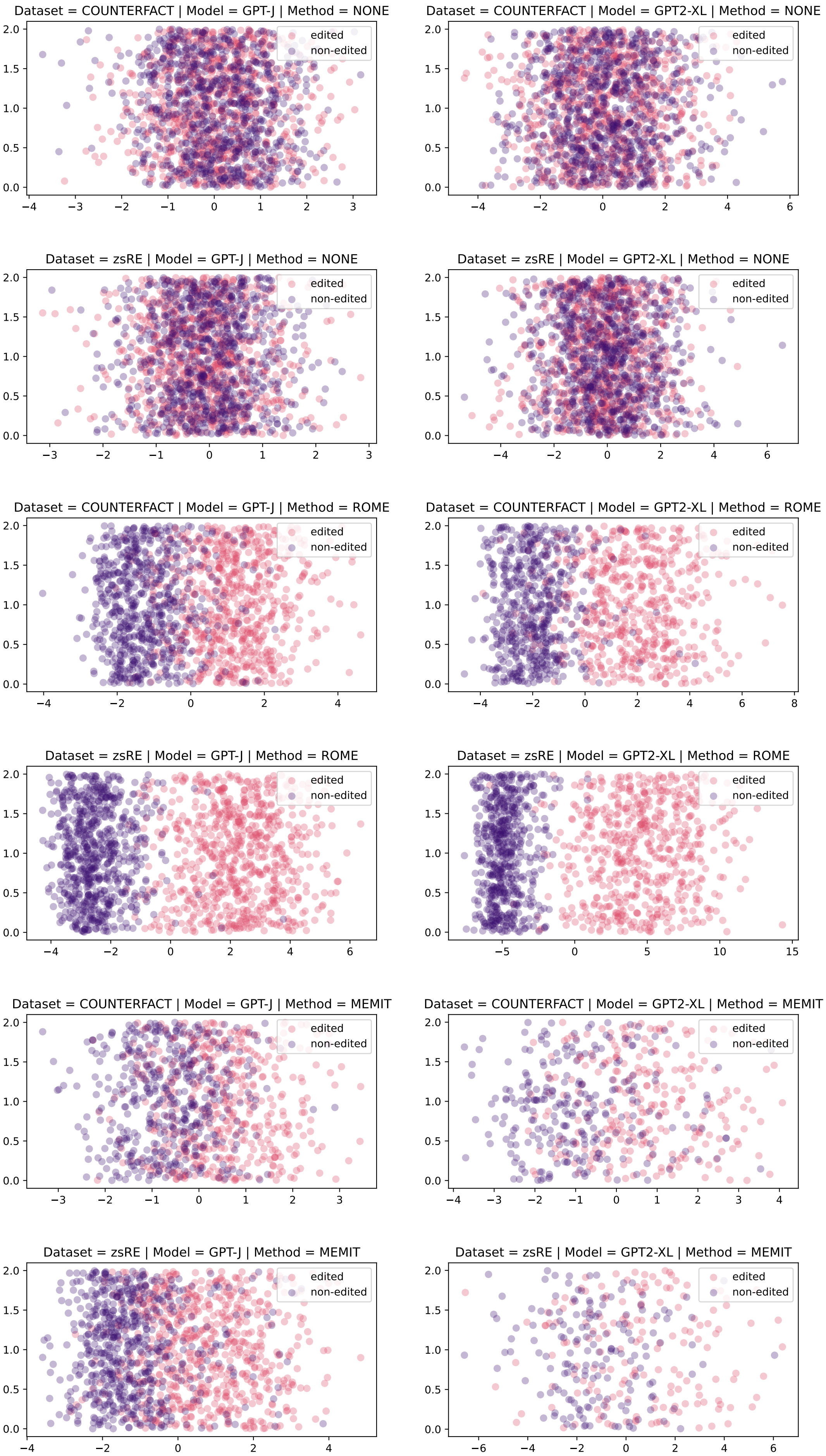

Figure 6: Representations of edited and non-edited facts in a two-dimensional space in the default setting.

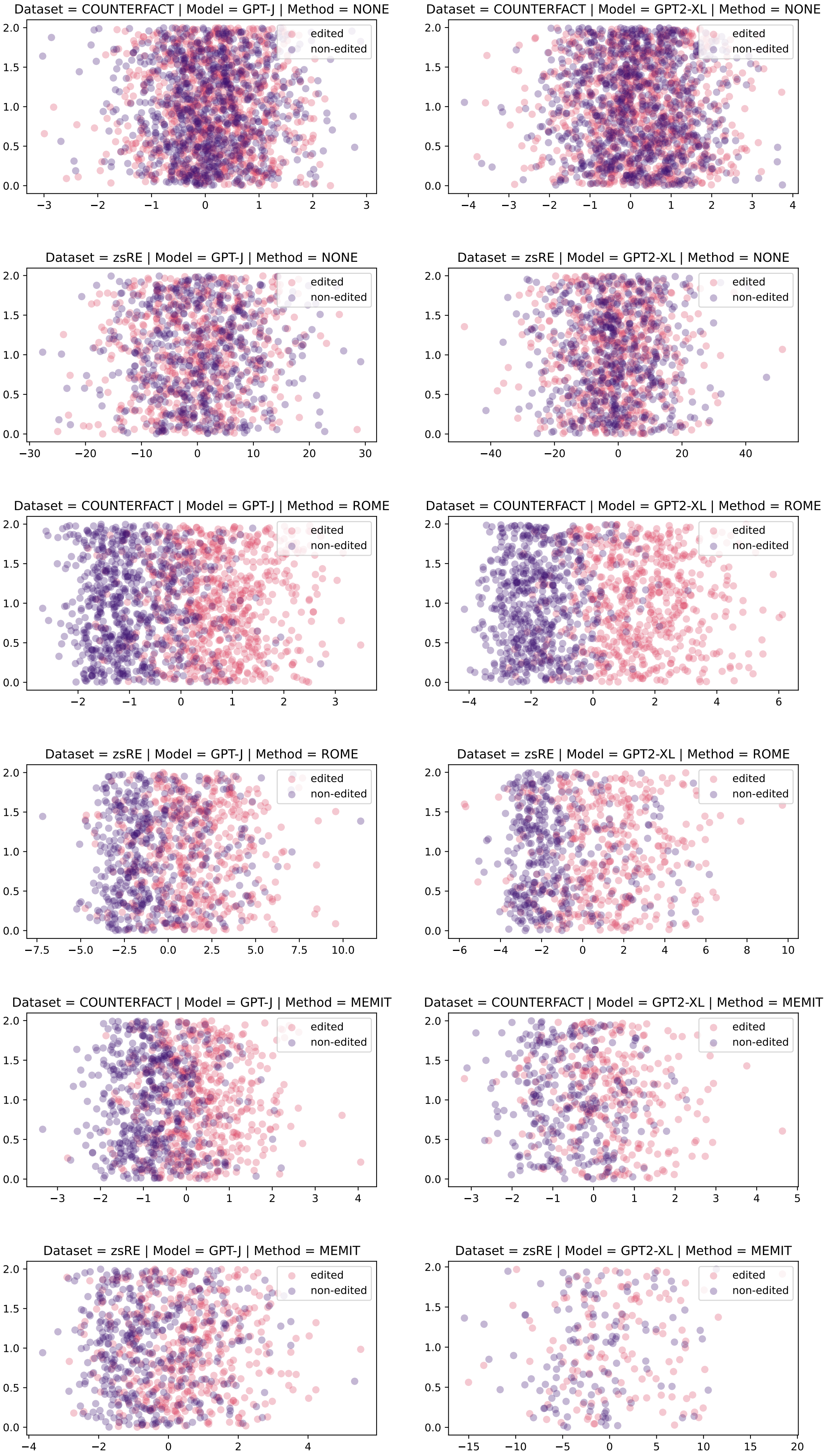

Figure 7: Representations of edited and non-edited facts in a two-dimensional space in the `same object` setting.

Generator		zsRE				CounterFact			
Model	Editor	Acc.	Pr.	Rec.	F1	Acc.	Pr.	Rec.	F1
Editing prompts									
GPT-J	NONE	47.86	47.63	42.93	45.16	47.92	47.98	49.29	48.62
	ROME	97.95	99.59	96.30	97.92	95.70	96.18	95.18	95.68
	MEMIT	86.43	87.76	84.67	86.19	88.85	89.21	88.39	88.80
GPT2-XL	NONE	48.83	48.80	47.34	48.06	48.31	48.32	48.64	48.48
	ROME	99.08	99.60	98.55	99.08	95.70	96.06	95.31	95.69
	MEMIT	76.42	78.14	73.36	75.68	78.96	78.75	79.32	79.03
Paraphrased prompts									
GPT-J	NONE	49.40	49.35	45.35	47.26	48.93	49.02	53.70	51.25
	ROME	96.78	98.83	94.68	96.71	91.19	94.27	87.70	90.87
	MEMIT	82.45	86.19	77.27	81.49	75.87	77.55	72.83	75.11
GPT2-XL	NONE	49.40	49.33	44.03	46.53	48.45	48.52	50.72	49.59
	ROME	98.08	99.46	96.69	98.06	94.11	96.08	91.96	93.98
	MEMIT	75.93	79.58	69.75	74.34	74.90	77.73	69.80	73.55

Table 9: Classification performance on detecting edited knowledge. Representations are based on the **editing prompts** that are used for editing, and **paraphrased prompts**.

Generator		zsRE				CounterFact			
Model	Editor	Acc.	Pr.	Rec.	F1	Acc.	Pr.	Rec.	F1
GPT-JT	NONE	49.64	49.76	74.22	59.58	49.11	49.22	56.56	52.64
	ROME	91.92	99.78	84.03	91.23	92.87	92.87	92.87	92.87
	MEMIT	74.53	85.81	58.77	69.76	76.80	72.61	86.08	78.77
GPT2-XL lima	NONE	49.22	49.22	48.69	48.95	48.99	49.03	51.31	50.15
	ROME	97.79	99.40	96.15	97.75	92.74	94.97	90.26	92.55
	MEMIT	75.79	78.82	70.53	74.44	74.56	77.45	69.30	73.15

Table 10: OOD classification performance. Training data is from GPT2-XL and GPT-J, whereas test data is from GPT-JT and GPT2-XL lima respectively.