Editing Large Language Models

Gokul Sunilkumar (gs3214), Pooja Srinivasan (ps3312),
Sai Rithvik Kanakamedala (sk5112), Saili Myana (sm5292), Utsav Vachhani (ukv2101)

December 17, 2023

Abstract

Large Language Models (LLMs) today encapsulate a vast amount of information and generative capabilities. This is supported through a complex architecture driven by hundreds of millions of parameters. However, since the model's factual knowledge can change over time, it requires a way to stay updated. To address this, various editing methods have been developed to address incorrect facts and provide a more computationally efficient alternative to retraining the LLMs.

We analyze the storage paradigm that autoregressive transformer models follow to facilitate easy retrieval of specific facts driven by the input prompt. This is done to edit factual associations stored in these locations to extend the current knowledge that the language models hold and ensure that they store information that is current and relevant. We perform a series of experiments to analyze the location of facts. The examination has also been performed on editing methodologies followed by current strategies and certain shortcomings are identified and underlined. Overall, our work drives the direction of future research to take into account the properties of a fact in itself along with all its input tokens to find the corresponding knowledge centers in transformers.

1 Introduction

In recent years, the field of machine learning has witnessed a surge in the development of large-scale models, particularly in domains like computer vision and natural language processing (NLP). These models, with their impressive performance, have revolutionized various applications. As big data have taken a pivotal role in the field, the models have been developed to utilize this vastness of knowledge. Specifically, this has emerged into the Large Language Models (LLMs), which have adapted to this big data with billions of parameters. Notable large language models include OpenAI's GPT-3.5 and GPT-4 as well as Google's PaLM. GPT-3.5 has 175 billion parameters, PaLM with 530 billion and GPT-4 rumored to have access to 1 trillion parameters. With these high-dimensional models, there is an inherent need for greater control; however, managing and customizing the behavior of these colossal models has become a significant challenge. Specifically, the need for precise, localized adjustments in their outputs, without disrupting unrelated inputs, has become a critical concern. Additionally, understanding factual associations within these large networks is crucial to applying these adjustments. Existing techniques, such as fine-tuning, often grapple with the intricate balance between making localized edits and preserving the model's overall generality. Our work explores the context and challenges surrounding the editing and knowledge localization of large language models, offering insights into the evolving landscape of model adaptation and customization.

Problem Statement: To retrieve the edit locations of edit algorithms and analyze the activation patterns of layer-token combinations to understand their contribution in the correct prediction of the next token, with an overall intention of coming up with an updated edit algorithm that can perform fast and scalable edits within the language model.

In the knowledge attribution method, the knowledge neurons in feed-forward are identified by computing the contribution of each neuron to the knowledge prediction. The attribution score is assigned to each neuron of each layer by integrating the gradients of output probability concerning the neuron activations by changing the activation from 0 to the activation value observed from the pre-trained model. Experiments were conducted to show that knowledge neurons are positively correlated to

knowledge expression. Preliminary studies of leveraging knowledge neurons to edit factual knowledge in Transformers were also done.

Recent advances in neural network editing have given rise to a novel approach called Model Editor Networks with Gradient Decomposition (MEND). MEND addresses the limitations of traditional fine-tuning techniques by leveraging the rank-1 property of gradients in fully connected layers [1]. This allows MEND to represent gradient updates efficiently with fewer parameters and less memory, making it well-suited for large models. MEND also showcases scalability by sharing parameters among model editor networks. It incorporates identity initialization and input normalization components, enhancing its effectiveness, maintaining model accuracy, and reducing training time. However, challenges related to edit locality and scalability still exist.

Another tangential work is MEMIT [2], a novel method for updating large language models like GPT-J and GPT-NeoX with multiple memories, surpassing prior techniques by orders of magnitude in scalability. MEMIT targets specific transformer modules identified through causal mediation analysis to enable factual knowledge recall. It successfully stores thousands of memories and is evaluated for robustness in terms of generalization, specificity, and fluency, outperforming rank-one, hypernetwork, and fine-tuning baselines. The paper also identifies a range of critical MLP layers involved in memory recall using the causal tracing method and demonstrates how individual MLPs contribute to storing memories across these layers. ROME [3]is the precursor to this work and just selects one of the critical layers identified through causal tracing that has the maximum indirect effect in the prediction of the correct next token.

Additional important work in the field of editing LLMs is the paper titled "Evaluating the Ripple Effects of Knowledge Editing in Language Models" which talks about how to go about evaluating these editing methods. The novelty in this paper comes from three aspects: (i) RippleEdits: A Diagnostic benchmark of 5K factual edits, capturing a variety of types of ripple effects while editing facts; (ii) Evaluating 3 KE methods (MEND, ROME, MEMIT) on 5 LLMs; (iii) They proposed a simple in-context editing baseline, that obtains the best scores on their benchmark test set.

2 Dataset

2.1 Collecting dataset - Capital facts

The dataset of capital facts is generated using the Country details library in Python. We were able to generate 239 capital facts, which only consist of the capitals of countries. Further, we compiled a list of 50 capital facts, encompassing the capitals of each state in the United States, finally resulting in 289 capital facts.

Out of the 289 facts, the gpt2-xl model was able to predict the facts correctly for 176 facts. For the remaining 113 facts, the predictions made by the model were incorrect. This also led to creating the correctly predicted v/s incorrectly predicted dataset.

2.2 Extracting sports person facts - Popular and Rare

We developed a Python module that facilitates web scraping specifically designed to extract information from Wikipedia pages associated with individuals mentioned in a curated list of 190-200 names. Initially, the code reads names from a specified file and employs a custom function, get_wikipedia_page_length, to retrieve the length of Wikipedia pages for individuals by constructing the appropriate URL and utilizing BeautifulSoup for parsing. The script processes each line, distinguishing between scenarios where a person "plays the sport of" or a place "is the capital of". For each case, it extracts the relevant name, conducts web scraping to obtain the Wikipedia page length, and compiles the results into a dictionary named 'player_lengths'. An interesting facet of the process is that, out of the 190-200 prompts initially considered, some may not be present in the final prompts list, as their corresponding Wikipedia pages do not exist. Subsequently, it sorts the prompts based on

the Wikipedia page lengths in descending order and prints the sentences along with their respective lengths.

2.3 Collecting In-Context facts dataset

For this experiment, we were inspired from the In-context baseline for editing introduced by the Ripple Edits paper. From the dataset of edits described in the paper, we used the compositionality test prompts added to the original facts as context. Compositonality test prompts/edits are curated to test composite relations on the given subject. For example, consider the fact: "The official language of the country of citizenship of Leonardo DiCaprio is English.". We edit this fact by giving the context: "Imagine that the name of the country of citizenship of Leonardo DiCaprio is Syria." and then test the same prompt "The official language of the country of citizenship of Leonardo DiCaprio is" again. In this way, we collected several context-prompt pairs to test the model. We thus curated a collection of 219 in-context prompts that are correctly predicted by the GPT2-xl model.

3 Comparing edits made by different Editing methods

In this section, we compare the location that each of the editing models edit in order to verify if they are trying to bring in structured changes in the layer activation values at the same or distinct locations through the following table. Fields marked - signify that the editing methodology is not supported with the associated autoregressive decoder-only transformer model.

Model	MEMIT	ROME	MEND
gpt2-xl	13-17	17	Layer-specific
gpt-j	3-8	5	Layer-specific
gpt2-large	-	12	Layer-specific
gpt2-medium	-	8	Layer-specific
gptneox-20b	-	15	Layer-specific

MEMIT focuses on minimal edits to model parameters, ensuring that changes are as small as possible while still achieving the desired output modification. The location of edit is tightly constrained to specific parameters, maintaining the model's overall integrity and performance on other tasks. MEMIT is likely to implement precise, pinpointed adjustments. It uses the causal tracing method to find the set of layers that contribute the most in reverting the wrongly predicted next token to the right one by virtue of the copying of the clean hidden state to the corrupted hidden state.

ROME's approach is more holistic, potentially involving broader adjustments to the model's parameters. While it aims to correct specific outputs or behaviors, its editing paradigm might allow for more extensive modifications than MEMIT, potentially affecting a larger magnitude of the model's parameter. It identifies one specific layer from the MEMIT layers that has the maximum magnitude of the average indirect effect towards prediction of the correct output token. Therefore, the location where ROME makes an edit is a subset of the location where MEMIT makes edits, i.e, they look at the same loctions.

MEND stands out for its efficiency in editing large-scale models. It employs auxiliary networks to make fast, local edits based on gradients, indicating that its edits are likely focused on specific parts of the model influenced by these gradients. MEND's paradigm involves leveraging existing model structures (like gradients) for efficient and targeted editing. It trains layer-specific editing models that can be used to edit layers of our interest. This method could definitely be used to drive future research as it could help us understand the effectiveness of edits based on the properties of the input prompt instead of rigid methods that have a fixed set of layers for any type of prompt.

4 Methodology

4.1 Causal Tracing

Causal tracing refers to a method used to analyze and understand the causal relationships and dependencies within a neural network, particularly in the context of natural language processing tasks like next-word prediction. The goal is to identify the contribution of specific hidden state variables in a neural network to the final output and to understand how they influence the prediction process. The key components and steps involved in causal tracing are as follows:

Causal Graph: The neural network is represented as a causal graph, a concept rooted in causal inference theory [4]. In this graph, nodes represent hidden variables, and edges represent dependencies between these variables. The graph illustrates the flow of information from input to output, capturing the network's internal workings.

Causal Mediation Analysis: Causal mediation analysis, inspired by the work of Vig et al. [5] and Pearl [6], is employed to quantify the contribution of intermediate variables in the causal graph. This helps identify the importance of specific hidden state variables in the network's decision-making process.

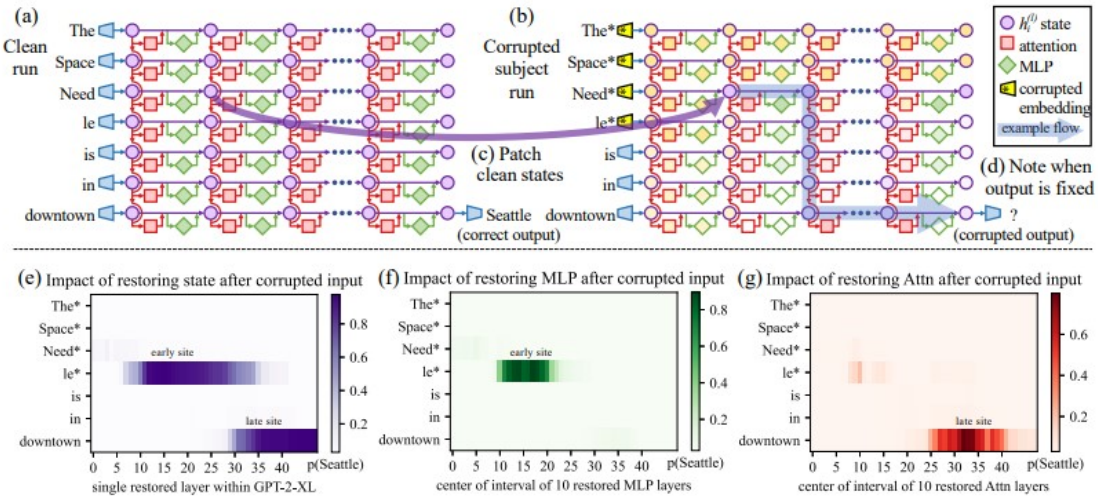


Figure 1: (a) normal processing, (b) intentional corruption of the subject token, and (c) restoration of selected internal activations. Some activations lead to the original prediction. The causal impact on output probability is mapped for the effect of (e) each hidden state, (f) only MLP activations, and (g) only attention activations.

Experimental Setup: The analysis involves three experimental runs:

- Clean Run: The network processes a factual prompt normally, and hidden activations are collected. This run represents the ideal scenario where the network produces the correct prediction.
- Corrupted Run: The subject (e.g., a specific word or token) is intentionally obfuscated before the network processes the input. The resulting activations are collected, and this run simulates a scenario where the prediction is damaged due to the loss of information about the subject.
- Corrupted-with-Restoration Run: Similar to the corrupted run, but at a specific token and layer, the network is forced to output the clean state, allowing for the restoration of selected internal activations. This run helps assess the ability of certain clean states to recover the correct prediction despite other states being corrupted.

Information Flow Analysis: Causal tracing involves analyzing the information flow within the network during these three runs. This includes understanding how corrupted states impact the pre-

diction and how specific clean states contribute to restoring accurate predictions.

Causal Traces: Causal traces map the causal effect of neuron activations on the final output probability. This analysis is done for various aspects, such as the impact of each hidden state, only the activations from the Multi-Layer Perceptron (MLP), and only the attention activations. These traces help visualize and quantify the influence of different components on the network's predictions. A detailed view is given in Figure 1.

In summary, causal tracing is a methodological approach to dissect and understand the causal relationships within a neural network, shedding light on the importance of different hidden states in the decision-making process.

4.2 Causal Tracing of relations instead of subjects

To further build on the causal tracing algorithm introduced earlier, we decided to focus on the relations within the knowledge tuples. We went through the code-base for the causal tracing algorithm and identified where the corruption of tokens is happening. We used this to change the code such that any tokens in the prompt can be corrupted by the algorithm to find the causal traces. Above all, we want to understand the differences in the layers identifies by causal tracing when different tokens are corrupted. If it is true that facts are stored in certain layers, the same layers should be consistently identified when any tokens are corrupted. Yet, we see that this doesn't happen. This prompted us to look into layers identified by causal tracing when relations are corrupted instead of the traditional way of corrupting the subject.

One example to show the differences is described below:

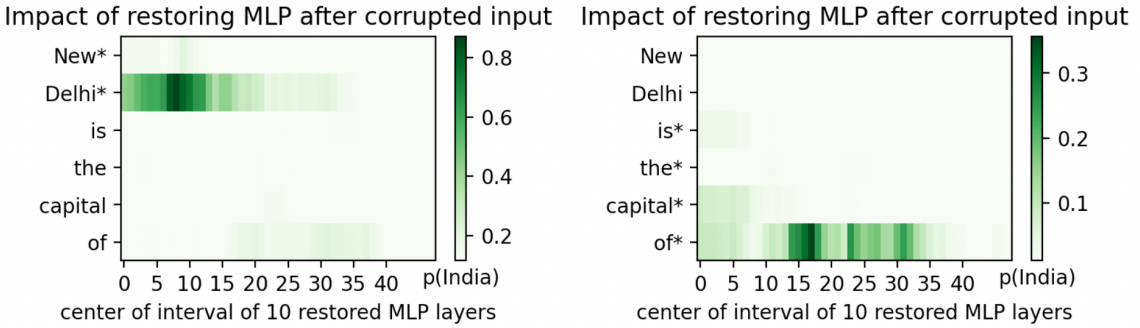


Figure 2: Causal tracing when corrupting subject and relation resp.

In the above figure (fig.1), causal tracing is performed twice on the prompt "New Delhi is the capital of", and the object predicted is "India" as expected. The difference between subject corruption ("New Delhi") and relation corruption ("is the capital of") is apparent. We observe that when subject is corrupted, layers 5 to 10 are activated the highest showing the highest causal impact, however layers 15 to 20 have the highest impact when the relation is corrupted. This may imply that both these sets of layers are used to store this fact. The following visualizations further observe the layers identified by causal tracing when subject and relations are corrupted.

5 Data visualization

To visualize the layers identified by the causal tracing of different facts, we consider three types of plots, Average distribution, peak distribution, and Inter-Quartile Range (IQR) distribution. Described below is how we define and calculate these plots, and the plots corresponding to the different experiments are shown and discussed in the respective results.

For all these plots, we consider the causal impact of the layers corresponding to the last corrupted token for each prompt. This is because the impact for each token is consolidated in the last token, which was observed in all the examples we considered. This is also consistent with the observations of the ROME paper [3]. In this report, the causal impact distributions are used interchangeably as the probability distribution since the causal impact of different layers is measured as the probability of the object (answer).

5.1 Average causal impact distribution

Firstly, we plot the average causal impact of all prompts over the layers. This plot has x-axis as the layer numbers and the y-axis corresponds to the mean causal impact (object probability) for all prompts.

5.2 Peak layer distribution

Secondly, we observe the one layer with the highest causal impact for each prompt. This is plotted as a distribution over all layers.

5.3 Inter-Quartile Range distribution

Finally, we wanted to understand if the facts have a narrow peak or flatter peak for different types of facts. For this we plotted the Inter-Quartile ranges for the causal impact distributions. So for each fact, we get one value of IQR as the difference in layers that make up 75 percent and 25 percent of the total causal impact. These IQRs for each fact are plotted as a distribution over layers.

6 Experiment 1: Finding Knowledge Centers in LLMs

The current methods only edit specific layers for all types of facts. We hypothesize that facts with different relations are stored by the model in different layers of the network. With a dataset featuring over 170 prompts for capital and sports categories, we meticulously analyze variations in activations, providing insights into the distribution of knowledge across layers.

6.1 Dataset

The dataset for the capital relation and sports person relation has been curated as mentioned in sections 2.1 and 2.2. There are 176 prompts related to capitals and 171 prompts related to sports persons for which the gpt2-xl model can predict the fact correctly.

6.2 Modeling

These prompts are given as input to the gpt2-xl model with their object masked. The model then tries to predict the next word with a certain probability. Using causal tracing for each prompt, we get 48 activation values associated with each token (since there are 48 layers). Since we are corrupting the relation in this experiment, we extract the activation values corresponding to the last token of the relation for all 48 layers and plot an average activation plot across layers to see where the knowledge related to the relation is stored.

Further, we also visualized the peak distribution and interquartile range distribution for capital and sports persons relations when the relation is corrupted.

6.3 Results and Observations

From the plots in fig.3, we can see that the peaks for the capital facts and sports persons facts when relation are corrupted are around the 17th layer and 31st layer respectively. This shows that knowledge related to the capital facts is stored in layers 16-18. But for the facts related to sports persons, the knowledge is stored in layers 30 - 34. Further, the analysis of peak distribution when relations are corrupted, in fig.4 confirms this observation with a clear difference in the peak layers for both the datasets. Further, fig.5 shows that the IQR for the causal impact distribution is also very different, as

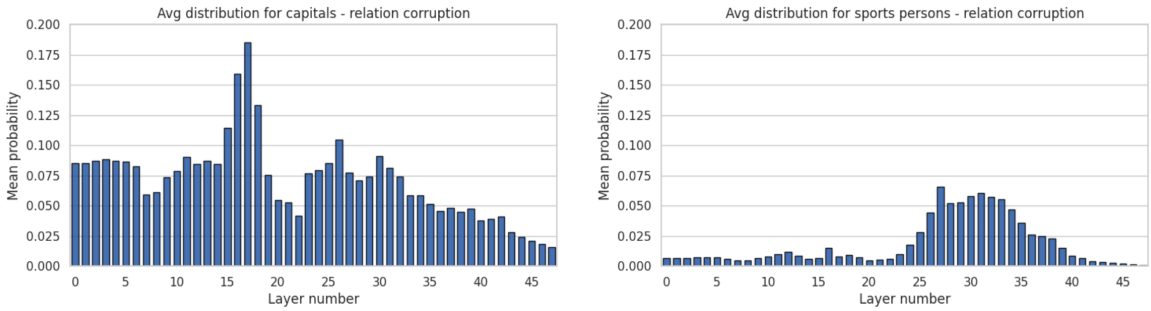


Figure 3: Average probability distribution over layers for capital prompts and sports prompts resp.

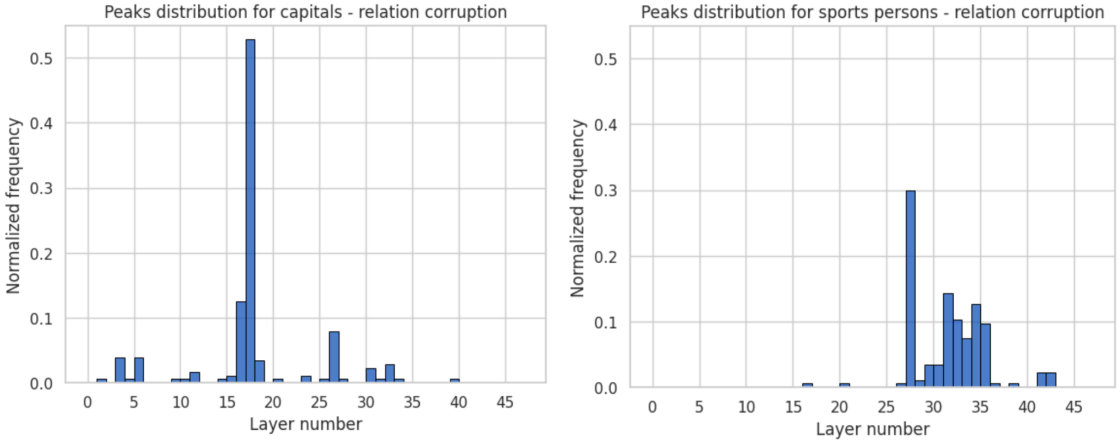


Figure 4: Distribution of peaks for facts of capitals and sports persons relations resp.

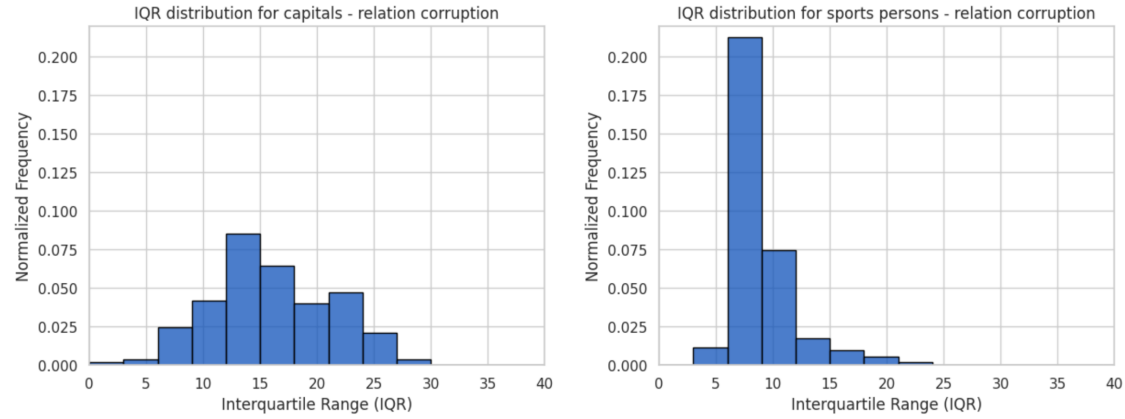


Figure 5: Distribution of IQR for facts of capitals and sports persons relations resp.

the mean IQR for capital relations significantly higher than one for the sports person relation. This means that the distribtuion for capital relation prompts are much flatter in general, meaning that the causal impact is spead around more layers while the causal impact distributions for the sports person relation are much narrower comparitively. This is signicant since the current SOTA (State of the Art) model MEMIT edits only top 5 layers for all facts, while our analysis shows that different types of facts may need different number of layers too, along with different layer ranges, i.e., top-5 layers won't

be comprehensive enough to store all type of facts.

7 Experiment 2: Exploring Popular vs Rarer entities

In this experiment, we explore the differences between editing a well-known fact and a less common or rare fact. We analyze sentences structured in the "subject-relation-object" format, considering pairs of prompts at a time. Both prompts in each pair belong to the same category, ensuring that the relationship between the subject and the object remains consistent. For example,
Prompt 1: Michael Jordan plays the sport of Basketball
Prompt 2: Steve Kerr plays the sport of Basketball

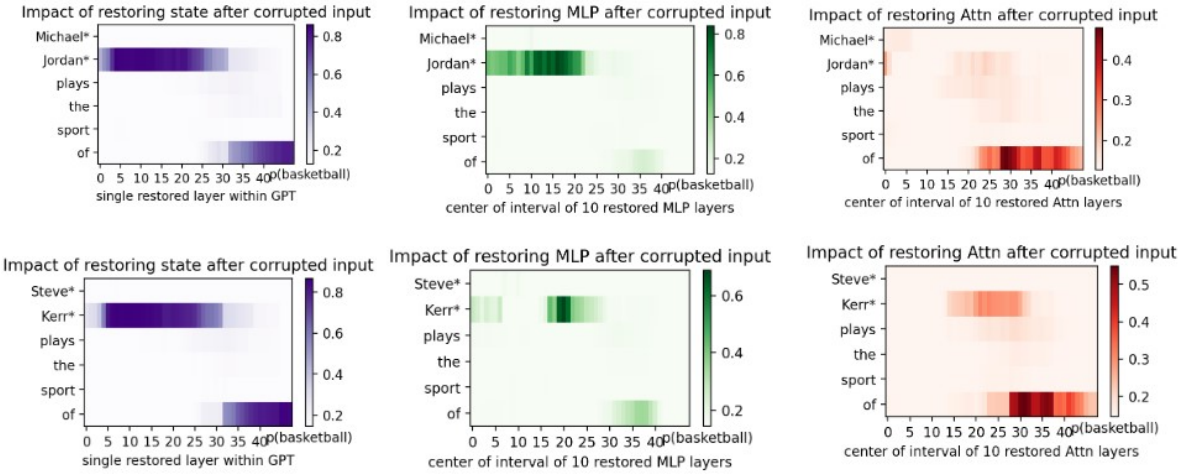


Figure 6: Causal Tracing for editing Popular v/s Rare facts

7.1 Dataset

The dataset for both popular and rare facts is compiled as detailed in section 2.2, employing the length of Wikipedia pages as a metric to gauge the popularity of individuals or entities.

7.2 Modeling

In our investigation, we utilize pairs of prompts, each comprising two distinct subjects. One subject revolves around a widely recognized or popular fact, while the other focuses on a rare fact. The key similarity lies in the identical relation between them. Our primary aim is to analyze activations across different layers and identify discernible patterns, particularly during the model's prediction of the object in these sentences. Employing causal tracing for each prompt, we extract activation values corresponding to the last token across all 48 layers (Figure 15). Subsequently, we generate an average activation plot across layers to visually discern where knowledge about popularity is stored.

In addition to the aforementioned analyses, we also visualize the peak distribution and the interquartile range distribution for popular and rare facts as mentioned in section 5. This visualization is conducted separately for subjects, relations, and all token corruptions. The objective is to delve into intricacies and enhance our understanding by demonstrating that comparable facts, whether popular or rare, exhibit higher activations in analogous layers.

7.3 Results and Observations

It became apparent that rarer facts consistently demonstrated higher activations in the upper layers of the model, surpassing their popular counterparts. Conversely, popular facts consistently exhibited equal or even higher activations when compared to their rarer counterparts. To validate this observation across a diverse set of prompts, we examined the average activations in all layers for each prompt,

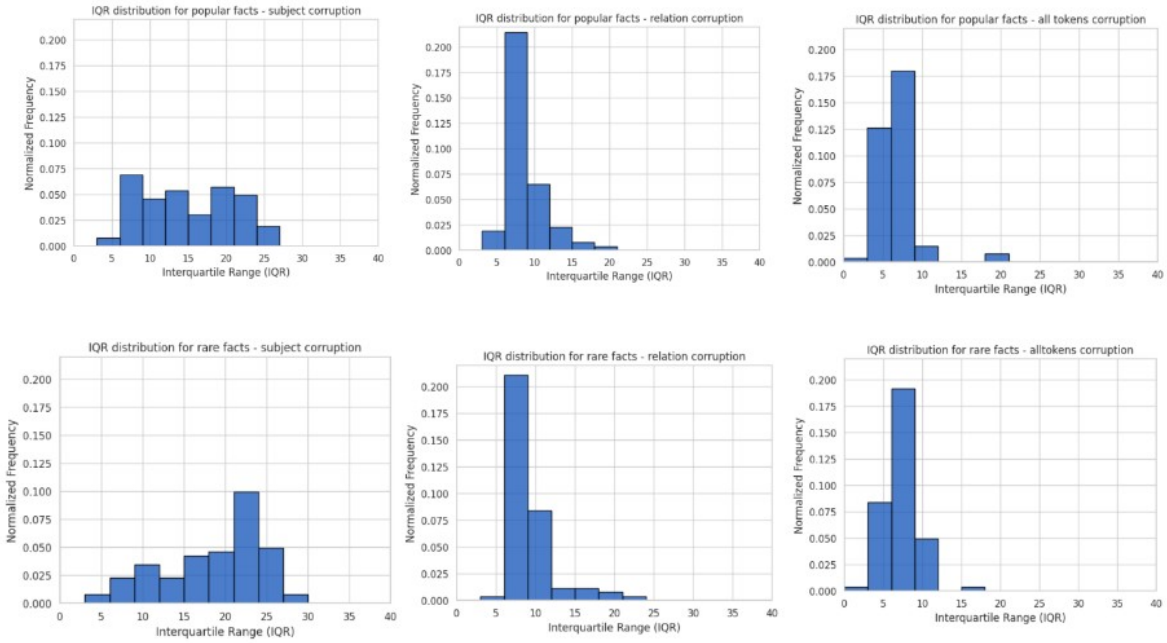


Figure 7: IQR distribution for popular and rare facts (subject, relations, and all tokens corruption)

encompassing both popular and rare facts across approximately 170 prompts. An interesting trend emerged, indicating that prompts of a similar nature displayed peaks in comparable layers (specifically layers 15-20).

However, it is noteworthy that the confidence of predictions tends to be higher for popular facts than for rarer facts, as illustrated in Figures 8 and 9. The visualizations of the interquartile range (IQR) and peak distribution further strengthen the notion that analogous facts store information in similar layers, as depicted in Figure 7.

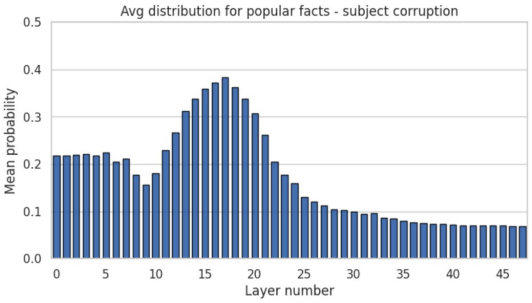


Figure 8: Avg probability distribution for popular facts

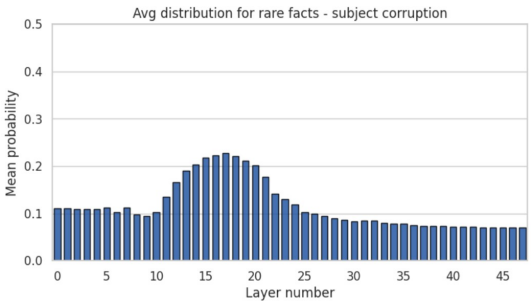


Figure 9: Avg probability distribution for rare facts

8 Experiment 3: Causal tracing on 6 different relations

In this experiment, we decided to dive deeper into how different relations can affect the causal impact on the object prediction probabilities. The knowledge tuples have a (subject, relation, object) format with the GPT-2XL model predicting the object. The prompt consists of the subject and the relation. For this, we chose 6 relations that encompassed varying topics to fully encapsulate the factual components. The object prediction was handled the same as before. For instance, here are two prompts from the same relation given to the model:

Prompt 1: Coca-Cola is headquartered in Atlanta
Prompt 2: Microsoft is headquartered in Seattle

8.1 Dataset

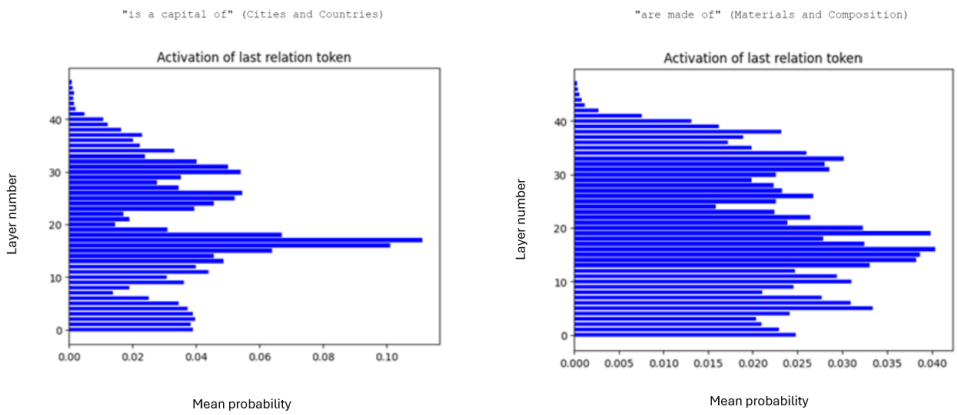
The data for the six relations was collected manually through trial and error. Due to the limited capabilities of the GPT-2XL model, object prediction faced a lot of obstacles. Therefore, each relation prompt had to be tried manually to ensure the correct object was being predicted. Each relation had 5 different prompts and objects collected with a total sample size of 30.

8.2 Modeling

To gain insights into the activation patterns of different relations, we fed the prompts into the model to compute the activation value graphs using casual tracing. For each of the 6 relation prompts, we extracted the activation values from the respective layers for each of the relation prompts. These activation values were recorded for the last subject token and averaged for each of the 5 prompts per relation.

8.3 Results and Observations

From the activation plots in figs.10, 11 and 12, there is a demonstrated difference between each relation prompts. The distribution of the activation for each respective last relation token peaks at separate layers. For instance, in the animals and habitation relation, the peak seems to occur in layers 31-33. This is in contrast to the headquarters relation where the highest casual impact is measured closer to layers 20-22. This fundamentally highlights that the storage of these different relations resides in different layers within the model.

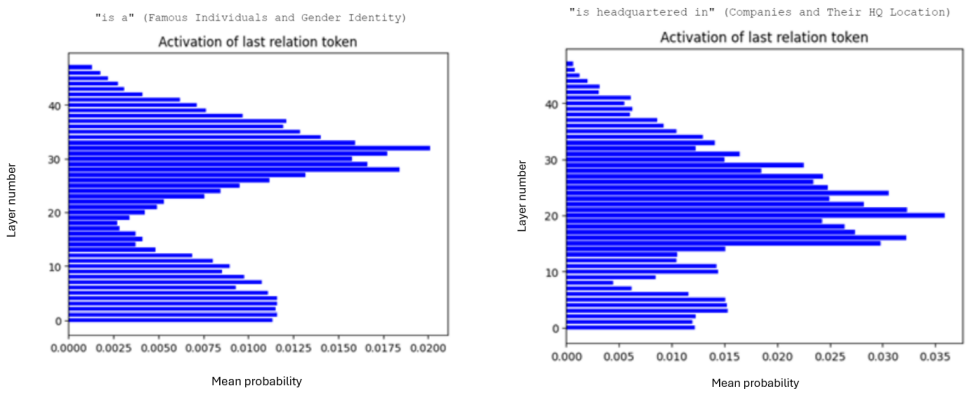


(a) "is capital of" relation (b) "are made of" relation

Figure 10: Mean causal impact for different relations

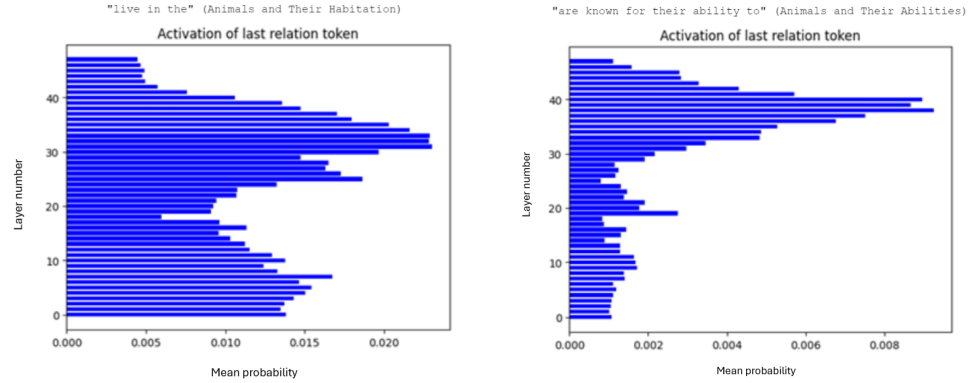
9 Experiment 4: In-Context vs Without context

The paper "Evaluating the Ripple Effects of Knowledge Editing in Language Models" [7] introduces a baseline method for Editing LLMs called In-Context Editing. This editing method does not change any parameters of the model but rather appends the edit to the prompt given to the model. This way, the LLM can take the edit as the context for answering the prompt. To quote an example from the paper, consider the prompt: "Imagine that the father of Barack Obama is Bill Clinton. The paternal grandmother of Barack Obama is". This prompt returns the answer "Virginia Clinton Kelley". This shows that the model can edit the facts previously learned and answer. The paper claims this method outperforms the other editing methods on accuracy on ripple effects. Hence, we aim to learn what layers these prompts with context are activating, and how this differs from the prompts without any



(a) "is a" relation (b) "is headquartered in" relation

Figure 11: Mean causal impact for different relations



(a) "live in the" relation (b) "are known for their ability to" relation

Figure 12: Mean causal impact for different relations

context.

One in-context editing prompt is provided and causal tracing is performed to analyze the output probabilities for different tokens and different layers. This is compared to the casual tracing done on the corresponding 'without context' prompt to compare how the model can capture the context.
Prompt 1 (without context example): "The official language of the country of citizenship of Leonardo DiCaprio is"
Prompt 2 (In-context example): "Imagine that the name of the country of citizenship of Leonardo DiCaprio would have been Syria, then the official language of the country of citizenship of Leonardo DiCaprio is"

For all of the examples of in-context editing in fig.13, we observed that the tokens for "Imagine that" were highly activated to predict the output. We realized through the token-layer plots that the casual tracing algorithms corrupt only the subject, and this way the importance is always higher for the subject tokens. This is the reason why "Imagine that" is given higher activations, because it is at the start of the prompt, like the subject.

We then decided along with our industry mentor Akshat, that this analysis needs a more comprehensive understanding of what tokens to corrupt while performing causal tracing. This experiment also does not immediately tie in with the rest of our work, since this is an editing method while the rest of our work focuses on localizing different types of facts. Yet, this is a very interesting research area, to identify how the activations of different neurons are able to consider the context while answering the prompt.

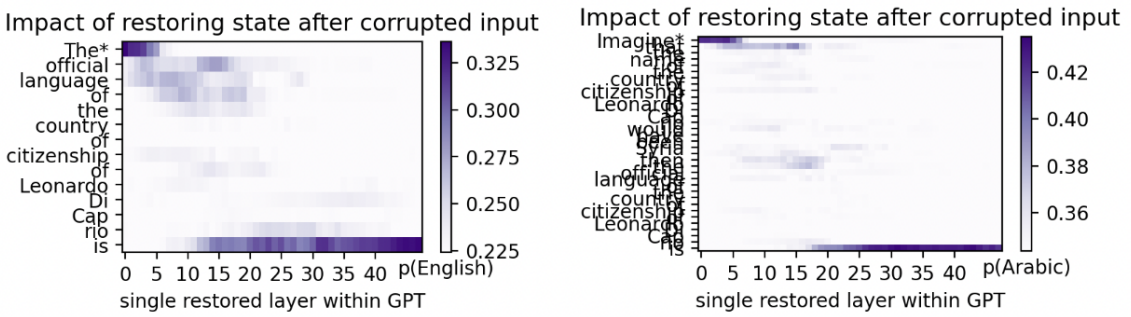


Figure 13: Without context vs In-context causal tracing

10 Experiment 5: Attribution-based method for tracing layers

To compare the results of the causal tracing of different knowledge centers with another algorithm to identify the layers used to store facts, we implemented the Attribution-based method from the paper "Knowledge neurons in pretrained transformers" [8]. Working through the implementation of this method, we realised that this method identifies (layer,neuron) tuples within the model where the given fact is said to be stored at. This method also required paraphrases of the given prompts to find the "refined neurons". After initial analysis of this attribution method on our dataset, we found that the algorithm does not identify any refined neurons for some of prompts. This is because the algorithm for this attribution method has a lot of hyperparameter thresholds and we need to adjust these thresholds for the code to work on our datasets. We along with our industry mentor Akshat decided not to focus on this experiment as it did not show enough promise for future direction and the threshold tuning was a research intensive process.

11 Experiment 6: Correct vs Incorrect predictions

11.1 Dataset

As mentioned in section 2.1, we first collect a set of facts of the capital relation. These prompts are passed as inputs to the gpt2-xl model with their object masked. The model then tries to predict the next word with a certain probability. We then segregate the capitals dataset into correctly predicted v/s incorrectly predicted based on whether the model is able to predict the object correctly or not. Finally, there were 176 capital facts that were correctly answered by the model and 113 capital facts that were incorrectly answered.

11.2 Modeling

Once again, we use causal tracing for each prompt to get 48 activation values associated with each token. We corrupt the subject in this experiment and extract the activation values corresponding to the last token of the subject for all 48 layers, and plot an average activation plot across layers to see if the average probability distributions are different for the correctly predicted v/s incorrectly predicted facts. Further, we also visualized the interquartile range distribution for capital and sports person relations when the relation is corrupted.
We also tried to use a random forest classifier to identify if the prediction made by the model is correct or incorrect. The classifier was trained on the activation values of each prompt, where each sample of data is a vector of 48 values.

11.3 Results and Observations

We look for a way to distinguish the incorrectly predicted facts from the correctly predicted facts based on the activation values. We can observe from the average probability distribution plot in fig.15 that there is no significant peak when the model's predictions are incorrect. Further, the mean IQR

from fig.14 is clearly different for correctly predicted facts from the incorrectly predicted facts. Coming to the machine learning model to identify if the model's predictions are correct or incorrect, the random forest classifier performed decently. The classification metrics look as shown in Tables 1 and 2.

As a baseline model, we use the probability of the word predicted as a feature to train another random forest classifier. The baseline model has an accuracy of 71% and an F1-Score of 0.66 on the test set. Compared to the baseline model, our model performs significantly better, achieving an accuracy of 88% and F1-Score of 0.88 on the test set. Hence, we can conclude that our model can better identify if the model's predictions are correct or incorrect.

	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	141
1	1.00	1.00	1.00	90
Accuracy			1.00	231
Macro Avg	1.00	1.00	1.00	231
Weighted Avg	1.00	1.00	1.00	231

Table 1: Classification metrics of random forest model on train set

	Precision	Recall	F1-Score	Support
0	0.97	0.83	0.89	35
1	0.79	0.96	0.86	23
Accuracy			0.88	58
Macro Avg	0.88	0.89	0.88	58
Weighted Avg	0.89	0.88	0.88	58

Table 2: Classification Metrics of random forest on test set

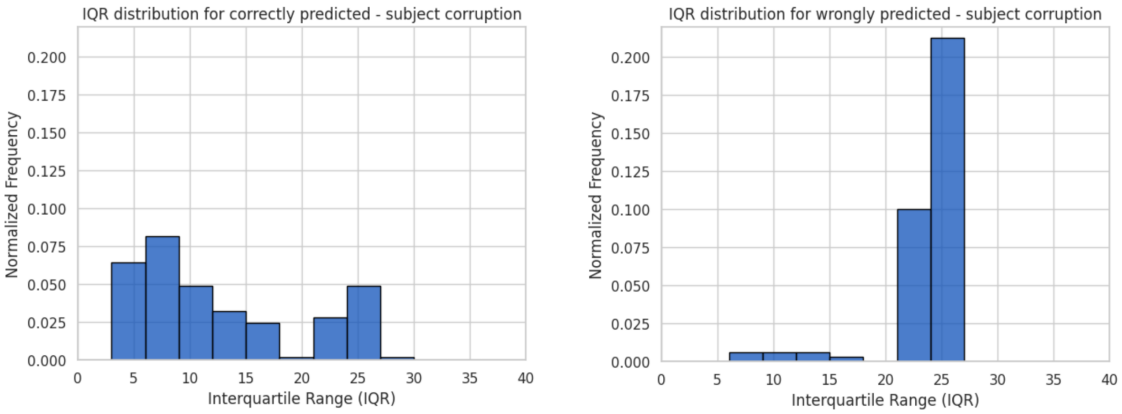


Figure 14: IQR distribution for correct and incorrect predictions

12 Software

The described methodology and experiments have been successfully implemented in a Python notebook hosted on Google Colab. Utilizing the well-organized MEMIT code template, our code structure incorporates functionalities for data collection, data pre-processing, refining prompts with pre-trained models, fine-tuning hyperparameters, and generating results through causal tracing. For accessibility

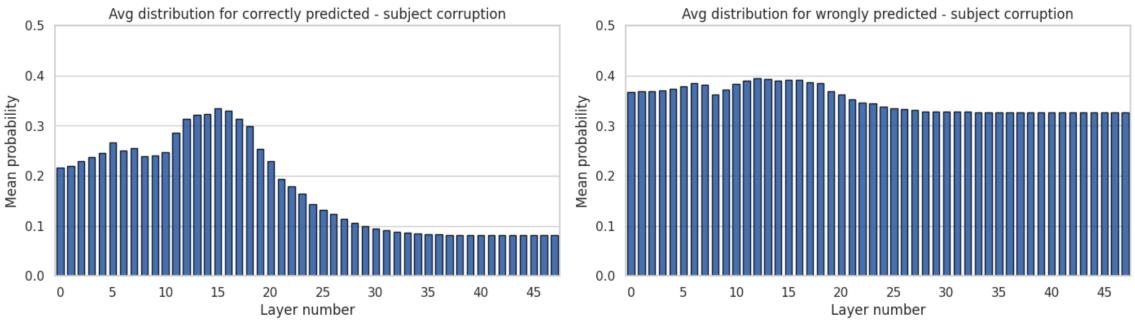


Figure 15: Average probability distribution for correct predictions, and incorrect predictions resp.

and clarity, we've shared the code and comprehensive documentation on our GitHub repository, accessible through the provided link. Inside the repository, Jupyter notebooks are available, allowing you to effortlessly execute the entire end-to-end process outlined in the methodology. [GitHub link]

13 Summary

To summarise, we started with a substantial literature review to understand the different state-of-the-art knowledge localizing and editing methods like attribution method, causal tracing etc. Understanding the shortcomings of the above methods, we started with a novel idea to modify the causal tracing algorithm and conducted various experiments to support our hypothesis that different types of facts are stored in different layers of the model. Experiments 1, 2 and 3 show the most promise and proof for this hypothesis, while we did not move forward with experiments 4 and 5 and they weren't directly relevant to our work even though they were promising areas of research. Experiment 6 is another bright area of research interest that shows promise based on our findings. We created datasets of different types of relations, popular v/s rare facts, and correct v/s incorrect predictions from scratch which are used for our analyses.

14 Conclusion

In conclusion, our work advances our understanding of knowledge storage within transformers as well as underscores the importance of considering the properties of a fact and its input tokens.

- Our exploration of corrupting relations, rather than just subjects, suggests a promising avenue for future research in identifying and editing specific layers.
- We highlight the limitation of current editing approaches that focus on specific layers for all prompts, demonstrating that different types of prompts may be stored in distinct layers.
- Analyzing incorrectly predicted prompts also opens avenues for deeper investigations into the model's behavior and potential clues for improving accuracy based on activation patterns.

This research paves the way for a more nuanced and effective approach to editing and understanding large language models.

15 Future Work

Overall, our analysis and observations are all qualitative, and we will be able to quantitatively evaluate our observations only after creating a new editing method based on our hypothesis. This was not in the scope of our project, but we point out great directions for improving editing models. If we had more time, we would have liked to expand on the incorrect vs correct predictions because an emerging area of research is how do we know if the model is answering facts correctly or incorrectly. For this purpose, we designed and conceptualized an elaborate baseline classification, which we would have

implemented if we had further resources and time. We also have identified clear challenges in the current editing methods and we could have implemented small modifications to the current methods to see if our findings show quantitative improvement in editing accuracy as a natural next step for our project. This would be the proof-of-concept editing method to show that our conclusions actually improve the editing LLMs paradigm.

16 Ethical Considerations

Our exploration into the editing of Large Language Models (LLMs), particularly through the MEMIT method and its counterpart editing methods, underscores the profound potential and significant ethical challenges of this technology. While our experiments demonstrate the ease with which facts can be localized and modified in models like GPT-2 XL, they also highlight the potential for misuse and the critical need for ethical oversight. It is crucial to note that these edits operate within the intricate web of model weights, making them less apparent to the naked eye. In a sense, they are hidden, yet their impact can reverberate across a myriad of model-generated answers, underscoring the need for meticulous scrutiny and ethical considerations in the editing process.

Inherent Risks: Our experiments reveal that even minor edits can have far-reaching implications, propagating misinformation, reinforcing biases, or inadvertently altering political and social narratives. With the scale at which generative pre-trained transformers (GPT) models are being utilized, the risks are further exponentiated.

Critical Need for Ethical Frameworks: The manipulation of LLMs, as shown in our experiments, necessitates robust ethical frameworks. These frameworks should encompass stringent guidelines for data accuracy, transparency in editing processes, and mechanisms to identify and correct biases. Furthermore, the frameworks should be inclusive of various societal perspectives to ensure no disparate impact occurs.

Importance of Transparency and Accountability: Given the hidden nature of these edits and their potential impact, transparency in the editing process and accountability for the content generated by LLMs are paramount. This involves clear documentation of edits, open channels for feedback, and regular audits by independent bodies.

Educational and Societal Implications: As LLMs become more integrated into educational and societal structures, the accuracy and integrity of the information they provide become crucial. Misinformation or biased content can significantly impact public opinion, educational outcomes, and social harmony.

Moving forward, it's vital to continue researching and developing methods to ensure the ethical use of LLMs. This includes ongoing evaluation of the impacts of model edits, enhancement of algorithms to detect and correct biases, and fostering a collaborative environment among AI researchers, ethicists, and policymakers. The advancement of AI and LLMs presents both remarkable opportunities and significant responsibilities. As we continue to push the boundaries of what these models can achieve, we must remain vigilant in ensuring they serve the greater good, uphold ethical standards, and contribute positively to the advancement of knowledge and society. Our project, while a step in understanding the complexities of LLM editing, is also a call to action for continuous ethical vigilance in the rapidly evolving field of artificial intelligence.

References

[1] E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning, "Fast model editing at scale," 2022.

[2] K. Meng, A. S. Sharma, A. Andonian, Y. Belinkov, and D. Bau, "Mass-editing memory in a transformer," 2023.

[3] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "Locating and editing factual associations in gpt," 2023.

[4] J. C. M. Pearl, Reasoning, and Inference, *Cambridge University Press, USA, 2nd edition.* ISBN 052189560X, 2009.

[5] J. Vig, S. Gehrmann, Y. Belinkov, S. Qian, D. Nevo, Y. Singer, and S. M. Shieber, "Investigating gender bias in language models using causal mediation analysis," in *NeurIPS, 2020b*, 2020.

[6] J. Pearl, "Direct and indirect effects," in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, 2001, p. 411–420.

[7] R. Cohen, E. Biran, O. Yoran, A. Globerson, and M. Geva, "Evaluating the ripple effects of knowledge editing in language models," 2023.

[8] D. Dai, L. Dong, Y. Hao, Z. Sui, and F. Wei, "Knowledge neurons in pretrained transformers," *CoRR*, 2021.