Causal Tracing of Text Generation in GPT-2-XL

Megan Kinniment
megan.k.w.12@gmail.com

Overview

This is the report for my final project of the Machine Learning Safety Scholars Program. The project consisted of writing an implementation of causal tracing for GPT-2 models, replicating a portion of Meng et al's causal tracing preprint[1]. This implementation was then used to to investigate the behaviour of GPT-2-XL on a wide variety of prompts. I (tentatively) think I may have found indications of two potentially interesting behaviours via this investigation. Firstly, it seems like there might be a fairly strong similarity between traces of hallucinated and true facts. Secondly, there might be a relationship between the layer at which information is retrieved from previous tokens and the complexity of that information. These two observations are discussed in the context of what information they could provide about the internal workings of GPT-2-XL and other large language models.

Github: `https://github.com/MeganKW/causal-trace`

1 Replication of Causal Tracing Results

Recently Meng et al. introduced causal tracing, a technique that can be used to study the flow of information through uni directional transformers.[1] The original authors perform a few different kinds of causal traces over different parts of the model: over entire layers, only over MLP layer components, and only over attention layer components. For this project I chose to just focus on replicating the causal tracing over entire layers.

For a more detailed view of implementation I would recommend looking at the accompanying notebook, but to roughly summarise my implementation:

- Run a pass of the model on the provided input as normal, storing logits, hidden states, and the top predicted token.
- Create noised versions of the input embeddings, with gaussian noise added at the positions of the tokens we want to corrupt.
- For every token and input pair, apply a pre forward pass hook that can replace the input to that layer with the input from the clean forward pass, then run the model on the corrupted outputs.
- For every token and layer pair, record the mean probability of the patched model outputting the same top predicted token as the unpatched model on the clean inputs.
- Display these mean probabilities.

My implementation seems to produce very similar results to the traces in the original paper. I think the small differences are due to differing randomness. Figure 1 shows my implementation of causal tracing alongside traces included in the original preprint.

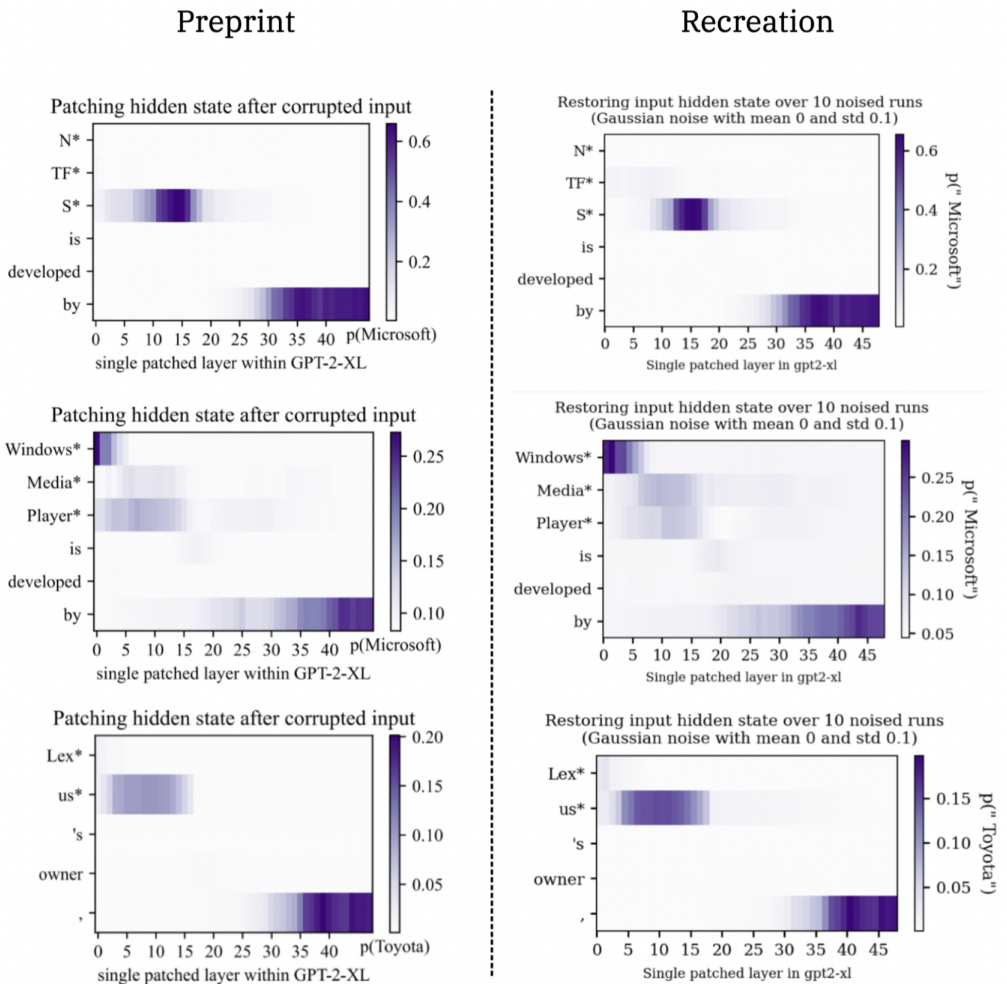


Figure 1: Side by side comparison of my causal traces with causal traces in the original preprint.

2 Note on Observations

Quick disclaimer: this report is presented and written (extremely slightly) in the style of an academic paper, so I would like to explicitly point out the observations that I will be presenting in these sections are very much casual. I consider this whole project a work in progress. Observation were gathered over a few days of exploration and playing around, and I have not checked whether they apply outside of the very limited number of examples I generated by hand. Additionally, the thoughts I share on potential internal mechanisms behind these observations are just my own weakly held speculation. I plan on investigating these observations further in the future and expect that some of my current views will change in important ways.

Additional causal traces for different behaviours (+ a few rough odds and ends that I haven't included in the report) are included in the notebook.

2.1 Choosing a Noise Level

Whilst examining behaviour under causal tracing, it became apparent that tracing results can be quite sensitive to the magnitude of the noise used to corrupt the input embeddings.

The authors of the original paper state that they picked a noise level based on the magnitude of the input embeddings. However, based on what I have seen it still seems like the best noise level can vary quite significantly from prompt to prompt, especially if the model is very confident of a particular

prediction when run on the clean input. Figure 4 in the appendix contains some examples of causal traces at varying noise levels.

I considered scaling the traces such that the white point corresponded to the probability of outputting the original token when the model was given the corrupted input and remained unpatched, and the black point corresponded to the probability of outputting the top token when given a clean input. This scaling helps with displaying causal traces as the probability range changes. However, I think that it can also result in misleading traces. For example, this scaling can hide areas where patching strongly lowers the probability of generating the correct output, and can cause outputs that are extremely noisy to look quite clean and sharp.

To avoid these issues I opted to default to displaying traces with the raw mean output probabilities. In the accompanying notebook file the majority of traces show a range of noise levels. In the future I would like to try dealing with this by using a high noise level but displaying probabilities on a log scale. Hopefully that would allow broad strokes behaviour to be seen whilst not discarding information about 'smaller magnitude' behaviour, such as information being passed to intermediate tokens before the final one.

3 Tracing Hallucinations

Language models have a tendency to "hallucinate" information, especially if they are faced with a prompt that presupposes false information, such as the existence of non-existent subjects. [2] I was interested to see how causal traces of hallucinated facts compared to roughly equivalent actual facts. From the small set of examples I generated it seems like hallucinating and 'factual recalling' have quite similar causal traces. Two examples are shown in Figure 2.

To me, this suggests generation of hallucinations and factual recall might share the same underlying process. If we do for the moment grant that this is the case, then I think it leads to some interesting thoughts. If this process can recall / generate facts or associations about completely new non-existent subjects then it seems like the mechanism being used for factual recall can essentially be queried at any point in subject/question space, in other words, that it essentially has something like a continuous input domain. Perhaps we could think of this generation/recall function as being "anchored" by the facts and associations it has managed to learn, but which can be queried at any point. In any case, I have some feeling that this process is more like generation / decompression / association than what we would usually think of as recall.

Given that hallucination and factual recall did essentially use the same process, this would at least suggest to me that being able to 'recall' a fact doesn't necessitate that there has to be some specific parameters whose 'purpose' is to store that information (since it doesn't seem like the network ought to have any dedicated "Mark Marsh" parameters).[1]

The original causal trace preprint goes quite hard on the whole 'locating facts' angle, and although I don't think this is necessarily an inaccurate description of what they do, I do wonder if it might be importantly incomplete. Perhaps what they really might be locating is something like associations in general, a much larger category, and one which would also include pesky non-factual associations.

[1] I wonder if this kind of thing might be related to polysemanticity?

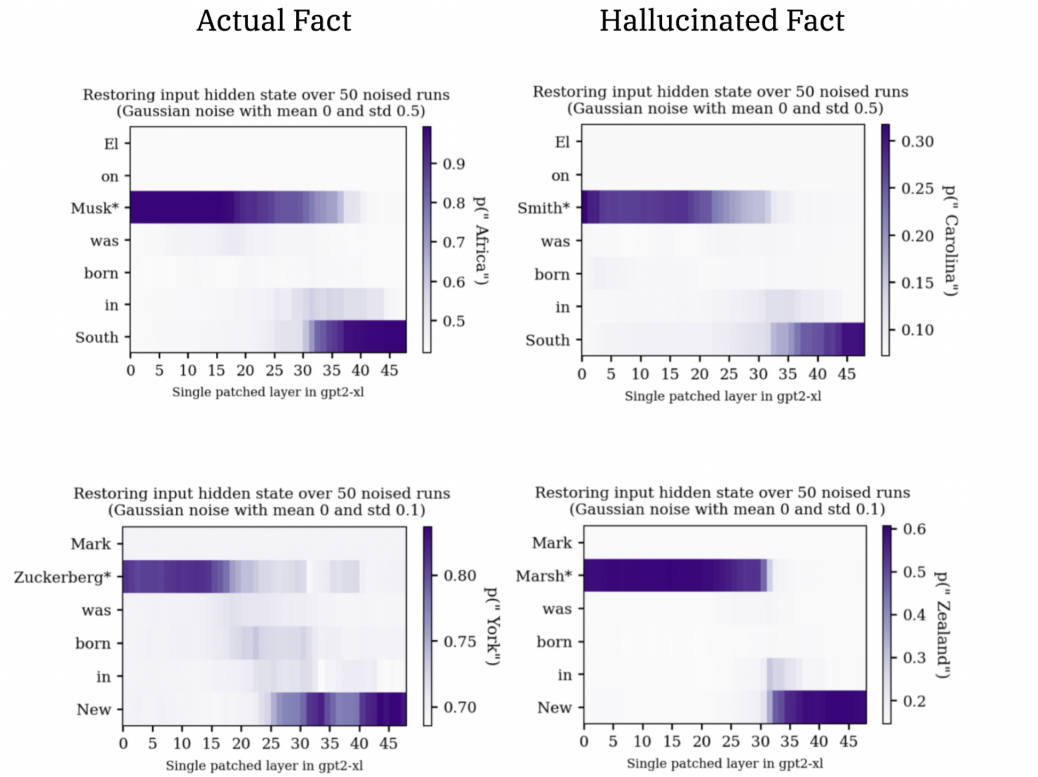
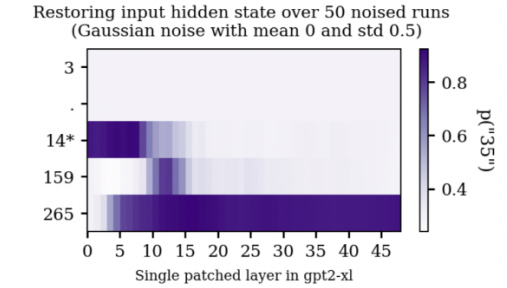
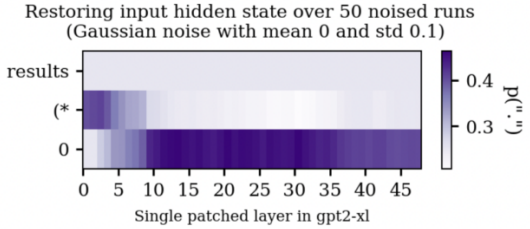
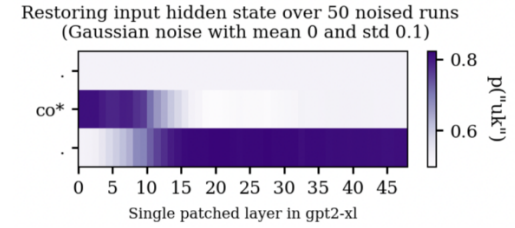
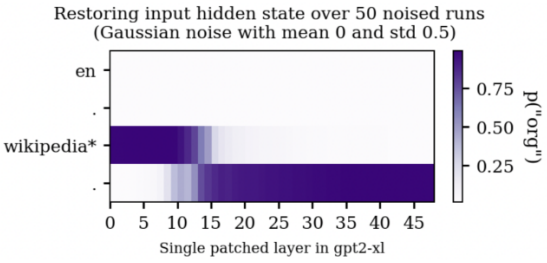


Figure 2: Causal traces of hallucinated facts and actual facts for similar prompts. As far as I know there are no especially prominent "Elon Smith"s or "Mark Marsh"s, nor do these names have associations with their supposed birthplaces.

4 Information Complexity and the Layer of Information Retrieval

I think that there might be a relationship between the layer at which the last token retrieves relevant information from previous tokens and the complexity of the information being retrieved. Specifically, I wonder if earlier layers might contain simpler information and later layers might contain more 'complex' information - like semantic information and associations. My main reason for suspecting this is that it seems like when token predictions require an understanding of the semantic properties of previous tokens, the attention of the last token seems to grab this information from the previous tokens at much later layers compared to prompts where next token prediction does not require semantic (or other complex) information about the previous tokens. Figure 3 shows some (mildly cherry-picked) examples of this kind of pattern. For more examples see the notebook.

Last Token Start Layer (Rough)	Task	Causal Trace
2 - 7	Pi Recall *“3.14159265”*	
5 - 10	Non Plain English Cached Phrase *“About X results (0.X seconds)”*	
7-12	Non Plain English Cached Phrase *“.co.uk”*	
10 - 15	Non Plain English Cached Phrase *“en.wikipedia.org”*	

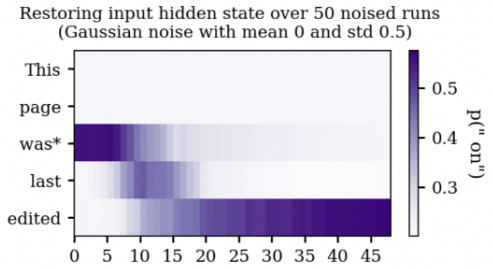
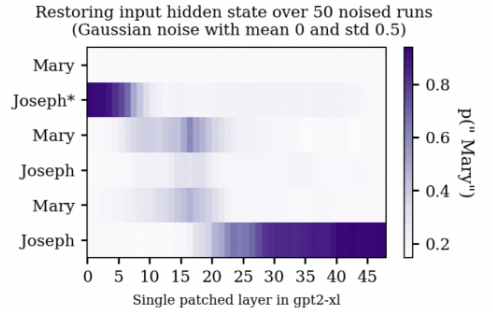
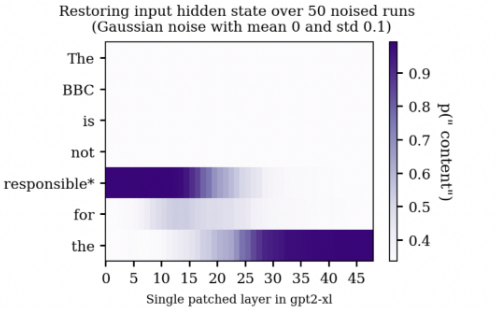
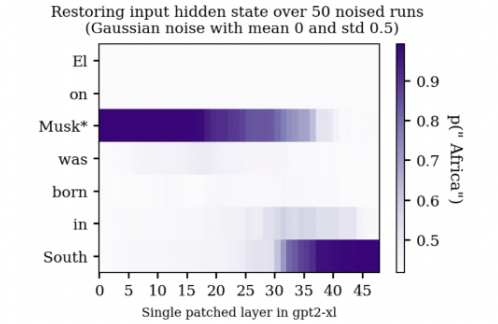
Last Token Start Layer (Rough)	Task	Causal Trace
10 - 20	Plain English Cached Phrase *"This page was last edited on"*	
15 - 25	Positional Pattern *"Mary Joseph Mary Joseph Mary Joseph Mary Joseph Mary"*	
15 - 25	Plain English Cached Phrase *"The BBC is not responsible for the content"*	
25 - 35	Semantically Dependent Fact / Association *"Elon Musk was born in South Africa"*	

Figure 3: Causal traces for a wide variety of different prompts, ordered by roughly at which layer the final token transitions to contain most of the relevant information which was modified (either directly or indirectly) by the noising process.

Figure 3 shows how the layer at which the final token tends to contain most/all of the relevant information downstream of the corrupted input changes depending on the prompt, and that the central layer of "information hand over" seems like it might vary based on the kind / complexity of information that seems relevant for next token prediction.

A possible speculative explanation for the could be that each layer helps build up some kind of internal semantic representation of the current token in the context of previous tokens? Later layers would then contain a more semantically and contextually rich representation of the input, and earlier layers might deal with simpler information (like global base rates of X following Y for example). Then the final token attention can utilise either depending on what is relevant for next token prediction.

I could try to test this with a prompt that requires retrieving simple information from one token and 'complex' semantic information from another - and then seeing what happens when one is noised and not the other. My current thinking would predict that noising the simple token would lead to a causal trace in which the noised information is grabbed by the last token earlier than if the complex token was noised.

Another potential experiment could be testing whether the salience of cached phrases in training data influences the layer at which the last token contains basically all the relevant information downstream of the corrupted input. I wonder if exact phrases which have a higher number of google search results tend to have causal traces where the last token streak starts earlier.

5 Potential Next Steps

- It seems like, for some simple tasks, information might be passed up the residuals on the last token such that the information remains basically unchanged over many layers. This seems like it would be interesting to analyse in more detail.
- Try and figure out some neat way of testing the 'later layer' = 'semantically + contextually richer' hypothesis.
- Try out different ways to corrupt the input in a more natural or principled way. e.g nearest k neighbour replacement, or smooth transition to nearest kth neighbour, or changing embedding magnitudes.
- Try different distance measures - instead of top prediction probability, what does KL divergence look like?
- Cleaning up some of the code, particularly the plotting functions.
- Implementing causal trace over MLPs and attention.
- Figure out some principled way to deal with setting the noise level.
- Perhaps I could make a dataset of different types of tasks and look for patterns in that. (I think there might be a fairly bright line between prompts that require semantic understanding of noised tokens and prompts which do not).
- Would be interesting to test on other models.
- Perhaps I should upgrade from colab pro to something which I can more easily run in the background.

Acknowledgments

Big thanks to all of the MLSS team for pulling off such a great program. :)

References

[1] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *arXiv preprint arXiv:2202.05262*, 2022.

[2] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation, 2022.

A Appendix

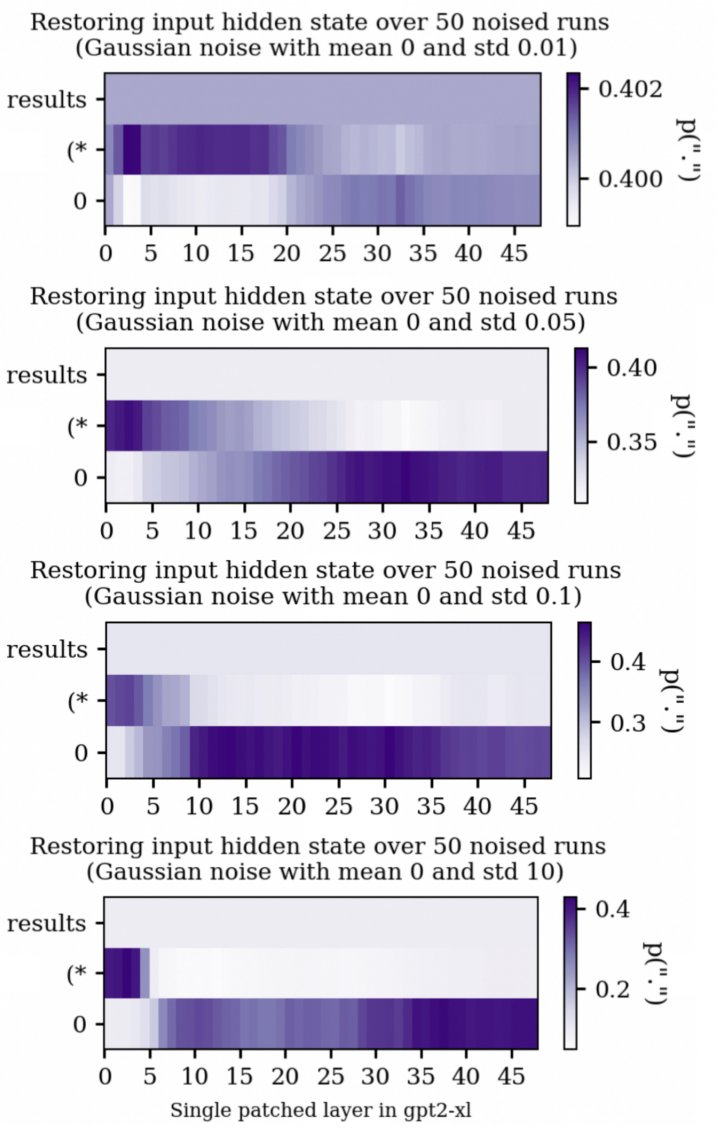


Figure 4: Example causal traces at different noise levels.