Hopping Too Late: Exploring the Limitations of Large Language Models on Multi-Hop Queries

Eden Biran[1] **Daniela Gottesman**[1] **Sohee Yang**[2] **Mor Geva**[1] **Amir Globerson**[1,3]

[1]Tel Aviv University [2]UCL [3]Google Research

edenbiran@mail.tau.ac.il

Abstract

Large language models (LLMs) can solve complex multi-step problems, but little is known about how these computations are implemented internally. Motivated by this, we study how LLMs answer multi-hop queries such as *"The spouse of the performer of Imagine is"*. These queries require two information extraction steps: a latent one for resolving the first hop (*"the performer of Imagine"*) into the bridge entity (John Lennon), and another for resolving the second hop (*"the spouse of John Lennon"*) into the target entity (Yoko Ono). Understanding how the latent step is computed internally is key to understanding the overall computation. By carefully analyzing the internal computations of transformer-based LLMs, we discover that the bridge entity is resolved in the early layers of the model. Then, *only* after this resolution, the two-hop query is solved in the later layers. Because the second hop commences in later layers, there could be cases where these layers no longer encode the necessary knowledge for correctly predicting the answer. Motivated by this, we propose a novel "back-patching" analysis method whereby a hidden representation from a later layer is patched back to an earlier layer. We find that in up to 66% of previously incorrect cases there exists a back-patch that results in the correct generation of the answer, showing that the later layers indeed sometimes lack the needed functionality. Overall, our methods and findings open further opportunities for understanding and improving latent reasoning in transformer-based LLMs.

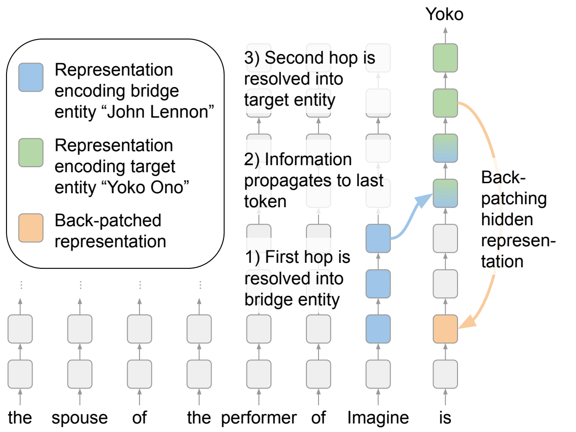


Figure 1: An illustration of our findings: we observe evidence of latent reasoning in two-hop queries where 1) During the early layers the first hop is resolved and the source entity Imagine now encodes the bridge entity John Lennon 2) During the middle layers critical information propagates to the last position 3) During the later layers the second hop is resolved and the last token now encodes the target entity Yoko Ono. We additionally illustrate back-patching: patching a hidden representation from a later layer back into an earlier layer in order to fix cases where the pathway fails.

1 Introduction

Despite groundbreaking performance in a multitude of tasks, large language models (LLMs) still struggle with complex knowledge queries (Press et al., 2023; Dziri et al., 2023). For example, LLMs often err when asked to complete multi-hop queries such as *"The spouse of the performer of Imagine is"*. Answering such queries requires composition and reasoning skills, which have been the focus of many recent works (Hou et al., 2023; Brinkmann et al., 2024; Yang et al., 2024; Li et al., 2024a; Wang et al., 2024; Petty et al., 2024). Besides the general interest in compositional skills, the investigation of latent multi-hop abilities of LLMs would also have significant implications for areas such as generalization (Lake and Baroni, 2018; Onoe et al., 2023) and model editing (Zhong et al., 2023; Cohen et al., 2024).

To strengthen the latent reasoning ability of LLMs, it is essential to first understand the internal mechanism of how LLMs successfully complete two-hop queries using latent multi-hop reasoning. While there have been several works that investigate such a mechanism, a precise latent reasoning pathway has only been found in relatively

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 14113–14130
November 12-16, 2024 ©2024 Association for Computational Linguistics

small language models trained on synthetic datasets (Brinkmann et al., 2024; Li et al., 2024a; Wang et al., 2024) and has not been thoroughly investigated in large pretrained models.

Throughout this work we consider and compare two complementing settings, one where the model correctly answers the two-hop query and the other when it does not. To this end, in §2, we start by creating and publishing a dataset containing 82,020 two-hop queries based on data from Wikidata (Vrandečić and Krötzsch, 2014).

Our approach to analyzing multi-hop reasoning begins with the hypothesis that it involves activating the same "knowledge-extraction module" twice: once for the first hop (*"the performer of Imagine is John Lennon"*) and once for the second (*"the spouse of John Lennon is Yoko Ono"*). The key question we ask in this work is where are these two procedures implemented in the LLM. A natural way to answer this question is to seek the first point in the network where the outputs of these two steps appear. The typical approach to this would employ vocabulary projections (nostalgebraist, 2020; Geva et al., 2021; Sakarvadia et al., 2023; Yang et al., 2024; Li et al., 2024a). However, we instead opt for a more recent method named Patchscopes (Ghandeharioun et al., 2024) which has clear advantages over the commonly used vocabulary projection.

Using this approach, in §3 we find that the outcome of the first hop is clearly evident in the hidden representation of the end-token of the first hop. We further find that resolving the first hop happens during the early layers, as depicted in Figure 1 (stage 1). Given the localization of the first hop, in §4 we then ask where the second hop is resolved. We find that it typically appears in the last token of the prompt *only* in layers after the resolution of the first hop, as further illustrated in Figure 1 (stage 3).

These observations suggest a mechanism where the first hop is resolved with the answer, that then propagates to the last token to resolve the second hop (as depicted by stage 2 in Figure 1). We explore this propagation in §5. Moreover, they suggest why this process may fail — If the two resolutions are done by different groups of layers, it must be that these two groups are both able to perform knowledge extraction, be it via their attention or MLP sublayers. However, since transformers have a limited amount of layers, and their functionality is different (Meng et al., 2022; Geva et al., 2023), it is quite likely that the layer at which the second hop is performed will not have the desired functionality. To verify this hypothesis, in §6, we propose an analysis method named *back-patching*. The crux of the method involves taking a hidden representation from a later layer, patching it back into the same position of the same prompt in an earlier layer, and then continuing the forward pass. In a way, this allows the model more layers to finish the computation without the need for additional parameters or training (Petty et al., 2024). Unlike previous methods (Sakarvadia et al., 2023; Li et al., 2024a), back-patching makes no assumption on the structure of the prompt and does not require any knowledge of the participating entities. Testing back-patching on queries initially predicted incorrectly, we find that up to 66% of incorrect cases have the potential to be correctly predicted using back-patching when choosing the optimal source and target layers.

Overall, our contributions can be summarized as follows:

- We provide a novel dataset of two-hop queries, which can serve to systematically study this important setting.
- We employ Patchscopes to inspect entities encoded in hidden representations while processing two-hop queries.
- We identify a sequential latent reasoning pathway in LLMs, where the first hop query is initially resolved into the bridge entity which is then used to answer the second hop.
- We propose an analysis method named back-patching, that verifies our results and could additionally be used to improve the performance of LLMs on multi-hop question answering.

We release our code and dataset at `https://github.com/edenbiran/HoppingTooLate`.

2 Experimental Setup

2.1 Two-Hop Queries

We consider facts denoted by a triplet (e, r, e') where e is a source entity (e.g., John Lennon), r is a relation (e.g., Spouse) and e' is a target entity (e.g., Yoko Ono). These facts can then be converted to factual statements (e.g., *"The spouse of John Lennon is Yoko Ono"*) and also into queries by omitting e' (e.g., *"The spouse of John Lennon is "*). We then prompt models with these queries in order to test their knowledge and reasoning.

We create two-hop queries by composing two facts where one's target entity is the other's source entity. Namely, $((e_1, r_1, e_2)$ and $(e_2, r_2, e_3))$. We refer to e_1 as the source entity, e_2 as the bridge entity, and e_3 as the target entity. For example, the two-hop query *"The spouse of the performer of Imagine is"* can be decomposed into (Imagine, Performer, John Lennon) and (John Lennon, Spouse, Yoko Ono).

We further define two tokens of specific interest: the last token of e_1[1] (e.g., *"The spouse of the performer of Imagine"*) and the last token of the whole two-hop prompt (e.g., *"The spouse of the performer of Imagine is"*), which we denote as t_1 and t_2, respectively.

2.2 Dataset

We create a dataset containing 82,020 two-hop queries based on data from Wikidata (Vrandečić and Krötzsch, 2014). First, we sample entities with a maximal amount of Wikidata statements from a set of predefined Wikidata entity types. These entities will act as e_2. Second, we use a predefined set of relations in order to construct the two triplets (e_1, r_1, e_2) and (e_2, r_2, e_3). Finally we convert both triplets into a single natural language phrase using manually crafted templates per relation. We refer to our codebase for further details including entity types and relations[2].

We next attempt to filter out cases where no latent reasoning is performed. Given a two hop query $((e_1, r_1, e_2), (e_2, r_2, e_3))$ we test two prompts constructed to detect and filter out cases where the model performs reasoning shortcuts (Xu et al., 2022; Wang et al., 2023a; Ju et al., 2024). The first prompt is the query $(("", r_1, e_2), (e_2, r_2, e_3))$ (i.e., the query without e_1), aimed at filtering out cases where the model predicts generally popular entities. The second prompt is $((e_1, "", e_2), (e_2, r_2, e_3))$ (i.e., the query without r_1), aimed at filtering out cases where the model predicts correctly due to a high correlation between e_1 and e_3. For example, given the two-hop query *"The spouse of the performer of Imagine is"* we filter out cases where the model predicts Yoko Ono for either *"The spouse of the performer is"* or *"The spouse of Imagine is"*. We perform this filtering for each model using greedy decoding creating a per model subset of the dataset. Table 1 displays the amount of examples left after this filtering process.

[1]This token is also the last token of the first hop.
[2]All data was retrieved during April 2024.

Model	Post Filtering	Cases Tested Correct	Cases Tested Incorrect
LLaMA 2 7B	70,625	388	351
LLaMA 2 13B	70,972	554	413
LLaMA 3 8B	71,569	379	371
LLaMA 3 70B	70,334	656	595
Pythia 6.9B	73,058	173	202
Pythia 12B	74,056	172	372

Table 1: The amount of examples post shortcut filtering and the final amount used throughout all experiments.

We are specifically interested in understanding the differences between cases where the model completes the two-hop query correctly and incorrectly. Therefore we generate two dataset subsets per model for these cases accordingly.

The first subset is made up of cases where the model correctly answers both the two-hop query and the first hop. For example, given the two-hop query *"The spouse of the performer of Imagine is"*, we verify that the model correctly predicts the answer Yoko Ono when prompted with the two-hop query and additionally correctly predicts John Lennon when prompted with the first hop *"The performer of Imagine is"*. We use the latter filter to make sure that the model indeed "knows" the answer to the first hop. We then randomly sample 100 examples of each bridge entity type, in order to create a more balanced subset (if less than 100 examples of a specific type exist we take them all).

The second subset includes cases where the model correctly answers both the first and second hop in isolation, but fails to answer the full two-hop query. As an example for the filtering procedure, given the same query above, we verify that the model correctly predicts John Lennon for the first hop *"The performer of Imagine is"* and Yoko Ono for the second hop *"The spouse of John Lennon is"*, but fails to predict Yoko Ono for the two-hop query *"The spouse of the performer of Imagine is"*. We focus on these cases, because we are interested in understanding why answering the two-hop query fails despite the model "knowing" the two separate facts. As we show later, this failure is likely due to the first fact being resolved too late. In a similar fashion to the first setting, we then randomly sample 50 examples of each bridge entity type (in order to keep the correct and incorrect subsets of approximately the same size).

Table 1 reports the amount of examples used in experiments by model and setting.

2.3 Models

We analyze the following models: LLaMA 2 7B and 13B (Touvron et al., 2023), LLaMA 3 8B and 70B (Meta AI, 2024), and Pythia 6.9B and 12B (Biderman et al., 2023). The 6.9B, 7B and 8B models have 32 layers, the 12B model has 36 layers, the 13B model has 40 layers, and the 70B model has 80 layers.

3 Localizing First Hop Resolution

One could imagine several possible ways for a model to solve multi-hop queries. One such way would be to effectively treat the two-hop query as a single relation, enabling the extraction of the final answer without the need to recall e_2. Despite this, if the model truly performs latent reasoning, the most intuitive way to do so would start by resolving the first hop at some point within the computation. In order to localize where this happens, we use the Patchscopes analysis method. Our results show that the model indeed resolves the first hop during the early layers, at the position of t_1.

3.1 Interpreting Hidden Representations

We use the Patchscopes framework (Ghandeharioun et al., 2024) to create a task that describes the entity encoded in a specific hidden representation. Patchscopes maps a given representation to a sentence in natural language, thus considerably extending vocabulary projections which map to a single token (nostalgebraist, 2020).

The procedure is as follows. First, given a source prompt, a source token and a source layer, the source prompt is passed through the model's forward computation and the hidden representation v of the source token at the source layer is recorded. This representation v is what we aim to probe in search for an encoded entity. Second, we pass the same prompt used by Ghandeharioun et al. (2024): "`Syria: Syria is a country in the Middle East, Leonardo DiCaprio: Leonardo DiCaprio is an American actor, Samsung: Samsung is a South Korean multinational corporation, x`" through the model, replacing the hidden representation of "`x`" with v at a specific target layer. Finally, the forward pass is continued and text is generated.

The chosen target prompt encourages the model to generate a continuation that states the semantic content (i.e., the entity name) encoded in the source hidden representation, along with a short description of it. We find this task is better fitting than other proposed probes such as vocabulary projections (nostalgebraist, 2020; Geva et al., 2022) or training linear classifiers on hidden representations (Belinkov and Glass, 2019; Belinkov, 2022). This is because exhibiting the ability to extract information from a hidden representation gives reason to believe that the same information can be extracted when answering the two-hop query. In addition Patchscopes has the clear advantage of decoding a representation into a natural language description.

Model	Subset	e_2 from t_1 Cases	e_2 from t_1 Layer	e_2 from t_2 Cases	e_2 from t_2 Layer	e_3 from t_2 Cases	e_3 from t_2 Layer
LLaMA 2 7B	Corr.	56%	8.9	41%	15.3	73%	16.2
	Incorr.	41%	9.1	36%	16.4	47%	17.5
LLaMA 2 13B	Corr.	49%	7.7	34%	18.2	71%	16.9
	Incorr.	48%	7.4	37%	17.9	33%	16.9
LLaMA 3 8B	Corr.	46%	8.9	37%	14.3	74%	13.5
	Incorr.	46%	11.9	25%	15.3	40%	17.2
LLaMA 3 70B	Corr.	63%	27.1	46%	35.2	86%	30.7
	Incorr.	57%	29.2	46%	34.5	50%	35.9
Pythia 6.9B	Corr.	75%	5.4	75%	11.4	80%	14.1
	Incorr.	78%	4.9	67%	9.2	65%	13.6
Pythia 12B	Corr.	73%	5.0	66%	12.2	77%	13.5
	Incorr.	61%	6.3	42%	12.9	52%	17.2

Table 2: The results of Patchscopes executions. The table reports the percentage of cases where a target entity was successfully decoded from a source position. The table additionally reports the mean of the layer where the target entity was first successfully decoded.

3.2 Experiment

We run Patchscopes on t_1, recording the hidden representation at each source layer and then patching it into all target layers. Following each patch we sample three generations. For each source layer we check whether one of the generations is of e_2, and if so, we consider the representation to encode the bridge entity at the source layer.

3.3 Results

Table 2 presents the percentage of cases where e_2 was observed in the hidden representation of t_1. We refer to these as cases where e_2 was successfully resolved. We note the non-trivial percentages of 41%-78% across all models and settings. Additionally, we note the general drop in resolved cases when the model fails to correctly complete the two-hop query, but that this setting still displays a surprisingly high success rate. Next, we examine the mean layer at which e_2 first appears

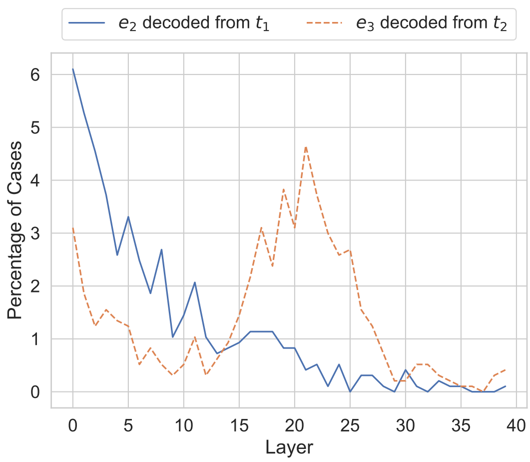


Figure 2: Percentage of cases per layer where target entities were first successfully decoded using Patchscopes. The percentages are out of all correctly answered cases for LLaMA 2 13B.

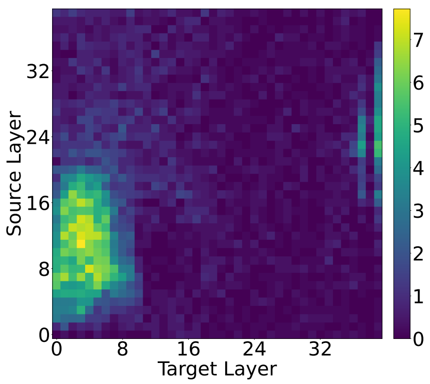


Figure 3: Heat-map of the layers where Patchscopes successfully decodes e_2 from the position of t_1. The percentages are out of all successful decodings for LLaMA 2 13B run on correctly answered cases.

in both correct and incorrect cases. We find that in the incorrect cases the resolving generally occurs at higher layers than those of the correct cases. This difference alludes to how early the resolving happens potentially playing a part in successfully answering two-hop queries. We further verify this using the back-patching analysis method in §6.

Figure 2 displays the proportion of cases per layer where e_2 is first successfully decoded from t_1 using Patchscopes. We observe that the resolving occurs almost exclusively during the early layers. Interestingly, this implies that the knowledge required for this resolution must reside in these earlier layers. Similar figures for additional models are provided in Appendix B. The plot additionally shows the curve regarding the resolving of e_3 from t_2, which is discussed in §4.

This finding can further be seen in Figure 3, which displays a heat-map of the layers where Patchscopes successfully decodes e_2 from t_1. Two hot-spots are apparent. The first corresponds to success cases with an early source layer, showing that e_2 generally appears at this position during the early layers. The second hot-spot corresponds to later source layers that display success almost exclusively when patched after the final layer, which is roughly equivalent to vocabulary projection (nostalgebraist, 2020). This is a clear advantage of Patchscopes, as despite the entity not being among the top tokens in the projection during early layers, it is in fact encoded in the hidden representation. See Appendix C for heat-maps of all models.

4 Second Hop is Resolved at Last Position

In §3, we observe that the first hop is often resolved in position t_1. Intuitively, one would expect the computation to use this result towards resolving the second hop (i.e., the final answer). In this section, we study this proposition by finding where e_3 first emerges and how it relates to the first emergence of e_2. We find that e_3 indeed emerges predominantly after e_2 at the position of t_2. Furthermore, we search for the final part of the pathway, the predicted token. By projecting the residual updates made by the attention and MLP sublayers on the hidden representation of t_2, we find that the MLP sublayers play the larger part in conjuring the predicted token.

4.1 Method

Previous works (Geva et al., 2022, 2023) have shown that the residual updates made by the MLP and attention sublayers can be interpreted by projecting them to the vocabulary using the layer norm and output embedding matrix. We use this method in order to locate specific entities in the flow of information by observing the rank of the entity in the vocabulary projection. We additionally use Patchscopes as presented in §3.

4.2 Experiment

We first employ Patchscopes on t_2, this time checking if the target entity e_3 is encoded at this position. Additionally, we project the sublayer updates performed on t_2, and check if the token with the highest probability in the projection is the first token of the generated text. If so, we conclude that the update of a particular sublayer promotes the final prediction.

Model	Subset	Attention Cases	Attention Layer	MLP Cases	MLP Layer
LLaMA 2 7B	Correct	25.5%	24.7	33.2%	25.2
	Incorrect	14.8%	23.4	28.4%	26.0
LLaMA 2 13B	Correct	28.8%	32.0	68.5%	33.1
	Incorrect	14.2%	27.0	51.5%	31.7
LLaMA 3 8B	Correct	9.7%	26.7	17.6%	27.8
	Incorrect	25.3%	27.4	11.8%	24.6
LLaMA 3 70B	Correct	21.9%	56.3	48.1%	68.0
	Incorrect	21.8%	65.7	36.1%	67.0
Pythia 6.9B	Correct	11.5%	21.8	36.4%	20.5
	Incorrect	28.2%	23.4	33.6%	21.2
Pythia 12B	Correct	27.9%	22.2	38.9%	22.4
	Incorrect	43.2%	23.6	47.0%	23.3

Table 3: Percentages and mean layers where for the first time the predicted token is the most probable token in the vocabulary projection of the sublayer update.

4.3 Results

Table 2 contains the percentage of cases e_3 was decoded from t_2, effectively resolving the second hop. As expected, these percentages are high for cases in which the model predicted correctly (71%-86%)[3] and low for cases the model predicted incorrectly (33%-65%). Additionally, it is apparent from both Table 2 and Figure 2 that the target entity is resolved by the mid-upper layers at the position of t_2. This suggests that the information required to perform the second hop exists in the parameters of these layers.

Inspecting the sublayer projections in Table 3, we find that the MLP sublayers play a larger role in promoting the predicted token compared to that of the attention sublayers. Despite this, the non negligible percentages achieved by the attention sublayers could suggest multiple pathways, in agreement with previous work (Geva et al., 2023; Merullo et al., 2024) that considered both possibilities. Looking at the mean layers at which the prediction is first promoted, one can see that these promotions occur in the upper layers, after the last token has been resolved to e_3.

5 Information Propagates to Last Token

As the model's prediction is emitted from the last position of the prompt, critical information regarding the bridge entity must propagate to this token. By using attention knockout, projection, and Patchscopes, we verify this claim. We find that in a large majority of cases relevant information can indeed be detected passing from t_1 to t_2.

[3]We attribute the gap from 100% to the approximations involved in decoding a hidden representation.

5.1 Method

In similar fashion to previous work (Wang et al., 2023b; Geva et al., 2023), we block hidden representations from attending to other representations at specific layers and test if the model's final generation changes. We perform the knockout by setting the mask of the attention from a source to a target position at a specific layer to $-\infty$, causing this attention edge to contribute nothing to the residual update performed on the source representation. We experiment with blocking a window of layers as previous work (Geva et al., 2023) shows that information propagation is spread among multiple layers.

We additionally employ Patchscopes and sublayer vocabulary projections, as described in §3 and §4 respectively.

5.2 Experiment

We run attention knockout blocking information originating at t_1 from propagating to t_2. We block a window of 7 layers in order to capture information flowing through multiple layers. If blocking a set of layers causes the model to no longer correctly predict the answer (in the incorrect setting we check for a change in the generated text), we consider these layers to contain information critical to the prediction.

We additionally project the updates made by the attention and MLP sublayers to the last token, as described in §4. In the vocabulary projection we check if e_2 is the token with the highest probability, and if so consider the update to "contain" the bridge entity.

Finally, we run Patchscopes on the last token of the prompt, as done in §3, checking if e_2 is encoded in t_2. Each one of these experiments gives a unique signal that information must have propagated from t_1 to t_2.

5.3 Results

Table 4 shows the percentage of cases where we detect critical information propagating from t_1 to t_2 using at least one of the methods stated above. We note the relatively high percentages, indicating the information indeed flows as expected. Additionally, the table shows the that the layers relevant to the propagation are mostly the middle layers.

Model	Subset	Detected	Mean Layer
LLaMA 2 7B	Correct	85.82%	12.83
	Incorrect	95.44%	8.10
LLaMA 2 13B	Correct	81.94%	15.45
	Incorrect	71.11%	16.99
LLaMA 3 8B	Correct	78.36%	10.15
	Incorrect	82.74%	7.45
LLaMA 3 70B	Correct	90.39%	29.14
	Incorrect	94.28%	20.06
Pythia 6.9B	Correct	96.53%	11.94
	Incorrect	95.04%	8.06
Pythia 12B	Correct	94.18%	11.23
	Incorrect	93.81%	9.28

Table 4: Percentages and mean layers of cases where critical information was first detected propagating from t_1 to t_2 using attention knockout, projection, or Patchscopes.

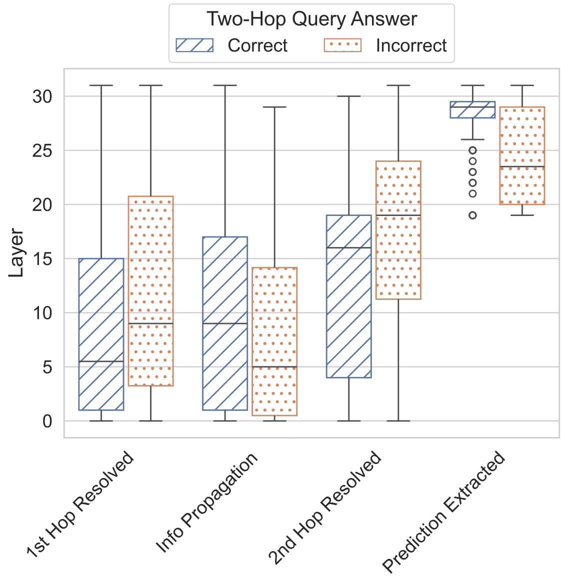


Figure 4: A comparison of the first layers of each stage in the pathway between correct and incorrect cases for LLaMA 3 8B.

Table 2 contains the percentage of cases e_2 was decoded from t_2, showing that this position indeed represents the bridge entity in 25%-75% of all cases. Comparing between the results of Patchscopes on t_2 to those on t_1, we observe a stark difference in the successful source layers. The results show that the t_2 first encodes e_2 during the mid-upper layers, while t_1 does so in the lower layers. This can further be seen in the Patchscopes heat-maps in Appendix C. This gives additional reason to believe that a sequential computation is taking place.

After describing all major pathway stages, we turn to a comparison between the correct and incorrect cases and observe the following pattern. Generally, in incorrect cases, the entities seem to be resolved later while the information seems to propagate and be extracted earlier. Figure 4 displays this pattern in LLaMA 3 8B, plotting the first observed layers of each stage (we refer to Appendix D for plots of additional models). This gives evidence of the importance of how early the first hop is resolved.

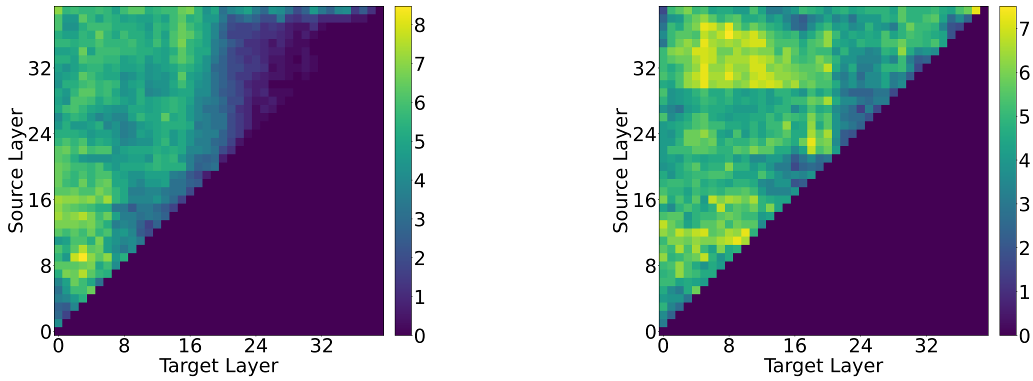
6 Back-patching Improves Two-Hop Performance

Our results consistently display a sequential nature of computation, where the first hop is resolved into the bridge entity by the lower layers, which is then used to answer the second hop in the upper layers. These results suggest that when answering two-hop queries fail, it could be because the first-hop is resolved at layers that no longer contain the information needed to resolve the second hop. In order to verify this hypothesis, we test the following intriguing conjecture. We know that the earlier layers certainly contain knowledge, since they are regularly used to resolve the first hop. Then, we ask what will happen if we take a hidden representation from a later layer and "feed" it into these earlier layers. Will the correct information now be extracted? As our results next show, this is often the case. This "back-patching" analysis thus provides strong support that the success in two hop queries require the later layers to have the same functionality as earlier ones.

Overall, this could point to an inherent limitation of the transformer architecture, where the model "runs out" of layers, propagates and extracts information before it is fully resolved, or is forced to use only the upper layers for knowledge extraction.

6.1 Method

Following our intuition above, we perform the following back-patching experiment. Given a prompt, we first record a hidden representation v from a source layer ℓ_s. Then, while rerunning the same prompt, we patch v into the same position at a target layer ℓ_t such that $\ell_t < \ell_s$. The forward pass then continues and text is generated using greedy decoding. This effectively feeds ℓ_t with information encoded in v that originated in ℓ_s.



(a) Back-patching t_1

(b) Back-patching t_2

Figure 5: Heat-map of the layers where back-patching succeeds. The percentages are out of all successful back-patching instances for LLaMA 2 13B.

Model	Subset	t_1	t_2
LLaMA 2 7B	Correct	100%	100%
	Incorrect	41.02%	42.45%
LLaMA 2 13B	Correct	100%	100%
	Incorrect	32.44%	36.07%
LLaMA 3 8B	Correct	100%	100%
	Incorrect	38.81%	47.16%
LLaMA 3 70B	Correct	100%	100%
	Incorrect	57.31%	57.81%
Pythia 6.9B	Correct	100%	100%
	Incorrect	66.33%	56.43%
Pythia 12B	Correct	100%	100%
	Incorrect	63.17%	61.82%

Table 5: Back-patching success rates by setting and back-patched token.

6.2 Experiment

We perform the back-patching experiment twice, once on the token t_1 and once on the token t_2. In order to quantify the potential of back-patching, we check whether one of the source-target pairs resulted in the generation of the correct answer. We then perform an analysis of the successful source and target layers.

6.3 Results

Table 5 presents the back-patching results. First we note the high success rates, correctly answering between 32%-66% of incorrect cases while keeping the correct cases at 100% across models and token positions. This means that in many cases rerunning specific parts of the computation is enough in order to successfully extract the target information. On a more practical note, this also shows that significant gains can be made by correctly choosing the source and target layers, which we leave for future work.

One can additionally observe that in LLaMA models back-patching t_2 achieves a higher success rate than back-patching t_1, which may point at the stages related to the last position of the prompt as being the more dominant failure point in answering two-hop queries. Moreover, this higher success rate is achieved despite the fact that back-patching t_2 does not assume any information about the structure of the prompt. We believe that this displays an additional strength of the method.

Figure 5a displays the heat-map of the layers where back-patching t_1 succeeds for LLaMA 2 13B (see Appendix E for additional models). We first note the higher success rate when the source layers are the model's lower layers. These lower layers correlate to the resolving of the first hop, therefore back-patching them allows the model a better chance at solving the first hop. Second, we note the non existent success rate of the higher target layers. Unsurprisingly, in this case the model does not gain enough time to correctly complete the pathway, resulting in a low success rate.

Figure 5b displays the heat-map of the layers where back-patching t_2 succeeds for LLaMA 2 13B. Unlike when back-patching t_1, in this case the higher success rate can be found in the the upper source layers. These layers correlate to the resolving of the second hop at the last token position. Hence, back-patching a hidden representation taken from a later layer allows the model a better chance at solving the second hop. Furthermore, we note the higher success rate in lower target layers. This allows the model to gain time to resolve the second hop. This also gives the model access to parametric knowledge that might only be encoded in the lower layers, allowing the model to fully utilize its whole knowledge-base.

7 Related Work

Tracing and locating factual knowledge has been the focus of many recent works aiming to undercover and control the way transformers make predictions. Several of these works are motivated by knowledge editing (De Cao et al., 2021; Mitchell et al., 2022; Meng et al., 2022; Zhang et al., 2024) while others focus on understanding the inner workings of transformers (Dai et al., 2022; Geva et al., 2022, 2023). Our work follows and expands on this second group by extending them to the more complex case of two-hop queries, which require multiple factual knowledge extractions and compositions.

The question whether models can implicitly reason over stored parametric knowledge has also gained recent popularity. One avenue of related research includes circuit discovery (Nanda et al., 2023; Wang et al., 2023b; Conmy et al., 2023; Brinkmann et al., 2024), but these works focus mainly on synthetic tasks. More closely related to this paper, recent works (Sakarvadia et al., 2023; Yang et al., 2024; Li et al., 2024a) have found existential evidence of latent reasoning in multi-hop queries, albeit without accounting for the flow of information throughout layers and positions. In a concurrent work by Wang et al. (2024) a multi-hop reasoning circuit similar to ours was discovered in a small transformer language model trained on synthetic data until grokking occurs. Our work finds further proof of such latent reasoning in a more practical setting that includes large language models, real world knowledge and a regular pretraining setup. We additionally use different mechanistic interpretability methods and account for the flow of information during the forward pass.

Even without fully identifying the mechanism behind latent reasoning, several works have tried to address model's shortcomings in this area. One such work by Sakarvadia et al. (2023) assumes prior knowledge of the bridge entity, whose hidden representation is then strategically injected directly into the computation. Likewise, recent work by Li et al. (2024a) also assumes knowledge of the bridge entity and then employs knowledge editing techniques to improve performance, which is known to have potentially disruptive effects on models (Zhang et al., 2024; Hsueh et al., 2024; Li et al., 2024b). Both these previously proposed methods have drawbacks, which a potential method based on back-patching would not share.

8 Conclusion

Our work uses mechanistic tools to study the latent reasoning abilities of LLMs on two hop queries. We observe strong evidence of a sequential latent reasoning pathway in which the second hop is answered in the mid-upper layers only after the first hop is answered by the lower layers. We find that this sequential nature could point to a possible limitation of transformers, where the second hop must be answered using only the knowledge encoded in the upper layers. To further validate this, we introduce and evaluate an analysis method named back-patching, based on extracting hidden representations from later layers and patching them back into earlier layers. Finally, we show that back-patching significantly improves performance on previously incorrect queries. Overall, our methods and findings open opportunities for understanding and improving latent reasoning in LLMs.

Limitations

Several of our experiments rely on mechanistic methods that decode hidden representations and residual updates in various ways. While these methods have been widely used in many recent works (Geva et al., 2022, 2023; Dar et al., 2023; Yang et al., 2024; Li et al., 2024a), they can only be seen as an approximation. However, using multiple different techniques (such as both Patchscopes and sublayer projections) helps alleviate this concern. Additionally, our experiments are generally designed to give positive signals that result in lower bound results, hence we believe that these approximations do not undermine our findings.

In this work we study one prominent latent reasoning pathway, although others most likely exist (McGrath et al., 2023; Yang et al., 2024). Additionally, we do not account for all possible parts of the discovered pathway (e.g., how the relations come in to play). Despite this, by focusing on the most relevant components we find sufficient proof for our main finding regarding the sequential nature of the computation.

Our work examines only two-hop queries, although we expect queries with three or more hops and even additional unrelated reasoning tasks to involve similar mechanisms to those we observed. This is due to previous (Wang et al., 2023b; Brinkmann et al., 2024) and concurrent (Wang et al., 2024) works in different (if sometimes small or synthetic) settings observing pathways that can be broken down into concrete computational steps that match intuitive reasoning steps. These results resemble our finding that two-hop questions are broken down and answered sequentially by LLMs. We leave such expansion to additional settings for future work.

Despite the promising results of back-patching, our current proposed method is not a practical inference method, as only a subset of back-patches generate the correct answer. However, if one could efficiently select a subset of source and target layers a priori, this could make for a viable method. Although we believe that if achieving the best performance in multi-hop question answering is the sole goal, methods such as chain of thought prompting (Wei et al., 2022; Press et al., 2023) would be far more effective.

Acknowledgements

This work was supported by the Tel Aviv University Center for AI and Data Science (TAD) and the Israeli Science Foundation.

References

David Bau. 2024. Baukit.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Jannik Brinkmann, Abhay Sheshadri, Victor Levoso, Paul Swoboda, and Christian Bartelt. 2024. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 4082–4102, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, pages 16318–16352. Curran Associates, Inc.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2023. Analyzing transformers in embedding space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16124–16170, Toronto, Canada. 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.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang (Lorraine) Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems*, volume 36, pages 70293–70332. Curran Associates, Inc.

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

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *Forty-first International Conference on Machine Learning*.

Yifan Hou, Jiaoda Li, Yu Fei, Alessandro Stolfo, Wangchunshu Zhou, Guangtao Zeng, Antoine Bosselut, and Mrinmaya Sachan. 2023. Towards a mechanistic interpretation of multi-step reasoning capabilities of language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4919, Singapore. Association for Computational Linguistics.

Cheng-Hsun Hsueh, Paul Kuo-Ming Huang, Tzu-Han Lin, Che-Wei Liao, Hung-Chieh Fang, Chao-Wei Huang, and Yun-Nung Chen. 2024. Editing the mind of giants: An in-depth exploration of pitfalls of knowledge editing in large language models. *arXiv preprint arXiv:2406.01436*.

Tianjie Ju, Yijin Chen, Xinwei Yuan, Zhuosheng Zhang, Wei Du, Yubin Zheng, and Gongshen Liu. 2024. Investigating multi-hop factual shortcuts in knowledge editing of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8987–9001, Bangkok, Thailand. Association for Computational Linguistics.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

Zhaoyi Li, Gangwei Jiang, Hong Xie, Linqi Song, Defu Lian, and Ying Wei. 2024a. Understanding and patching compositional reasoning in LLMs. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9668–9688, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2024b. Unveiling the pitfalls of knowledge editing for large language models. In *The Twelfth International Conference on Learning Representations*.

Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. 2023. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372. Curran Associates, Inc.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Language models implement simple Word2Vec-style vector arithmetic. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5030–5047, Mexico City, Mexico. Association for Computational Linguistics.

Meta AI. 2024. Introducing meta llama 3: The most capable openly available llm to date.

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

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*.

nostalgebraist. 2020. interpreting gpt: the logit lens.

Yasumasa Onoe, Michael Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. 2023. Can LMs learn new entities from descriptions? challenges in propagating injected knowledge. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5469–5485, Toronto, Canada. Association for Computational Linguistics.

Jackson Petty, Sjoerd Steenkiste, Ishita Dasgupta, Fei Sha, Dan Garrette, and Tal Linzen. 2024. The impact of depth on compositional generalization in transformer language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7239–7252, Mexico City, Mexico. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Mansi Sakarvadia, Aswathy Ajith, Arham Khan, Daniel Grzenda, Nathaniel Hudson, André Bauer, Kyle Chard, and Ian Foster. 2023. Memory injections: Correcting multi-hop reasoning failures during inference in transformer-based language models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 342–356, Singapore. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. 2024. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. *arXiv preprint arXiv:2405.15071*.

Fei Wang, Wenjie Mo, Yiwei Wang, Wenxuan Zhou, and Muhao Chen. 2023a. A causal view of entity bias in (large) language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15173–15184, Singapore. Association for Computational Linguistics.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023b. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Nan Xu, Fei Wang, Bangzheng Li, Mingtao Dong, and Muhao Chen. 2022. Does your model classify entities reasonably? diagnosing and mitigating spurious correlations in entity typing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8642–8658, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024. Do large language models latently perform multi-hop reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10210–10229, Bangkok, Thailand. Association for Computational Linguistics.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Zexuan Zhong, Zhengxuan Wu, Christopher Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15686–15702, Singapore. Association for Computational Linguistics.

A Technical Details

We use the Baukit library (Bau, 2024) for several of the experiments. Each experiment was run on 1-4 A100 or H100 GPUs and lasted at most 24 hours. The models were retrieved and run using the HuggingFace Transformers library (Wolf et al., 2020). When generating with sampling we use the default parameters from the Transformers library. We use half precision for the 70B model and full precision for all other models.

B Patchscopes First Decoded Layers

Figure 6 depicts the percentage of cases per layer where target entities were first successfully decoded using Patchscopes for all models (as described in §3).

C Patchscopes Heat-maps

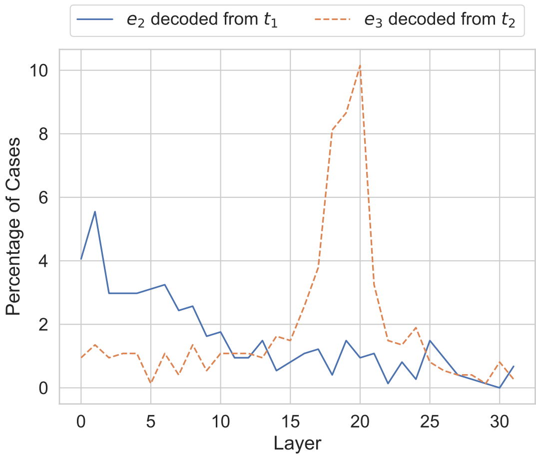
Figures 7 and 8 depict the heat-maps of Patchscope success cases as introduced in §3. We report Patchscope experiments for decoding e_2 from t_1, e_2 from t_2, and e_3 from t_2 for all models.

D First Observed Layers of Pathway Stages

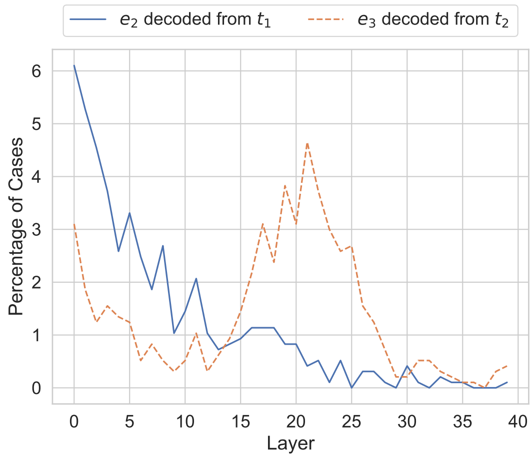
Figure 9 depicts a comparison of the first layers of each stage in the observed pathway between correct and incorrect cases (as shown in §5) for all models.

E Back-patching Heat-maps

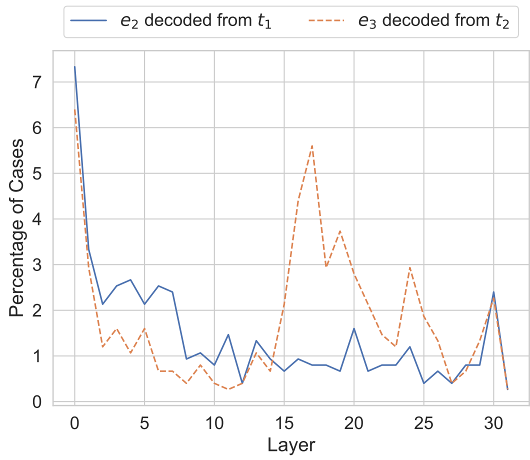
Figures 10 and 11 depict the back-patching heap-maps of successful source and target layers (as shown in §6) for all models.



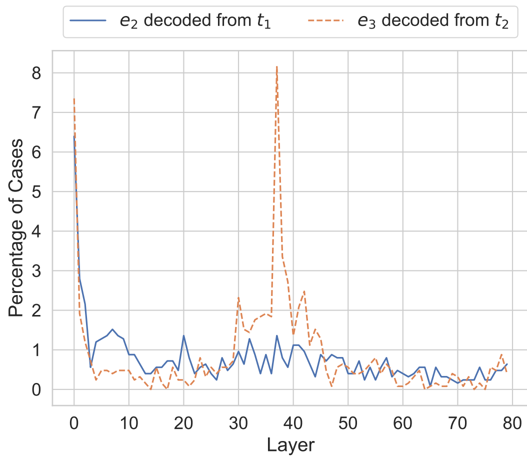
(a) LLaMA 2 7B



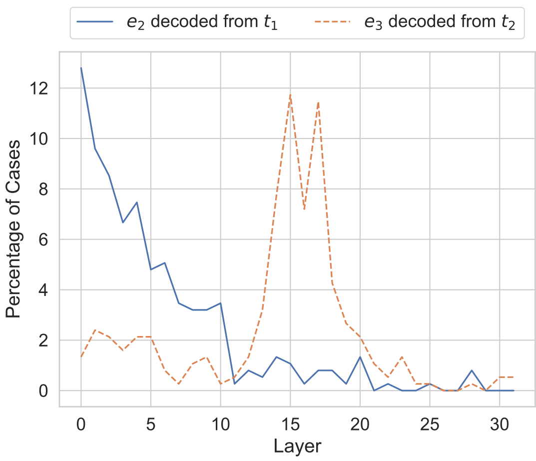
(b) LLaMA 2 13B



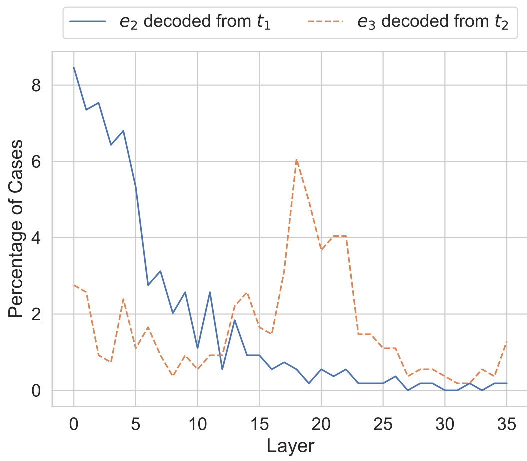
(c) LLaMA 3 8B



(d) LLaMA 3 70B

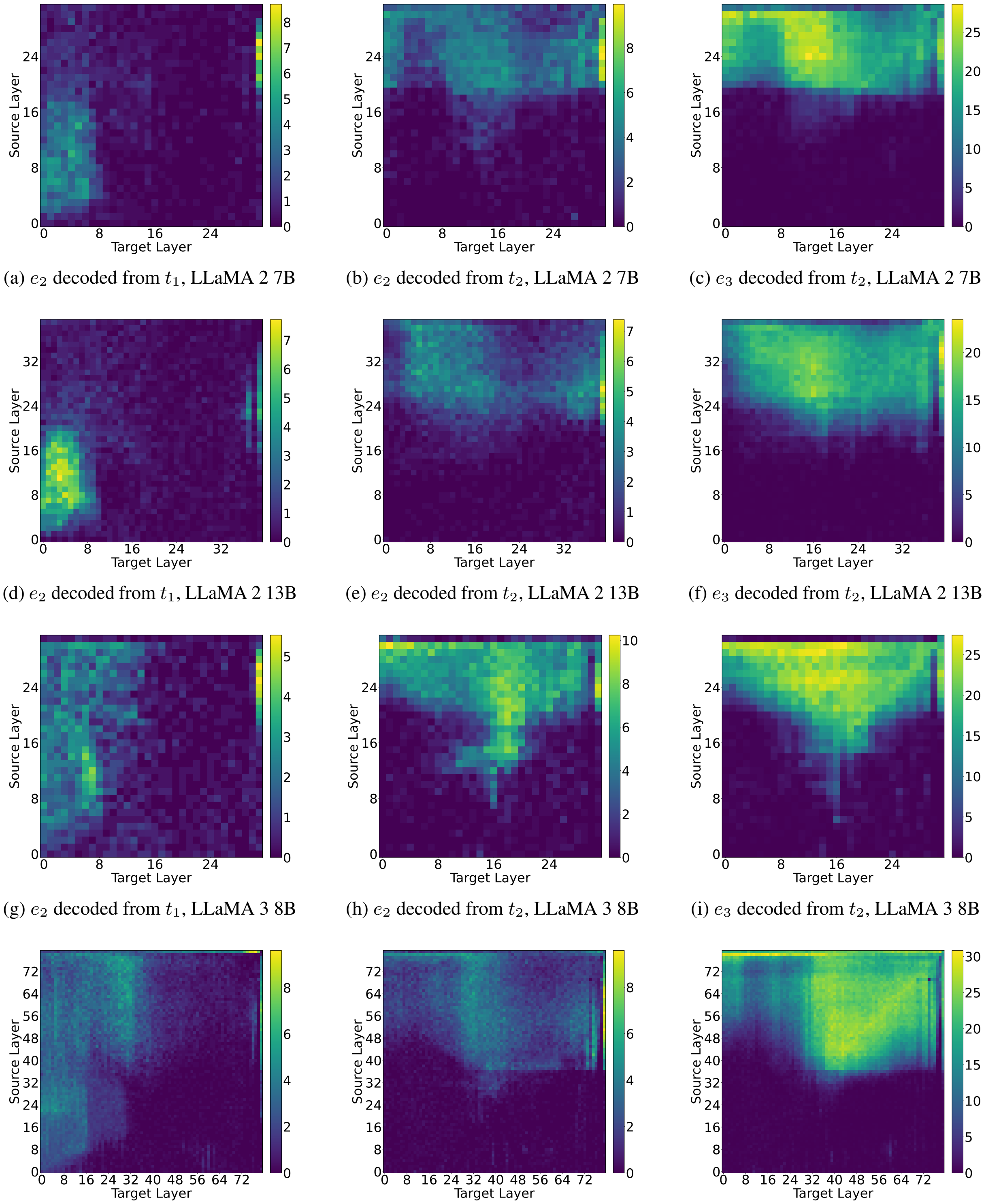


(e) Pythia 6.9B



(f) Pythia 12B

Figure 6: Percentage of cases per layer where target entities were first successfully decoded using Patchscopes.



(a) e_2 decoded from t_1, LLaMA 2 7B

(b) e_2 decoded from t_2, LLaMA 2 7B

(c) e_3 decoded from t_2, LLaMA 2 7B

(d) e_2 decoded from t_1, LLaMA 2 13B

(e) e_2 decoded from t_2, LLaMA 2 13B

(f) e_3 decoded from t_2, LLaMA 2 13B

(g) e_2 decoded from t_1, LLaMA 3 8B

(h) e_2 decoded from t_2, LLaMA 3 8B

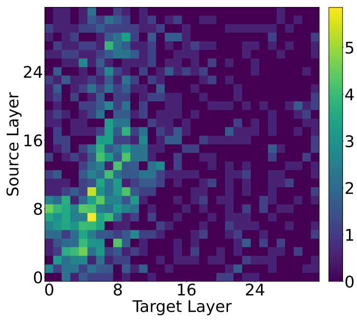
(i) e_3 decoded from t_2, LLaMA 3 8B

(j) e_2 decoded from t_1, LLaMA 3 70B

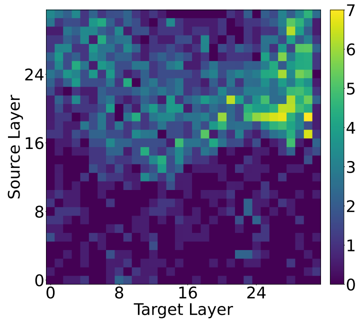
(k) e_2 decoded from t_2, LLaMA 3 70B

(l) e_3 decoded from t_2, LLaMA 3 70B

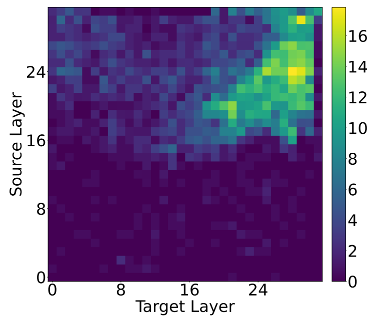
Figure 7: Patchscopes success heat-maps for LLaMA models.



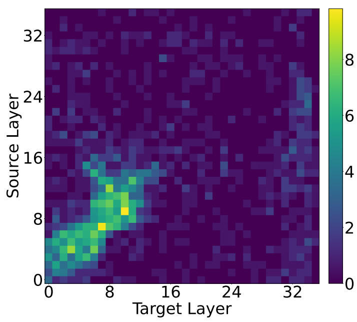
(a) e_2 decoded from t_1, Pythia 6.9B



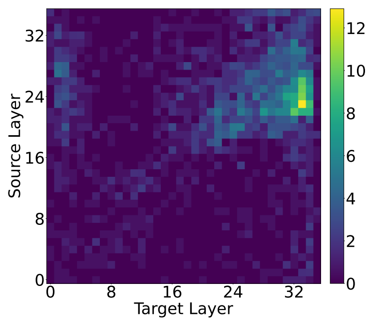
(b) e_2 decoded from t_2, Pythia 6.9B



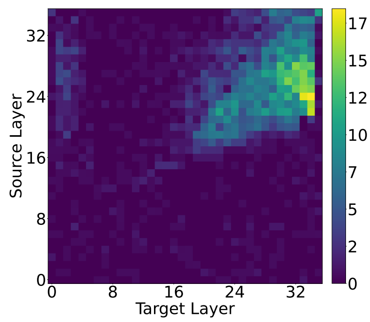
(c) e_3 decoded from t_2, Pythia 6.9B



(d) e_2 decoded from t_1, Pythia 12B

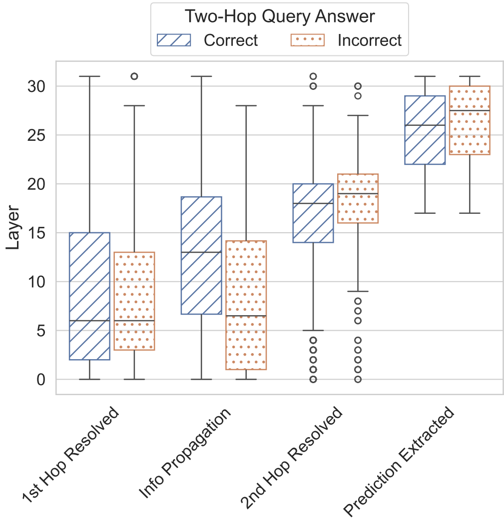


(e) e_2 decoded from t_2, Pythia 12B

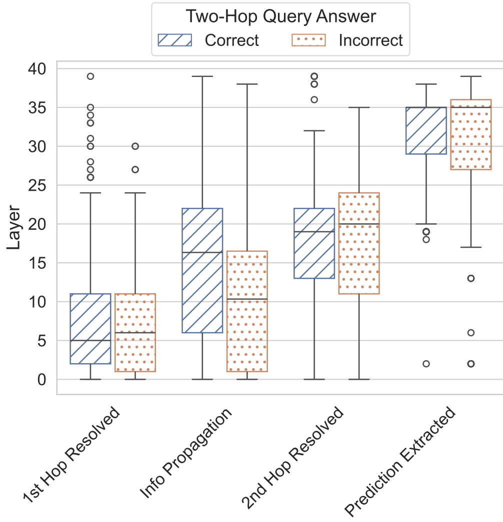


(f) e_3 decoded from t_2, Pythia 12B

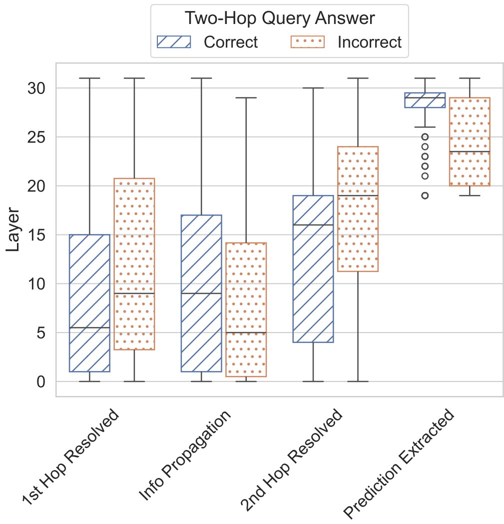
Figure 8: Patchscopes success heat-maps for Pythia models.



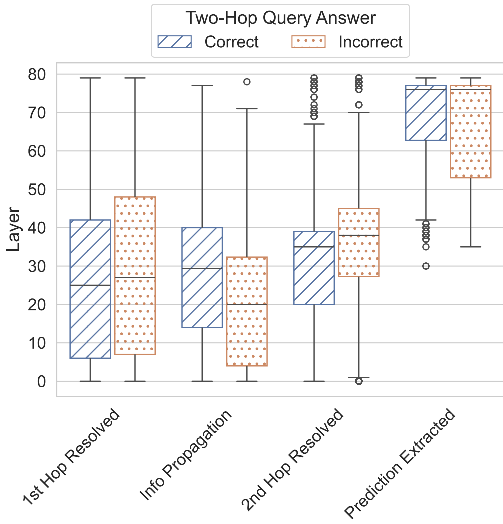
(a) LLaMA 2 7B



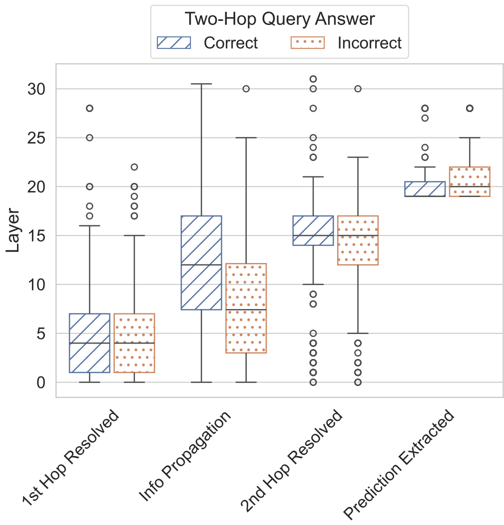
(b) LLaMA 2 13B



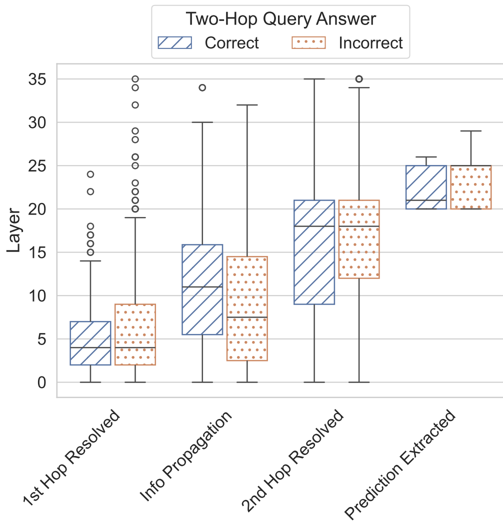
(c) LLaMA 3 8B



(d) LLaMA 3 70B

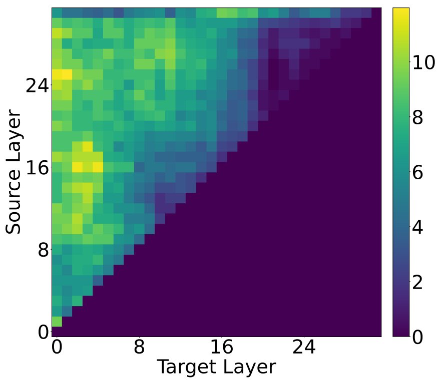


(e) Pythia 6.9B

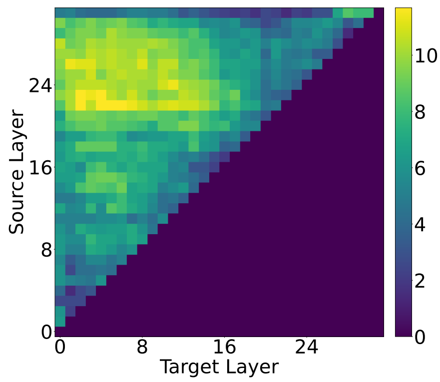


(f) Pythia 12B

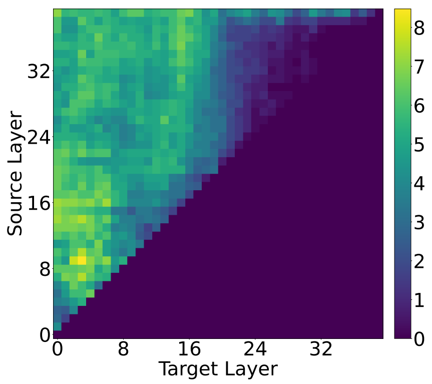
Figure 9: A comparison of the first layers of each pathway stage between correct and incorrect cases for all models.



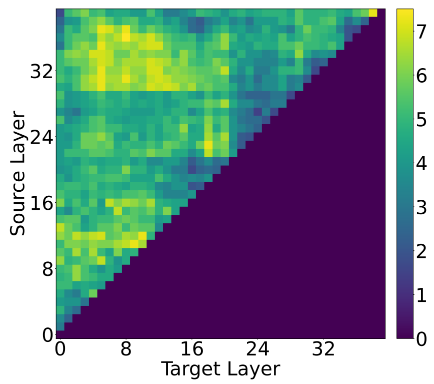
(a) Source token is t_1, LLaMA 2 7B



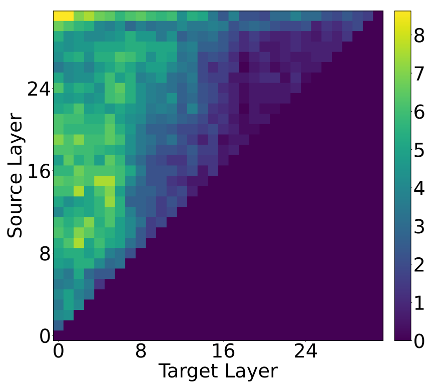
(b) Source token is t_2, LLaMA 2 7B



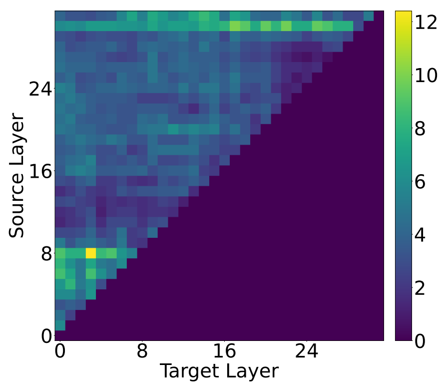
(c) Source token is t_1, LLaMA 2 13B



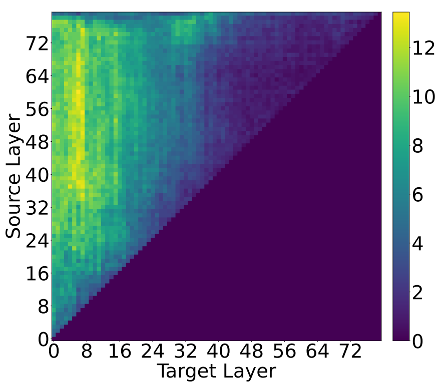
(d) Source token is t_2, LLaMA 2 13B



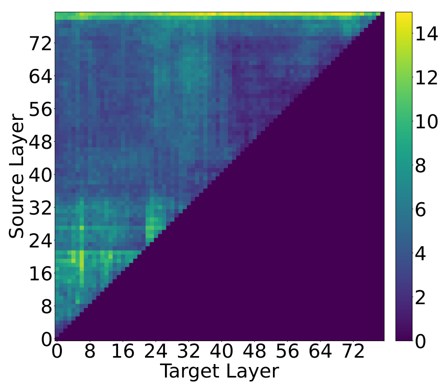
(e) Source token is t_1, LLaMA 3 8B



(f) Source token is t_2, LLaMA 3 8B

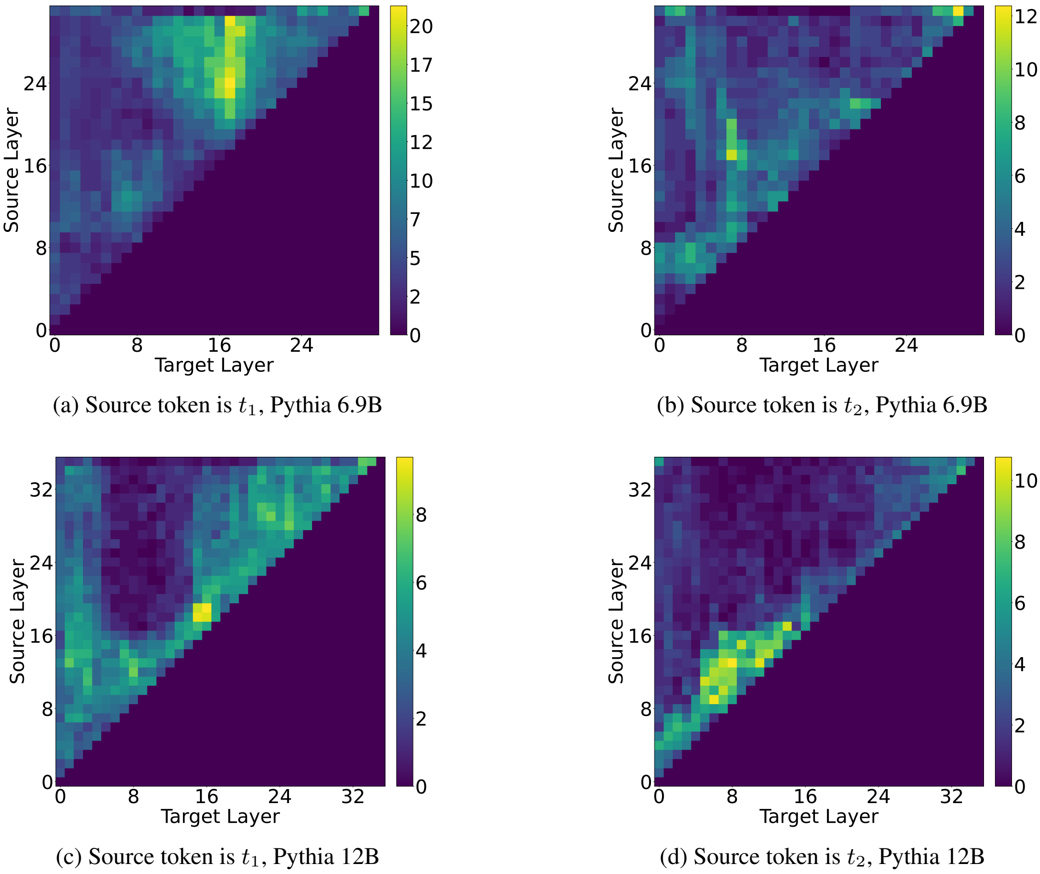


(g) Source token is t_1, LLaMA 3 70B



(h) Source token is t_2, LLaMA 3 70B

Figure 10: Back-patching heat-maps of successful source and target layers for LLaMA models.



(a) Source token is t_1, Pythia 6.9B

(b) Source token is t_2, Pythia 6.9B

(c) Source token is t_1, Pythia 12B

(d) Source token is t_2, Pythia 12B

Figure 11: Back-patching heat-maps of successful source and target layers for Pythia models.