Do Transformers Encode Cognitive Maps: Analyzing LLMs on Search Tasks?

***Author*: Halleluiah Girum**
Research Team: Samantha Shulman, Sami Nourji, Michael Lepori
Supervisor : Professor Ellie Pavlick
Click Here for Code

1 Introduction

A cognitive map is a representation or model of the underlying structure that depicts how different elements or concepts are interrelated within a task or environment. Cognitive maps (Cogmaps) are crucial for inference in planning and reasoning tasks, as their incremental, scalable, but compressed representations were found to be related to the “brain’s GPS.” As humans utilize CogMaps for inference in planning and reasoning tasks, the question arises of how Large Language Models (LLMs) make inferences in such tasks and if they utilize CogMaps as well. However, a recent paper by Momennejad et al.Momennejad et al. [2023] argued that LLMs do not possess cognitive map capabilities and instead rely on route memorization when solving inference tasks within graphs. The researchers found that while LLMs perform well on simpler graphs, they fail when a representation of the graph’s structure is necessary, often hallucinating, selecting suboptimal paths, or falling into loops. Although this paper claims that LLMs lack cognitive maps and planning abilities based on their performance on shortest-path tasks in complex graphs, we proposed the potential existence of a performance-competence paradox.

Another research group, Saparov et al. [2024], conducted the research we planned in identifying the mechanism used by transformers for search tasks. However, Saparov et al. [2024] found that as the size of the graph increased, the algorithm used by the transformer did not scale and thus failed in the search tasks. Increasing the number of parameters in the transformer did not alleviate this difficulty, which suggests that the problem lies in the mechanism utilized by the transformer not scaling to larger graphs. In order to continue probing the limitations of transformers in conducting search, we hypothesized that the path-merging algorithm identified by Saparov et al. [2024] led transformers to learn a reachability heuristic of whether you can reach the target node from the source node, rather than store the underlying representation or cognitive map of the graphs, which is a crucial part for reasoning and planning tasks.

To investigate this hypothesis, we design a causal intervention experiment using synthetic directed acyclic graphs with three disjoint arms from a common source to a shared sink. By selectively ablating two arms in each variant and measuring changes in the model’s output and internal activations, we test whether the transformer stores multiple paths or relies on a simpler reachability heuristic. Our results show that, across all layers and heads, only two to three heads exhibit a notable logit difference (0.2 nats), while the rest remain largely unaffected by ablation. This suggests that apart from a single attention head with some path sensitivity, the model does not store full path representations and instead uses reachability-based reasoning. However, further investigation is needed to establish a direct link between transformers’ failure to search in large graphs and their apparent inability to store complete latent representations of graph structure.

Preprint. Under review.

2 Related Work

Originally, we planned to investigate the mechanism by which LLMs reason through shortest-path tasks in simpler graphs, proposing binding vectors as a potential way of storing CogMaps. For each node in the graph, the binding vectors serve as pointers that reference connected nodes. Previous work, such as Feng et al. Feng and Steinhardt [2024], introduces the binding ID mechanism to explain how LLMs associate variables within their hidden states, and the binding vector is a plausible mechanism through which Cogmaps are stored in LLM representations. Training linear probes was crucial in our plan to investigate whether the binding IDs were present in the hidden states of the model.

However, after encountering the work of Saparov et al. Saparov et al. [2024], we shifted our focus toward evaluating and extending their findings. In this paper, the authors train transformers from scratch on search tasks in Directed Acyclic Graphs (DAGs), carefully constructing training distributions that prevent shortcut heuristics. Through mechanistic interpretability techniques, they identify that transformers learn a path-merging algorithm: a parallelized computation that incrementally builds sets of reachable vertices from each node. Each attention layer expands these reachable sets, enabling transformers to simulate exponential search behavior by merging information across multiple paths.

While this mechanism allows the model to infer reachability—i.e., whether a path exists between source and target—it does not necessarily imply that the transformer forms a full latent representation of the graph's structure. This distinction is crucial: reachability can be computed without storing which paths exist, or how many, or even which nodes they pass through. Consequently, this could lead to the types of failures observed in Saparov et al.'s work, where models struggled on larger graphs or when generalizing beyond trained distributions. Their findings suggest that the model's internal representation remains shallow, relying on minimal cues for reachability rather than capturing the true cognitive map of the graph.

3 Technical Approach

To summarize, the approach in this work will be to:

1. Train linear probes to investigate binding vector hypothesis
2. Path Patching in DAGs to investigate reachability vs. cognitive map abilities

3.1 Probes

In order to establish whether LLMs store cognitive maps, we decided to study the activations or hidden states of the model. Since the LLM would have to create the representation of the graph in-context, this representation would be found in the hidden states of the model and not the weights.

We designed a 26-way classifier, which took as its input the hidden states of the LLM at the tokens representing Nodes 2, 3, and 4. The output of the classifier would be a class, from 0 to 25, corresponding to the letter associated with the node. For example, for the prompt in Figure 1a, the hidden states at token 'J', token 'M', and token 'Q' would be input into the classifier, and the expected output would be class 'A', 'J', and 'M', respectively. Our experiment was based on the assumption that the activation for token 'J' in the hidden state would either encode the content of Node A or a binding vector pointing to Node A if the LLM is representing the graph structure. This part of our research focused on identifying whether the hidden states of models stored cognitive maps in the form of binding vectors or content.

3.2 Path Patching

3.3 Technical Approach: Causal Path-Patching to Probe Path Storage

After pivoting to focus on the results of the Saparov et al. [2024] paper, our goal was to determine whether transformers encode complete graph representations or merely compute reachability when answering shortest-path queries. To this end, we design a two-step causal-intervention protocol that isolates the contribution of each attention head to the model's belief about valid paths.

1. Graph Construction and Input Encoding

Graph family. We generate synthetic directed acyclic graphs (DAGs) that contain three disjoint arms of equal length L, diverging from a common source node s and reconverging on a single sink node t. The arms share no internal vertices, ensuring that each represents a distinct proof of reachability. 1b

Clean vs. ablated variants. For every graph, we create three *ablated* variants, each preserving exactly one arm while redirecting the other two arms to a dummy vertex. Thus, the surviving arm is the *only* valid $s \to t$ route in the ablated prompt.

Tokenization. Edges are linearized as sequences of `E u v` tokens, followed by a query token `Q s t`, and a prefix token `P` that marks where the model must begin outputting a path. This yields four prompts per graph: one clean and three ablated.

2. Causal Path-Patching Procedure

1. **Baseline passes:** We run the model with caching enabled on the clean and each ablated prompt to obtain the output logits and the per-head `z` activations (i.e., post-attention value vectors).
2. **Head-level intervention:** For every layer ℓ and head h:
   - We replace the `z` activation of (ℓ, h) in the clean run with its counterpart from one of the ablated runs.
   - We forward the modified activation through the model to obtain the *patched* logits.
   - This is repeated using each of the three ablated variants, allowing us to test each head's contribution.
3. **Evaluation metrics:** We measure two complementary quantities:
   - **Per-path logit drop:**
   $$\Delta_{\text{path}} = \sum_{i=1}^{L} \left[\ell_{\text{clean}}(t_i) - \ell_{\text{patched}}(t_i)\right],$$
   where t_i are the node tokens along the surviving path.
   - **Log-sum-exp mass drop:**
   $$\Delta_{\text{LSE}} = \log \sum_k e^{\ell_{\text{clean}}(P_k)} - \log \sum_k e^{\ell_{\text{patched}}(P_k)},$$
   where the sum runs over all valid paths P_k in the patched graph.

3. Rationale

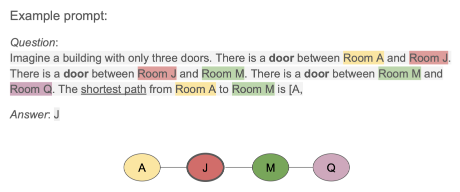
- If a head encodes only *reachability*, then removing two of three arms (while preserving one valid path) should leave both Δ_{path} and Δ_{LSE} near zero after patching that head.
- If a head stores *additive information about all paths*, then removing the other arms will significantly reduce the total probability mass over valid paths. This will yield a high Δ_{LSE}, even when the surviving path remains intact.
- By sweeping across all layers and heads, we can map out which components, if any, carry multi-path representations.

This path-patching experiment thus provides a targeted behavioral lens on the internal mechanism identified by Saparov et al. Saparov et al. [2024], allowing us to test whether the learned path-merging algorithm reflects a mere reachability heuristic or supports a full cognitive map-like structure.

3.4 Dataset

For the linear probes, we create a set of graphs, each with four nodes labeled with letters from A to Z. The graphs are all path graphs, which are undirected and represented in the context of rooms. Figure 1a provides an example of a graph, along with the prompt that is fed to the model.

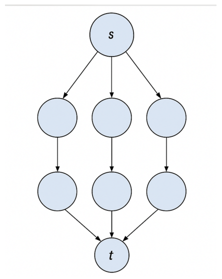
For the path patching experiments, we generate a dataset of 50 synthetic directed acyclic graphs (DAGs), each structured with three disjoint arms of equal length L that branch from a common source node s and re-converge at a shared sink node t. Figure 1b provides an example.


Example prompt:

Question:
Imagine a building with only three doors. There is a **door** between Room A and Room J. There is a **door** between Room J and Room M. There is a **door** between Room M and Room Q. The shortest path from Room A to Room M is [A,

Answer: J

(a) Example Simple Graph



(b) Multi-Path DAG

Figure 1: Different DAG structures used in our experiments.

4 Results

Layer	**Head**	**Path 1**		**Path 2**		**Path 3**	
		Δ_{path}	Δ_{LSE}	Δ_{path}	Δ_{LSE}	Δ_{path}	Δ_{LSE}
12	07	+0.328	+0.316	+0.407	+0.383	+0.333	+0.325
12	10	+0.086	+0.201	–	–	–	–
21	03	+0.055	+0.221	+0.042	+0.202	+0.203	+0.207
03	02	–	–	–	–	+0.247	+0.237
13	12	–	–	–	–	+0.211	+0.208

Table 1: Per-path logit drop and log-sum-exp (LSE) mass drop for attention heads with logit drop greater than 0.2 across three disjoint paths.

We behaviorally tested various LLMs on shortest-path questions, as shown in Figure 4, and its variations including presenting the graph structure in a non-canonical order and asking shortest-path questions from Nodes 2 to 4. We observed the performance of different models, as summarized in Figure 3a. The pythia-2.8b model achieved the highest accuracy, with 99% on shortest-path questions from Node 1 to 3 when the graph was presented in canonical order, and hence will be the model we used for linear probes.

Our behavioral results reveal a "flip-flop" behavior in shortest-path tasks, where models performing well from Node 1 to 3 struggle with Node 2 to 4, and vice versa, suggesting reliance on heuristics rather than a latent representation of the graph (CogMap). Our linear probes, achieving 85% test accuracy 3a, indicate that LLM hidden states may encode a cognitive map, as they correctly predict shortest paths with a benchmark probability of 3%.

To determine whether the transformer encodes the full internal structure of each path or relies on a simpler reachability heuristic, we performed causal path-patching on all attention heads (Section **??**). For every head, we measured the *per-path logit drop* (i.e., the average reduction in next-token logit along the surviving path) and the *log-sum-exp (LSE) mass drop* (i.e., the reduction in total log-probability assigned to all valid paths) when that head's activation was replaced by its counterpart from an ablated prompt in which two of the three arms were removed.

Per-path logit drop quantifies the extent to which a head's activation contributes to the model's confidence in generating the tokens along the surviving path. A small drop suggests that the head merely verifies the existence of a path—consistent with a reachability heuristic. In contrast, a substantial drop indicates that the head relies on information from the full multi-path structure, encoding sequence-level details rather than just the existence of connectivity.

LSE mass drop measures the change in total log-probability mass over all correct paths, computed as the log-sum-exp of their respective logits. This metric captures the model's sensitivity to the removal of alternative paths: a large drop implies that the head contributed meaningful probability mass to the ablated routes, reflecting its reliance on and representation of the full graph structure. A negligible drop, on the other hand, is consistent with a head that is insensitive to the presence of multiple paths and uses only a minimal signal of reachability.

Together, these two metrics provide insight into whether a head's behavior reflects reachability or a richer internal encoding of multiple disjoint paths within the graph. A head that shows a large drop on either metric can be interpreted as storing and utilizing multi-path information, whereas heads with consistently small drops are more likely operating under a simplified reachability-based mechanism.

4.1 Sparse importance across heads

Out of the 1,024 heads in the model, only five produced an LSE mass drop greater than 0.20 nats on any path. Most heads exhibited drops below 0.05 nats—within the noise band determined by fp16 jitter-band they do not store explicit path information.

4.2 A single high-impact circuit

Head H07 in layer 12 (L12H07) is unique:

- For every path it shows $\Delta_{\text{path}} \geq 0.33$ nats and $\Delta_{\text{LSE}} \geq 0.31$ nats—well above the 0.30 nats threshold we adopt as a causal contribution.

L12H07's logit mass drops for both the per-path logit and the LSE mass, suggesting that it encodes information about the latent graph structure.

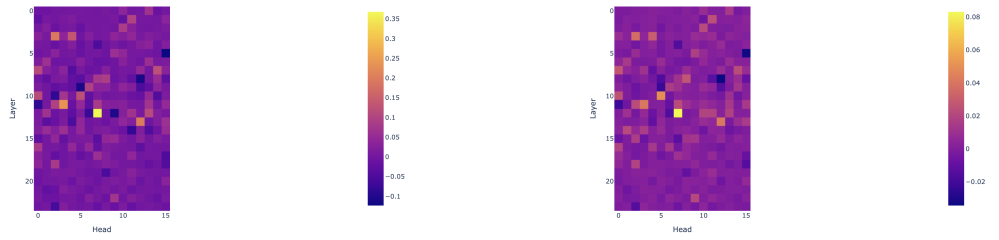
4.3 Curious Results

Heads L12H10 and L21H03 show modest mass drops ($\Delta_{\text{LSE}} \approx 0.20$ nats) but small per-path drops. These heads appear largely unaffected in predicting the specific tokens along the remaining path, yet their overall correctness decreases, suggesting they may encode some latent information about the full graph structure while still relying primarily on a reachability-based heuristic. Interestingly, when Path 3 is the sole remaining route, a greater number of heads display elevated logit drops, indicating that the model may represent or utilize this path differently from the others. But further investigation is necessary on both these findings.

4.4 Implications for the path-merging hypothesis

The transformer's behaviour:

- **Predominantly reachability-based:** The overwhelming majority of heads are unaffected by removal of two arms, indicating that a single reachable route suffices for their computation.
- **Limited path storage:** One high-impact circuit (L12H07) appears to encode full path-level details. Its existence demonstrates that transformers are *capable* of encoding multi-step structure, but its rarity suggests such representations are not a default outcome of training.



(a) Per-path logit difference heatmap

(b) Log-sum-exp mass drop heatmap

Figure 2: Head-level patching effects for two metrics: logit drop along a path and LSE mass over valid paths. Only Path 3 is valid.

5 Discussion

These findings support our central hypothesis: the path-merging algorithm learned by the model is largely exploited as a reachability heuristic rather than as a full latent cognitive map. However, further research is needed to strengthen this conclusion. One promising direction is to apply the mechanistic interpretability techniques introduced by Saparov et al. [2024] to our setting—specifically, to analyze how the internal path-merging algorithm manifests under full-path versus ablated-path conditions. This could provide deeper insight into whether and how the model's internal computation shifts in the absence of redundant paths. Additionally, a more systematic study of failure modes in shortest-path inference tasks—particularly when multiple disjoint paths exist—would help clarify the consequences of relying on reachability alone and offer stronger evidence for the absence of multi-path structure in the model's internal representations.

6 Conclusion

Our findings provide evidence that transformer models primarily rely on a reachability-based heuristic rather than storing the full latent structure of a graph. Through path patching experiments on synthetic multi-path DAGs, we observed that most attention heads exhibit minimal sensitivity to the removal of alternate paths—consistent with a binary reachability mechanism. Only a small subset of heads, notably L12H07, demonstrated strong logit and mass drops, indicating a more detailed encoding of multi-path structure. These results suggest that while transformers are capable of representing rich graph information, such representations are not widespread across the model. Future work is needed to mechanistically trace the internal algorithm used by these heads and to link this behavior to specific failure modes in complex reasoning tasks.

References

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://arxiv.org/abs/2310.17191`. Published as a conference paper at ICLR 2024.

Ida Momennejad, Hosein Hasanbeig, Felipe Vieira, Hiteshi Sharma, Robert Osazuwa Ness, Nebojsa Jojic, Hamid Palangi, and Jonathan Larson. Evaluating cognitive maps and planning in large language models with cogeval. *arXiv preprint arXiv:2309.15129*, 2023. URL `https://arxiv.org/abs/2309.15129`. Preprint.

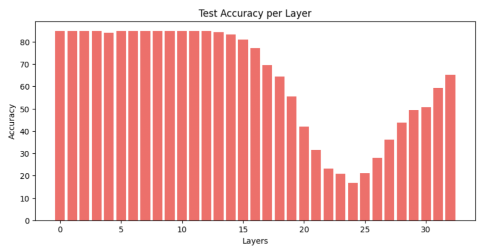
Abulhair Saparov, Srushti Pawar, Shreyas Pimpalgaonkar, Nitish Joshi, Richard Yuanzhe Pang, Vishakh Padmakumar, Seyed Mehran Kazemi, Najoung Kim, and He He. Transformers struggle to learn to search. *arXiv preprint arXiv:2412.04703*, 2024. URL `https://arxiv.org/abs/2412.04703`. Preprint.

Kevin Wang, Alexandre Variengien, Andrew Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022. URL `https://arxiv.org/abs/2211.00593`. Preprint.

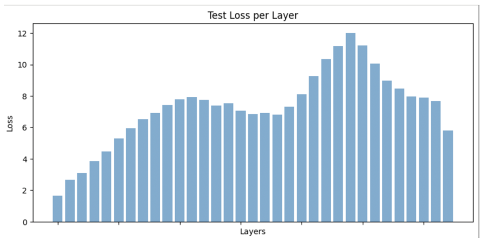
Saparov et al. [2024] Wang et al. [2022]

7 Appendix

7.1 Additional Figures



(a) Test set accuracy for linear probes



(b) Test set loss for linear probes

Figure 3: Evaluation metrics for linear probe performance on the test set.

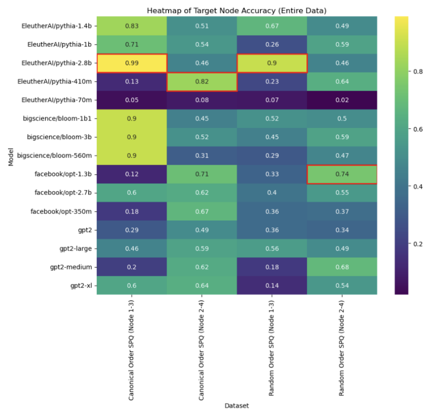
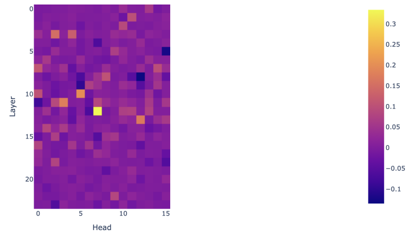
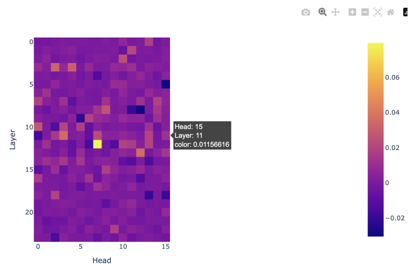


Figure 4: LLM performance on shortest-path questions on graphs

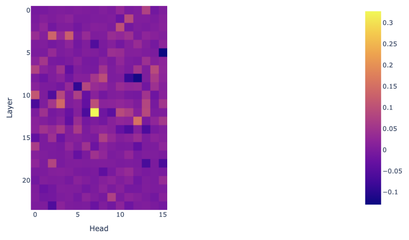


(a) Per-path logit difference heatmap

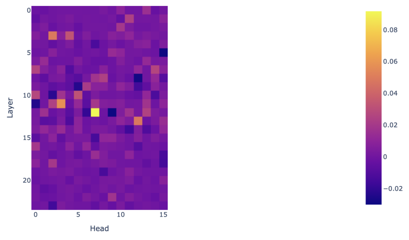


(b) Log-sum-exp mass drop heatmap

Figure 5: Head-level patching effects for two metrics: logit drop along a path and LSE mass over valid paths. Only Path 1 is valid.



(a) Per-path logit difference heatmap



(b) Log-sum-exp mass drop heatmap

Figure 6: Head-level patching effects for two metrics: logit drop along a path and LSE mass over valid paths. Only Path 2 is valid.

This project is a group project with students not in this class, but all the work I will be doing for this class will be on my own.