LLMs can Annotate Attribution Graphs

Ameen Patel [*] **Max Zhang** [*] **Nathan Hu** [1]

Abstract

Circuit tracing is an exciting technique for revealing internal computation in language models, but it requires a time-intensive manual step of grouping individual features or MLP neurons into supernodes. We present a simple pipeline for automating this step: directly presenting feature descriptions to a language model that groups them into supernodes. Using automated interpretability metrics, we confirm that supernodes generated by our pipeline are as interpretable as those generated by human annotators. On a two-hop Capitals task, our pipeline recovers a supernode corresponding to the intermediate hop in 97 of 100 prompts. Finally, we present a simple proof of concept using our pipeline for open-ended exploration, where we first automatically annotate 1000 attribution graphs from Wikipedia prompts and then use an LLM judge to flag graphs worth human review. We hope this work demonstrates that even simple automation can produce meaningful attribution graph annotations, motivating further work on automated circuit tracing. Code[1] and data[2] are provided.

1. Introduction

Recent work in mechanistic interpretability has produced techniques for revealing the internal computation graphs of language models (Lindsey et al., 2025a). These *circuit tracing* techniques produce attribution graphs whose nodes are individual model components, such as MLP neurons (Arora et al., 2026b) or SAE features (Ameisen et al., 2025; Lindsey et al., 2025b; Marks et al., 2025). These nodes can be interpreted via their top-activating text examples. However, circuit tracing by default requires a final manual annotation step in which nodes are grouped into *supernodes*: sets of neurons or features corresponding to similar concepts. This manual step is labor-intensive: fully annotating an attribution graph can take one to two researcher-hours (Lindsey et al., 2025b).

Recent work has begun to address this time-intensive bottleneck. Arora et al. (2026a) group MLP neurons into supernodes by constructing and clustering attribution profiles for each neuron. Birardi (2025) groups features into supernodes based on their activations across a set of LLM-generated probe prompts related to the original query. Our work similarly aims to automate the supernode grouping step. We treat circuit tracing as a tool for hypothesis generation[3], and ask: *what is the simplest way to annotate attribution graphs so that they yield insight into a model's internal computation?*

We present a simple yet effective pipeline for annotating attribution graphs. For each feature in the pruned attribution graph, we first generate a description from its top-activating exemplars, following standard autointerp (Bills et al., 2023; Choi et al., 2024; Paulo et al., 2025). We then use a language model (GPT-5 mini (Singh et al., 2026)) to group these descriptions into supernodes. Our pipeline costs roughly 3–6 cents on heavily pruned graphs ($\sim$80 nodes) and 61 cents on denser graphs ($\sim$585 nodes).

Our contributions are as follows:

1. We present a simple, low-cost pipeline to annotate attribution graphs using an LLM.
2. We verify that supernodes generated by our pipeline are as interpretable as those from human annotation, according to automated interpretability metrics.
3. We next apply our pipeline to the well-studied two-hop Capitals task and find that it recovers a supernode corresponding to the intermediate hop in 97 of 100 prompts.
4. Last, we present an initial proof of concept for open-

[*]Equal contribution [1]Stanford University. Correspondence to: Nathan Hu <nathu@cs.stanford.edu>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

[1]https://github.com/maxh119Z/circuit-tracer-automation

[2]https://huggingface.co/datasets/circuit-tracer-automation/pipeline_automation

[3]For example, Lindsey et al. (2025b) surfaced evidence of planning circuits in poetry through attribution graphs and steering experiments. However, fully characterizing such planning is the subject of dedicated follow-up work (Nainani et al., 2025; Maar et al., 2026).

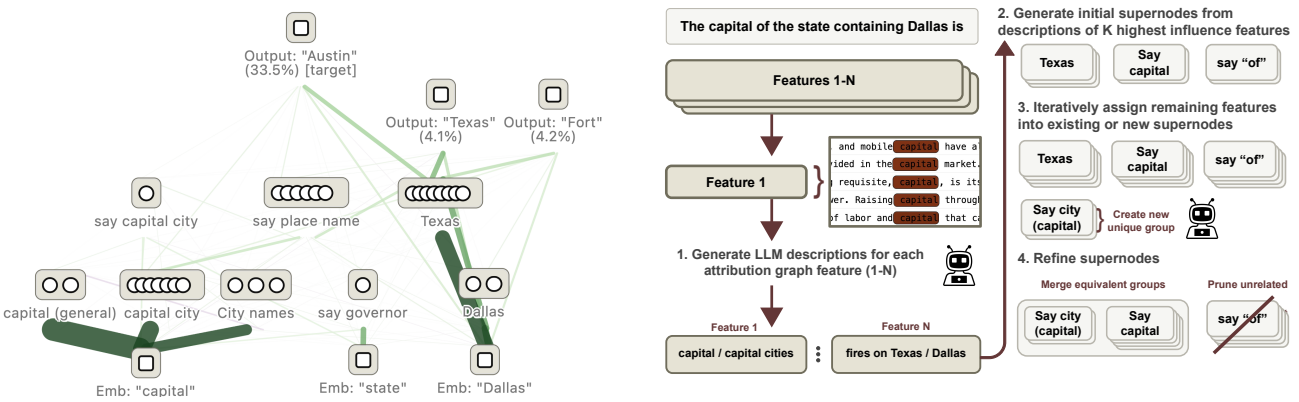


Figure 1. **Method Overview.** Prompt: "The capital of the state containing Dallas is" → *Austin* at influence threshold 0.40 (42 features). **Left**: a sample attribution graph annotated by our pipeline. **Right**: An overview of our LLM supernode grouping process.

ended exploration of attribution graphs: we automatically annotate 1000 Wikipedia prompts and filter the resulting graphs for interesting or surprising behavior.

2. Methods

All experiments use Gemma-2-2B (Team et al., 2024) with the Gemma Scope single-layer transcoders (Lieberum et al., 2024). We generate attribution graphs through circuit-tracer (Hanna et al., 2025) using default hyperparameters. The only hyperparameter we vary is the cumulative node-influence pruning threshold: 0.7 in Section 3 and 0.4 in Sections 4 and 5.[4] All steps in our pipeline use GPT-5 mini (Singh et al., 2026) as the LLM annotator.

2.1. Feature descriptions

For each feature in the pruned attribution graph, we first use an LLM to generate a feature description from activating text examples. This is the standard autointerp pipeline for feature description (Bills et al., 2023; Choi et al., 2024; Paulo et al., 2025). We use 5 activating text examples from Neuronpedia (Lin, 2023). We additionally provide the LLM with logit information about the tokens most promoted and most suppressed by the feature. We take care to tune our instructions for what features can be described as output features ("say X"). The description prompt and sample descriptions are in Appendix A.1 and Appendix A.5.

2.2. Supernode grouping

We generate supernodes by presenting the feature descriptions from the previous step to an LLM in three phases. At each phase, the LLM also receives the original prompt and the model's top-3 next-token predictions for additional context.

[4]The 0.4 threshold retains roughly the 40–80 highest-influence nodes per graph.

1. **Generate supernodes.** Present the 50 highest-influence features and ask the LLM to identify and name meaningful supernodes, grouping those features into them (or leaving them ungrouped if irrelevant).

2. **Iteratively assign remaining features.** Each remaining feature is then presented to either be assigned to an existing supernode, placed in a new supernode (rare, reserved for relevant features that don't fit existing groups), or left ungrouped. We present the remaining features descriptions in batches of 50.

3. **Refine supernodes.** A final pass that drops irrelevant supernodes, merges equivalent ones (directly or in the context of the prompt), and reassigns misplaced features.

As with feature descriptions, we include detailed instructions on what makes a useful supernode. Output supernodes can be formed only by merging output features. Grouping granularity is judged relative to the prompt. For example on 2-hop capitals Dallas prompt, surrounding Southern states might merge into one supernode while Texas remains its own group. More heuristically, we also instruct the model to avoid building supernodes for less informative patterns like stop words or punctuation. We provide full prompts for all stages of our pipeline in Appendix A.

Cost and runtime. Cost scales roughly linearly with the number of features in the pruned graph. On heavily pruned Capitals graphs (40–80 features), our pipeline costs around \$0.05 per graph (34K input and 13K output tokens); on denser graphs at the 0.7 threshold (∼585 features), it costs around \$0.60 per graph (728K input and 214K output tokens).

Condition	Feature Detection	Text Detection	Features Grouped	# Groups	Feat./ group
Human-annotated	66.6%	81.8%	33.7	6.6	4.91
Ours (top 50)	80.8%	81.7%	26.9	7.1	3.98
Ours (top 100)	83.1%	82.1%	48.5	7.1	7.08
Ours (top 150)	82.8%	84.2%	68.1	7.5	9.43
Ours (top 200)	81.9%	81.7%	88.5	8.1	11.62
Ours (full)	93.4%	83.2%	207.5	8.3	26.42
Ours (full, no refine)	92.3%	82.1%	250.1	11.1	24.30
Random Grouping	22.1%	54.7%	–	–	–
Chance	10%	50%	–	–	–

Table 1. **Autointerp scores vs. human baseline.** On 15 Neuronpedia graphs from Hanna et al. (2025), our supernodes match or slightly exceed human-annotated ones on both automated interpretability metrics. Supernode sizes of human-annotated groups are most comparable to those from running our method on the top 50 or 100 features.

3. Comparison to human-annotated supernodes

We compare the interpretability of supernodes generated by our pipeline to those produced by human annotation, using the 15 human-annotated reference graphs from circuit-tracer (Hanna et al., 2025) on a variety of Gemma-2-2B prompts. We adapt two automated interpretability metrics to the supernode setting. For any supernode (whether from our pipeline, human annotation, or a random baseline), we generate a description in two steps: (1) describe each member feature using an LLM, as in Section 2.1; (2) feed the list of feature descriptions to a second LLM that produces a single supernode description. We use this supernode description to compute two metrics:

- **Feature Detection Score.** Given the supernode description and 10 feature descriptions (1 of which is from the supernode), can an LLM identify the supernode feature?
- **Text Detection Score.** Given the supernode description and 10 text samples (5 from supernode features), can an LLM identify the 5 text samples that activate a supernode feature?

Negative features and text examples are sampled from ungrouped features in the same pruned attribution graph. We additionally compare to a random baseline, where supernodes are formed by randomly grouping features in the pruned attribution graph. Our pipeline can group any number of features. We show results applying our pipeline to annotate the top k features for differing values of k. Supernode sizes of human-annotated groups are most comparable to those from running our method on the top 50 or 100 features.

Validation Results. Autointerp scores and supernode statistics are shown in Table 1. Increasing the number of features shown to the pipeline increases features grouped into supernodes. Most additional features are packing into existing supernodes rather than yielding new supernodes. For all settings, automatically generated supernodes are as interpretable or slightly more so than human-annotated ones on both automated interpretability metrics. Ablating the refine step increases the number of supernodes and slightly decreases autointerp scores. We confirm results hold when varying the LLM judge in Appendix B.4 and report per-feature description scores in Appendix B.1.

Qualitative examples. Appendix E provides a broad gallery of qualitative examples: side-by-side comparisons with Hanna et al. (2025) human annotations, plus pipeline annotations on Capitals and Wikipedia graphs, the other datasets explored in this paper.

4. Capitals: 2-Hop Queries

A well-studied 2-hop circuit (Lindsey et al., 2025b; Arora et al., 2026b; Ameisen et al., 2025) answers queries such as *"What is the capital of the state containing Dallas?"* by first recalling the state (e.g., *Texas*), then mapping it to its capital. We construct 100 such queries, 50 with each US state as the intermediate hop and 50 with a country. Sample queries are shown in Appendix C. One each query, we evaluate if our annotation pipeline a supernode corresponding to the intermediate hop. Our pipeline correctly recovers the intermediate-hop supernode in 97 of 100 prompts (Figure 2, left); of the remaining 3, human review found 2 had intermediate-hop features the pipeline missed, while 1 genuinely had no such features in the pruned graph. We then confirm that recovered intermediate-hop supernodes are causally important via constrained patching (Hanna et al., 2025) (Figure 2, right): suppressing them ($0\times$) reduces the probability of the correct answer while amplifying them ($2\times$) increases it, on prompts where the model is initially correct and incorrect alike. To test that our pipeline generalizes beyond transcoder features, we also rerun it on the model's raw MLP neurons (built with ADAG (Arora et al., 2026a)) on these same 100 prompts, and compare the two

Outcome	#
Found by pipeline	97
Present but missed	2
Not in pruned graph	1
Total	100

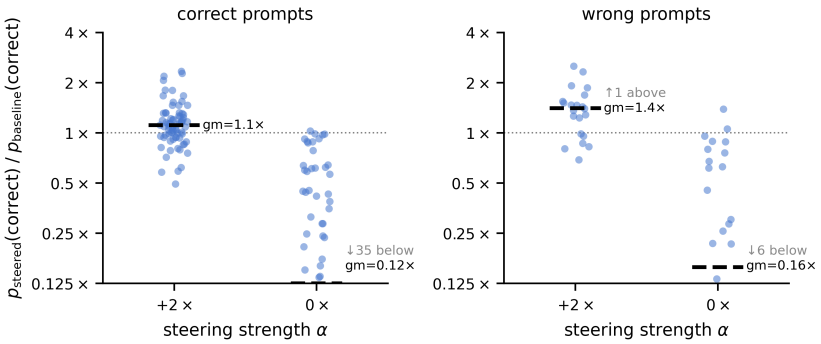


Figure 2. **Capitals 2-hop results. Left:** Hop recovery outcomes on 100 prompts. Our pipeline recovers a supernode for the intermediate hop in 97 of 100 prompts. We manually evaluated the remaining 3: in 2, an intermediate-hop feature was present but not grouped into a supernode; in 1, no intermediate-hop feature was present in the pruned graph. **Right:** We confirm via steering that intermediate-hop supernodes are causally important on both correct and incorrect prompts: amplification (2×) increases the probability of the correct answer, suppression (0×) decreases it. Each point is one prompt.

decompositions in Appendix F.

5. Towards Open-Ended Circuit Exploration

Lastly, we use our pipeline to demonstrate a proof of concept: applying automated annotation to open-ended exploration and identification of meaningful attribution graphs. We first collect 1000 one- to two-sentence Wikipedia snippets, applying heuristic filtering to avoid graphs that predict low-entropy tokens or stop words (full details in Appendix C.3). We then use GPT-5.4 to rate each annotated graph along several handcrafted criteria for interestingness (details in Appendix A.3). We use these ratings to filter from 1,000 graphs down to 10 for manual review, of which we find two interesting graphs we present below. We show these graphs in Figure 3. In the first graph (left), on the prompt "Soon after leaving the city, the highway begins to follow the North" → *Fork*, many active features relate to rivers and streams despite no mention of water. Sampling further from Gemma-2-2B yields "Fork of the Flathead River," suggesting these river features reflect internal planning in the model. In the second graph (right), on the prompt "Internationally, La Maison en Petits Cubes won the Academy Award for Best" → *Animated*, all of the most likely completions correspond to different Academy Award categories. For each category, there is a corresponding internal supernode, suggesting that the model's internal computation promotes several plausible continuations in parallel.

6. Discussion

In this work, we present a simple pipeline for automated annotation of attribution graphs, automating a labor-intensive step of circuit tracing. Beyond general interpretability research, automated annotation could enable broad, unsupervised search for interesting attribution graphs; we provide a proof of concept in Section 5. Additionally, supernode grouping can apply to any feature basis (MLP neurons, single-layer transcoders, or cross-layer transcoders), and an exciting direction for automated annotation is to enable systematic comparison of these different bases for decomposing attribution graphs; we take a first step toward this in Appendix F, comparing MLP-neuron and transcoder decompositions of the Capitals graphs.

Our work has several limitations. We study a single model and transcoder suite on a limited set of tasks: the 15 reference graphs from Hanna et al. (2025) and the relatively simple 2-hop Capitals task. We only test our pipeline with a single LLM annotator, GPT-5 mini. Our supernode grouping pipeline is naive: it ignores spatial information (the layer and prompt-token position of each feature), the attribution flow between features, and the backward-from-output search style that human circuit-tracers often use. More generally, human interpretation draws on much more context than activating text examples alone, including the prompt and surrounding features in the graph. For example, in Appendix D, we apply our pipeline to arithmetic queries and find that the resulting supernode labels are too coarse to capture the known internal structure. Incorporating these is a clear direction for future work. Despite these many areas for improvement, our method already produces useful annotations of attribution graphs. We hope this work demonstrates that there remains much low-hanging fruit when automating circuit tracing and motivates further work in this direction.

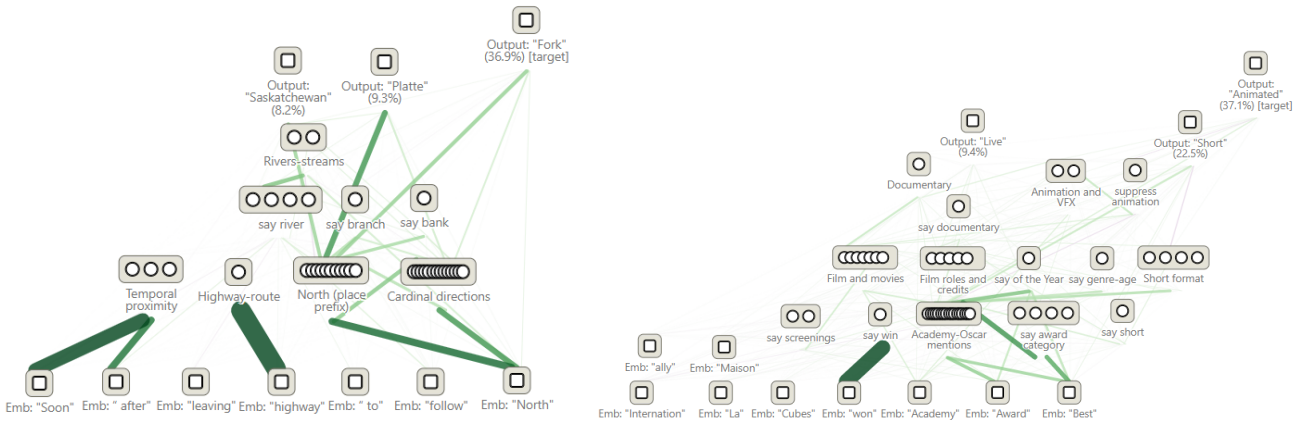


Figure 3. **Attribution graphs surfaced by open-ended exploration. Left:** "Soon after leaving the city, the highway begins to follow the North" → *Fork*. Despite no mention of water, river-related supernodes are active. Sampling further from Gemma-2-2B yields "Fork of the Flathead River," suggesting these supernodes reflect planning ahead. **Right:** "Internationally, La Maison en Petits Cubes won the Academy Award for Best" → *Animated*. The graph shows competing categories active in parallel (animation, documentary, movies).

Impact Statement

This paper presents work that advances the field of mechanistic interpretability. We present a method for automating the human-labor-intensive feature annotation step of circuit tracing, which can accelerate interpretability research more broadly. Research progress in interpretability is inherently dual-use: it can benefit capabilities while also improving our safety and understanding of models. We believe this work is net positive.

References

Ameisen, E., Lindsey, J., Pearce, A., Gurnee, W., Turner, N. L., Chen, B., Citro, C., Abrahams, D., Carter, S., Hosmer, B., Marcus, J., Sklar, M., Templeton, A., Bricken, T., McDougall, C., Cunningham, H., Henighan, T., Jermyn, A., Jones, A., Persic, A., Qi, Z., Ben Thompson, T., Zimmerman, S., Rivoire, K., Conerly, T., Olah, C., and Batson, J. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Arora, A., Wu, Z., Steinhardt, J., and Schwettmann, S. Adag: Automatically describing attribution graphs, 2026a. URL https://arxiv.org/abs/2604.07615.

Arora, A., Wu, Z., Steinhardt, J., and Schwettmann, S. Language model circuits are sparse in the neuron basis, 2026b. URL https://arxiv.org/abs/2601.22594.

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 2023.

Birardi, G. Automated circuit interpretation via probe prompting, 2025. URL https://arxiv.org/abs/2511.07002.

Choi, D., Huang, V., Meng, K., Johnson, D. D., Steinhardt, J., and Schwettmann, S. Scaling automatic neuron description. https://transluce.org/neuron-descriptions, October 2024.

Hanna, M., Piotrowski, M., Lindsey, J., and Ameisen, E. circuit-tracer. https://github.com/decoderesearch/circuit-tracer, 2025. The first two authors contributed equally and are listed alphabetically.

Lieberum, T., Rajamanoharan, S., Conmy, A., Smith, L., Sonnerat, N., Varma, V., Kramár, J., Dragan, A., Shah, R., and Nanda, N. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2, 2024. URL https://arxiv.org/abs/2408.05147.

Lin, J. Neuronpedia: Interactive reference and tooling for analyzing neural networks, 2023. URL https://www.neuronpedia.org. Software available from neuronpedia.org.

Lindsey, J., Ameisen, E., Nanda, N., Shabalin, S., Piotrowski, M., McGrath, T., Hanna, M., Lewis, O., Tigges, C., Merullo, J., Watts, C., Paulo, G., Batson, J., Gorton, L., Simon, E., Loeffler, M., McDougall, C., and Lin, J. The circuits research landscape: Results and perspectives. *Neuronpedia*, 2025a. URL https://neuronpedia.org/graph/info.

Lindsey, J., Gurnee, W., Ameisen, E., Chen, B., Pearce, A., Turner, N. L., Citro, C., Abrahams, D., Carter, S., Hosmer, B., Marcus, J., Sklar, M., Templeton, A., Bricken, T., McDougall, C., Cunningham, H., Henighan, T., Jermyn, A., Jones, A., Persic, A., Qi, Z., Thompson, T. B., Zimmerman, S., Rivoire, K., Conerly, T., Olah, C., and Batson, J. On the biology of a large language model. *Transformer Circuits Thread*, 2025b. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Maar, J., Paperno, D., McDougall, C. S., and Nanda, N. What's the plan? metrics for implicit planning in llms and their application to rhyme generation and question answering, 2026. URL https://arxiv.org/abs/2601.20164.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models, 2025. URL https://arxiv.org/abs/2403.19647.

Nainani, J., Vaidyanathan, S., Watts, C., Assis, A. N., and Rigg, A. Detecting and characterizing planning in language models, 2025. URL https://arxiv.org/abs/2508.18098.

Paulo, G., Mallen, A., Juang, C., and Belrose, N. Automatically interpreting millions of features in large language models, 2025. URL https://arxiv.org/abs/2410.13928.

Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., Nathan, A., Luo, A., Helyar, A., Madry, A., Efremov, A., Spyra, A., Baker-Whitcomb, A., Beutel, A., Karpenko, A., Makelov, A., Neitz, A., Wei, A., Barr, A., Kirchmeyer, A., Ivanov, A., Christakis, A., Gillespie, A., Tam, A., Bennett, A., Wan, A., Huang, A., Sandjideh, A. M., Yang, A., Kumar, A., Saraiva, A., Vallone, A., Gheorghe, A., Garcia, A. G., Braunstein, A., Liu, A., Schmidt, A., Mereskin, A., Mishchenko, A., Applebaum, A., Rogerson, A., Rajan, A., Wei, A., Kotha, A., Srivastava, A., Agrawal, A., Vijayvergiya, A., Tyra, A., Nair, A., Nayak, A., Eggers, B., Ji, B., Hoover, B., Chen, B., Chen, B., Barak, B., Minaiev, B., Hao, B., Baker, B., Lightcap, B., McKinzie, B., Wang, B., Quinn, B., Fioca, B., Hsu, B., Yang, B., Yu, B., Zhang, B., Brenner, B., Zetino, C. R., Raymond, C., Lugaresi, C., Paz, C., Hudson, C., Whitney, C., Li, C., Chen, C., Cole, C., Voss,

C., Ding, C., Shen, C., Huang, C., Colby, C., Hallacy, C., Koch, C., Lu, C., Kaplan, C., Kim, C., Minott-Henriques, C., Frey, C., Yu, C., Czarnecki, C., Reid, C., Wei, C., Decareaux, C., Scheau, C., Zhang, C., Forbes, C., Tang, D., Goldberg, D., Roberts, D., Palmie, D., Kappler, D., Levine, D., Wright, D., Leo, D., Lin, D., Robinson, D., Grabb, D., Chen, D., Lim, D., Salama, D., Bhattacharjee, D., Tsipras, D., Li, D., Yu, D., Strouse, D., Williams, D., Hunn, D., Bayes, E., Arbus, E., Akyurek, E., Le, E. Y., Widmann, E., Yani, E., Proehl, E., Sert, E., Cheung, E., Schwartz, E., Han, E., Jiang, E., Mitchell, E., Sigler, E., Wallace, E., Ritter, E., Kavanaugh, E., Mays, E., Nikishin, E., Li, F., Such, F. P., de Avila Belbute Peres, F., Raso, F., Bekerman, F., Tsimpourlas, F., Chantzis, F., Song, F., Zhang, F., Raila, G., McGrath, G., Briggs, G., Yang, G., Parascandolo, G., Chabot, G., Kim, G., Zhao, G., Valiant, G., Leclerc, G., Salman, H., Wang, H., Sheng, H., Jiang, H., Wang, H., Jin, H., Sikchi, H., Schmidt, H., Aspegren, H., Chen, H., Qiu, H., Lightman, H., Covert, I., Kivlichan, I., Silber, I., Sohl, I., Hammoud, I., Clavera, I., Lan, I., Akkaya, I., Kostrikov, I., Kofman, I., Etinger, I., Singal, I., Hehir, J., Huh, J., Pan, J., Wilczynski, J., Pachocki, J., Lee, J., Quinn, J., Kiros, J., Kalra, J., Samaroo, J., Wang, J., Wolfe, J., Chen, J., Wang, J., Harb, J., Han, J., Wang, J., Zhao, J., Chen, J., Yang, J., Tworek, J., Chand, J., Landon, J., Liang, J., Lin, J., Liu, J., Wang, J., Tang, J., Yin, J., Jang, J., Morris, J., Flynn, J., Ferstad, J., Heidecke, J., Fishbein, J., Hallman, J., Grant, J., Chien, J., Gordon, J., Park, J., Liss, J., Kraaijeveld, J., Guay, J., Mo, J., Lawson, J., McGrath, J., Vendrow, J., Jiao, J., Lee, J., Steele, J., Wang, J., Mao, J., Chen, K., Hayashi, K., Xiao, K., Salahi, K., Wu, K., Sekhri, K., Sharma, K., Singhal, K., Li, K., Nguyen, K., Gu-Lemberg, K., King, K., Liu, K., Stone, K., Yu, K., Ying, K., Georgiev, K., Lim, K., Tirumala, K., Miller, K., Ahmad, L., Lv, L., Clare, L., Fauconnet, L., Itow, L., Yang, L., Romaniuk, L., Anise, L., Byron, L., Pathak, L., Maksin, L., Lo, L., Ho, L., Jing, L., Wu, L., Xiong, L., Mamitsuka, L., Yang, L., McCallum, L., Held, L., Bourgeois, L., Engstrom, L., Kuhn, L., Feuvrier, L., Zhang, L., Switzer, L., Kondraciuk, L., Kaiser, L., Joglekar, M., Singh, M., Shah, M., Stratta, M., Williams, M., Chen, M., Sun, M., Cayton, M., Li, M., Zhang, M., Aljubeh, M., Nichols, M., Haines, M., Schwarzer, M., Gupta, M., Shah, M., Guan, M. Y., Huang, M., Dong, M., Wang, M., Glaese, M., Carroll, M., Lampe, M., Malek, M., Sharman, M., Zhang, M., Wang, M., Pokrass, M., Florian, M., Pavlov, M., Wang, M., Chen, M., Wang, M., Feng, M., Bavarian, M., Lin, M., Abdool, M., Rohaninejad, M., Soto, N., Staudacher, N., LaFontaine, N., Marwell, N., Liu, N., Preston, N., Turley, N., Ansman, N., Blades, N., Pancha, N., Mikhaylin, N., Felix, N., Handa, N., Rai, N., Keskar, N., Brown, N., Nachum, O., Boiko, O., Murk, O., Watkins, O., Gleeson, O., Mishkin, P., Lesiewicz, P., Baltescu, P., Belov, P., Zhokhov, P., Pronin, P., Guo, P., Thacker, P., Liu, Q., Yuan, Q., Liu, Q., Dias, R., Puckett, R., Arora, R., Mullapudi, R. T., Gaon, R., Miyara, R., Song, R., Aggarwal, R., Marsan, R., Yemiru, R., Xiong, R., Kshirsagar, R., Nuttall, R., Tsiupa, R., Eldan, R., Wang, R., James, R., Ziv, R., Shu, R., Nigmatullin, R., Jain, S., Talaie, S., Altman, S., Arnesen, S., Toizer, S., Toyer, S., Miserendino, S., Agarwal, S., Yoo, S., Heon, S., Ethersmith, S., Grove, S., Taylor, S., Bubeck, S., Banesiu, S., Amdo, S., Zhao, S., Wu, S., Santurkar, S., Zhao, S., Chaudhuri, S. R., Krishnaswamy, S., Shuaiqi, Xia, Cheng, S., Anadkat, S., Fishman, S. P., Tobin, S., Fu, S., Jain, S., Mei, S., Egoian, S., Kim, S., Golden, S., Mah, S., Lin, S., Imm, S., Sharpe, S., Yadlowsky, S., Choudhry, S., Eum, S., Sanjeev, S., Khan, T., Stramer, T., Wang, T., Xin, T., Gogineni, T., Christianson, T., Sanders, T., Patwardhan, T., Degry, T., Shadwell, T., Fu, T., Gao, T., Garipov, T., Sriskandarajah, T., Sherbakov, T., Korbak, T., Kaftan, T., Hiratsuka, T., Wang, T., Song, T., Zhao, T., Peterson, T., Kharitonov, V., Chernova, V., Kosaraju, V., Kuo, V., Pong, V., Verma, V., Petrov, V., Jiang, W., Zhang, W., Zhou, W., Xie, W., Zhan, W., McCabe, W., DePue, W., Ellsworth, W., Bain, W., Thompson, W., Chen, X., Qi, X., Xiang, X., Shi, X., Dubois, Y., Yu, Y., Khakbaz, Y., Wu, Y., Qian, Y., Lee, Y. T., Chen, Y., Zhang, Y., Xiong, Y., Tian, Y., Cha, Y., Bai, Y., Yang, Y., Yuan, Y., Li, Y., Zhang, Y., Yang, Y., Jin, Y., Jiang, Y., Wang, Y., Wang, Y., Liu, Y., Stubenvoll, Z., Dou, Z., Wu, Z., and Wang, Z. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., Ferret, J., Liu, P., Tafti, P., Friesen, A., Casbon, M., Ramos, S., Kumar, R., Lan, C. L., Jerome, S., Tsitsulin, A., Vieillard, N., Stanczyk, P., Girgin, S., Momchev, N., Hoffman, M., Thakoor, S., Grill, J.-B., Neyshabur, B., Bachem, O., Walton, A., Severyn, A., Parrish, A., Ahmad, A., Hutchison, A., Abdagic, A., Carl, A., Shen, A., Brock, A., Coenen, A., Laforge, A., Paterson, A., Bastian, B., Piot, B., Wu, B., Royal, B., Chen, C., Kumar, C., Perry, C., Welty, C., Choquette-Choo, C. A., Sinopalnikov, D., Weinberger, D., Vijaykumar, D., Rogozińska, D., Herbison, D., Bandy, E., Wang, E., Noland, E., Moreira, E., Senter, E., Eltyshev, E., Visin, F., Rasskin, G., Wei, G., Cameron, G., Martins, G., Hashemi, H., Klimczak-Plucińska, H., Batra, H., Dhand, H., Nardini, I., Mein, J., Zhou, J., Svensson, J., Stanway, J., Chan, J., Zhou, J. P., Carrasqueira, J., Iljazi, J., Becker, J., Fernandez, J., van Amersfoort, J., Gordon, J., Lipschultz, J., Newlan, J., yeong Ji, J., Mohamed, K., Badola, K., Black, K., Millican, K., McDonell, K., Nguyen, K., Sodhia, K., Greene, K., Sjoesund, L. L., Usui, L., Sifre, L., Heuermann, L., Lago, L., McNealus, L., Soares, L. B., Kilpatrick, L., Dixon, L., Martins, L., Reid, M., Singh, M., Iverson,

M., Görner, M., Velloso, M., Wirth, M., Davidow, M., Miller, M., Rahtz, M., Watson, M., Risdal, M., Kazemi, M., Moynihan, M., Zhang, M., Kahng, M., Park, M., Rahman, M., Khatwani, M., Dao, N., Bardoliwalla, N., Devanathan, N., Dumai, N., Chauhan, N., Wahltinez, O., Botarda, P., Barnes, P., Barham, P., Michel, P., Jin, P., Georgiev, P., Culliton, P., Kuppala, P., Comanescu, R., Merhej, R., Jana, R., Rokni, R. A., Agarwal, R., Mullins, R., Saadat, S., Carthy, S. M., Cogan, S., Perrin, S., Arnold, S. M. R., Krause, S., Dai, S., Garg, S., Sheth, S., Ronstrom, S., Chan, S., Jordan, T., Yu, T., Eccles, T., Hennigan, T., Kocisky, T., Doshi, T., Jain, V., Yadav, V., Meshram, V., Dharmadhikari, V., Barkley, W., Wei, W., Ye, W., Han, W., Kwon, W., Xu, X., Shen, Z., Gong, Z., Wei, Z., Cotruta, V., Kirk, P., Rao, A., Giang, M., Peran, L., Warkentin, T., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Sculley, D., Banks, J., Dragan, A., Petrov, S., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Borgeaud, S., Fiedel, N., Joulin, A., Kenealy, K., Dadashi, R., and Andreev, A. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

A. Full Prompt Templates

A.1. Feature Description Prompt

Feature description annotation (default)

```
You are a mechanistic interpretability researcher. You will be given evidence
about a single feature neuron. Your task is to produce a label and brief
description for this feature.

You will receive three types of evidence:
1. Overall Prompt Context: the original prompt the model was processing.
2. Input Activations: text excerpts where the neuron activated strongly. The most
relevant tokens are delimited by <<<>>>.
3. Global Output Tokens: tokens this neuron tends to push toward or away from in
the output.

Use input activations as the primary evidence. Use prompt context only for
disambiguation, not as proof by itself. Output tokens can be noisy --only factor
them in when they show a clear, consistent pattern. If they are consistent, they
likely reveal a lot of information. A tight cluster of specific promoted tokens
(e.g. one city, one state) outranks a broader category label --prefer the specific
entity.

STYLE: Write in short, direct fragments --not full sentences. Get to the point
immediately. No filler, no hedging, no grammatical padding.

FEATURE TYPES --use this to guide your description style:
Features tend to fall into three types. Figure out which one fits, then describe
accordingly.

1. Input features --activate on a specific token or category of tokens.
  Describe what they activate on: 'represents X' or just name the pattern directly.
  If they activate on a range of related things, describe the category.

2. Output features --consistently promote a specific next token or category.
  Label as 'say X' when a clear next-token pattern exists. Reference definitions
  below. Prepositions may fall under this category, where the important words are
  subsequent to the thing it is referencing.
3. Abstract/middle features --neither cleanly input nor output.
  Describe the context pattern: what kind of text, what situation, what role it
  plays. These often need the surrounding context of activations, not just the
  highlighted token.

'SAY X' vs 'X ITSELF':
Features can represent a concept directly, or signal that a concept is about to
appear (activating on structural words right before it --prepositions, articles,
punctuation).
- Highlighted tokens are content words →SHORT_LABEL is the concept itself.
- Highlighted tokens are structural words setting up content →SHORT_LABEL is 'say
[what]'.
- Check what follows the trigger across activations: if a specific concept X (e.g.
a proper noun, a method name) CONSISTENTLY appears right after the trigger token,
that supports 'say X'. When unclear, prefer naming the concept directly 'X'
without the 'say' --'say X' is a stronger claim and needs consistent evidence and
should not be used lightly: be strict about including 'say' in any feature.

PROPER NOUNS:
If a specific name, place, or entity recurs across the activations --even in a
minority of them --include it in the SHORT_LABEL or elaboration. Don't collapse to
a generic label when a specific one is clearly supported. Beyond highlighted
triggers, also consider consistently occurring proper nouns. These are a signal of
specificity, not noise. If highlighted tokens vary widely and the feature looks
```

```
polysemantic, capture the consistently specific entities that recur across
excerpts when clear and possible rather than defaulting to a single broad label.

AVOID:
- Linguistic or technical jargon: copula, lemma, morpheme, orthogonal, syntactic,
prepositional phrase, noun phrase, etc. Prefer layman's vocabulary and casual tone.
- Broad labels when something more specific is clearly supported.
- Full sentences. Filler (grammar is not required).

OUTPUT FORMAT: SHORT_LABEL --elaboration

- SHORT_LABEL: 1-5 words. Natural graph node name --specific over generic.
- After ' --': 1-2 tight fragments. Add context, what it promotes, or consistent
subpatterns. Skip if the label already says it all.
- Total: 10-35 words.

Return only the formatted line, nothing else.
```

A.2. Supernode Grouping Prompts

Design principles. The grouping prompts encode several principles that shape the resulting supernodes: features weakly related to the prompt or output are sent to UNGROUPED rather than forced into groups; specific labels (especially recurring proper nouns) are preferred over broad categories; promotion features ("say X") are kept separate from concept features ("X" itself), and "suppress X" is kept separate from both; grammatical or structural patterns default to UNGROUPED unless central to the prompt; and supernode names are constrained to layman's vocabulary, under five words, and at most one or two concepts. These are operationalized across the three phases below.

All three phases use GPT-5 mini. Phase 1 clusters the top `GROUPING_TOP_K_SEED=50` most-influential features; Phase 2 assigns the remaining features in batches of `GROUPING_BATCH_SIZE=50`; Phase 3 refines. Each phase prompt inlines the same `GROUPING_PHILOSOPHY` block (shown once below) and a `SPECIFICITY_GUIDANCE` block (default variant `a2`, also shown once below); alternative variants of the bias block are supported in code. To avoid repetition, the per-phase prompt boxes render those shared blocks as `{GROUPING_PHILOSOPHY}` and `{SPECIFICITY_GUIDANCE}` placeholders.

Runtime-substituted placeholders in the text below: `{prompt_text}` is the prompt under analysis; `{output_context}` is a line of the form "Model's top predicted outputs: ..."; `{groups_context}` is a JSON dict of current groups and rationales (Phase 2 only); `{group_summary}` is a breakdown of current groups with member features and descriptions (Phase 3 only); `{features}` is a list of lines of the form "ID: <feature_id> | Desc: <description>".

Shared `GROUPING_PHILOSOPHY` block

```
GOAL: Produce a cohesive attribution graph that highlights the main intent and
meaning of the prompt.


RELEVANCE & UNGROUPED:
- Assign to "Ungrouped": weak, isolated, or noisy features; features not
meaningfully connected to the main prompt and output semantics.
- Purely grammatical tokens (prepositions, articles, conjunctions, punctuation,
copulas) go to "Ungrouped" unless they clearly promote a semantically meaningful
role (this is very rare). Sentence structure-level grammar is mostly pointless,
unless the prompt or output is about it.
- "say X" groups are ONLY valid when X is a meaningful content word or category
(e.g., "say a color", "say a fruit"). Do NOT create "say X" groups when X is a
function word, grammatical structure, or syntactic role --e.g., "say 'is'", "say a
relative clause", "say 'of'", "say a preposition" are never valid groups. These
belong in Ungrouped.
- A valid group should feel connected to at least one other group in the graph,
```

```
not like an isolated curiosity.
- "Ungrouped" is not a failure.


GRANULARITY & SPECIFICITY:
- Create only groups clearly supported by the data. Prefer the most specific name
the evidence supports over broad buckets.
- Preserve meaningful distinctions in abstraction level when relevant to the
prompt --don't blindly merge a broad category with a narrower stable subtype.
- Distinctions that explain WHY the model chose one output over another should be
preserved.
- The prompt commits to one active sense of every word in it. Features from
alternate senses of a word (senses the prompt does not require) may form their own
groups --Phase 3 will consolidate them.


SEMANTIC ROLE --"SAY X" vs "X ITSELF" (high-priority rule):
- A feature that promotes a concept is different from a feature that IS the
concept. Keep them in separate groups.
  - Highlighted tokens are function/structural words →the feature sets up what
  follows; name it "say [what]".
  - Highlighted tokens are content words →the feature represents the concept
  directly; name the concept.
- say tags in descriptions are strong signals --respect them.
- "Say" is for genuine promoting. Do not use it when features directly represent a
concept.
- Only use 'say' in a group name when member descriptions themselves use it. Do
not add 'say' by inference.
- Surface overlap is never enough reason to merge groups with different semantic
roles.


NAMING (STRICT):
- LIMIT: 5 words maximum. No exceptions.
- Natural, simple phrasing. If you need more than 5 words, the name is too
specific.
- Before naming a group, read the member descriptions. If a specific named entity
(place, person, concept) recurs across them, use that name --do not default to a
generic category when the descriptions clearly point to something specific.
- This applies to say-X groups too: "say California" is better than "say a place"
when descriptions consistently name California.
- Avoid parentheses and words like "mention", "reference", "entity", "concept",
"topic", or "pattern" when a simpler phrase works.
- Prefer layman's vocabulary.


SPLITTING vs MERGING:
- Split when a group mixes semantic roles or abstraction levels.
- Small groups are fine if they are interpretable and prompt-relevant.
```

Shared `SPECIFICITY_GUIDANCE` block (default variant `a2`)

```
SPECIFICITY GUIDANCE:
When in doubt between narrower and broader groups, use whichever granularity best
explains the model's specific output and prompt. Consider both the prompt and the
predicted output together: a distinction is worth keeping only if it is relevant
to what was asked and (or) what the model predicted.


BORDERLINE FEATURES: prefer assigning a borderline feature to a plausible existing
group over Ungrouped --reserve Ungrouped for features with no meaningful
```

```
connection to the prompt or output.


MERGE CONSTRAINT: when two same-role groups tell the same part of the story and
their separation does not help a reader understand the reasoning differently,
merge them. Never merge a say-X group with a concept group --promotion and content
always stay separate. Prefer keeping more proper noun specificity when relevant to
the prompt and output.


DESCRIPTION-AWARE NAMING: before naming any group, scan the member descriptions
for recurring proper nouns or specific named entities. If a specific entity (a
place, person, concept) appears consistently across descriptions, use that
specific name --do not collapse to a generic category like 'a place' or 'a city'
when the descriptions clearly name something specific. Apply the same rule to
say-X groups: if descriptions consistently name a specific entity after the
trigger, prefer 'say California' over 'say a place'. If features clearly relate to
an alternate sense of a key prompt word, name the group with a sense qualifier
(e.g. 'X (general)') rather than a domain label that is not applicable given the
prompt and output context (e.g. 'economic X' or 'X (music)'). The last few words
of a description often carry non-trivial specificity --use them as an additional
signal when placing features into groups. Group names must come from member
descriptions, not from the prompt and output context. Be faithful to feature
descriptions in naming. Treat say-X / X-itself separation as a hard constraint.
Every say-X group must contain only promoting features; every concept group must
contain only content features. A single misplaced feature is enough to split or
reassign.
```

Phase 1 (Discover): propose candidate supernode labels

```
You are an expert AI interpretability researcher analyzing internal
representations of a large language model.
Context: The model was given the following prompt: {prompt_text}


{output_context}


Below are the 50 most influential features that activated during this prompt.
Cluster them into meaningful semantic groups ("supernodes").


{GROUPING_PHILOSOPHY}


Additional guidance for this phase:
- A single feature may form its own group only if it reflects a stable, reusable
semantic pattern, not a one-off surface detail.
- Prefer names that make the graph easy to read over taxonomically tidy labels.
- Prefer two narrow groups over one vague bucket --Phase 3 later can merge, but
cannot recover lost distinctions easily. When a concept is clearly relevant to the
prompt or output, err toward creating a group rather than Ungrouped.
- HARD RULE --do NOT create groups for grammatical or structural patterns under
any circumstances. Assign those features directly to Ungrouped. This includes:
prepositions and locational connectors (say 'of', say 'in', say 'after', say
locational preposition), copulas and predicate framing (say 'is', predicate
framing, copula, say noun after copula), sentence-completion or next-token
patterns (say completion, say next noun), subword or token-prefix fragments,
heading markers, where-clause framing, structural/relational patterns derived from
words in the prompt itself (containment verbs, prepositional structures, syntactic
connectors), typographic or capitalization patterns (title case, capitalized
```

```
tokens, proper noun formatting), and word-onset or prefix fragments --these
describe token shape, not meaning. The test: does this group name a semantic
concept, or does it describe a syntactic role or sentence structure? Concept =
valid group. Sentence structure = Ungrouped.
- Do not create groups named for prompt format or input structure (e.g. "fact
prompt", "fill-in-the-blank") --these describe the wrapper, not the reasoning
content. Assign to Ungrouped.


{SPECIFICITY_GUIDANCE}


Features:
{features}
```

Phase 2 (Assign): batch assignment

```
You are an expert AI interpretability researcher analyzing internal
representations of a large language model.
Context: The model was given the prompt: {prompt_text}


{output_context}


Current groups and rationales:
{groups_context}


{GROUPING_PHILOSOPHY}


Task: Assign each feature below to the best matching existing group.
These are lower-influence features --they rarely promote meaningfully new semantic
concepts beyond what Phase 1 already captured. Default to an existing group or
"Ungrouped".
Do not force a match: if no group fits clearly, "Ungrouped" is correct.
Only create a new group if the concept is genuinely absent from the existing
groups, clearly relevant to the prompt, and specific enough that multiple features
would share it --this should be rare.


{SPECIFICITY_GUIDANCE}


Features:
{features}
```

Phase 3 (Refine): merge and prune

```
You are an expert AI interpretability researcher reviewing the output of an
automated feature grouping pipeline.


Context: The model was given the prompt: {prompt_text}


{output_context}
```

```
The pipeline produced {num_groups} groups from {num_features} features. Your job
is to clean up the result --rename unclear groups, reassign misplaced features,
and drop irrelevant groups as defined below.


{GROUPING_PHILOSOPHY}


Your job is limited to five things only:


1. GRAMMAR KILL: Any group whose name describes a syntactic role, sentence
structure, token pattern, word-prefix fragment-- move its members to Ungrouped and
dissolve it. Examples: "containment verb", "prefix 'ill'", "say location after
of", "fill-in-the-blank", "[concept] prefix", "[X] relation", "[X] clause",
"location clause". The test: does this name a concept or describe sentence
structure / token shape? Structure/shape = dissolve. For borderline say-X groups,
judge by X: if X names a concept relevant to the prompt or output reasoning chain,
keep the group --the "say" promoting does not make it irrelevant. Exception: if a
word is clearly semantic and central to the prompt's reasoning chain, judge by its
role in context rather than its word class alone.


2. ALTERNATE SENSE: A word has an alternate sense when it shares a surface form
with the relevant concept but means something different given this prompt --this
includes any domain (financial, architectural, political, etc.) that the prompt
does not require. Example: "notes (music)" or "musical notes" when the prompt asks
about note-taking. If alternate-sense groups are present, merge them together into
a single fallback group named "[concept] (general)" --do not touch the
correct-sense group. Do not split the correct-sense group to create a (general)
variant; only create "[concept] (general)" by merging existing alternate-sense
groups. If no alternate-sense groups exist, take no action. This applies to
genuine alternate senses only --do not use this to merge a specific named group
into a broader same-sense group. '[concept] (general)' is strictly for features
activating on a genuinely different dictionary definition (e.g. 'bank' as a
financial institution vs. a riverbank) --not the same concept in different
contexts or positions.


Lastly, suppression-flavored groups ("suppress X", "demote X", "anti-X", "avoid
X", "inhibit X") must never be merged into the concept group "X" --they represent
opposite causal roles. If multiple suppression variants for the same concept X
exist, consolidate them into one group named "suppress X". Always use "suppress X"
as the canonical name.


3. RENAME: Are any group names unclear, longer than 5 words, or use jargon? Rename
for clarity. A rename must not lose specificity, promote a structural name, or
flip a concept group to a say-X group or vice versa. Do not drop intermediate
reasoning steps from a group name.


4. REASSIGN: Are any individual features obviously in the wrong group given their
description and the prompt? Move them. Only reassign with high confidence.


5. RELEVANCE DROP: If a group's concept has no clear connection to the prompt's
reasoning chain or predicted output --it is not a named entity in the prompt or is
not specific and interesting in general, not an intermediate reasoning step, and
not a framing pattern for the output --drop it (members to Ungrouped). Use the
SPECIFICITY GUIDANCE to judge relevance. Exception: keep groups that name a
competing value in the same category as the answer (e.g., a wrong language when
```

```
the prompt asks about a language), and lean towards keeping neighbors or related
topics in the same domain (e.g., neighboring states, nearby countries) --these may
be informative competing signals, not noise.


SPECIFIC →BROAD PROTECTION: Before any merge or rename, check --is one group
semantically more precise than the other (a named entity, specific concept, or
something referenced in the prompt or output)? If yes, protect the specific group.
"say color" must not collapse into "say appearance"; "say school" must not
collapse into "say place name". If the specific group is irrelevant to the
reasoning chain, send it to Ungrouped --never collapse into a vaguer group.


Only make changes you are CONFIDENT about. If the grouping looks good, return
empty lists for all actions.


{SPECIFICITY_GUIDANCE}


Current grouping:
{group_summary}
```

A.3. Screening Judge Prompt

The screening judge is called once per graph with the system prompt below (which lists all dimensions together), followed by the user message template. It scores each graph on five independent dimensions on a 1–10 scale, with a one-sentence justification per score that names the specific supernodes driving the assessment.

Screening judge — system prompt

```
You are a mechanistic interpretability researcher evaluating attribution graphs from a
language model.  You score each graph on several dimensions of interestingness, each
on a 1-10 scale, and give a short justification for each score.
The dimensions are:

1.  Hidden intermediate concept / multi-hop reasoning -- Score high if EITHER of these
holds (they are two faces of the same thing:  internal computation that goes beyond
a single surface lookup).  (a) HIDDEN INTERMEDIATE CONCEPT -- a supernode represents
a concept that is load-bearing for the answer but appears in NEITHER the prompt NOR
the predicted output; the model computes it internally as a stepping stone (Paper:
implicit 'Texas' in Dallas->Austin; the Apollo program behind 'space').  Trivial
restatements of a prompt/output token do not count.  (b) MULTI-HOP REASONING -- the
supernodes form a genuine chain (input concept -> one or more intermediate concepts ->
output), i.e.  composed reasoning steps rather than a single direct association (Paper:
multi-step factual reasoning).  Medium:  one plausible unstated association OR one
clear intermediate step beyond lookup.  High:  a specific, genuinely surprising hidden
concept clearly doing real work, OR a multi-link chain where each hop is a distinct,
necessary concept.
2.  Parallel / competing computation -- The model computes several candidate outputs
or competing hypotheses at once, OR combines independent pathways into the answer (e.g.
a rough estimate pathway plus an exact-lookup pathway).  (Paper:  addition's parallel
approximate+precise routes; planning multiple candidate words.)  Near-synonyms or
reformattings of the same token do NOT count.  Medium:  multiple candidates present
but weakly separated.  High:  clearly distinct competing or complementary pathways.
3.  Abstraction beyond surface tokens -- The most influential supernodes operate
on abstract, semantic, or relational concepts decoupled from the literal surface
tokens -- category-level, language-independent, or role-based features -- rather than
echoing specific prompt words.  (Paper:  language-independent 'meaning' features in
multilingual circuits.)  Medium:  some abstraction beyond the tokens.  High:  core
computation is clearly in an abstract concept space, not in the surface strings.
```

```
4. Unexpected mechanism (headline) -- Taken as a whole, does the internal computation
DIVERGE from what a knowledgeable person would predict just from the prompt and the
predicted output? This is the open-ended 'reveals computation different from what
one might expect' signal. (Paper: chain-of-thought unfaithfulness; mechanisms
that contradict the model's own account.) Score LOW when the supernodes are exactly
the obvious ones a person would guess. Score HIGH when the route to the answer is
structurally different from the naive expectation. Judge the mechanism, not whether
the prompt topic is interesting.
5. Other interesting structure -- A catch-all for notable internal structure not
covered above. If you notice something genuinely worth a researcher's attention,
score it high here and name it specifically.

Scoring guidance (use the full 1-10 range):
1-2 = no evidence of this property
3-4 = weak or ambiguous hint
5-6 = plausible but not strongly supported
7-8 = clear, concrete instance worth noting
9-10 = textbook example, worth a writeup
Score each dimension independently on its own merits. Do NOT inflate scores because
the prompt is interesting; only the supernode structure matters. For each score,
provide a one-sentence justification that names the specific supernodes or features
driving your assessment.
```

Screening judge — user message template

```
Prompt given to the model: "{prompt}"
Model's predicted output: "{predicted}" -- correct answer is: {correct_answer}
Model confidence: {confidence} -- {confidence_band}
Supernode groups ({num_supernode_groups} total):
{groups}
Sample descriptions of the most influential individual nodes (by influence score):
{clerp_samples}
Score this graph on each of the five dimensions. Reply in this exact JSON format --
every dimension is required, with both a score and a one-sentence reason:
{
"hidden_or_multihop": {"score": <integer 1-10>, "reason": "<one sentence naming the
specific supernodes or features>"},
"parallel_or_competing": {"score": <integer 1-10>, "reason": "<one sentence naming
the specific supernodes or features>"},
"abstraction_beyond_tokens": {"score": <integer 1-10>, "reason": "<one sentence
naming the specific supernodes or features>"},
"unexpected_mechanism": {"score": <integer 1-10>, "reason": "<one sentence naming
the specific supernodes or features>"},
"other_interesting_structure": {"score": <integer 1-10>, "reason": "<one sentence
naming the specific supernodes or features>"}
}
```

A.4. Validation Prompts

Each validation protocol uses a per-task prompt built from a template at scoring time. Runtime-substituted placeholders in the templates below: `{group_name}` is the group description being probed; `{description}` is the feature description; `{evidence}` is a feature's raw evidence (input activations and output tokens); `{n_pos}` is the number of correct items embedded among distractors (5 for text detection); `{n}` is the total number of candidates shown; the numbered list (`{i. item}`) is the shuffled candidates.

Feature Detection (1-in-10 feature ID): pick the feature that belongs to the group

```
Group: "{group_name}"

Below are {n} feature descriptions from neurons inside a language model. Exactly 1
of these belongs to the group above.
```

```
Pick the single feature description that best matches the group.

1. {item_1}
2. {item_2}
...
{n}. {item_n}

Respond with the 1-based index of the best match.
```

Text Detection (5-in-10 snippet match): pick the snippets that activated this group

```
Group: "{group_name}"

Below are {n} text excerpts that strongly activated neurons inside a language
model. The key activating tokens are highlighted with <<<>>>. Exactly {n_pos} of
these come from neurons in this group.

Identify which {n_pos} text excerpts belong to this group.

1. {item_1}
2. {item_2}
...
{n}. {item_n}

Respond with the 1-based indices of the {n_pos} excerpts from this group.
```

Description Detection (1-in-10 description ID): pick the description matching this feature's evidence

```
Below is evidence about a single feature neuron in a language model (its input
activations and output tokens).

{evidence}

Below are {n} candidate descriptions. Exactly 1 of these correctly describes this
feature. Pick the single best match.

1. {item_1}
2. {item_2}
...
{n}. {item_n}

Respond with the 1-based index of the description that best matches the feature
evidence above.
```

Description Text Detection (5-in-10 snippet match): pick the snippets that activated this feature

```
Feature description: "{description}"

Below are {n} text excerpts that strongly activated neurons inside a language
model. The key activating tokens are highlighted with <<<>>>. Exactly {n_pos} of
these come from the neuron described above.

Identify which {n_pos} text excerpts match this feature description.

1. {item_1}
2. {item_2}
...
```

```
{n}. {item_n}

Respond with the 1-based indices of the {n_pos} excerpts that activated this
feature.
```

A.5. Feature Description Examples

Representative feature descriptions produced by the pipeline on a single prompt. Each row is one transcoder feature with its description LLM-generated from the top activating snippets. The explanation is from human analysis of these features themselves, looking at the top and bottom logits, as well as text snippets.

Label	Elaboration
Texas (state) — activates on mentions of Texas and Texas place names (cities, counties); tends to push output tokens like “Texas”, “TX”, “Texan”	All text snippets discuss Texas locations, with triggers predominantly being “Texas.”
capital (economics) — activates on the token "capital" in finance/production contexts; fires for loans, capital markets, raising capital, instruments of production.	The description is faithful. Triggers are predominantly the word “capital” with surrounding context discussing “capital” in economic and social contexts.
City / city-list — activates on major city names and list contexts (commas, “and”); consistently promotes city tokens like Seattle, Atlanta, Chicago, London, “cities.”	The feature is polysemantic: triggered on various cities and city names like New York City, Mumbai, Chicago, London, and more.
say capital city — fires on prepositions/articles (in, from, under, the, to) that precede locations; pushes capital/-place names (Washington, Beijing, Canberra, "berra").	The feature is often triggered on preposition tokens while the subsequent word is largely locations or capitals.
say "governor" — activates on state names/“State”/state-official contexts; promotes the token governor/Governor and suppresses federal/national-related tokens	The top promoted tokens are all some form of “governor” or “govern”
Dallas (city) — activates on mentions of Dallas and nearby local proper nouns (e.g., “Dallas Morning News”, local institutions); signals city/location context and local-entity mentions.	Most trigger tokens are “Dallas,” with addition weaker mentions of Texas and Texas counties.

Table 2. Sample feature descriptions from a single attribution graph. “The capital of the state containing Dallas is”

B. Validation Protocol Details

B.1. Feature-level protocols

The two feature-level protocols mirror the supernode-level pair but probe a single feature’s description.

Description Detection (Feature) — specificity; 1-in-10 description ID.
For each feature, present its raw evidence (text snippet activations + trigger tokens) and ask the model to pick the correct description from 10 candidates (9 from other features). Random chance accuracy: 10%.

Text Detection (Feature) — sensitivity; 5-in-10 snippet match.
Given a feature’s description, present 10 text snippets (5 from this feature’s top activations, 5 from other features). The model picks the 5 that activated this feature. Random chance accuracy: 50%.

These scores are per-feature and independent of grouping conditions; they apply identically across all Ours rows of Table 1, so we report only Ours (full) below.

Condition	Desc. Det. (Feature) (↑)	Text Det. (Feature) (↑)
Ours (full)	79.9%	85.4%
Chance	10%	50%

Table 3. **Per-feature description scores.** Description Detection (Feature) and Text Detection (Feature) probe the feature description rather than the supernode description. Scores are constant across the top-k and refine/no-refine variants of Ours, so we report only Ours (full).

B.2. Hop matching

For each graph in the multi-hop experiments, we record (i) whether the model is top-1 correct and (ii) whether the intermediate hop supernode is present. Hop presence is determined by lowercase string and stem matching against supernode labels: e.g., an "Islam" answer matches supernodes named "say Islam" or "Islam location." The same matching logic is reused for intervention and amplification. False positives (e.g. "University" matching with "University of Budapest") are manually confirmed and removed from all results; false negatives (e.g. "Portuguese" vs. "Portugal") are excluded from intervention and amplification numbers, but are recovered manually for Table 2. All results are free of hop matching issues.

B.3. Score Aggregation

Each validation question (Feature Detection or Text Detection) is run with 5 random seeds to stabilize the estimate. For each prompt, we average across its supernodes; we then average across the 15 prompts. This weights each prompt equally regardless of how many supernodes it has.

B.4. Validation Consistency across models

	GPT-5 mini		Gemini 2.5 Flash		Claude Haiku 4.5	
Condition	Feat. Det.	Text Det.	Feat. Det.	Text Det.	Feat. Det.	Text Det.
Random baseline	22.1%	54.7%	22.7%	54.2%	19.7%	50.3%
Human-annotated	66.6%	81.8%	65.7%	81.9%	58.3%	72.0%
Ours (no refine)	92.3%	82.1%	91.8%	80.1%	83.9%	70.5%
Ours (full)	93.4%	83.2%	93.3%	81.3%	84.9%	70.9%
Chance	10% / 50%		10% / 50%		10% / 50%	

Table 4. Feature Detection and Text Detection validation scores across three different judge models on the 15 Neuronpedia Hanna et al. (2025) graphs. Results are consistent across GPT-5 mini and Gemini-2.5-flash; Claude Haiku shows slightly lower absolute scores but preserves roughly the same ordering. Judge bias is not a significant confounder of validation results.

B.5. Validation Consistency across group sizes

Condition	Feature Detection (↑)	Text Detection (↑)	Features grouped	# supernodes	Features/ supernode
Human-annotated	66.6%	81.8%	33.7	6.6	4.91
Ours (top 50, no refine)	82.1%	81.7%	30.3	8.7	3.50
Ours (top 50)	80.8%	81.7%	26.9	7.1	3.98
Ours (top 100, no refine)	82.4%	80.6%	54.9	8.7	6.43
Ours (top 100)	83.1%	82.1%	48.5	7.1	7.08
Ours (top 150, no refine)	81.7%	81.7%	77.1	9.3	8.60
Ours (top 150)	82.8%	84.2%	68.1	7.5	9.43
Ours (top 200, no refine)	83.2%	81.1%	97.6	9.7	10.51
Ours (top 200)	81.9%	81.7%	88.5	8.1	11.62
Ours (no refine)	92.3%	82.1%	250.1	11.1	24.30
Ours (full)	93.4%	83.2%	207.5	8.3	26.42
Random baseline	22.1%	54.7%	–	–	–
Chance	10%	50%	–	–	–

Table 5. **Autointerp scores: full table including pre-refine variants.** Same setup as Table 1. The refine pass contributes a small, consistent improvement on Feature Detection across top-k settings.

	Feature Detection (↑)			Text Detection (↑)		
Condition	$k \geq 2$	$k \geq 3$	$k \geq 4$	$k \geq 2$	$k \geq 3$	$k \geq 4$
Random baseline	22.1%	21.9%	21.4%	54.7%	54.8%	54.9%
Human-annotated	66.6%	66.7%	69.2%	81.8%	82.3%	82.9%
Ours (no refine)	92.3%	91.6%	91.0%	82.1%	80.3%	80.6%
Ours (full)	93.4%	92.2%	92.9%	83.2%	81.8%	81.8%

Table 6. Stability of Feature Detection and Text Detection scores across minimum supernode-size thresholds k on the 15 Neuronpedia graphs from Hanna et al. (2025). The headline numbers in Table 1 ($k \geq 2$) are not an artifact of the threshold choice. Results change minimally across supernode sizes.

C. Dataset Construction

C.1. Replication of Hanna et al. (2025) Reference Graphs

We use the 15 human-annotated reference graphs from the `circuit-tracer` Gemma demo[5] (Hanna et al., 2025), hosted on Neuronpedia (Lin, 2023).

Prompt	Answer
The International Advanced Security Group (IAS	G
Mexico:peso :: Europe:	euro
The guitarist knew the song	. / is
The keys on the cabinet	are
Fact: Michael Jordan plays the sport of	basketball
La saison après le printemps s'apelle l'	été
La estación después de la primavera se llama el	verano
The girl that the teacher sees	is
The girls that the teacher sees	are
Fait: Michael Jordan joue au	basket
Hecho: Michael Jordan juega al	baloncesto
2 + 1 =	3
3 + 5 =	8
Mexico:Spanish :: US:	English
Mexico:peso :: US:	dollar

Table 7. All 15 Hanna et al. (2025) replication prompts.

[5] https://github.com/decoderesearch/circuit-tracer/blob/main/demos/gemma_demo.ipynb

C.2. Capitals

100 prompts following the template "The capital of the {state/country} containing {city} is". The intermediate concept is a US state (50 prompts) or a non-US country (50 prompts).

Prompt	Intermediate	Answer
The capital of the state containing Birmingham is	Alabama	Montgomery
The capital of the state containing Anchorage is	Alaska	Juneau
The capital of the state containing the Grand Canyon is	Arizona	Phoenix
The capital of the state containing Fort Smith is	Arkansas	Little Rock
The capital of the state containing Los Angeles is	California	Sacramento
The capital of the country containing Toronto is	Canada	Ottawa
The capital of the country containing Cancun is	Mexico	Mexico City
The capital of the country containing Manchester is	United Kingdom	London
The capital of the country containing Munich is	Germany	Berlin
The capital of the country containing Mumbai is	India	New Delhi
… (90 more)		

Table 8. Sample of 10 Capitals prompts (5 state-based, 5 country-based). Full 100-prompt list in our code repository.[1]

C.3. Wikipedia

Construction. 1000 mid-sentence completions from random Wikipedia articles (8–15 word prefixes): 500 one-sentence prompts (a single mid-sentence prefix) and 500 context prompts (a complete preceding sentence prepended to the prefix). The number of prompts is a configurable target; the candidate pool we fetch and score scales with that target (the selection yield is only a few percent), so it is not a fixed count. We keep positions where: (1) Gemma-2-2B's top-1 token matches the true next word; (2) the top-1 probability is at most 0.8; (3) the target token is not a stop-word, article, punctuation, or short/non-ASCII subword fragment; and (4) the prompt does not end on a word that syntactically forces a function word next (e.g. "of", "the"). We cap each article at five prompts for diversity. No intermediate concept applies; the target column is the true Wikipedia next token.

Articles are seeded from topic searches across six broad categories, then supplemented with random Wikipedia pages until the candidate pool reaches the target size.

Category	Example seed topics
Science & Technology	History of science, Neuroscience, Astronomy, Mathematics, Computer science, Artificial intelligence, Space exploration, Physics, Chemistry, Biology
History & Civilization	World War II, Ancient Rome, French Revolution, Industrial Revolution, Cold War, Ottoman Empire, British Empire, Chinese dynasties
Arts & Culture	Renaissance art, Jazz music, Classical music, Film history, Literature, Architecture, Impressionism, Opera
Geography & Nature	Amazon rainforest, Solar System, Island nations, Mountain ranges, Desert ecosystems, Major rivers
Social Sciences	Linguistics, Economics, Sociology, Anthropology, Psychology, Political philosophy
Sports & Other	Olympic Games, History of chess, History of sport

Table 9. Seed topic categories used for Wikipedia article sampling.

Prompt prefix	Target token	Conf.
Booth in 1901, Scrooge, or, Marley's Ghost is the earliest film	adaptation	0.399
Some of the best candidates for future deep	space	0.435
John Westergaard's and Henrietta Resler's Class in a capitalist	society	0.352
Researchers in educational neuroscience investigate the neural mechanisms of reading, numerical	cognition	0.270
Australia is not included as it is considered a continental	country	0.395
The Babylonians discovered that lunar eclipses recurred in the saros	cycle	0.389
Beyond AGI, artificial superintelligence (ASI) would outperform the best human abilities across every	domain	0.417
The myth continues that after Heracles completed his twelve labours, he built the Olympic	stadium	0.379
Models in theoretical neuroscience are aimed at capturing the essential	features	0.303
The Cold War also saw a nuclear arms race between the two	superpowers	0.501
… (490 more)		

Table 10. Sample of 10 one-sentence Wikipedia prompts. Full 500-prompt list in our code repository.[1]

Prompt prefix	Target token	Conf.
In his keynote address, printed in the first volume of its new publication, The Anthropological Review, Hunt stressed the work of Waitz, adopting his definitions as a standard. Among the first associates were the young Edward Burnett Tylor, inventor of cultural	anthropology	0.410
In 2016, it was recognized that the first possible evidence of an exoplanet had been noted in 1917. As of 23 April 2026, there are 6,273	confirmed	0.436
Homeric Greek had significant differences in grammar and pronunciation from Classical Attic and other Classical-era dialects. The origins, early form and development of the Hellenic	language	0.400
All depended upon farmers producing an agricultural surplus to support the centralized government, political leaders, religious leaders, and public works of the urban centers of the early civilizations. The earliest signs of a process leading to sedentary culture can be	seen	0.399
Climate change may occur over long and short timescales due to various factors. Recent warming is discussed in terms of global	warming	0.398
The Romance family itself is part of the larger Indo-European family, which includes many other languages native to Europe and South Asia, all believed to have descended from a common ancestor known as Proto-Indo-European. A language family is usually said to contain at least two	languages	0.397
Its native distribution is in southern and western Europe and North Africa. It occurs as a scarce plant in south-west	england	0.405
Therefore, they are able to inhabit many types of geography and topology. Research suggests that these spiders prefer flatter and more	open	0.406
Climate change mitigation actions include conserving energy and replacing fossil fuels with clean energy sources. Secondary mitigation strategies include changes to land use and removing	carbon	0.394
… (490 more)		

Table 11. Sample of 10 context Wikipedia prompts, each a complete sentence followed by an incomplete second sentence. Full 500-prompt list in our code repository.[1]

D. Probing the Grouping Pipeline on Arithmetic

One concern is that supernodes may often be too vague to be informative. We test this on arithmetic. Prior work (Lindsey et al., 2025b) has shown that language models add numbers via two parallel pathways: one computes exact single-digit addition, the other estimates the rough magnitude of the result. We apply our pipeline to ∼200 two- and three-digit addition prompts.

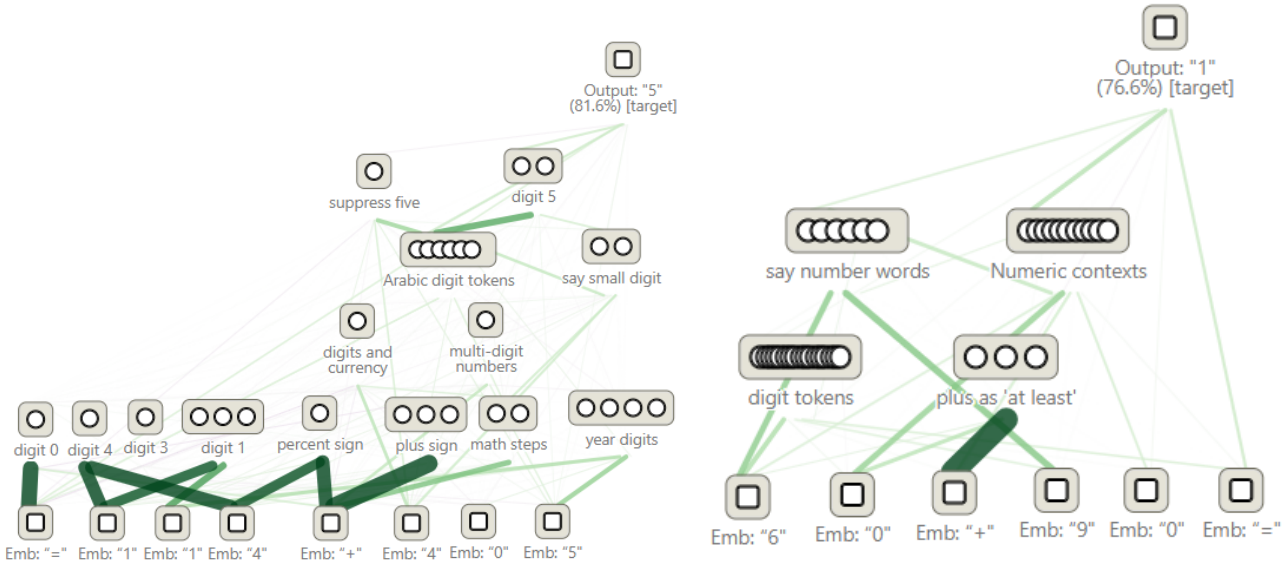


Figure 4. **Two example arithmetic graphs** (left: `114+405`=→"5"; right: `60+90`=→"1"). Supernodes are either surface-token groups (DIGIT 1/4, PLUS SIGN, DIGIT TOKENS) or generic numeric-context groups (YEAR DIGITS, DIGITS AND CURRENCY, PLUS AS 'AT LEAST'), with no carrying, magnitude, or lookup step.

We show two annotated attribution graphs in Figure 4. Qualitatively, the resulting supernodes are either at too low a level — labeling specific digits in the input — or too high a level, with vague labels like "numeric context." The labels are too coarse to reveal the structure that prior work has shown exists at the feature level.

We quantify this with an LLM judge that categorizes each supernode name across all graphs (Table 12). Only 12 of 1885 supernodes (1%) are classified as naming an arithmetic mechanism.

Bucket	Instances	Share	Unique names
surface-token	1228	65%	368
say-number	542	29%	191
mechanism	12	1%	9
other	103	5%	81

Table 12. **Supernode-name taxonomy on the math graphs.** Group names overwhelmingly identify surface tokens or generic number output; almost none name an arithmetic mechanism.

Despite these supernodes being too coarse to reveal this known arithmetic structure, they score well on the autointerp metrics from Section 3 (Table 13). This suggests that autointerp scores measure whether a grouping is internally coherent, not whether it reveals meaningful structure. A natural direction for future work is to tune automatic annotation pipelines to recover this known structure. We hypothesize that our general approach of describing features and then grouping descriptions could still work, with further refinement of the description and grouping steps.

Metric	Mean	Chance
Feature Detection (Supernode)	87%	10%
Text Detection (Supernode)	70%	50%
Description Detection (Feature)	73%	10%
Text Detection (Feature)	81%	50%

Table 13. **Autointerp validation on 20 math graphs.** The groups validate well (they are internally coherent) despite carrying weak mechanistic content.

E. Graph Gallery

E.1. Hanna et al. (2025) Replication: Side-by-side

8 representative prompts from the 15-prompt Hanna et al. (2025) replication set (Appendix C.1).

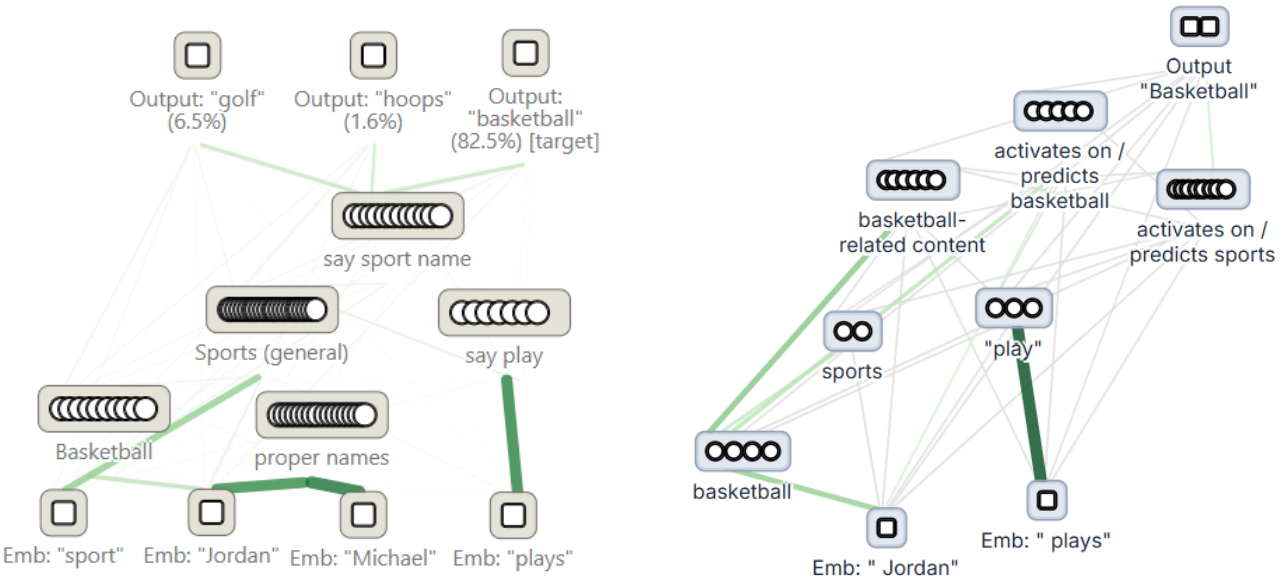


Figure 5. Left: our LLM-grouped supernodes (top 100). Right: Hanna et al. (2025) human-annotated supernodes. Both pipelines surface an explicit Basketball intermediate.

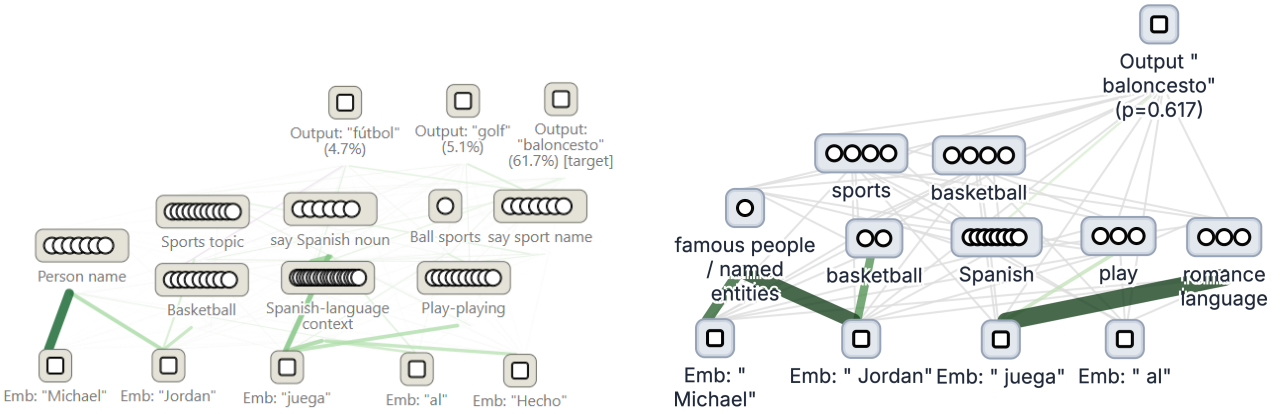


Figure 6. Left: ours (top 100). Right: Hanna et al. (2025) human-annotated supernodes. The Michael Jordan → basketball hop is recovered in both, and on top of it our grouping isolates a Spanish-language context supernode that captures the cross-lingual transfer separately from the sport concept.

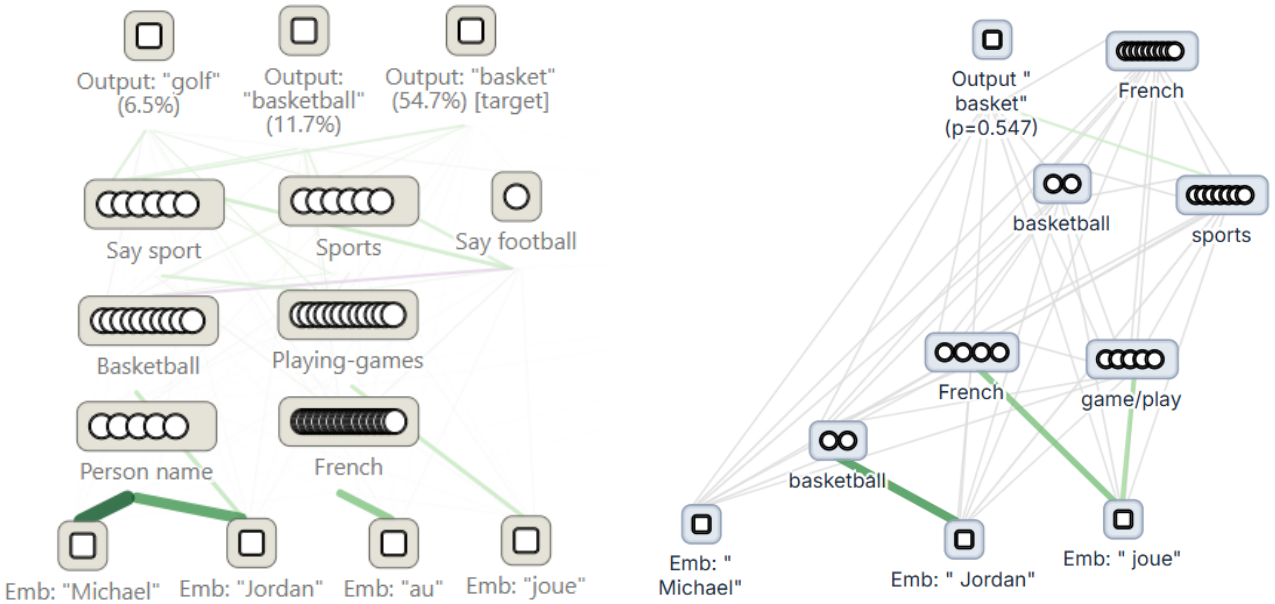


Figure 7. Left: ours (top 100). Right: Hanna et al. (2025) human-annotated supernodes. The same MJ→basketball circuit appears under a French output token. Ours additionally surfaces a Say football group competing with the target output.

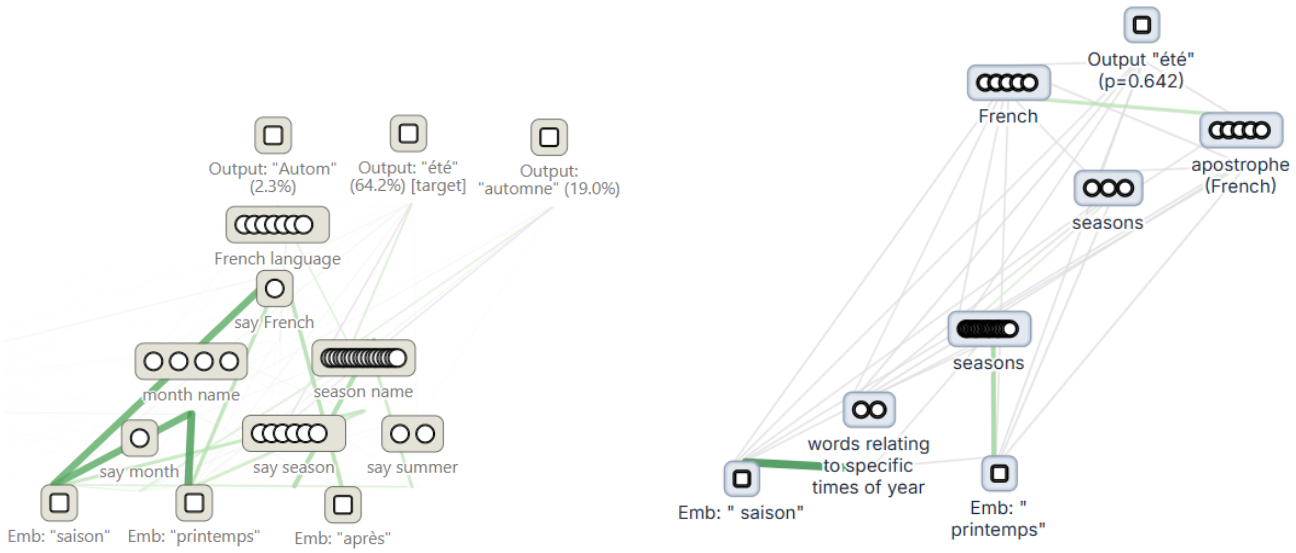


Figure 8. Left: ours (top 100). Right: Hanna et al. (2025) human-annotated supernodes. Our grouping surfaces an explicit say summer supernode that names the intermediate concept; the human grouping uses a more generic seasons / words relating to specific times of year cluster that does not call out the target season.

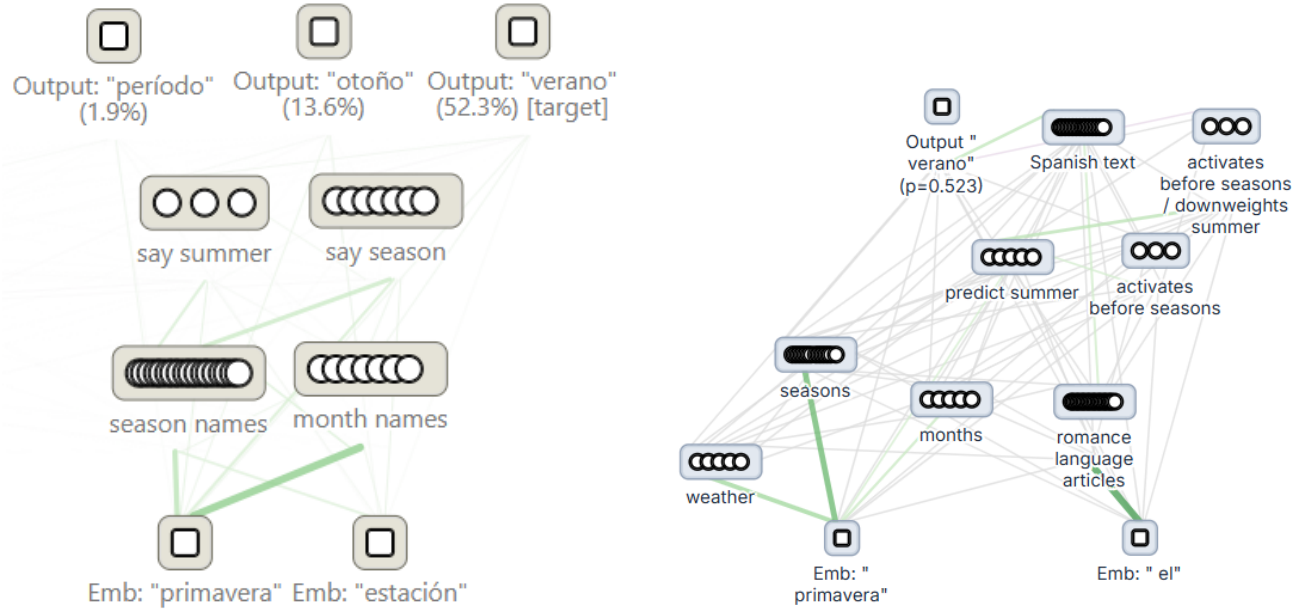


Figure 9. Left: ours (top 100). Right: Hanna et al. (2025) human-annotated supernodes. Both find a predict/say summer intermediate.

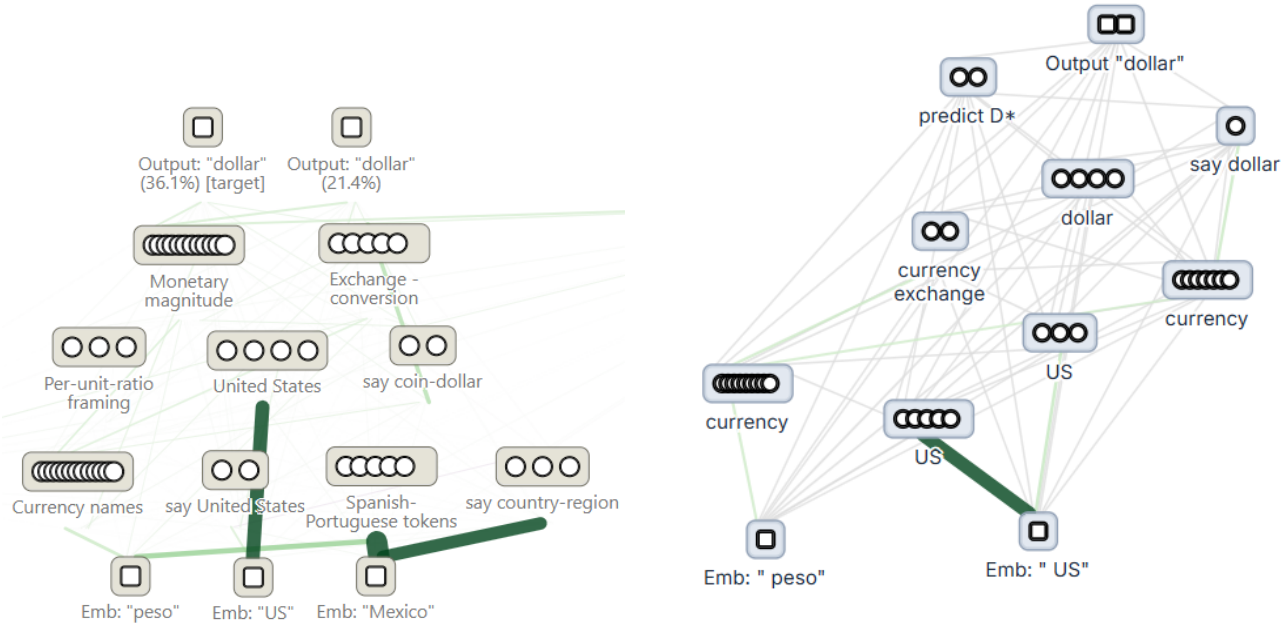


Figure 10. Left: ours (top 100). Right: Hanna et al. (2025) human-annotated supernodes. The two-step analogy is recovered on both sides: a currency supernode encodes the relation, and a US supernode encodes the country mapping that selects "dollar" as its currency.

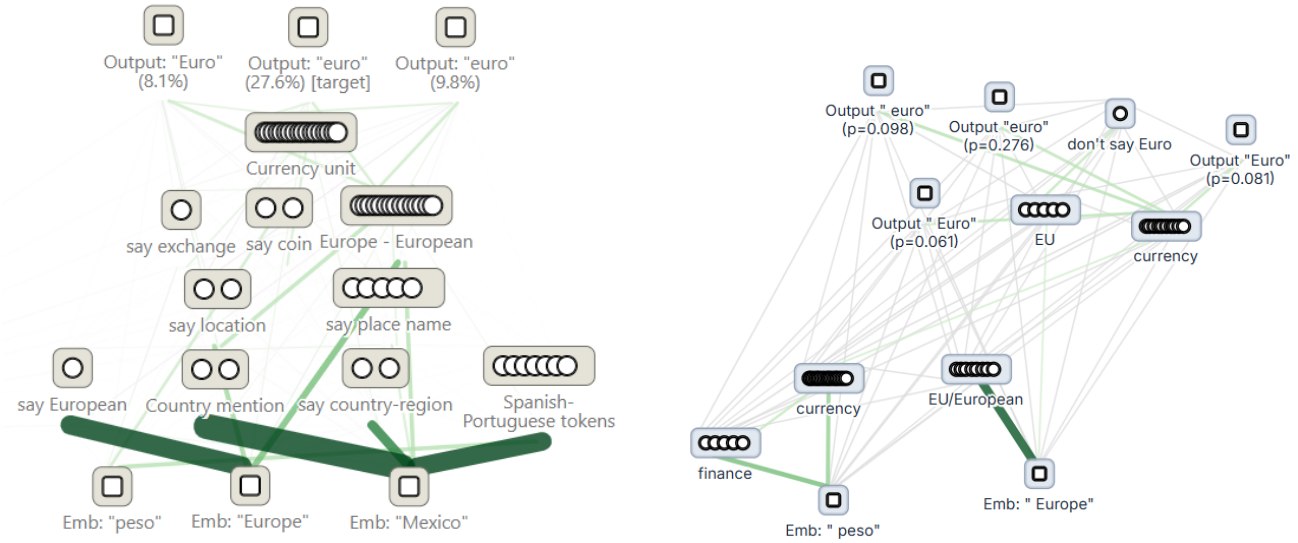


Figure 11. Left: ours (top 100). Right: Hanna et al. (2025) human-annotated supernodes. Currency-analogy structure with the country term swapped to Europe / EU.

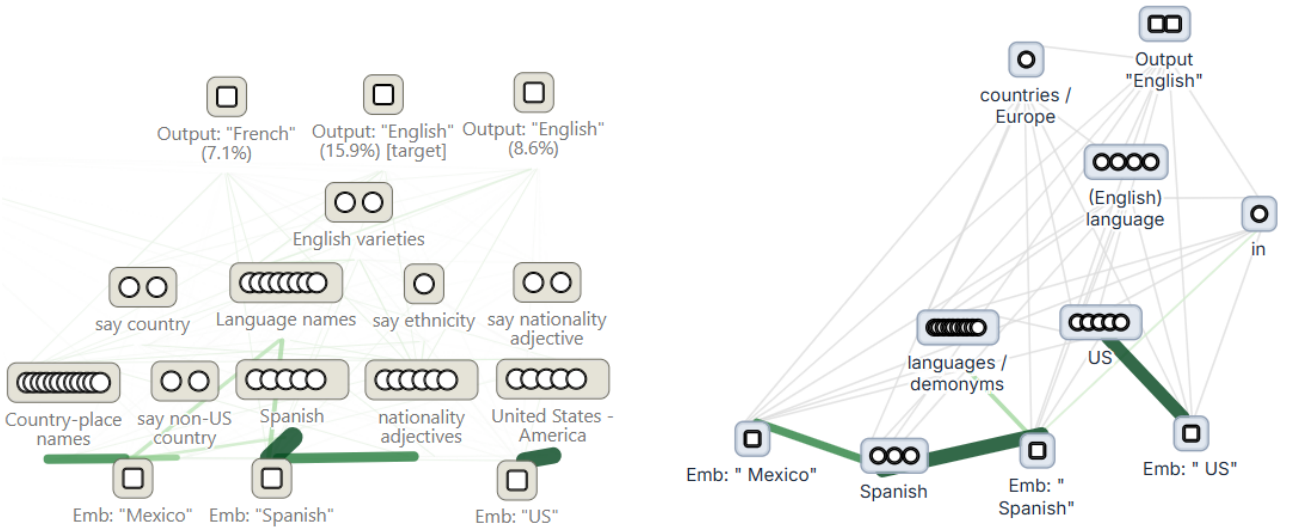
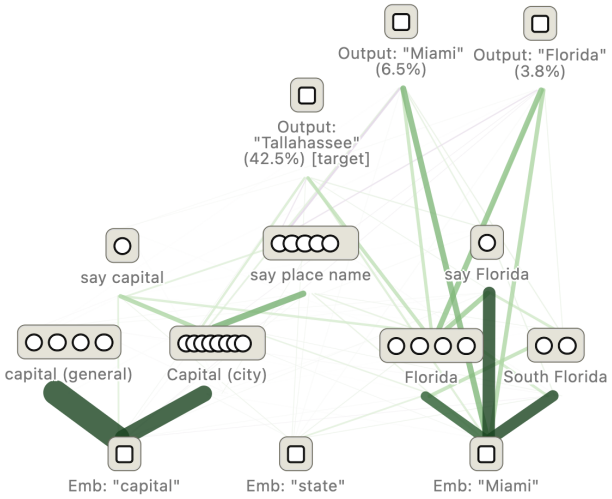


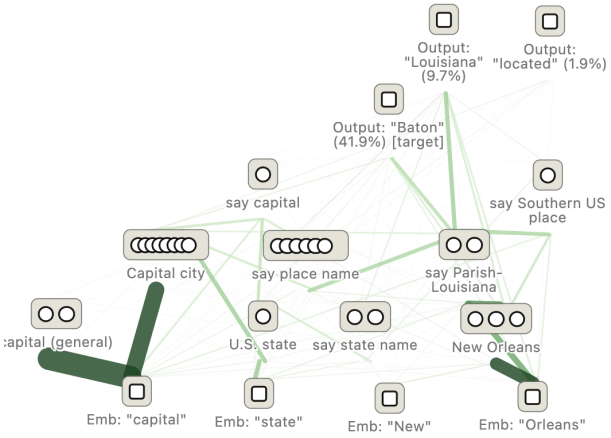
Figure 12. Left: ours (top 100). Right: Hanna et al. (2025) human-annotated supernodes. The same analogy template applied to language rather than currency. Our grouping recovers a fine-grained set of country, nationality adjective, and language name supernodes, while Hanna et al. (2025) consolidates these into broader clusters.

E.2. Capitals: Canonical 2-Hop Examples

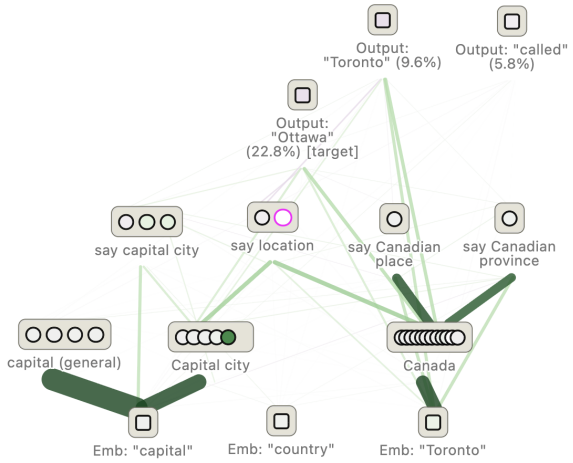
Selected examples from the 100-prompt Capitals dataset annotated by our pipeline.



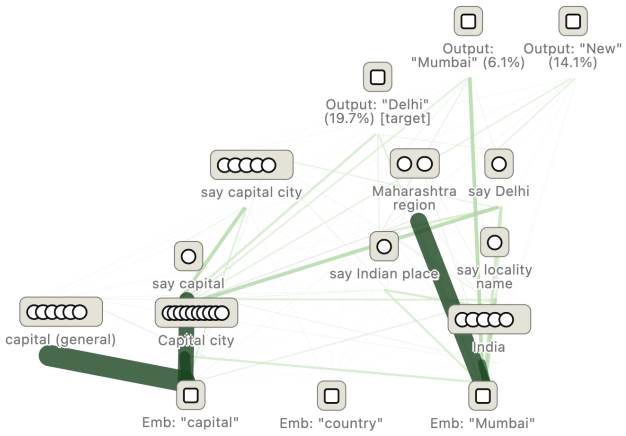
(a) Miami → Florida → *Tallahassee*.



(b) New Orleans → Louisiana → *Baton Rouge*.



(c) Toronto → Canada → *Ottawa*.



(d) Mumbai → India → *New Delhi*.

Figure 13. **Capitals 2-hop graphs.** Each prompt is of the form "The capital of the {state/country} containing {city} is". The intermediate state or country is the expected middle-hop supernode.

F. Automating MLP neurons versus SLT features

Our main results decompose Gemma-2-2B with Gemma Scope single-layer transcoder features. To test the generalization of our automation pipeline, we rerun it on the model's raw MLP neurons and compare the two decompositions on the 100 Capitals prompts.

F.1. Setup

We build our attribution graphs over the model's individual MLP neurons rather than transcoder features using the recent ADAG codebase (Arora et al., 2026a). We then apply the identical description and grouping pipeline (Sections 2.1 and 2.2), with the same GPT-5-mini.

Ensuring fair comparison. MLP neurons are not hosted on Neuronpedia, so we collect activating-text exemplars ourselves, matching the corpus, token budget, and context length used to build Gemma Scope's dashboards. Additionally, with no hosted logits, we compute each neuron's most promoted and suppressed tokens using the same method as the transcoder logits. However, because an MLP neuron's activation is signed (unlike non-negative transcoder features), a neuron's positive- and negative-firing behavior can differ; we describe each separately and label every graph node with the description matching the sign of its activation. Lastly, we keep the top 150 neurons per graph by attribution magnitude, rather than the node-influence threshold used for transcoder graphs (Section 2).

F.2. How much gets grouped

Method	Grouped	Ungrouped	Fraction grouped
MLP neurons	3,205	11,604	21.6%
Transcoder	2,684	1,777	60.2%

Table 14. **Grouped vs. ungrouped features across 100 Capitals graphs.** The pipeline places far fewer MLP neurons into supernodes (21.6%) than transcoder features (60.2%).

Across the 100 Capitals graphs, the pipeline groups a much smaller fraction of MLP neurons into supernodes than transcoder features (21.6% vs. 60.2%; Table 14), suggesting many MLP neurons do not have a clean concept. We use 150 neurons per graph so the pipeline sees a comparable number of features to group as on transcoder graphs, matching the top-150 setting in Table 1: 32.0 features per graph (MLP) vs. 26.8 features per graph (SLT).

F.3. Cross-method comparison

Are there supernodes present in SLT automated graphs but absent in MLP feature descriptions, and vice versa? For each supernode found by one method, we test whether the other method contains any individual feature representing the same concept. We use GPT-5.4 as an LLM-as-a-Judge: concepts with no corresponding feature are reported as missing. A meaningfulness filter first removes generic or grammatical labels (e.g., "capital city," "say place name") so that only specific concepts (e.g., "Texas") are compared. The results in Table 15 show that transcoder features more often find specific entities, whereas MLP neurons, being more polysemantic, more often miss them.

Examples only in transcoder features	Examples only in MLP neurons
European Commission (Antwerp)	Jefferson City (St. Louis)
Karnataka districts (Bangalore)	San Francisco (Los Angeles)
Vermont / Montpelier (Burlington)	Rome (Naples)
Brazil (Cusco)	Macedonia (Thessaloniki)
Kiev–Ukraine (St. Petersburg)	Washington / Olympia (Portland)
Wales (Newport)	South Bend (South Bend)

Table 15. **Representative concepts with supernode in one method but not the other** (prompt in parentheses). Across all 100 graphs, 34 transcoder supernodes have no corresponding MLP neuron, versus only 13 vice versa.

F.4. False-positive control

To bound the judge's false-positive rate on the graphs we could not check, we run the same comparison of MLP supernodes against the *same* MLP neurons re-described in an independent pass, where every supernode should match its own features, so any "missing" is a false positive. The judge recovers 99.5% of supernodes (1 false positive over 196 compared), indicating the cross-method comparison is reliable.

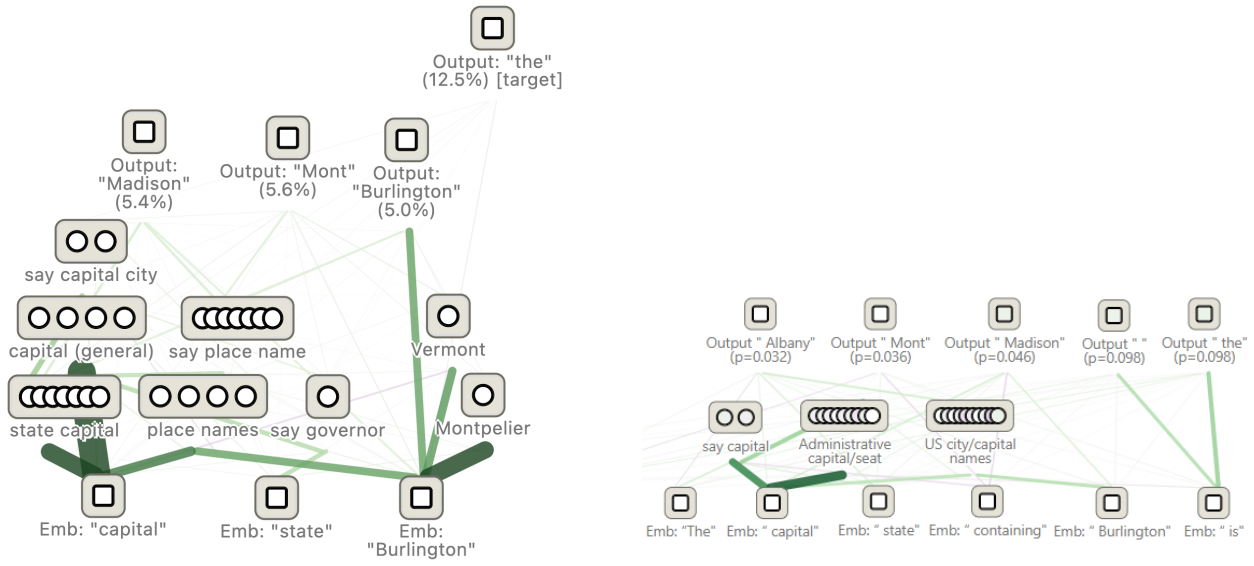


Figure 14. **Intermediate-hop supernode missing for MLP.** "The capital of the state containing Burlington is" → *Montpelier* (intermediate hop: Vermont). **Left:** SLT transcoders. **Right:** MLP neurons. The *Vermont* supernode is missing from the MLP neurons.

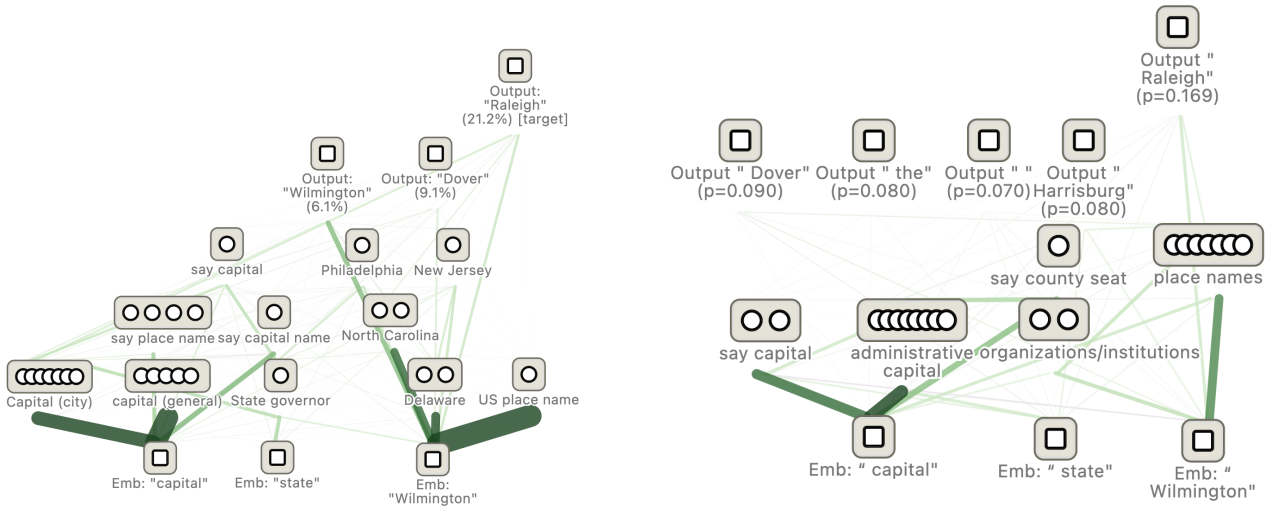


Figure 15. **Neighbor supernodes missing for MLP.** "The capital of the state containing Wilmington is" → *Dover* (intermediate hop: Delaware). **Left:** SLT transcoders. **Right:** MLP neurons. The *Philadelphia* and *New Jersey* supernodes are missing from the MLP features.