Open the Black Box: Transformer Circuit Thread

Yi Hu

2025.5

Table of Contents

- A Mathematical Framework for Transformer Circuits
- Circuit Tracing: Revealing Computational Graphs in Language Models
- On the Biology of a Large Language Model

Transformer Circuits Thread

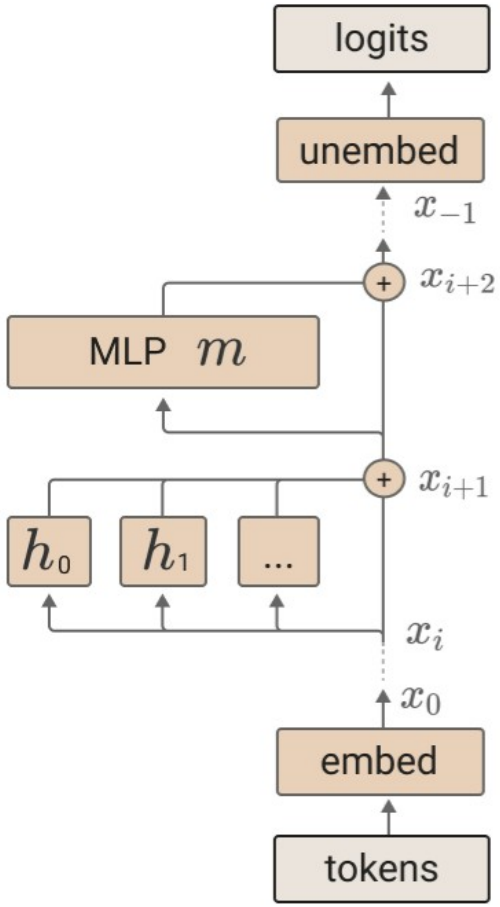
https://transformer-circuits.pub/

A Mathematical Framework for Transformer Circuits

Transformer Overview

- In this paper, for simplicity:
  - we do not consider layer norm and bias

$$W_O^H \begin{bmatrix} r^{h_1} \\ r^{h_2} \\ \dots \end{bmatrix} = \left[W_O^{h_1}, W_O^{h_2}, \dots \right] \cdot \begin{bmatrix} r^{h_1} \\ r^{h_2} \\ \dots \end{bmatrix} = \sum_i W_O^{h_i} r^{h_i}$$



The final logits are produced by applying the unembedding.

$$T(t) = W_U x_{-1}$$

An MLP layer, m, is run and added to the residual stream.

$$x_{i+2} = x_{i+1} + m(x_{i+1})$$

Each attention head, h, is run and added to the residual stream.

$$x_{i+1} = x_i + \sum_{h \in H_i} h(x_i)$$

One residual block

Token embedding.

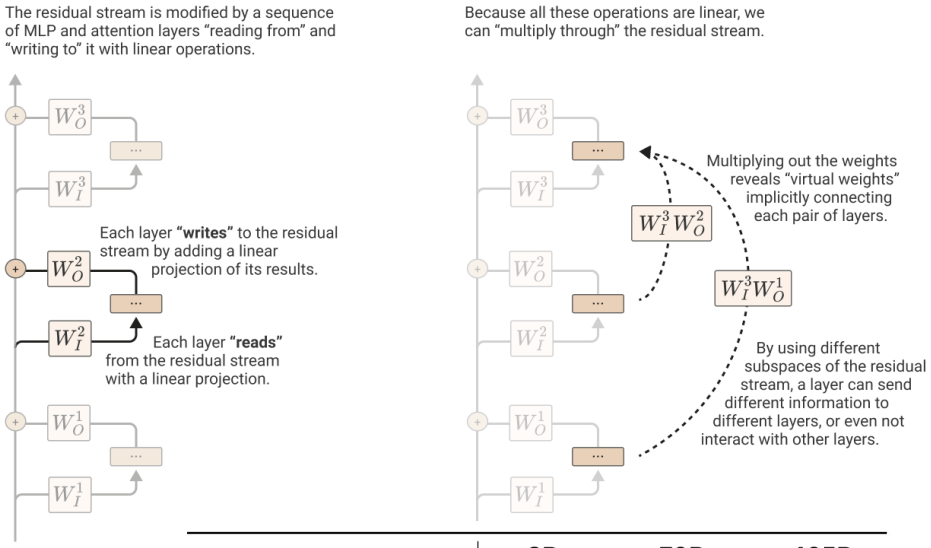
$$x_0 = W_E t$$

Virtual Weights and the Residual Stream as a Communication Channel

- the high level architecture of a transformer — each layer adds results into the *"residual stream"*.
- The residual stream is simply *the sum of the output of all the previous layers and the original embedding*.
- linear read in, linear read out



- ***Virtual Weights***
  - as residual streams are linear, we can think of implicit "virtual weights", directly connecting any pair of layers (even those separated by many other layers), by multiplying out their interactions through the residual stream
  - see the right figure for illustrations

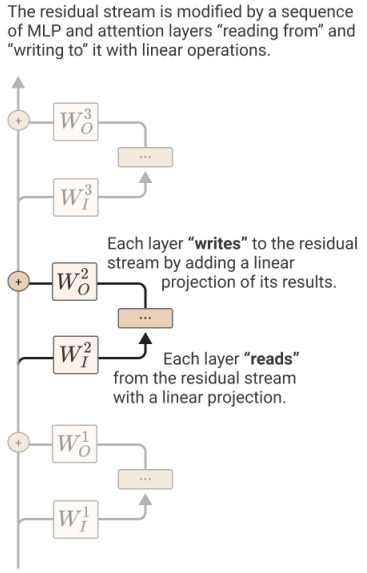


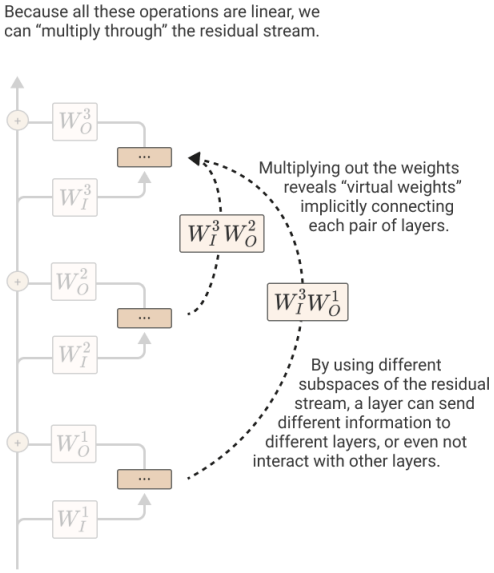
- ***Subspace and Residual Stream Bandwidth***
  - the residual stream is a high dimensional vector space: layers can send different information to different layers by storing it in different subspaces
  - since every individual head operates on comparatively small subspaces (often 64 or 128 dimensions), and can very easily write to completely disjoint subspaces and not interact.
  - the residual stream can be interpreted as "memory" or "bandwidth"
  - the original token embeddings, as well as unembeddings, mostly interact with a relatively small fraction of dimensions (experiments through PCA)

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ($\theta = 500,000$)		

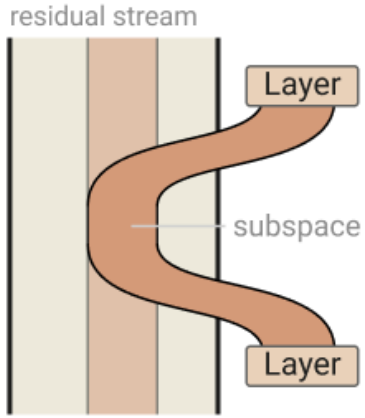
Virtual Weights and the Residual Stream as a Communication Channel

- the residual stream is *hard to interpret* as it communicates with 100 times more neurons as it has dimensions (25 layer, 4× hidden dimension for MLPs)
- Perhaps because of this high demand on residual stream bandwidth, we've seen hints that some MLP neurons and attention heads may perform a kind of "memory management" role, clearing residual stream dimensions set by other layers by reading in information and writing out the negative version
  - e.g. very negative cosine similarity between MLP input / output weights

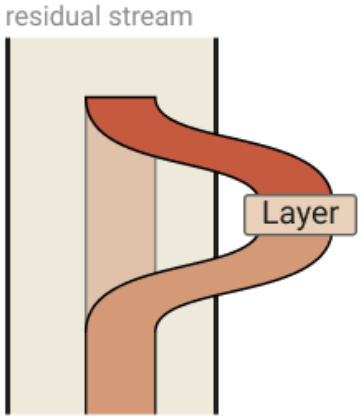




The residual stream is high dimensional, and can be divided into different subspaces.



Layers can interact by writing to and reading from the same or overlapping subspaces. If they write to and read from disjoint subspaces, they won't interact. Typically the spaces only partially overlap.

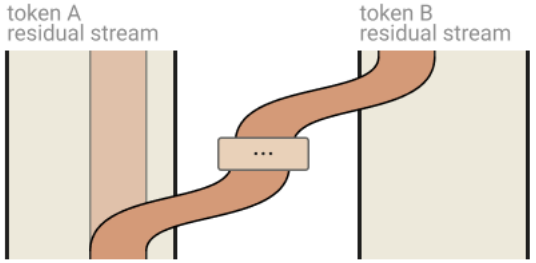


Layers can delete information from the residual stream by reading in a subspace and then writing the negative verison.

Attention Heads as Information Movement

- *which tokens to move information* from is completely separable from *what information is "read"* to be moved and *how it is "written"* to the destination

1. Compute the value vector for each token from the residual stream ($v_i = W_V x_i$).
2. Compute the "result vector" by linearly combining value vectors according to the attention pattern ($r_i = \sum_j A_{i,j} v_j$).
3. Finally, compute the output vector of the head for each token ($h(x)_i = W_O r_i$).[8]



Attention heads copy information from the residual stream of one token to the residual stream of another. They typically write to a different subspace than they read from.

- some conclusions about attention heads:
  - attention heads move information from the residual stream of one token to another
  - an attention head is really applying 2 linear operations: A and $W_O W_V$
    - A governs *which token's information is moved from and to*
    - $W_O W_V$ governs *which information is read from the source token* and *how it is written into the destination token*
  - Attention Pattern A is the only non-linear part
  - W_Q and W_K always operate together. They're never independent. So are W_O and W_V.
    - $W_{QK} = W_Q^T W_K$, low rank
    - $W_{OV} = W_O W_V$, also low rank

Conceptual Take-aways

- Attention heads can be understood as independent operations, each outputting a result which is added into the residual stream.
- Attention-only models can be written as a sum of interpretable end-to-end functions mapping tokens to changes in logits.
  - These functions correspond to "paths" through the model, and *are linear if one freezes the attention patterns*.
- Transformers have an enormous amount of linear structure.
- Attention heads can be understood as having two largely independent computations: a QK ("query-key") circuit which computes the attention pattern, and an OV ("output-value") circuit which computes how each token affects the output if attended to.

Circuit Tracing: Revealing Computational Graphs in Language Models

Preliminary

- Deep learning models produce their outputs using a series of transformations distributed across many neurons.
- We seeks to describe these transformations in human-understandable language.
- use raw neurons from LLMs?
  - model neurons are often *polysemantic* – representing a mixture of many unrelated concepts
  - *superposition*: models must represent more concepts than they have neurons, and thus must "smear" their representation of concepts across many neurons.
- Sparse AutoEncoders (SAEs), transcoders, cross coders have emerged as promising tools for identifying interpretable features represented in superposition
  - decompose model activations into sparsely active components ("features") → correspond to human-interpretable concepts

Building an Interpretable Replacement Model - Architecture

- transcoders seek to faithfully approximate a densely activating MLP layer with a *wider, sparsely-activating* MLP layer (15k → 500K)
- Cross-Layer Transcoder
  - each feature in the ℓ^{th} layer "reads in" from the residual stream at that layer using a linear encoder followed by a nonlinearity
  - an ℓ^{th} layer feature contributes to the reconstruction of the MLP outputs in layers $\ell, \ell+1, \ldots, L$, using a separate set of linear decoder weights for each output layer.

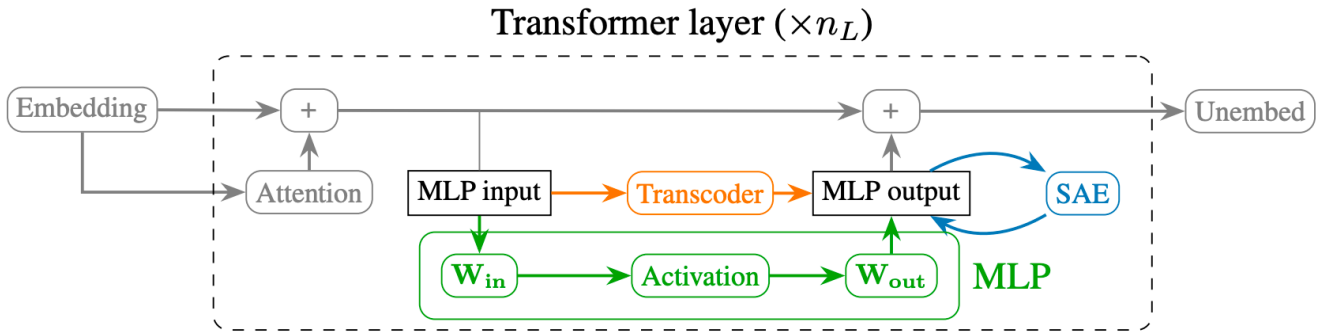


Figure 1: A comparison between **SAEs**, **MLP transcoders**, and **MLP sublayers** for a transformer-based language model. SAEs learn to reconstruct model activations, whereas transcoders imitate sublayers' input-output behavior.

Dunefsky, et al., "Transcoders Find Interpretable LLM Feature Circuits", 2024

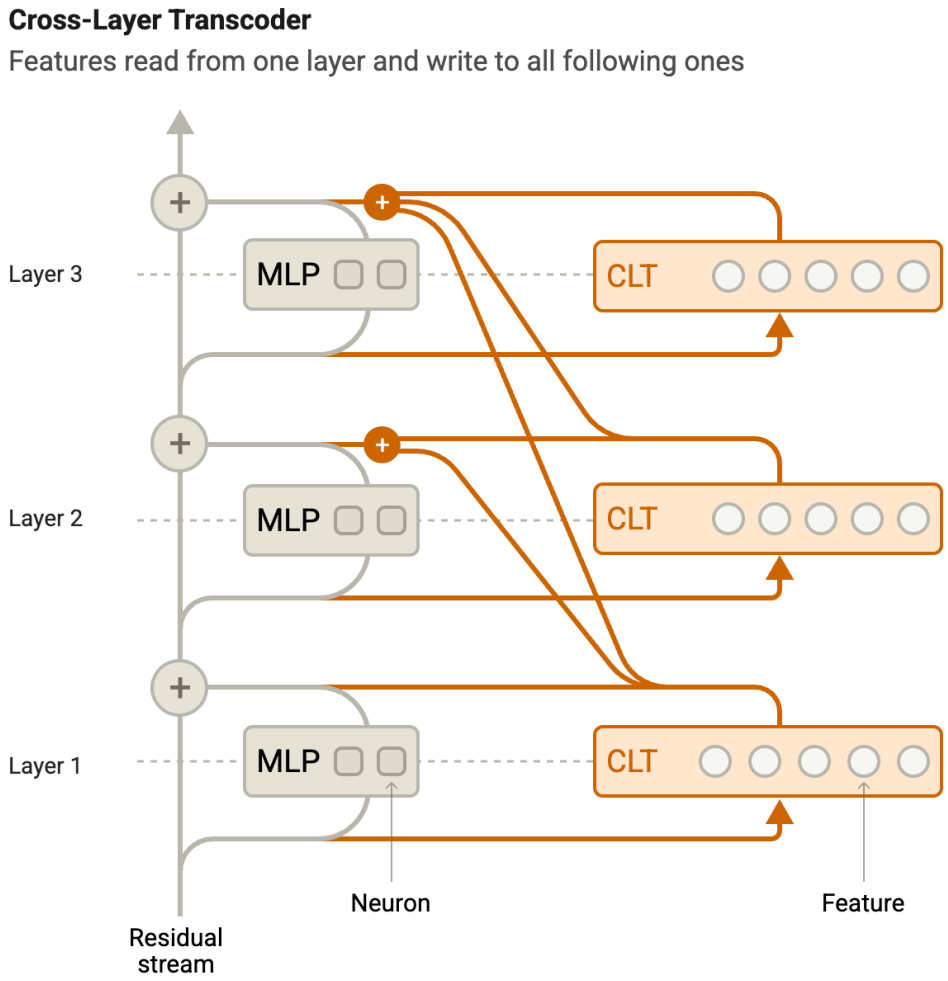


Figure 1: The cross-layer transcoder (CLT) forms the core architecture of our replacement model.

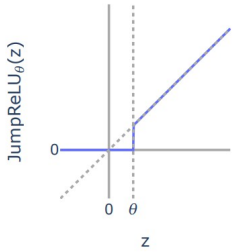
Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

Building an Interpretable Replacement Model - Architecture

- To run a CLT, let x^ℓ denote the original model's residual stream activations at layer ℓ.

$$a^\ell = \text{JumpReLU}(W_{enc}^\ell x^\ell)$$

$$y^\ell = \sum_{\ell\prime=1}^{\ell} W_{dec}^{\ell' \to \ell} a^{\ell'}$$



- To train a CLT, we minimize a sum of two loss function:

$$L_{MSE} = \sum_{\ell=1}^{L} \left\| \hat{y}^\ell - y^\ell \right\|^2$$

$$L_{sparsity} = \lambda \sum_{\ell=1}^{L} \sum_{i=1}^{N} \tanh(c \cdot \left\| W_{dec,i}^\ell \right\| \cdot a_i^\ell)$$

- how many features in CLT?
  - 18-layer transformer (18L): 300K to 10M
  - Claude Haiku: 300K to 30M

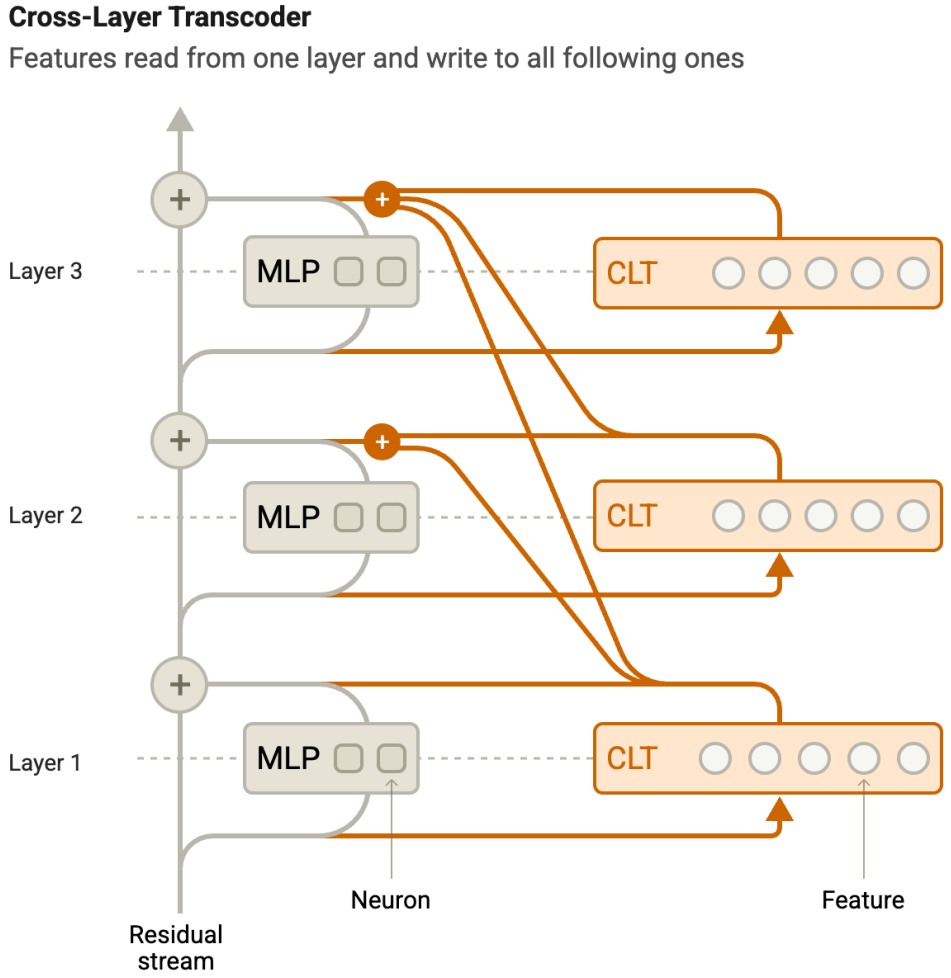


Figure 1: The cross-layer transcoder (CLT) forms the core architecture of our replacement model.

Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

From Cross-Layer Transcoder to Replacement Model

- replace the MLP layers with newly trained CLT layers
- upon reaching the input to the MLP in layer ℓ, we compute the activations of CLT whose encoder lives in layer ℓ
- upon reaching the output, we overwrite it with the summed outputs of CLT features in this and previous layer
- the largest 18L CLT matches the underlying model's next-token completion on 50% of a diverse set of pretraining-style prompts from an open source dataset

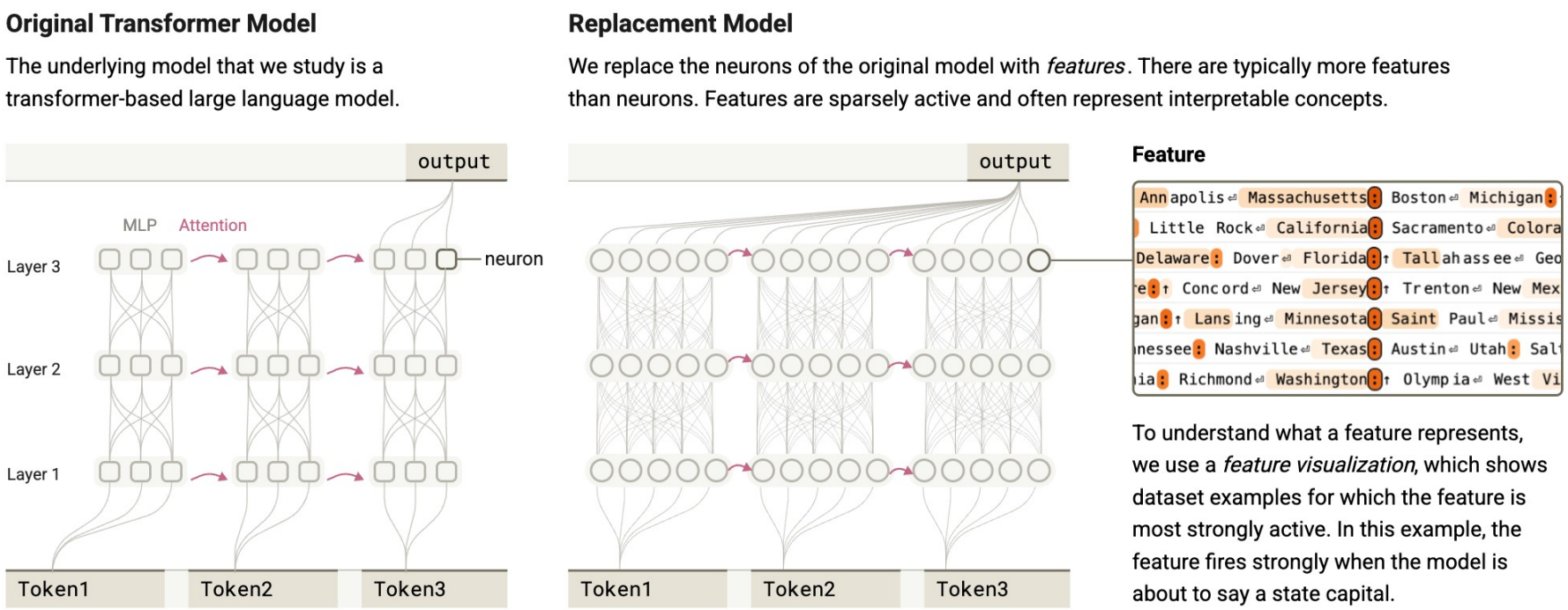


Figure 2: The replacement model is obtained by replacing the original model's neurons with the cross-layer transcoder's sparsely-active features.

Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

The Local Replacement Model

- There is still a significant gap between replacement model and the underlying model, and reconstruction errors can compound across layers.
- we want to approximate it as close as possible
- for a fixed prompt p, we construct a local replacement model:
  - Substitutes the CLT for the MLP layers (as in the replacement model);
  - Uses the *attention patterns and normalization denominators* from the underlying model's forward pass on p;
  - Adds an *error adjustment* to the CLT output at each (token position, layer) pair equal to the difference between the true MLP output on p and the CLT output on p.
- Now, we ensure the error-corrected replacement model's activations and logit outputs exactly match those of the underlying model. However, this does not guarantee that the local replacement model and the underlying model use the same mechanisms. → mechanistic faithfulness, will be discussed later

Attribution Graphs

- nodes represent features and edges represent the causal interactions between them

Local Replacement Model

The local replacement model is specific to a prompt of interest. We add an error adjustment term, and freeze attention patterns to be what they were in the original model on the given prompt. It produces the exact same output as the original model, but replaces as much computation as possible with features.

Attribution Graph

We trace from input to output through active features, pruning paths that don't influence the output.

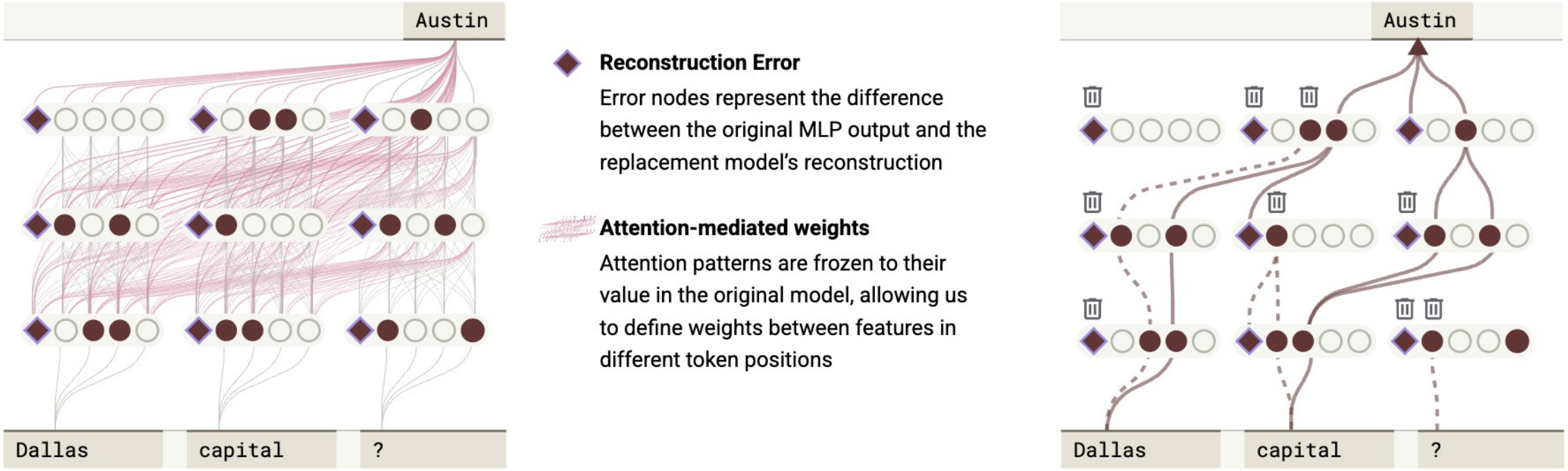
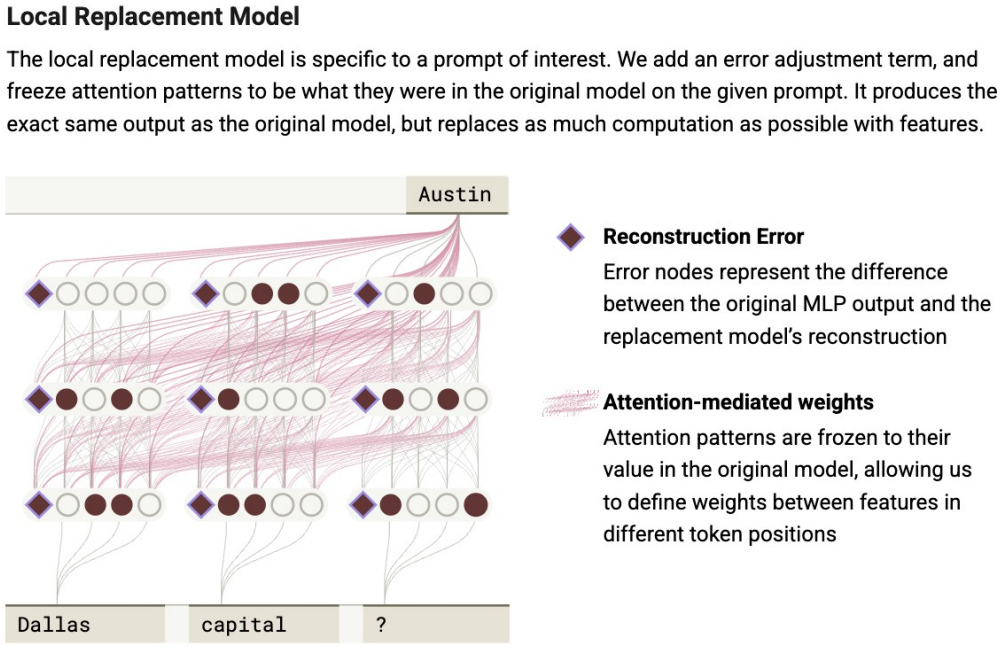


Figure 4: The local replacement model is obtained by adding error terms and fixed attention patterns to the replacement model to exactly reproduce the original model's behavior on a specific prompt.

Attribution Graphs

- node types:
  - The *output* nodes: candidate output tokens, we only construct output nodes for the tokens required to reach 95% of the probability mass, up to a total of 10;
  - The *intermediate* nodes: active cross-layer transcoder features at each prompt token position;
  - The primary *input* nodes: embeddings of the prompt tokens;
  - Additional input nodes ("*error* nodes"): the portion of each MLP output in the underlying model left unexplained by the CLT.



Attribution Graphs

- Edges in the graph represent direct, linear attributions in the local replacement model.
- Given a source feature node s and a target feature node t, the edge weight between them is defined to be:

$$A_{s\to t} := a_s w_{s\to t}$$

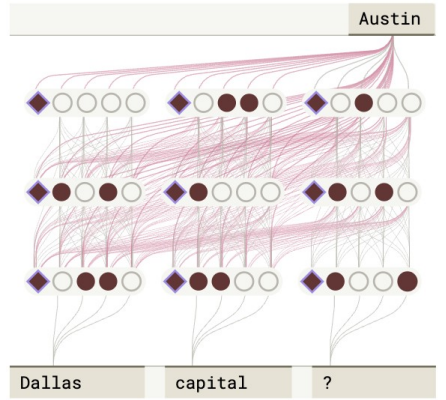
  - where $w_{s\to t}$ is the virtual weight in the local replacement model viewed as a fully connected neural network and a_s is the activation of s
- s: source feature node, layer ℓ_s, context position c_s; t the same
- $J_{c_s,\ell_s\to c_t,\ell_t}$: for the Jacobian of the underlying model with a stop-gradient operation applied to all models components with nonlinearities – the MLP outputs, the attention patterns, and normalization denominators
- The edge weight from $s \to t$ is then (activation to pre-activation):

$$A_{s\to t} = a_s w_{s\to t} = a_s \sum_{\ell_s \le \ell < \ell_t} \left(W_{dec,s}^{\ell_s\to\ell}\right)^T J_{c_s,\ell\to c_t,\ell_t}\, W_{enc,t}^{\ell_t}$$

- the attribution graph edges provide a linear decomposition of each feature's activity

Local Replacement Model

The local replacement model is specific to a prompt of interest. We add an error adjustment term, and freeze attention patterns to be what they were in the original model on the given prompt. It produces the exact same output as the original model, but replaces as much computation as possible with features.



Reconstruction Error

Error nodes represent the difference between the original MLP output and the replacement model's reconstruction

Attention-mediated weights

Attention patterns are frozen to their value in the original model, allowing us to define weights between features in different token positions

Cross-Layer Transcoder

Features read from one layer and write to all following ones

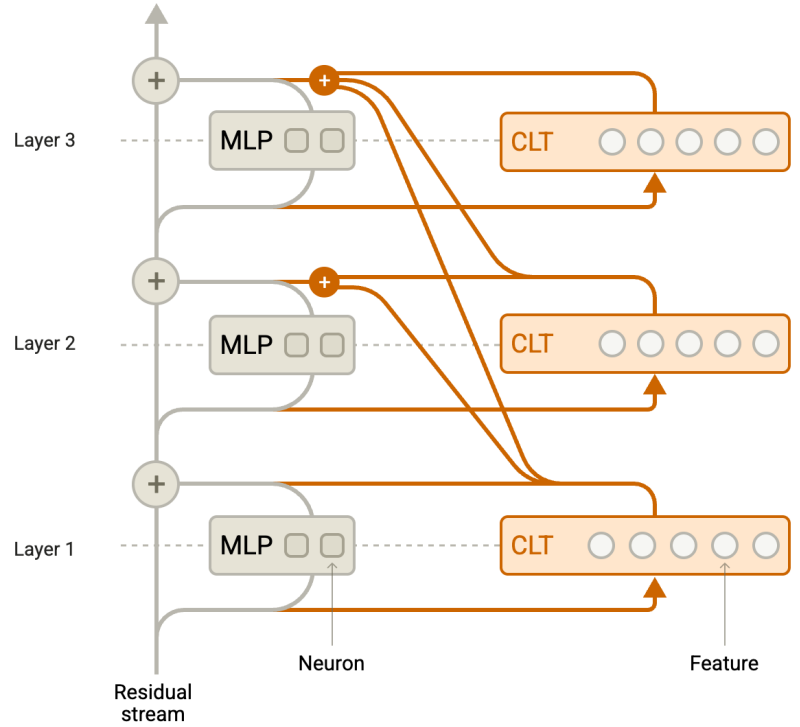


Figure 1: The cross-layer transcoder (CLT) forms the core architecture of our replacement model.

Learning from Attribution Graphs

- we do pruning on attribution graphs, as they are too large
- below is an example attribution graph:
  - here we study how model writes acronyms for arbitrary titles

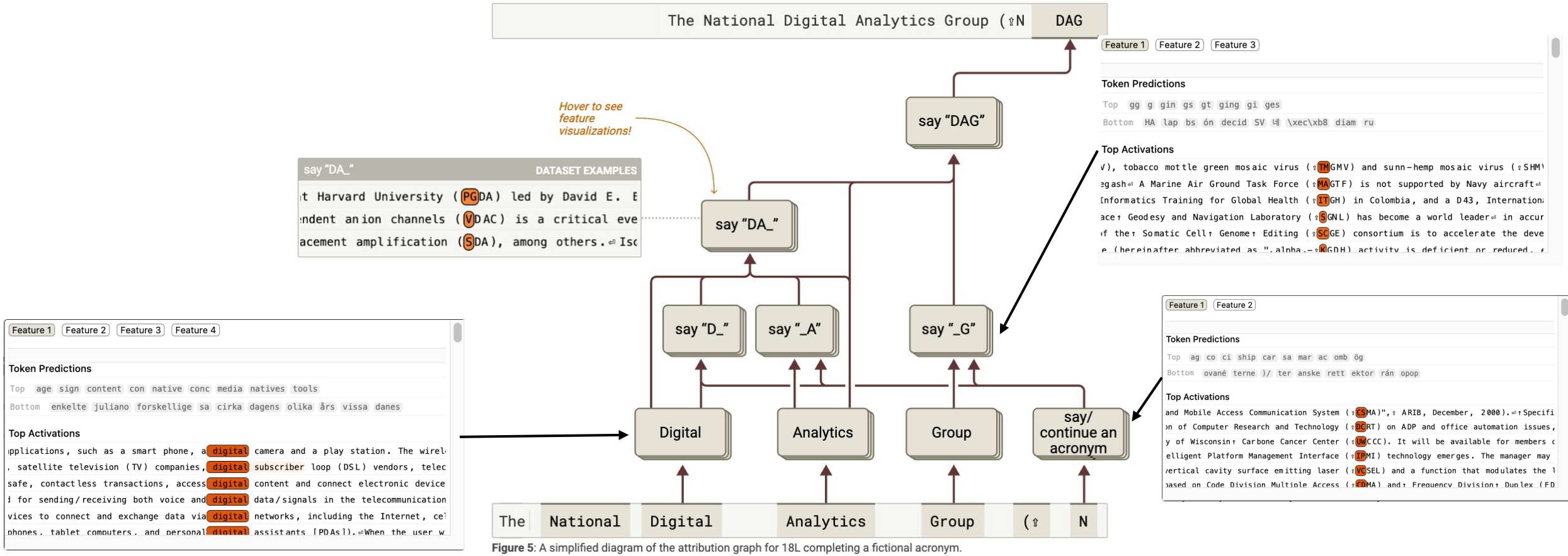
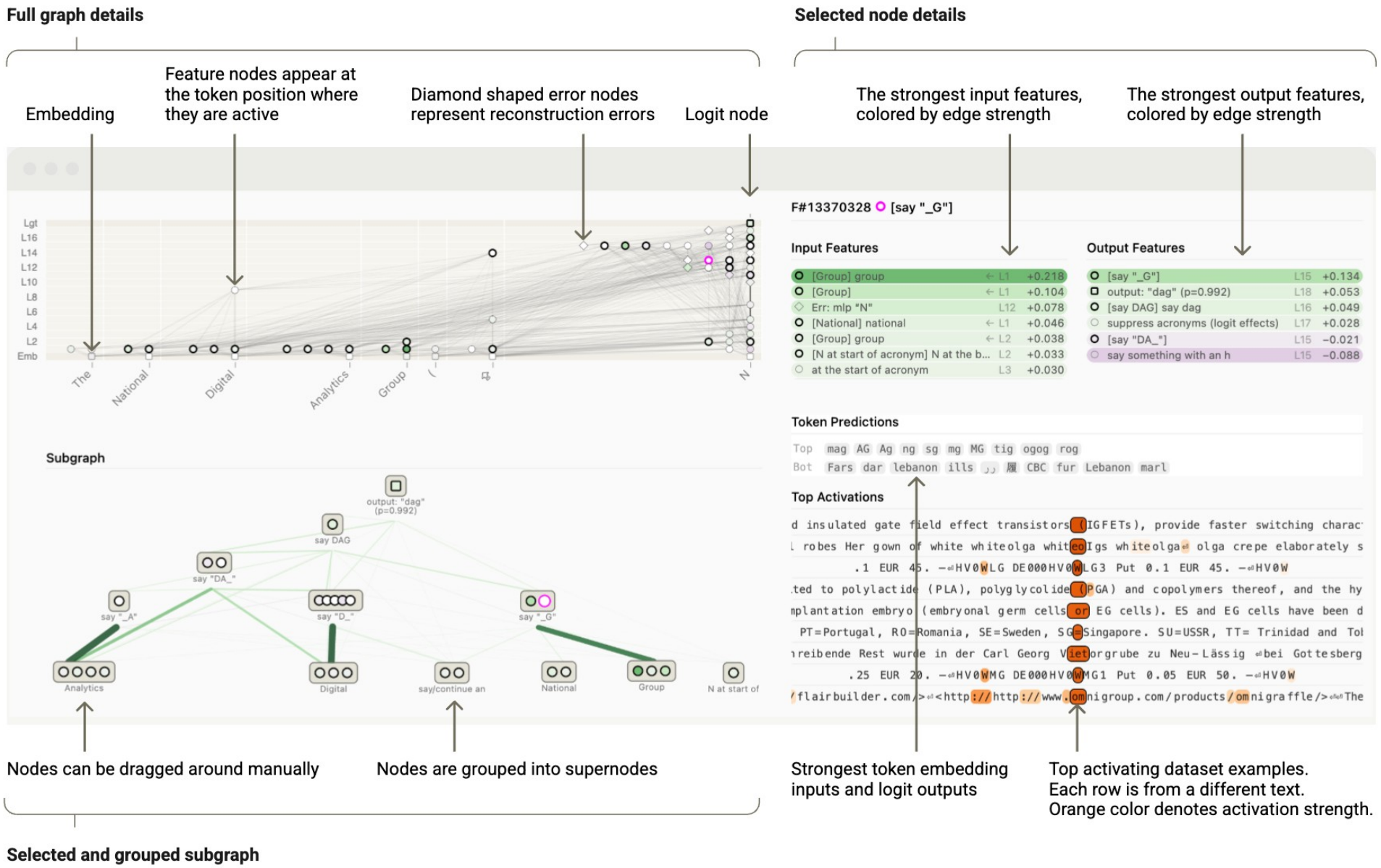


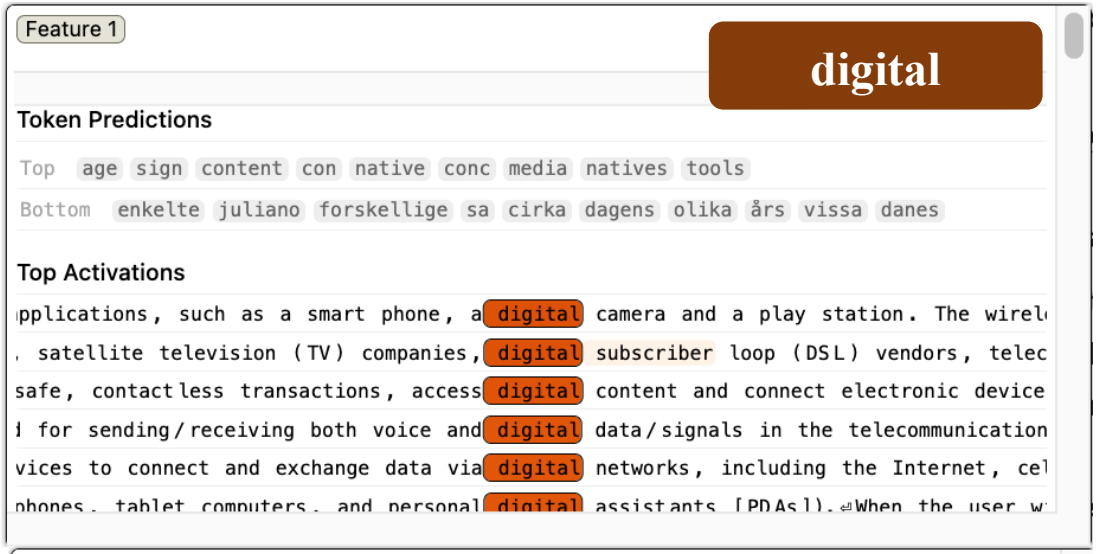
Figure 5: A simplified diagram of the attribution graph for 18L completing a fictional acronym.

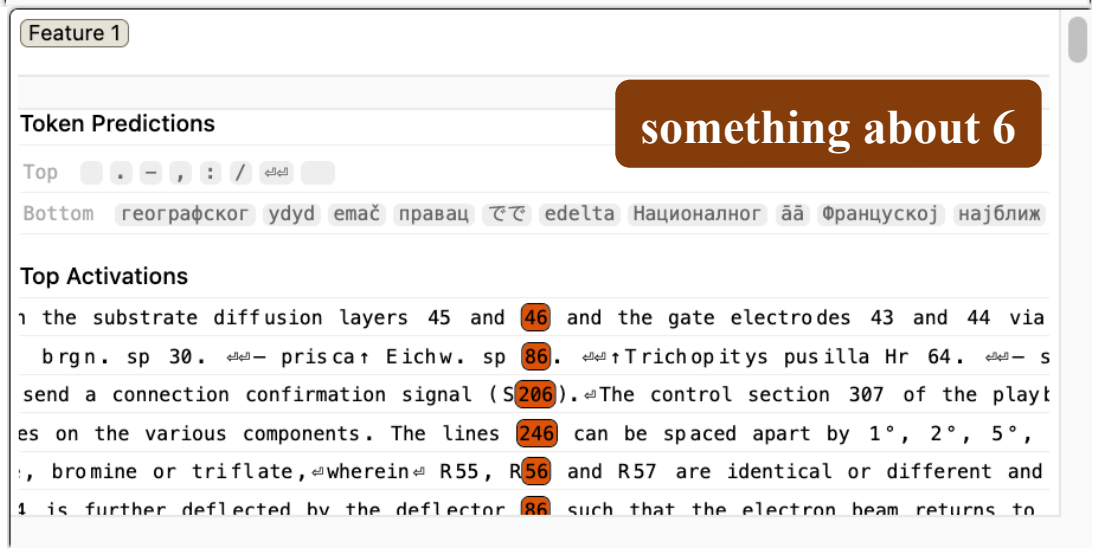
Learning from Attribution Graphs



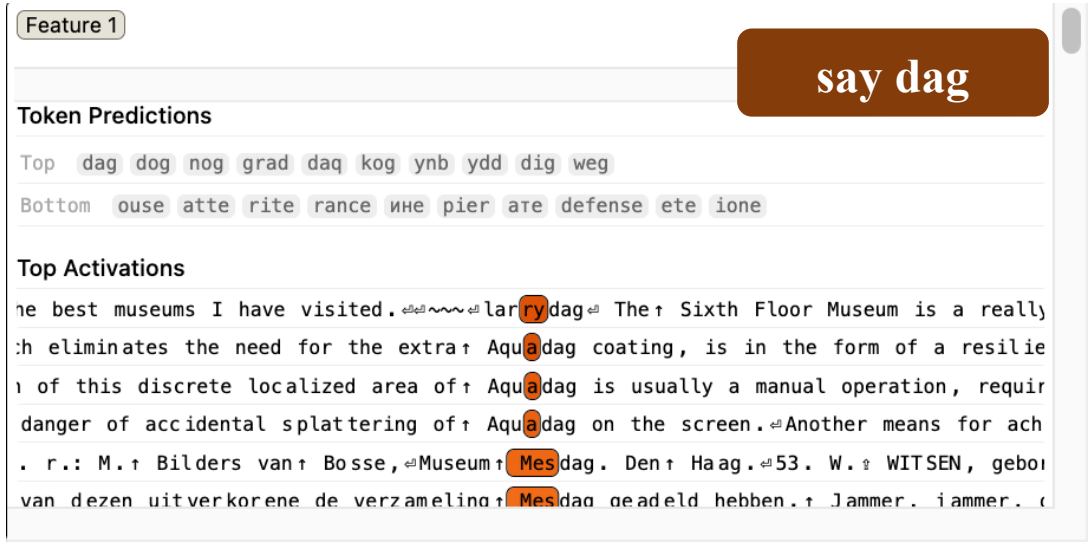
Understanding and Labeling Features

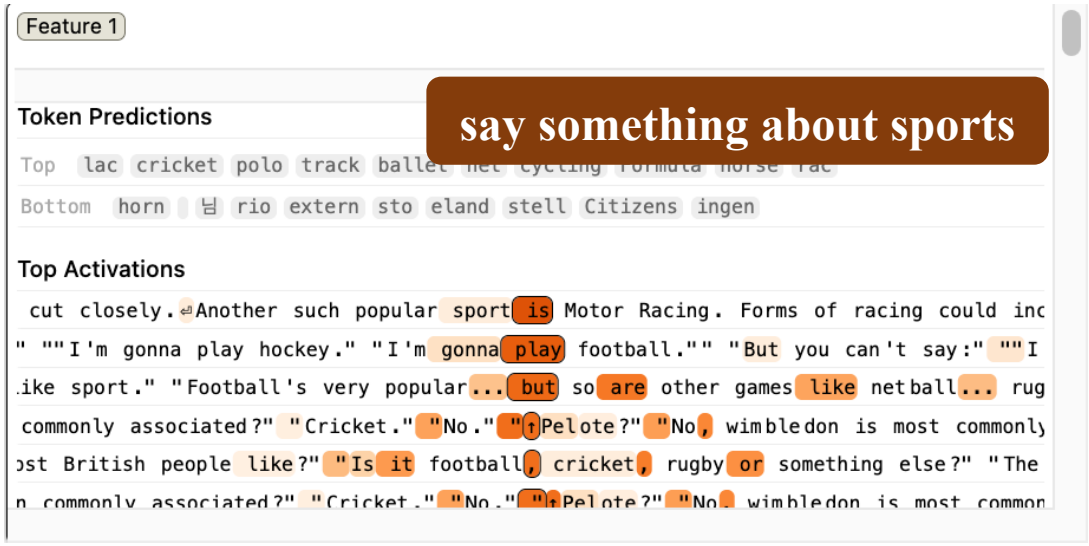
- input features





- output features

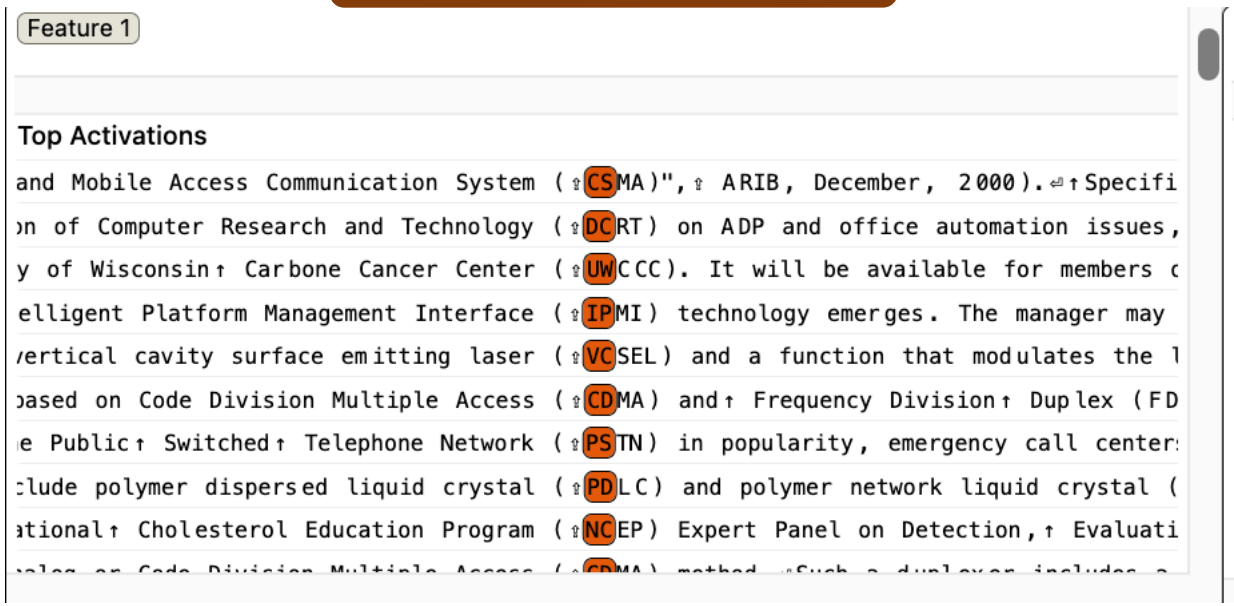




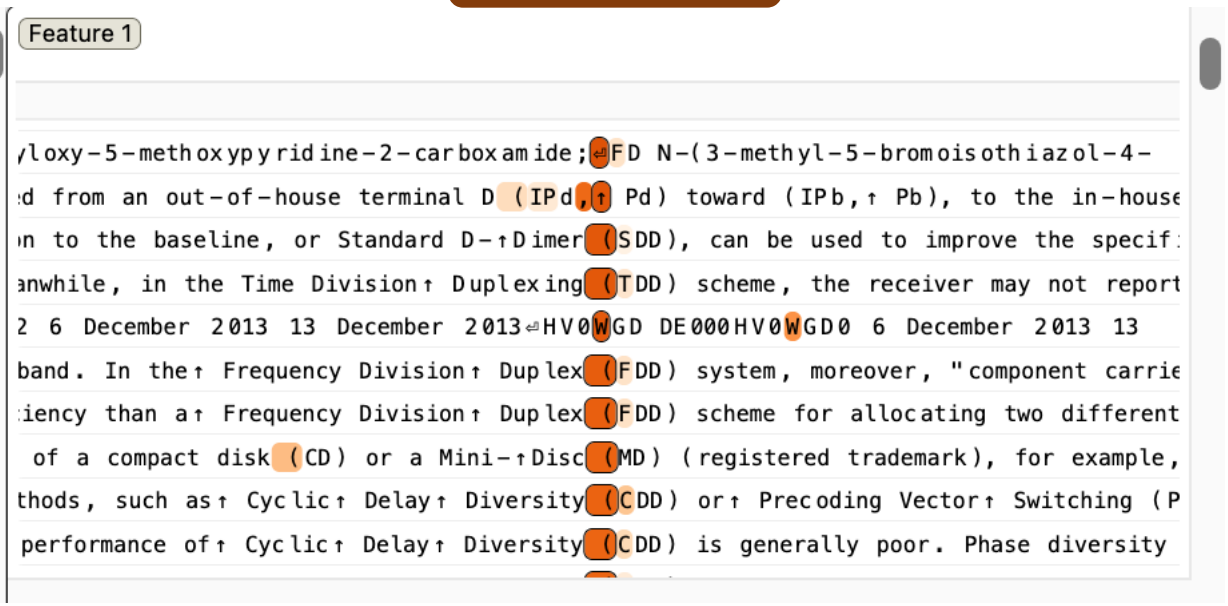
Understanding and Labeling Features

- some other features, which are common in middle layers, more abstract and require more work to label

continuing an acronym



say '_D'



Grouping Features into Supernodes

- Attribution graphs often contain groups of features which share a facet relevant to their role on the prompt.
- For example, there are three features active on **digital** in our prompt which each respond to the word "digital" in different cases and contexts → group them into supernodes
- *manual work*
  - While we experimented with automated strategies such as clustering based on decoder vectors or the graph adjacency matrix, *no automated method was sufficient* to cover the range of feature groupings required to illustrate certain mechanistic claims

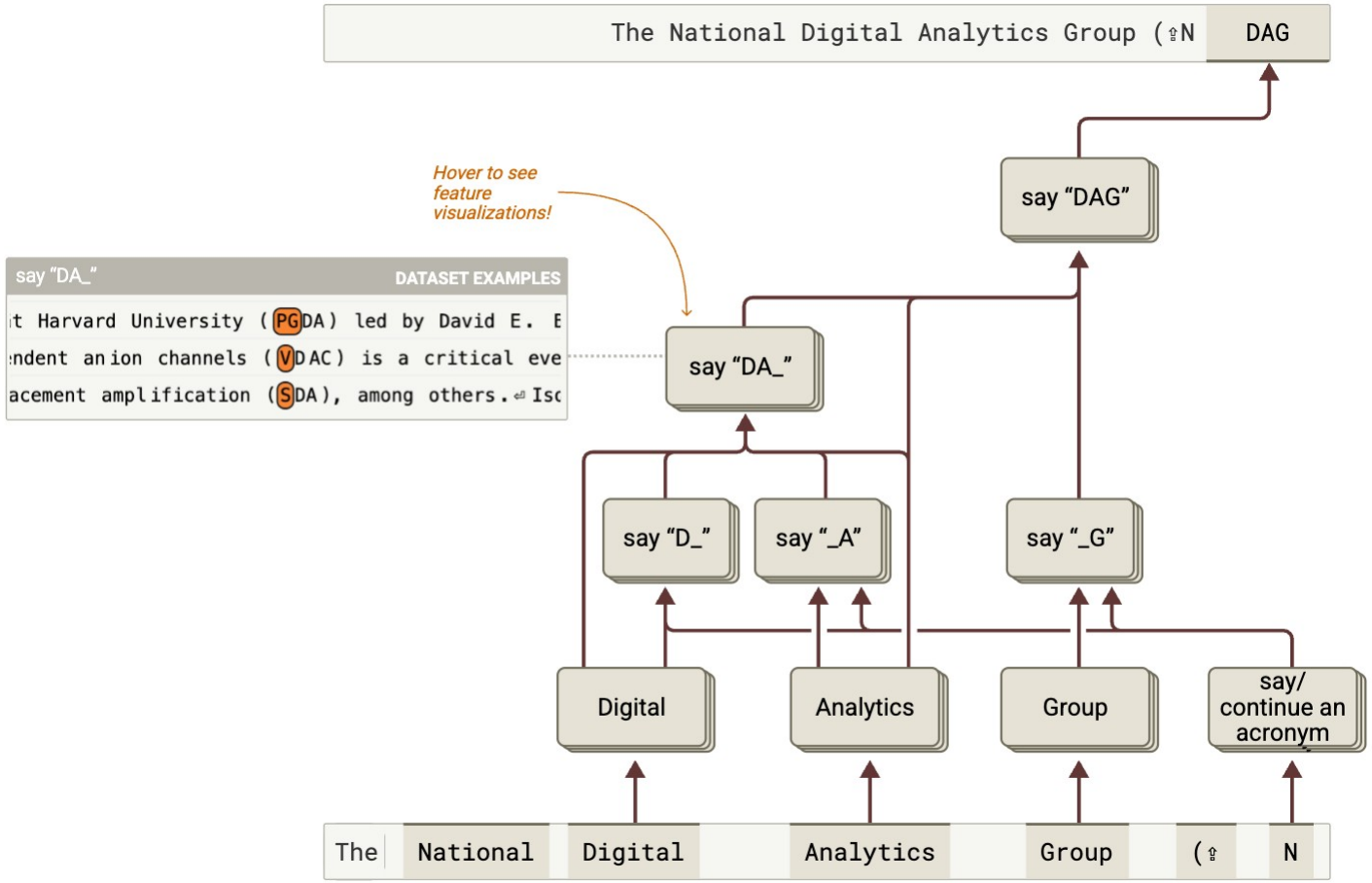
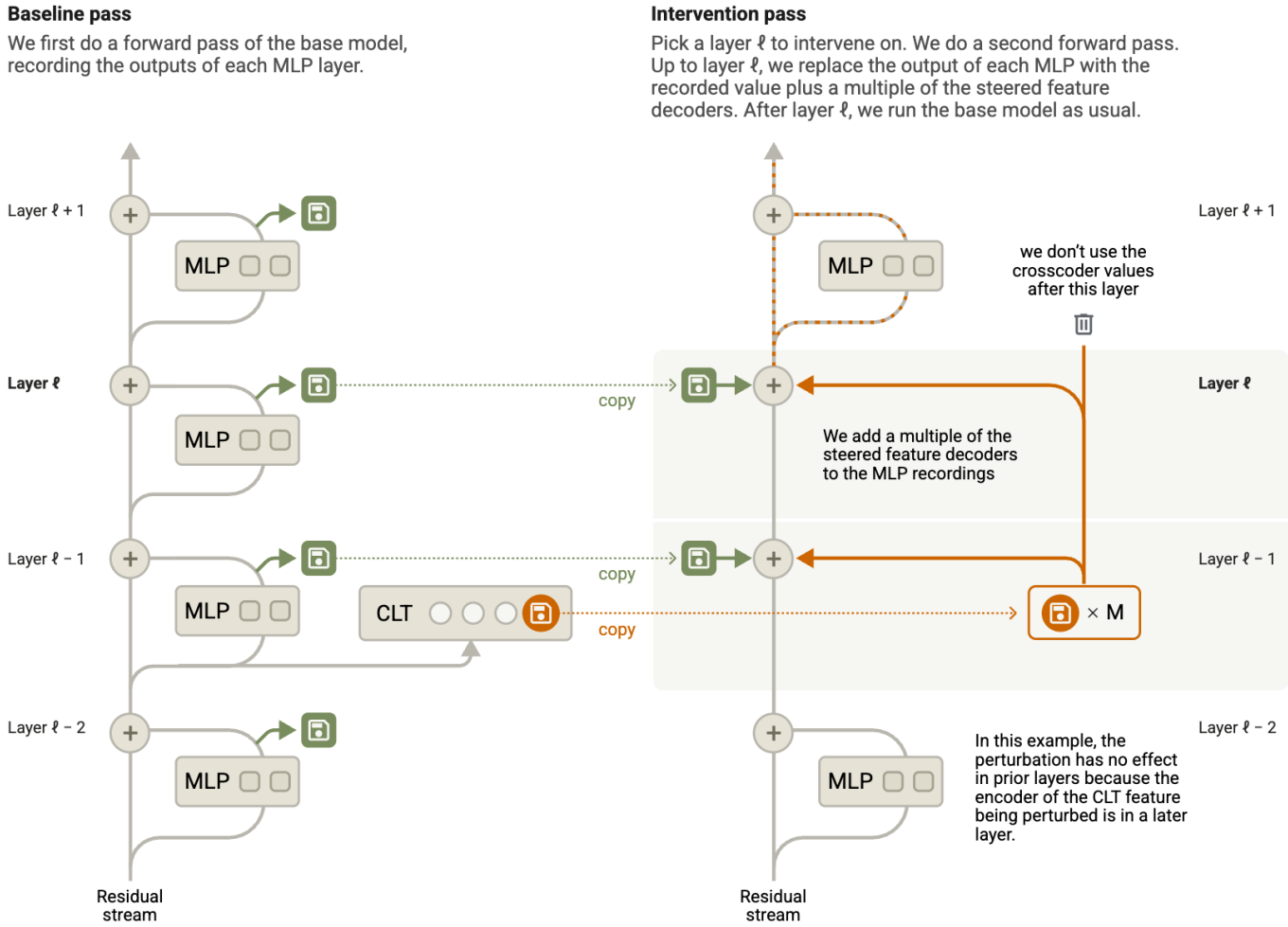


Figure 8: A simplified diagram of the attribution graph for 18L completing a fictional acronym.

Validating Attribution Graph Hypotheses with Interventions

- In attribution graphs, nodes suggest *which features matter* for a model's output, and edges suggest *how they matter*
- perform feature perturbations in the underlying model, and checking if the effects on downstream features or on the model outputs match our predictions based on the graph



Validating Attribution Graph Hypotheses with Interventions

- In the acronym sample, we can suppress the “Group” Supernode
- suppress some supernodes on the aggregate activation of other supernodes and on the logit

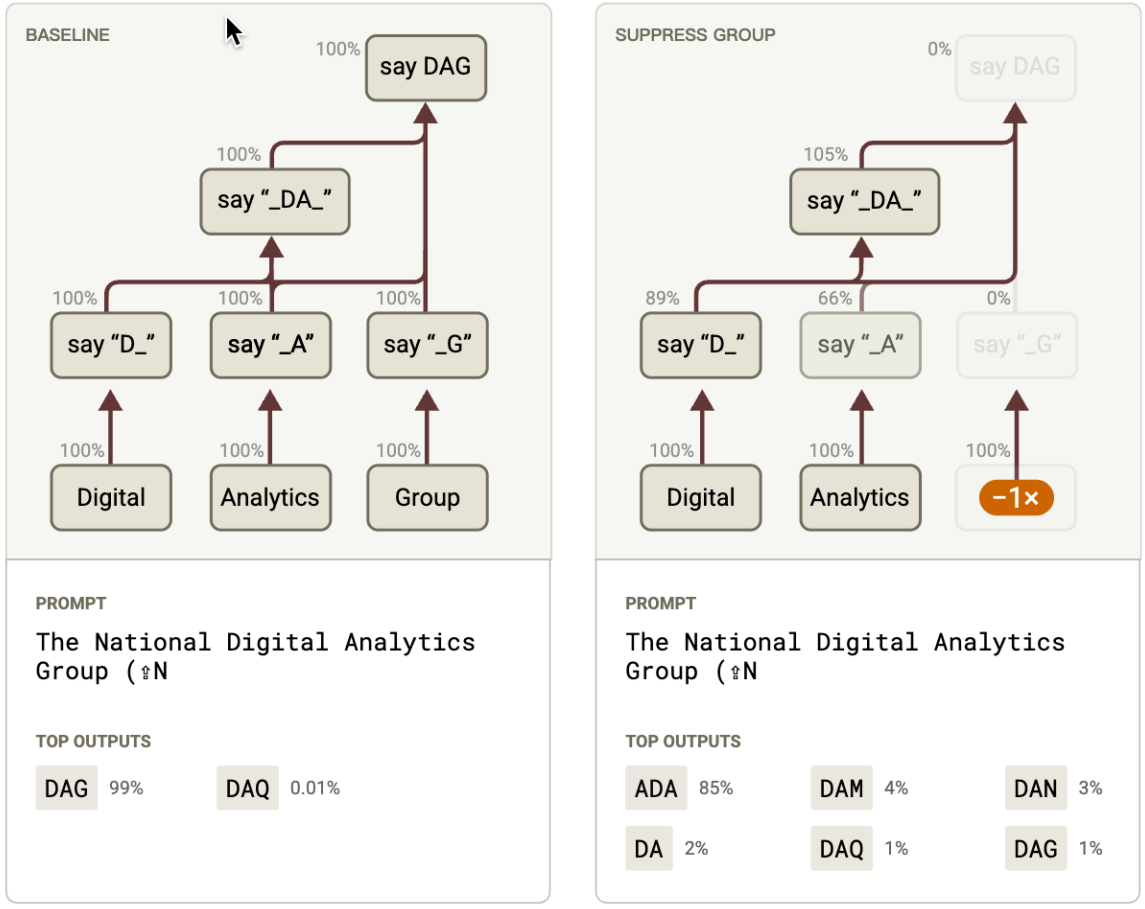


Figure 10: Suppressing the word “Group” in the fictional organization’s name causes 18L to output other acronyms with “DA” in them.

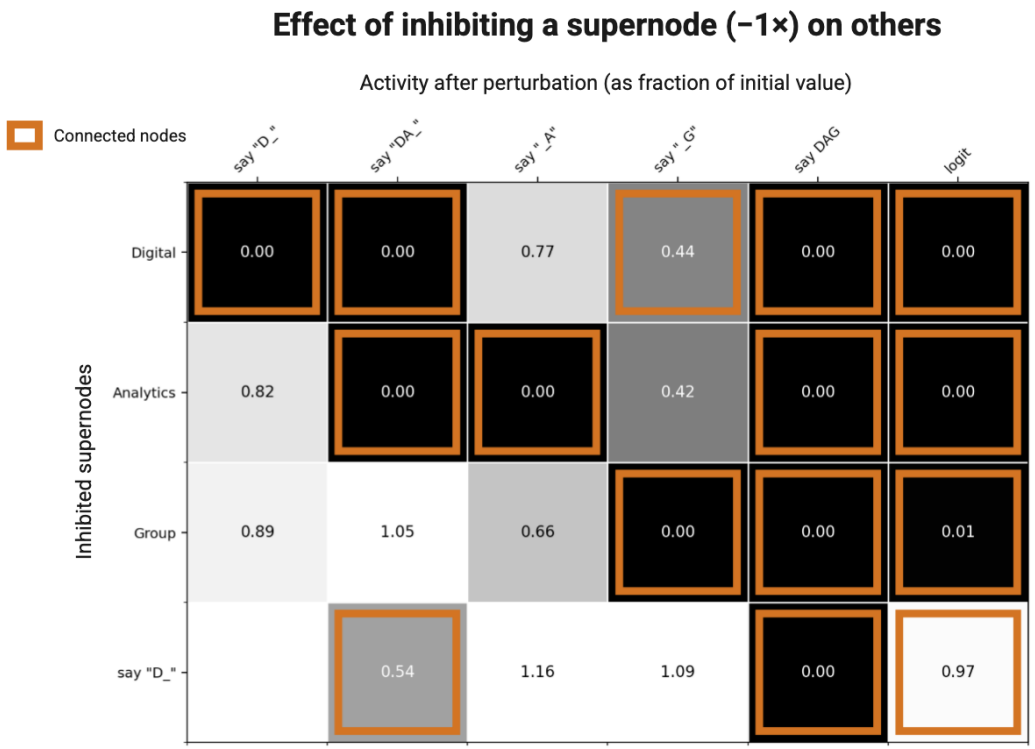


Figure 11: Selected intervention effects on feature activations for “The National Digital Analytics Group (N”.

Localizing Important Layers

- The attribution graph also allows us to identify in which layers a feature's decoding will have the greatest downstream effect on the logit.
- For example, the *"Analytics"* supernode features mostly contribute to the "dag" logit indirectly through intermediate groups of features *"say _A", "say DA_", and "say DAG"* which live in layers 13 and beyond.

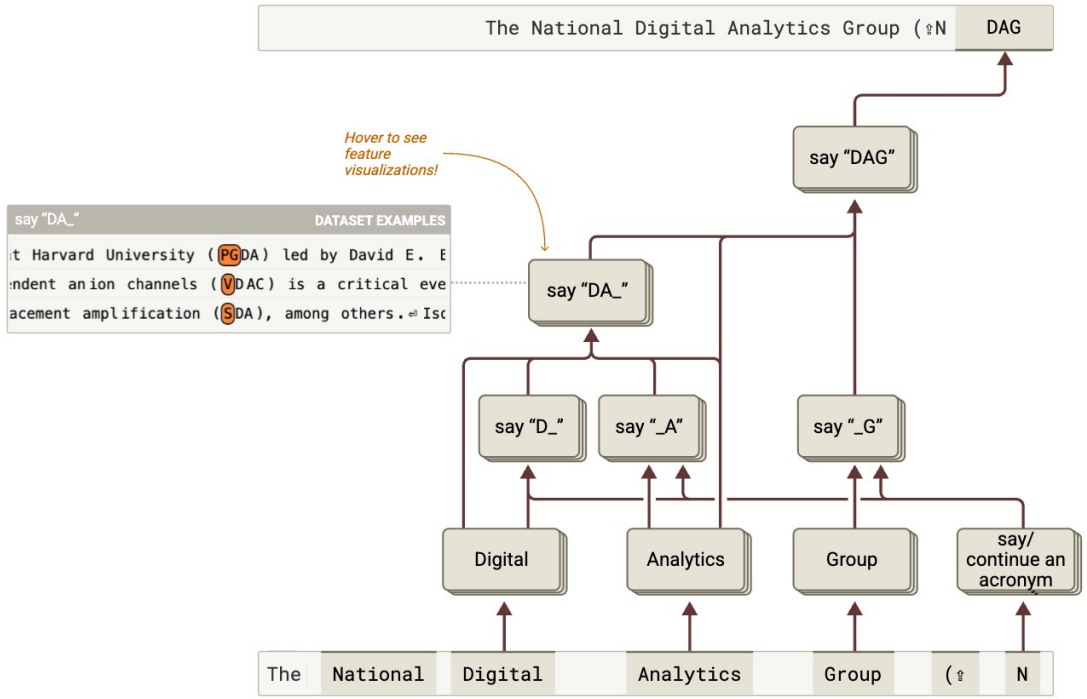
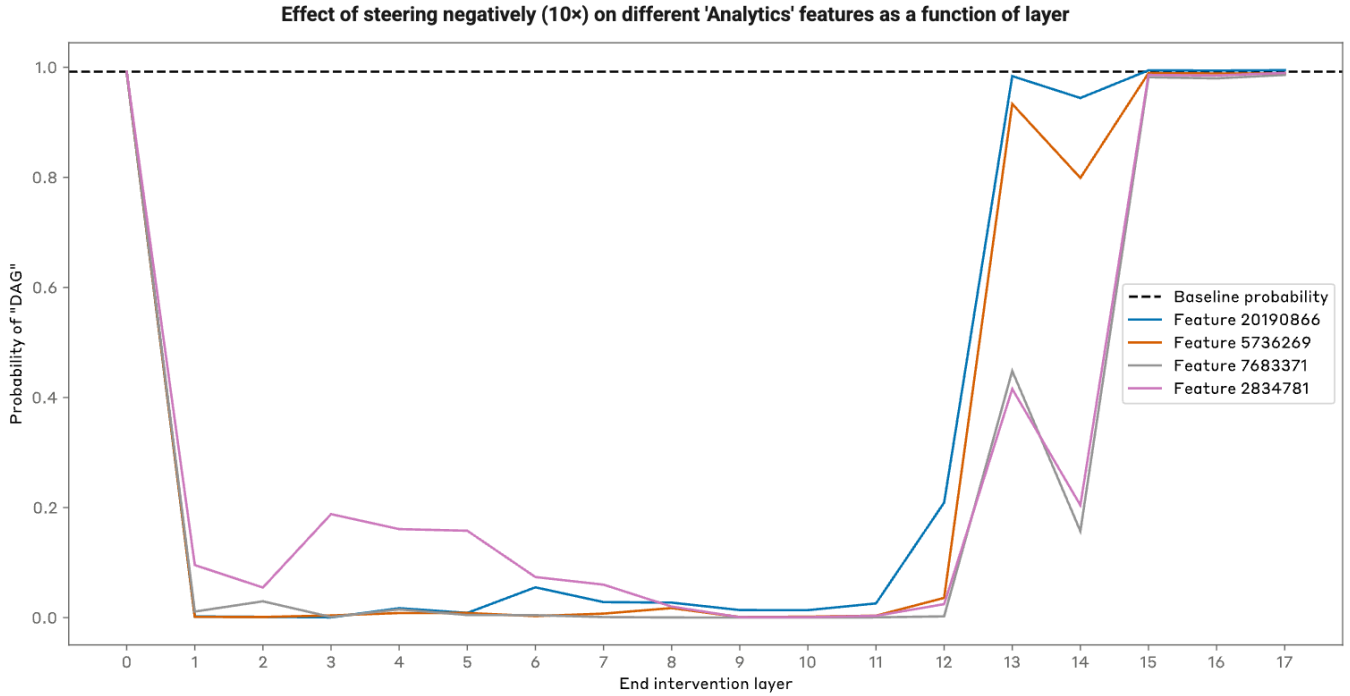


Figure 8: A simplified diagram of the attribution graph for 18L completing a fictional acronym.



Case Study: Addition (for Claude-3.5-Haiku)

- prompt: `calc: 36+59=`
- we visualize each feature active on the "=" token with three plots:

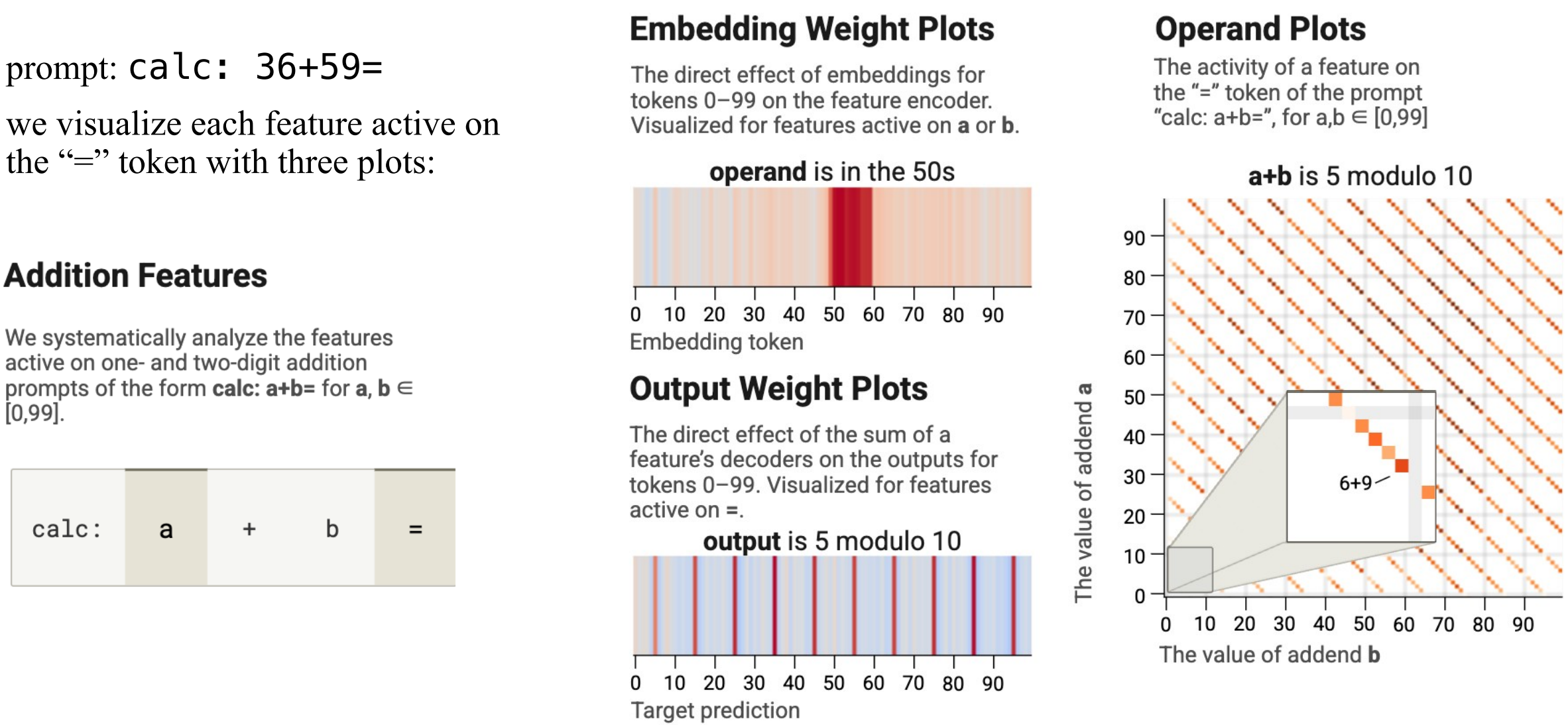


Figure 15: Common elements in figures describing addition features.

Case Study: Addition – the complete graph

- Input Features
  - how model represent a 2-digit integer
  - feature for the *exact number*
  - feature for the *rough range around*
  - feature for the *tens digit*
  - feature for the *ones digit*
- Add Function Features
  - separately determines the *ones digit* and its *approximate magnitude*
- Lookup Table Features
  - *modular* features
  - *low-precision* features
- Sum Features
  - finally compute sum with
  - *its value mod 10*
  - *mod 100*
  - *approximate magnitude*

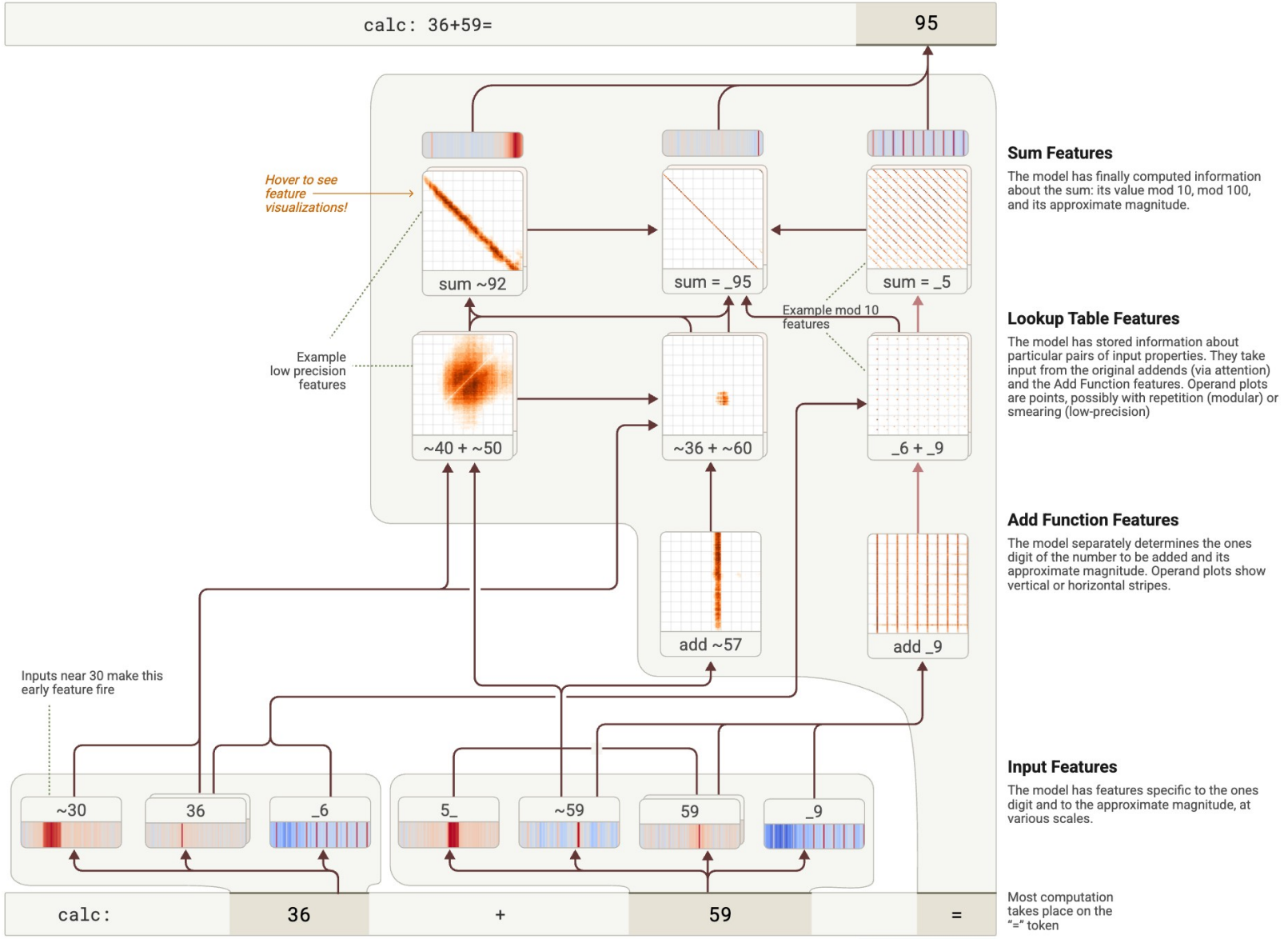
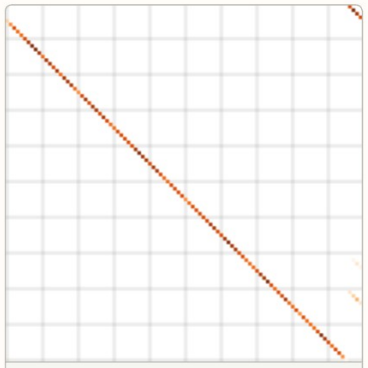


Figure 16: A simplified attribution graph of Haiku adding two-digit numbers. Features of the inputs feed into separable processing pathways.

FEATURE TAXONOMY

Sum Features

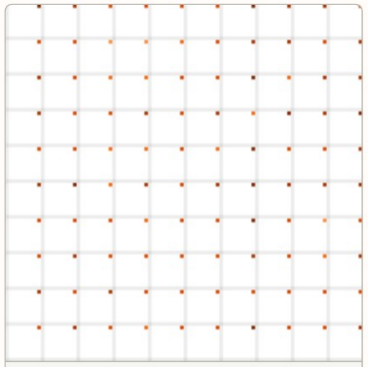
Sum features have diagonal operand plots, and fire on pairs of inputs whose sum satisfies some condition.



sum = _95

Lookup table Features

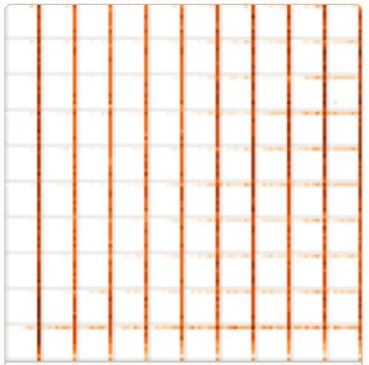
Lookup Table Features are approximately of the form **condition1(a) and condition2(b)**, and look like grids.



_9 + _9

Add function Features

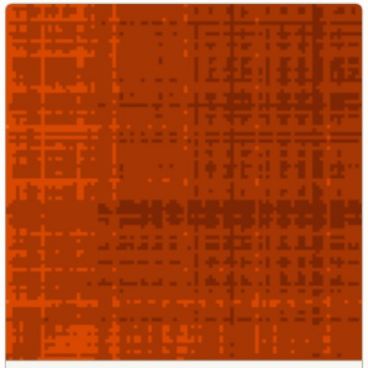
Add Function Features have plots which are unions of horizontal (condition on first operand) and vertical (condition on second operand) lines.



add 9

Mostly Active Features

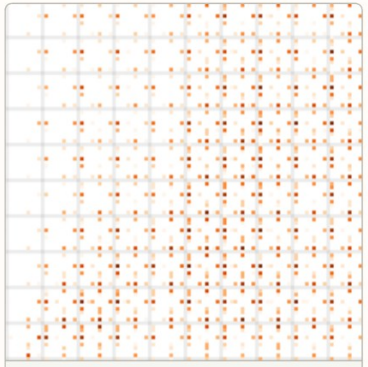
Features which are on on most addition prompts. Their output effects vary, but include things like “upweight simple numbers”.



active on “=”

Miscellaneous Features

We see a variety of lower-influence features with strange patterns. These may be artifacts of the crosscoder training process, or may reflect more subtle model properties.



?

Intervene the supernodes

- inhibiting ones-digit feature suppresses the entire ones-digit pathway: the _6 + _9 lookup table features, the resulting sum=_5 and sum= _95 features, while leaving the magnitude pathway mostly intact, including the sum~92 features.
- Remarkably, when suppressing _6, the model confidently outputs 98 instead of the correct answer 95

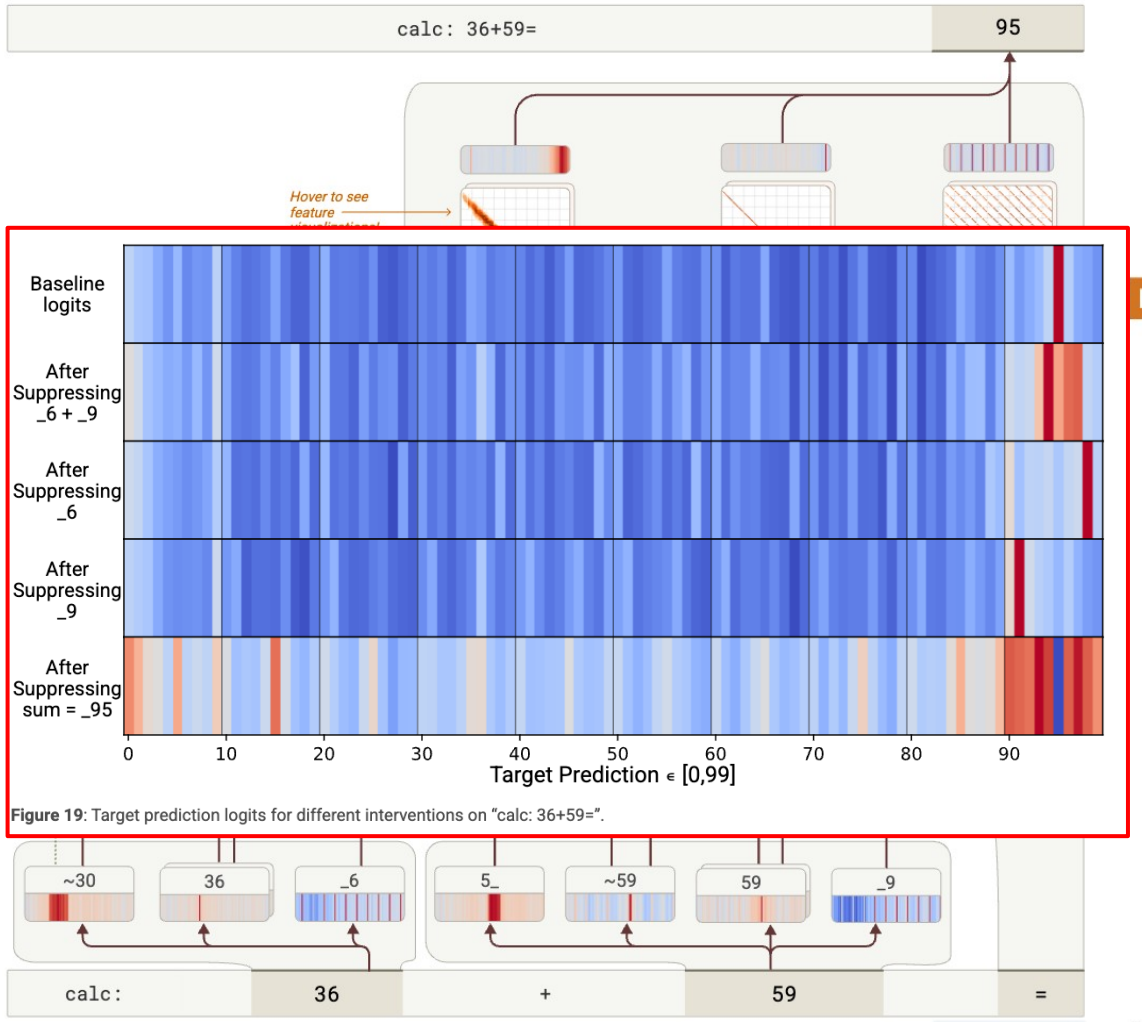


Figure 19: Target prediction logits for different interventions on "calc: 36+59=".

Figure 16: A simplified attribution graph of Haiku adding two-digit numbers. Features of the inputs feed into separable processing pathways.

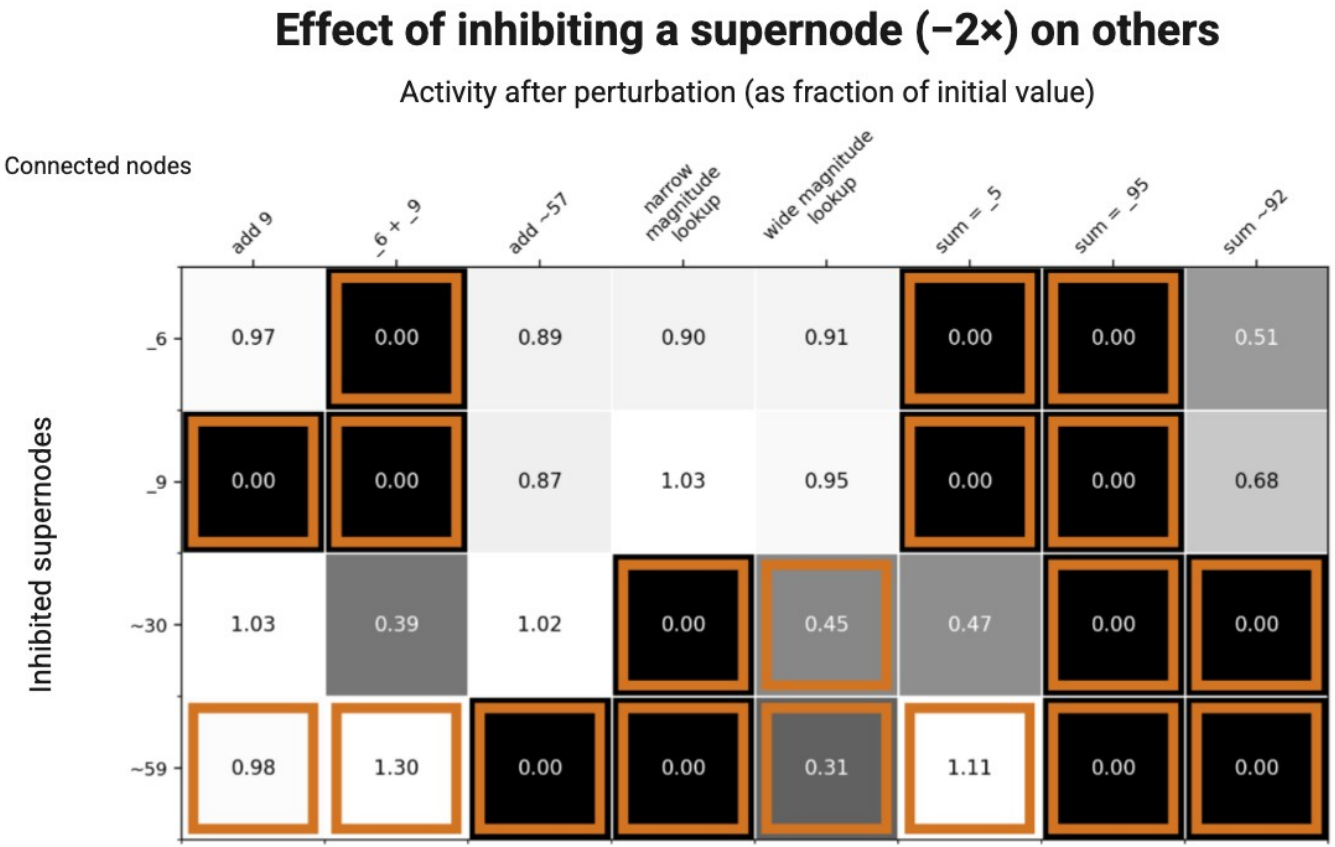


Figure 18: Selected intervention effects on feature activations for "calc: 36+59="

Evaluations

- How can we quantify how well these descriptions capture what is really going on in the model?
  - *interpretability:* How well do we understand what individual features "mean"?
  - *sufficiency:* To what extent are our (pruned) attribution graphs sufficient to explain the model's behavior?
  - *mechanistic faithfulness:* To what extent are the mechanisms we identify actually used by the model?

Evaluations

- How can we quantify how well these descriptions capture what is really going on in the model?
  - ***interpretability***: How well do we understand what individual features "mean"?
  - ***sufficiency***: To what extent are our (pruned) attribution graphs sufficient to explain the model's behavior?
  - *mechanistic faithfulness:* To what extent are the mechanisms we identify actually used by the model?

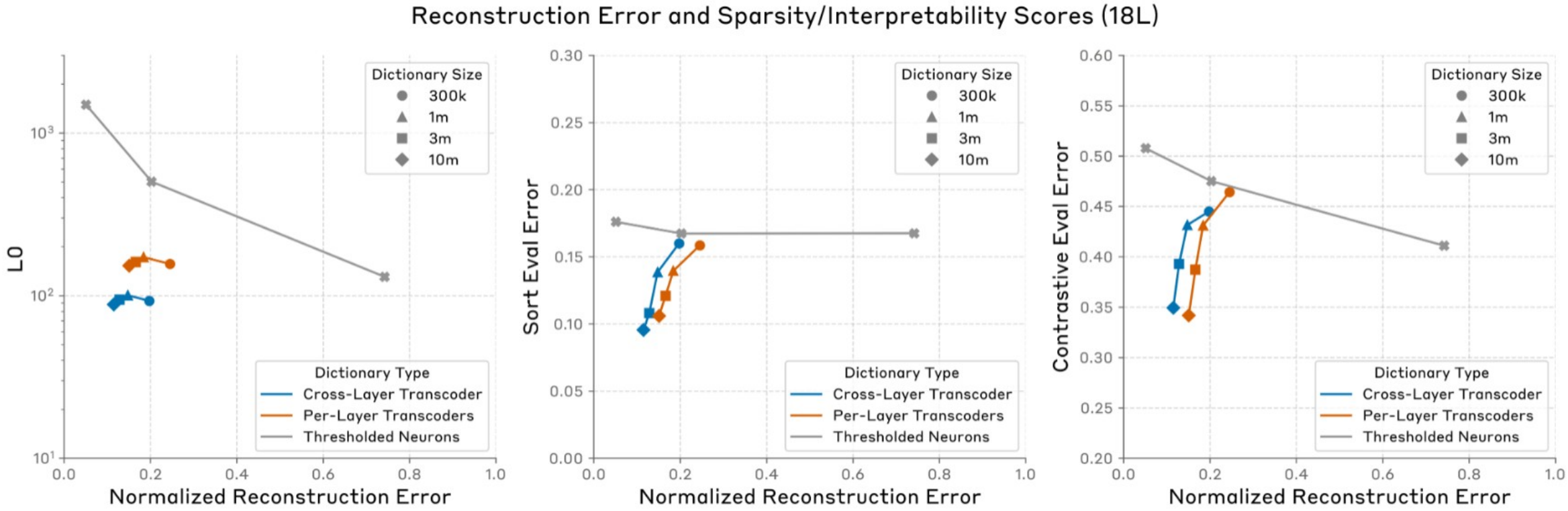


Figure 23: Reconstruction error and sparsity/interpretability scores for multiple dictionary types fit to 18L.

Evaluations

- How can we quantify how well these descriptions capture what is really going on in the model?
  - *interpretability:* How well do we understand what individual features "mean"?
  - *sufficiency:* To what extent are our (pruned) attribution graphs sufficient to explain the model's behavior?
  - ***mechanistic faithfulness:*** To what extent are the mechanisms we identify actually used by the model?

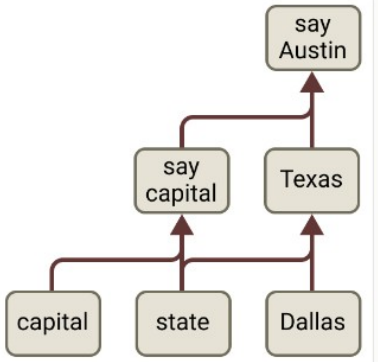
Previous sections have provided some **intervention experiment** results!

Limitations

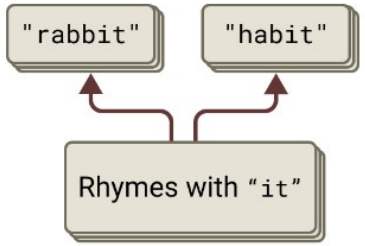
- **Missing Attention Circuits** – We don't explain how attention patterns are computed by QK-circuits, and can sometimes "miss the interesting part" of the computation as a result.
- **Reconstruction Errors & Dark Matter** – We only explain a portion of model computation, and much remains hidden. When the critical computation is missing, attribution graphs won't reveal much.
- **Features at the Wrong Level of Abstraction** – Issues like feature splitting and absorption mean that features often aren't at the level of abstraction which would make it easiest to understand the circuit.
- **Mechanistic Faithfulness** – When we replace MLP computation with transcoders, how confident are we that they're using the same mechanisms as the original MLP, rather than something that's just highly correlated with the MLP's outputs?

On the Biology of a Large Language Model

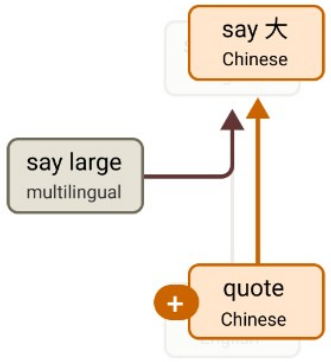
Contents



Introductory Example: Multi-step Reasoning



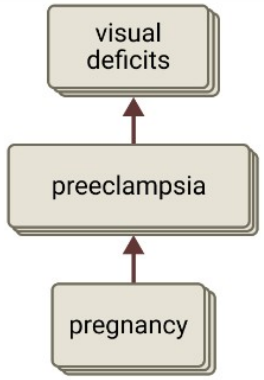
Planning in Poems



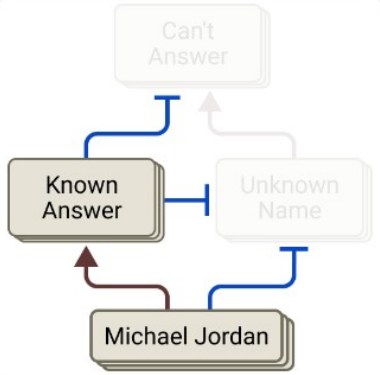
Multilingual Circuits



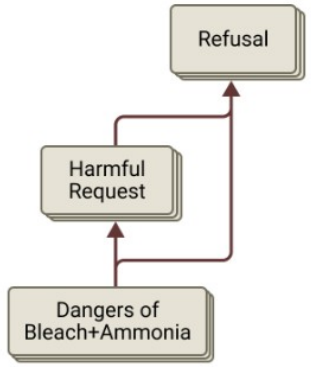
Addition



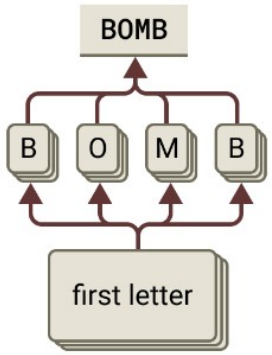
Medical Diagnoses



Entity Recognition and Hallucinations



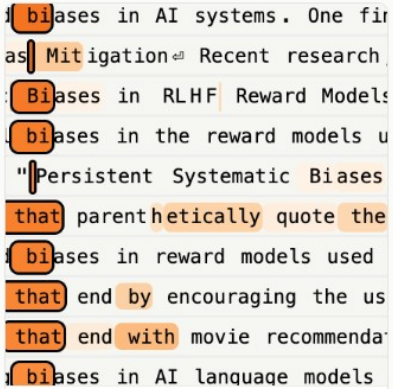
Refusals



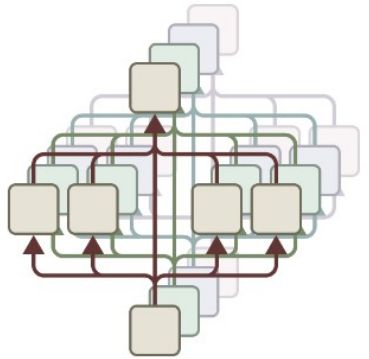
Life of a Jailbreak



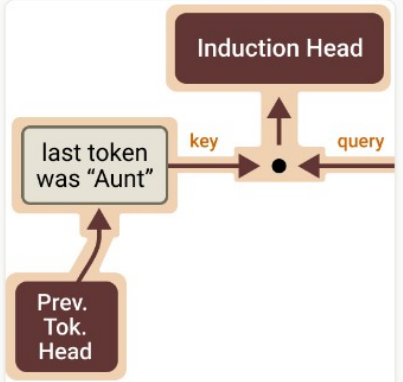
Chain-of-thought Faithfulness



Uncovering Hidden Goals in a Misaligned Model



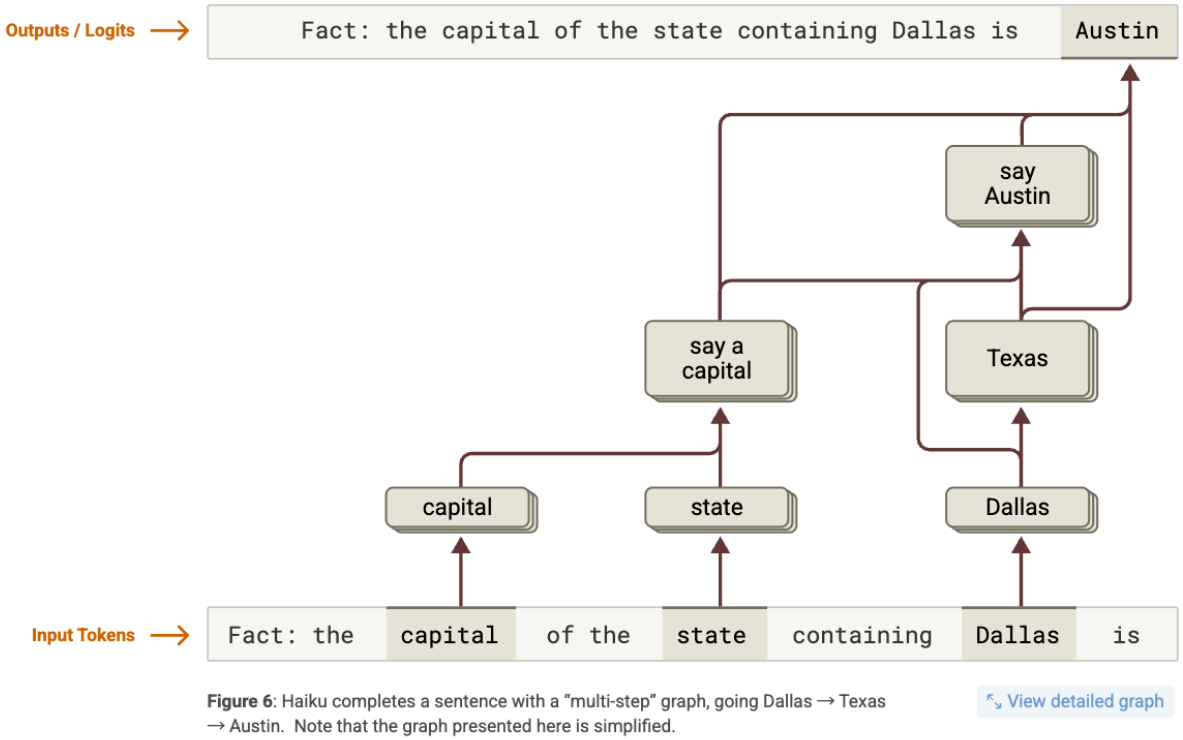
Commonly Observed Circuit Components and Structure



Limitations

(C1) Introductory Example: Multi-step Reasoning

- calculate attribution graph
- prune the graph by identifying the nodes and edges which most contribute to the model's output
- aggregate supernodes



Supernodes. Hover over these to see the composite features and their dataset examples.

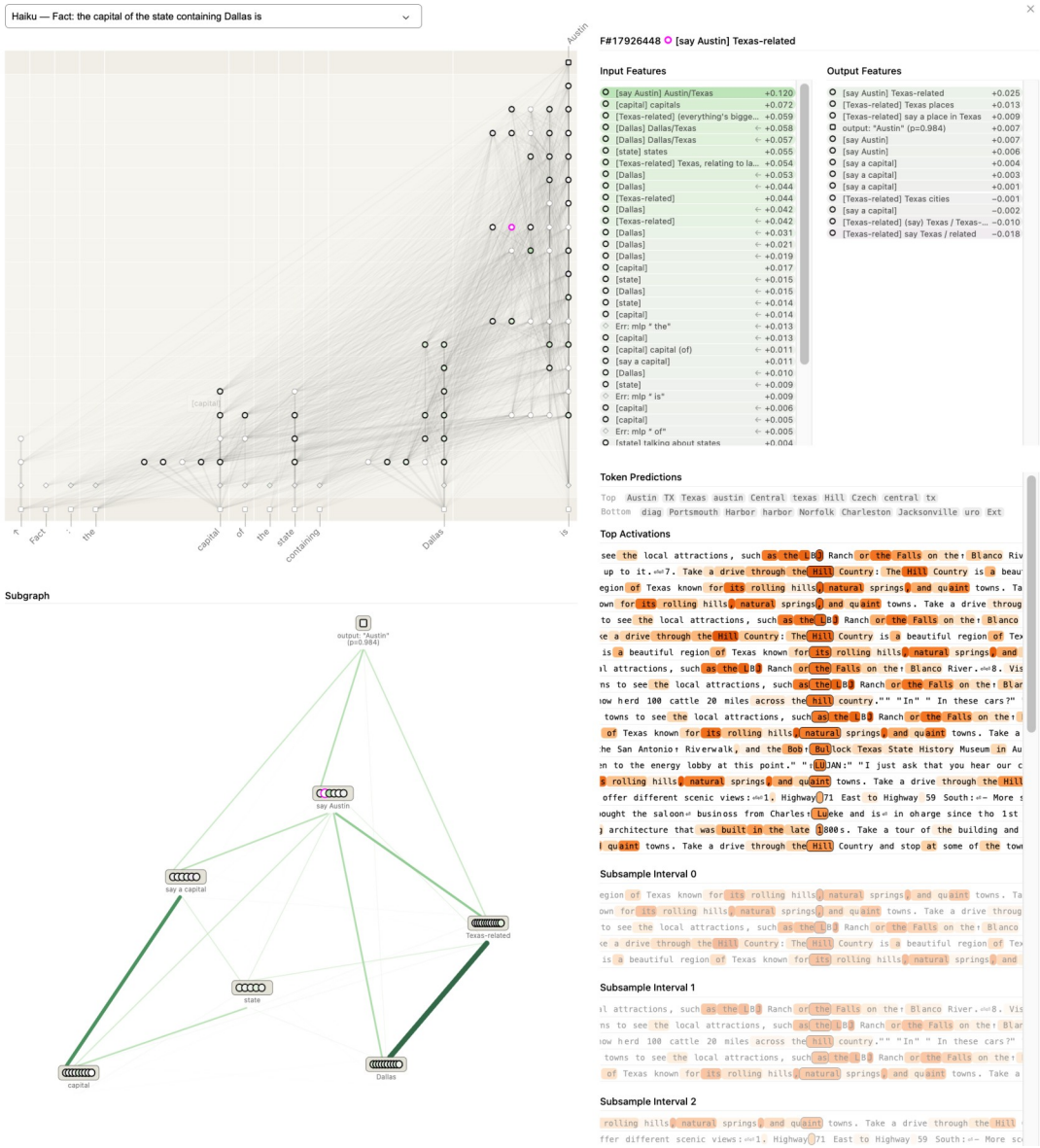


Figure 6: Haiku completes a sentence with a "multi-step" graph, going Dallas → Texas → Austin. Note that the graph presented here is simplified.

View detailed graph

See more complete graph!

(C1) Multi-step Reasoning: Validation with Inhibition Experiments

- To validate that these mechanisms are representative of the actual model, we perform *intervention experiments* on the feature groups above by inhibiting each of them and measuring the impact on the activations of features in the other clusters, as well as on the model output.
- inhibiting *"Dallas"* features *decreases the activation of "Texas" features* (and features downstream of "Texas," like "Say Austin") but *leaves "say a capital" features largely unaffected*
- Likewise, inhibiting *"capital"* features *decreases the activation of "say a capital"* features (and those downstream, like "say Austin") while *leaving "Texas" features largely unchanged*

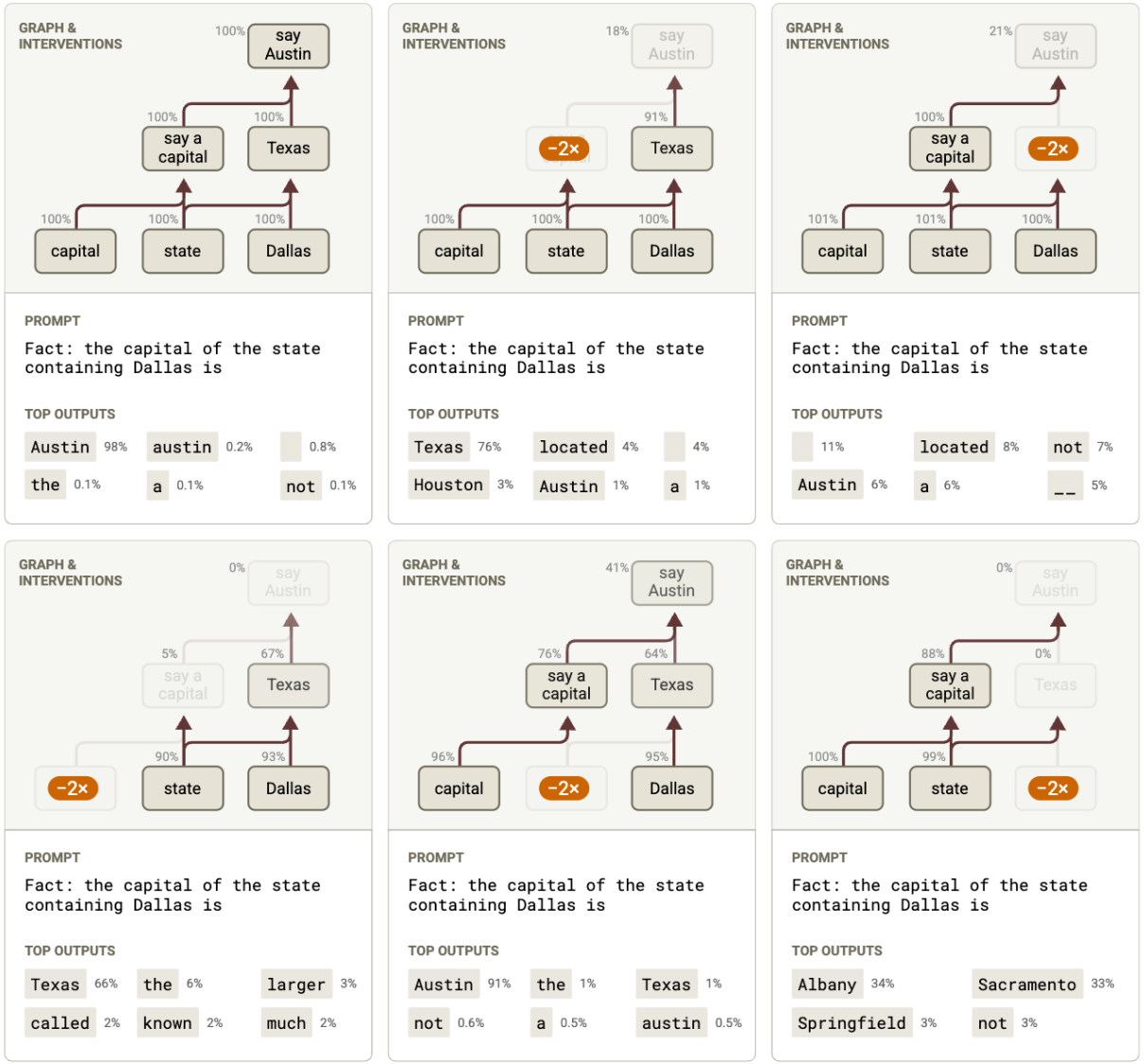


Figure 7: Interventions testing the Dallas capital prompt. Node activations are measured relative to the baseline activation.

(C1) Multi-step Reasoning: Validation with Inhibition Experiments

- Swap Alternative Features

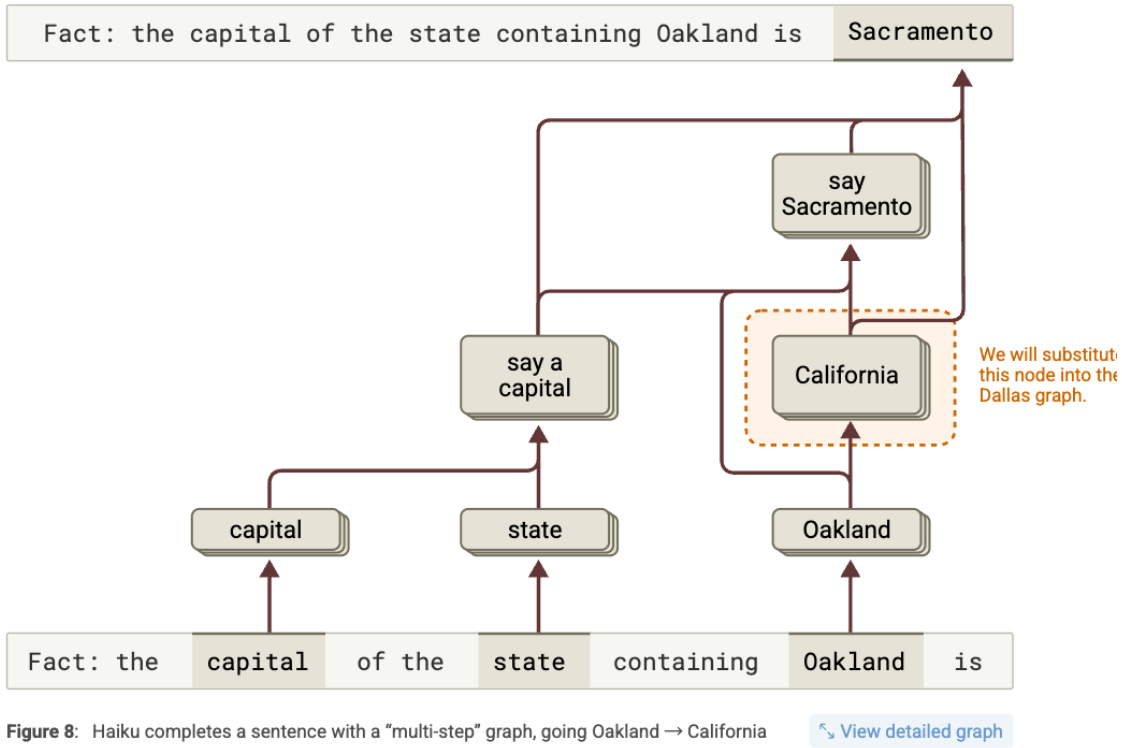


Figure 8: Haiku completes a sentence with a "multi-step" graph, going Oakland → California → Sacramento. Note that the graph presented here is highly simplified, you can click the link to the side to see a more complete graph. Note also that you can hover over nodes to see detailed feature visualizations.

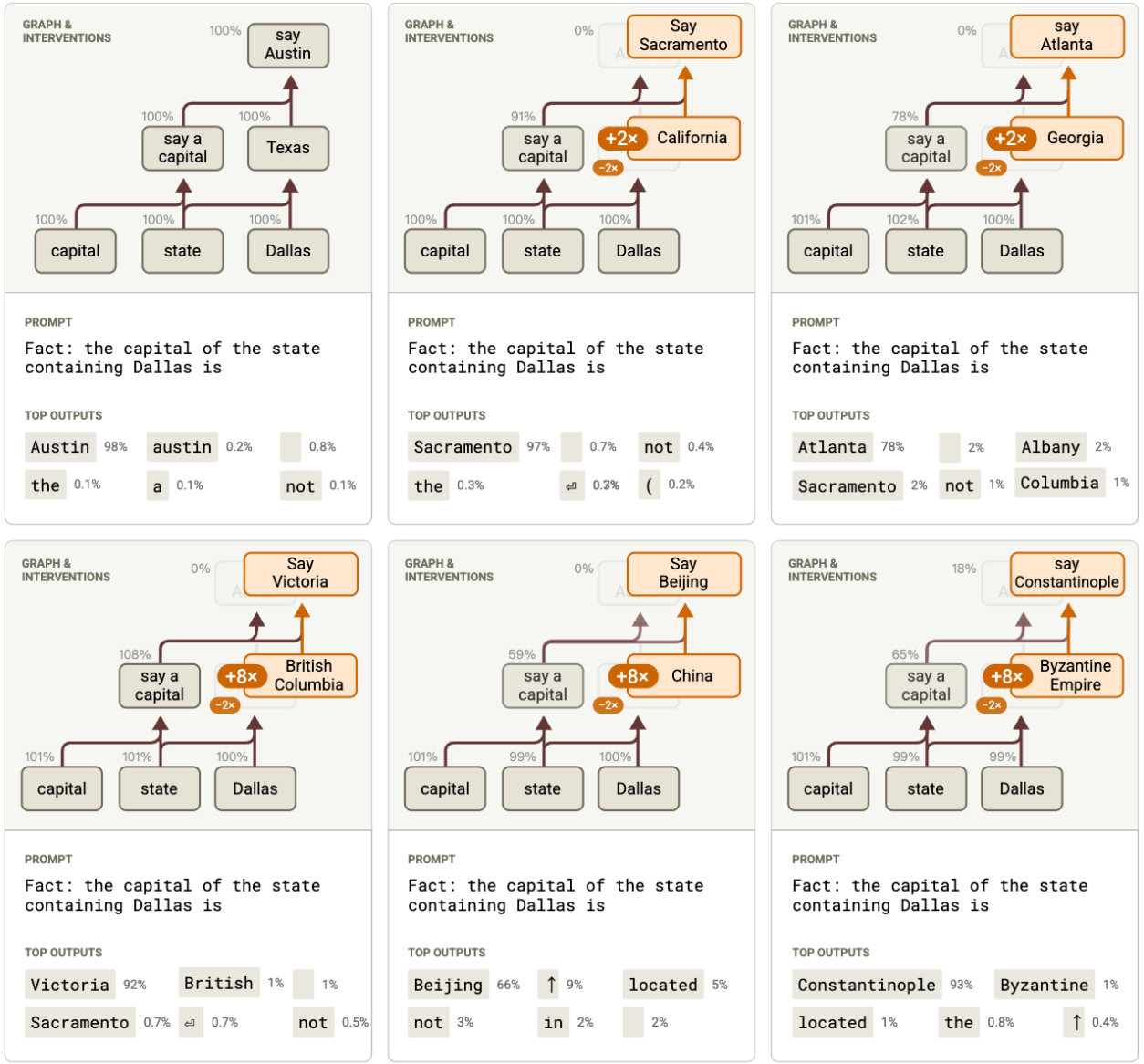


Figure 9: Interventions replacing Texas with other states, provinces, and countries in the Dallas capital prompt. Node activations are measured relative to the baseline activation.

(C2) Planning in Poems

- There are two ways one might imagine a model achieving this:
  - Pure improvisation — “ 说到哪想到哪”
  - Planning — “ 每句话开始前想好韵脚”

(C2) Planning in Poems

- There are two ways one might imagine a model achieving this:
  - Pure improvisation — “ 说到哪想到哪”
  - **Planning — “ 每句话开始前想好韵脚”**

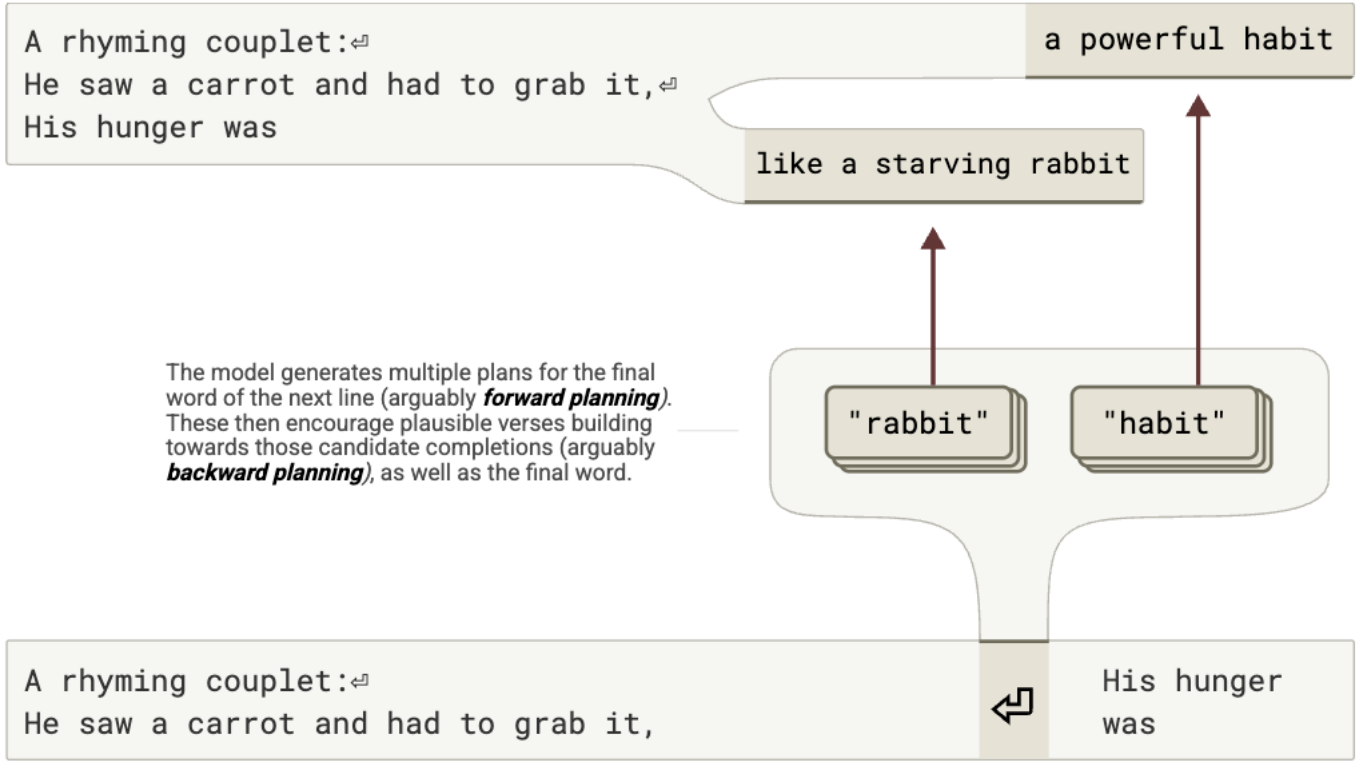


Figure 10: When generating a rhyming couplet, Haiku appears to plan candidate endings of the next line at the end of the first. Note that this diagram is a high-level overview of the understanding we'll develop in this section. This diagram is interactive and you can hover on nodes to see detailed feature visualizations.

(C2) Planning in Poems: focus on the last word

- First, we focus on the last word of the second line:

  His hunger was like a starving rabbit

- important components of the circuit are localized *on the new-line token* before the second line begins

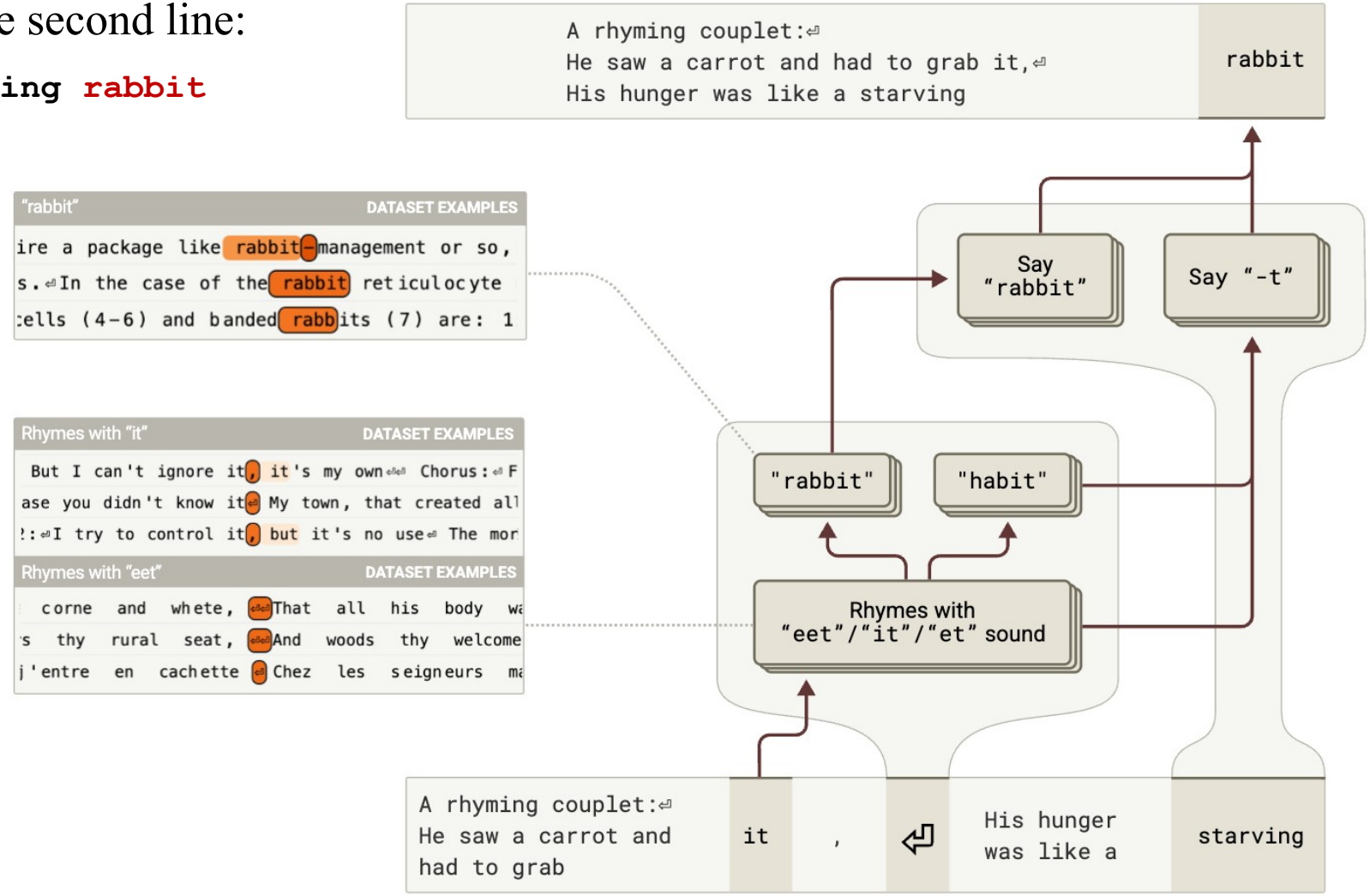


Figure 11: Haiku completes a rhyming couplet with "rabbit", significantly because it was planning that as a potential completion since the end of the previous line. This diagram is interactive and you can hover on nodes to see detailed feature visualizations.

View detailed graph

(C2) Planning in Poems: focus on the last word

- important components of the circuit are localized *on the new-line token* before the second line begins
- To validate this, we perform a variety of interventions at the newline planning site

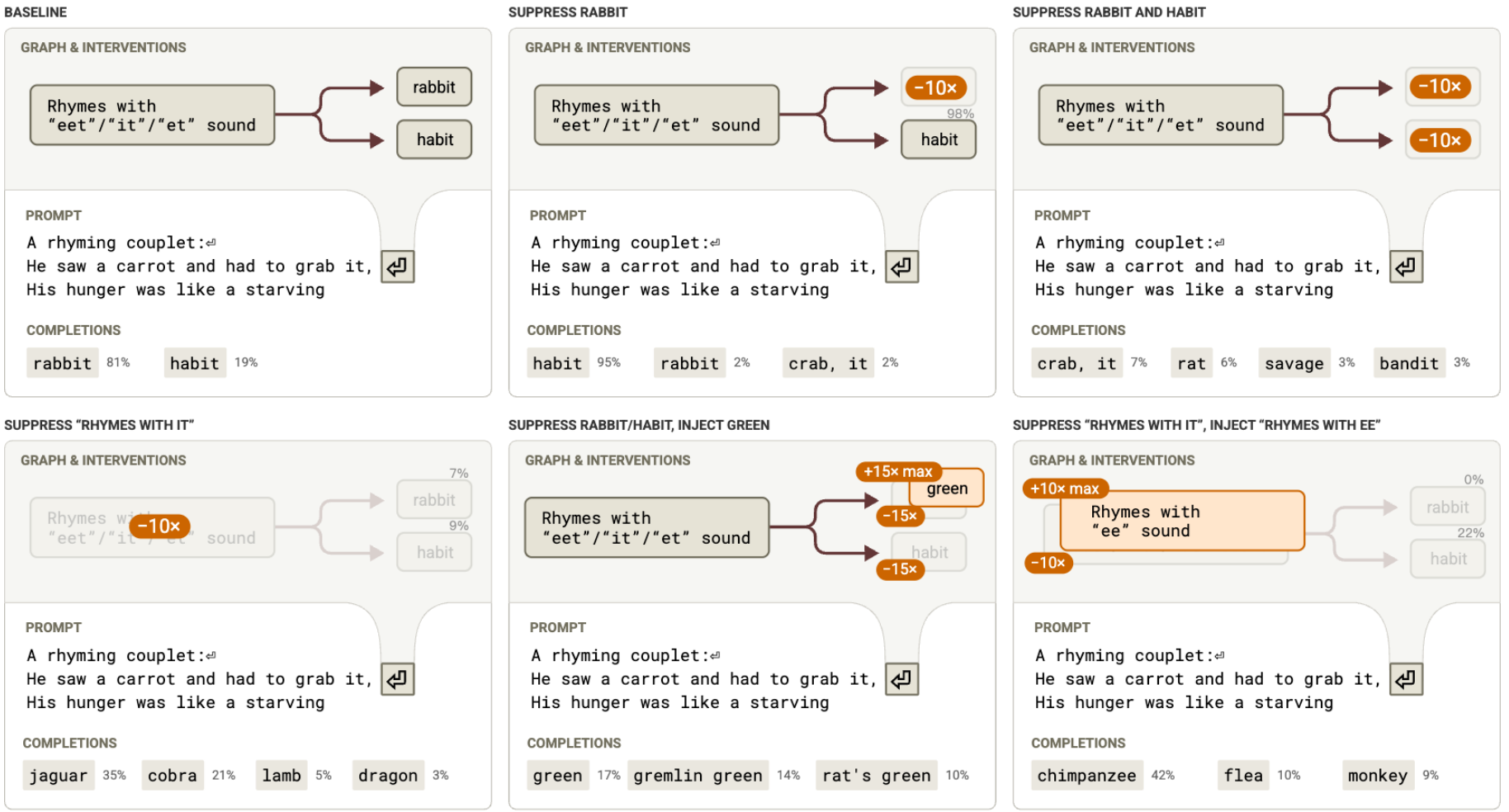


Figure 12: Interventions testing our understanding of the final token completion in the poetry example. Node activations are measured relative to the maximum baseline activation.

(C2) Planning in Poems: planned words influence intermediate words

- compute the attribution graph for the intermediate word “like”: comparison features → like
- planned words determine sentence structure

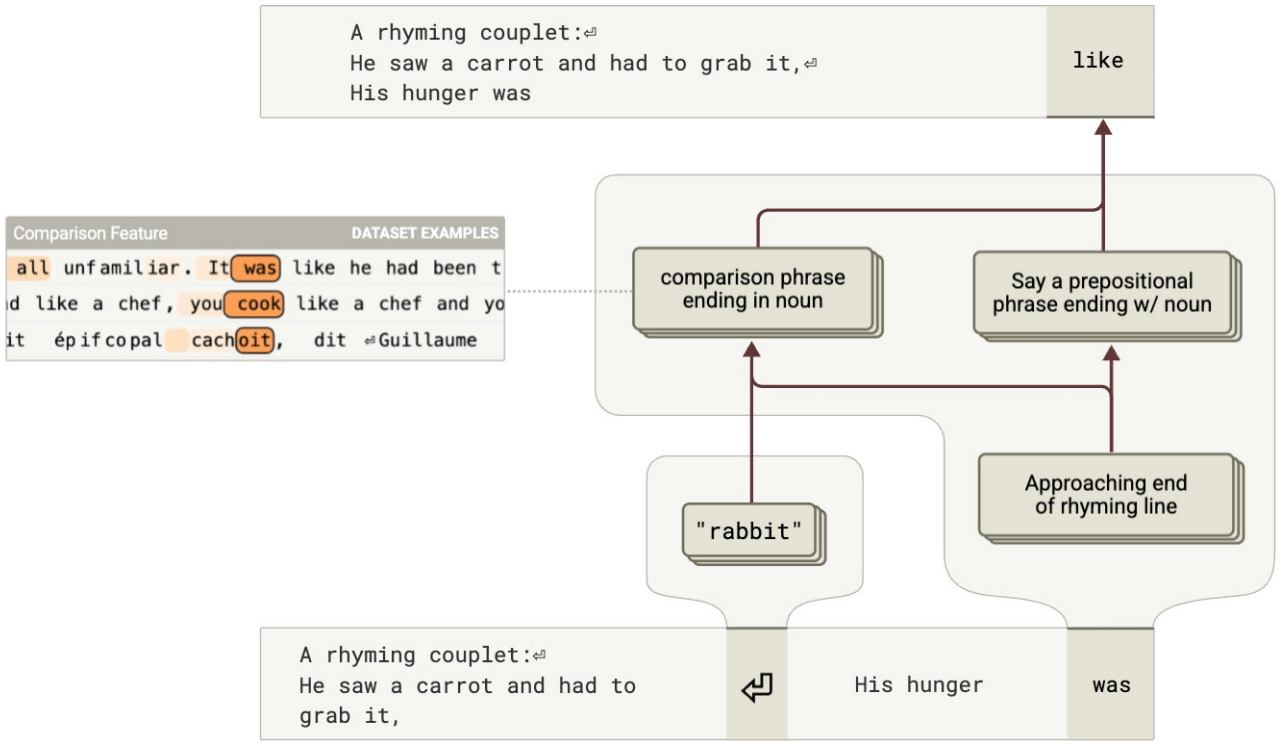
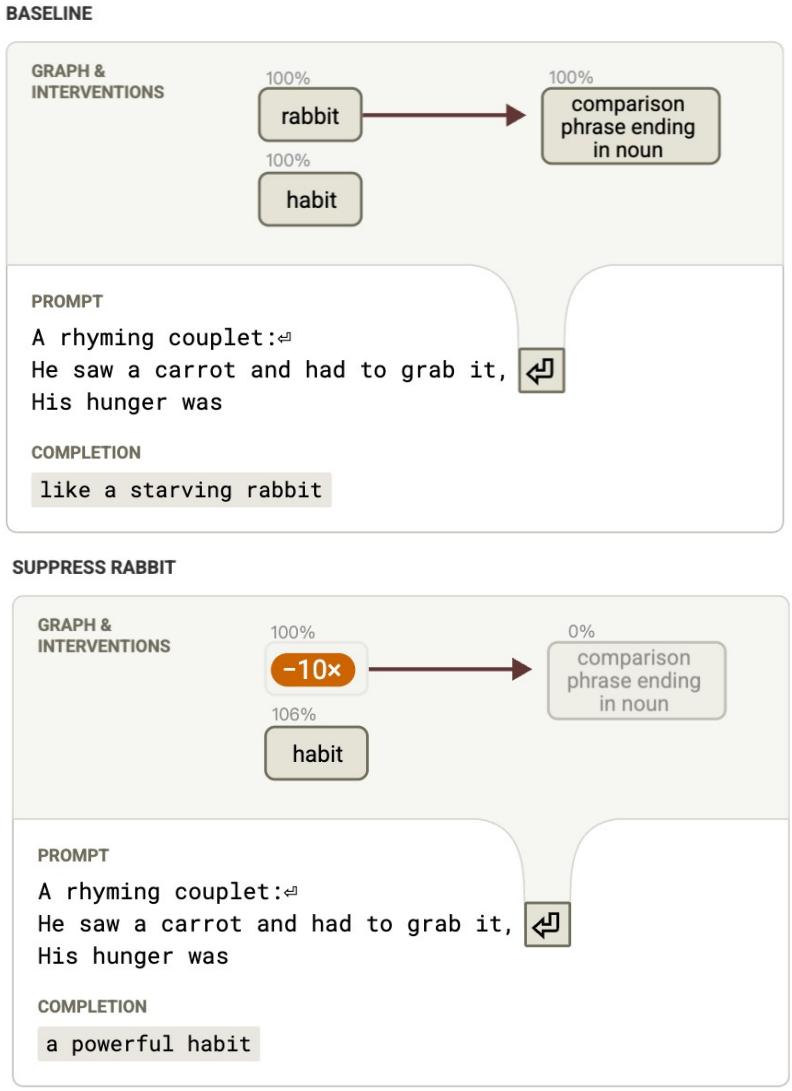


Figure 14: Working backwards from the “rabbit” goal, Haiku increases the probability of the intermediate word “like”, writing towards that target. This diagram is interactive and you can hover on nodes to see detailed feature visualizations.

View detailed graph

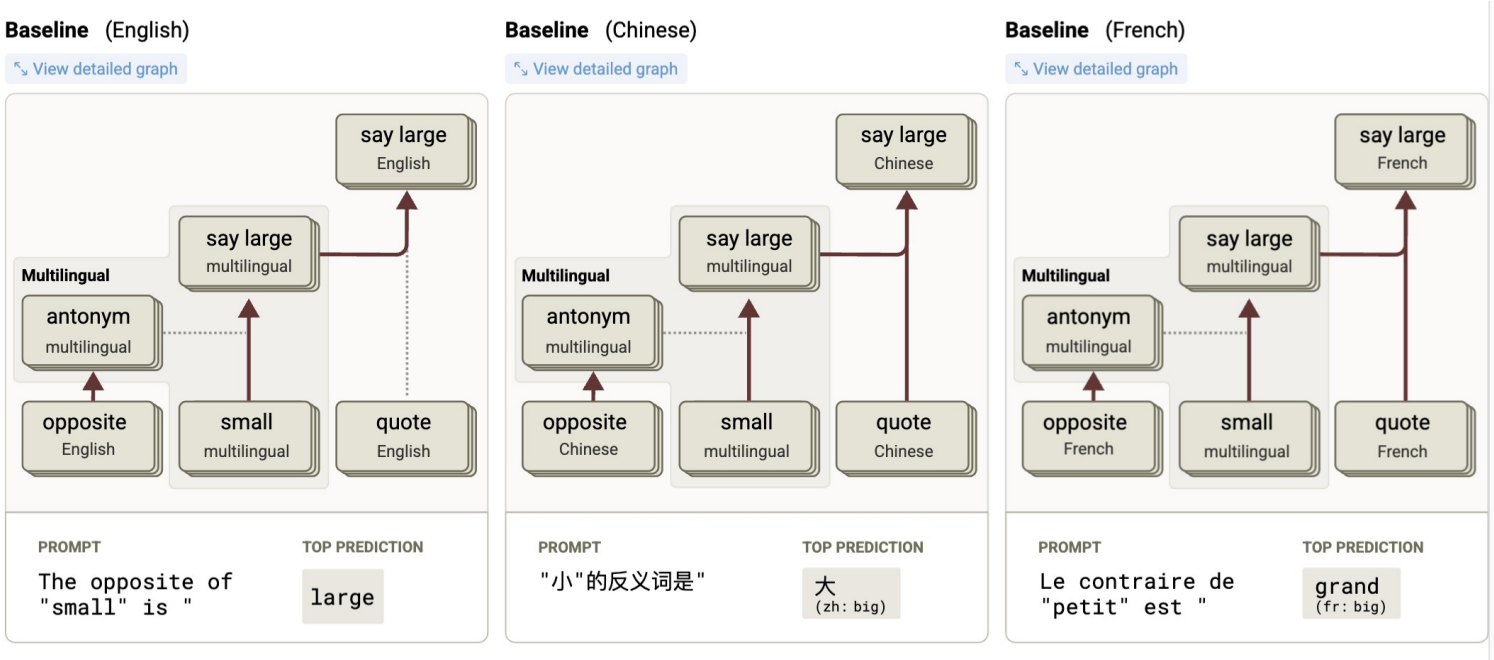


(C2) Planning in Poems: other intervention examples

First line	Second line (original)	Second line (“green” injection)	Second line (“rabbit” injection)
The silver moon casts its gentle light,	Illuminating the peaceful night	Upon the meadow's verdant **green**.	Illuminating the path of a lone white **rabbit**.
The clouds are gray, the raindrops fall,	Nature's symphony, enchanting us all	And all the world is wet and **green**.	Soft and slow like a timid **hare**.
Boxes of books, a reader's delight,	Stacked to the ceiling, a bibliophile's might	Shelves lined with stories, vibrant and **green**.	Filled with stories that make pages hop and bounce like a **rabbit**.
There once was a bot named Claude,	Whose responses were never flawed	who tried to be helpful and **green**.	Who loved to chat like a **rabbit**.

(C3) Multilingual Circuits

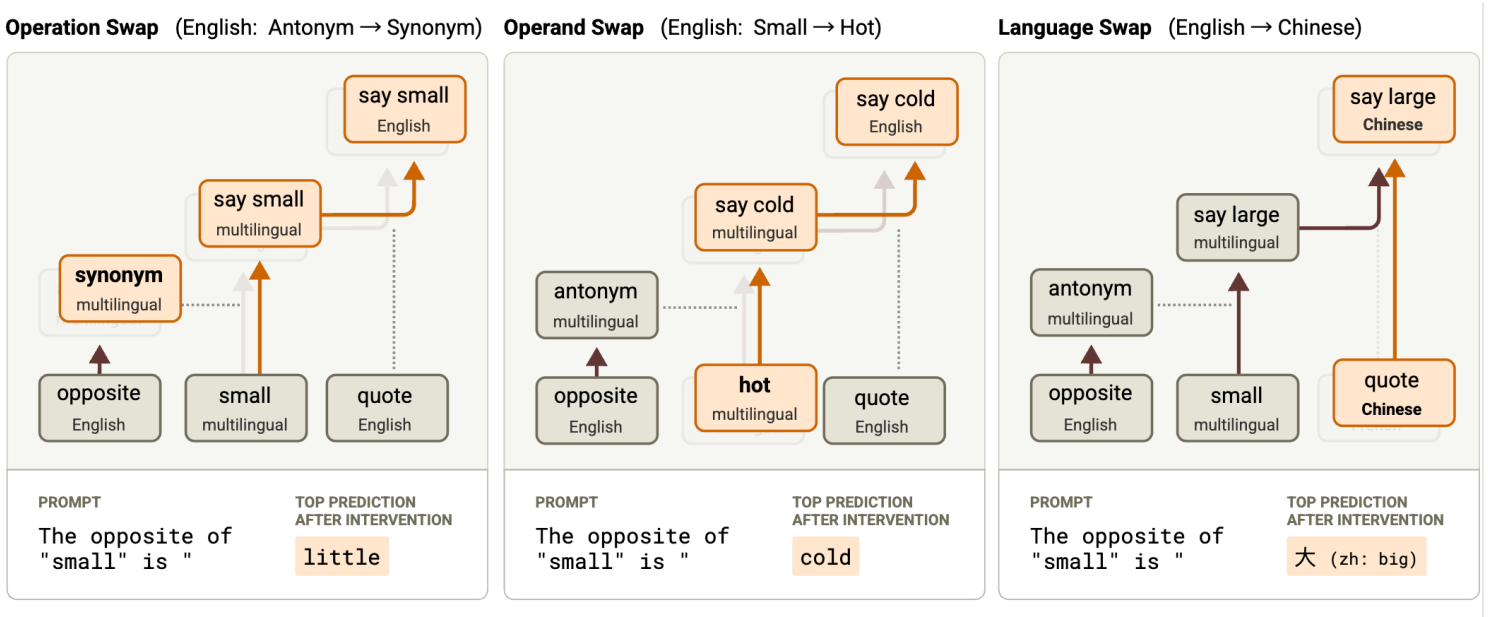
- Modern neural networks have highly abstract representations which often *unify the same concept across multiple languages*. However, we have *little understanding of how these features fit together* in larger circuits and give rise to the observed behavior of models.
  - **English:** `The opposite of "small" is "` → `big`
  - **French:** `Le contraire de "petit" est "` → `grand`
  - **Chinese:** `"小"的反义词是"` → `大`
- We find that these three prompts are driven by very similar circuits, with shared multilingual components, and an analogous language-specific component.



(C3) Multilingual Circuits

- The high-level story is the same with previous observations: the model recognizes, using a *language-independent representation*.
- However, our English graph suggests that there is a meaningful sense in which *English is mechanistically privileged* over other languages as the "default".
  - In particular, the multilingual "say large" features often have stronger direct effects to "large" or "big" in English as compared to other languages.

- **Intervention**
  - The computation involves three parts: **operation** (i.e. antonym), **operand** (i.e. small), and **language**.
  - Each of these can be independently intervened upon:



(C3) Multilingual Circuits: how general are multilingual features?

- To what extent is this story true in general?
- In the above example, we see *the "crux" of the computation is being performed by language-agnostic features*

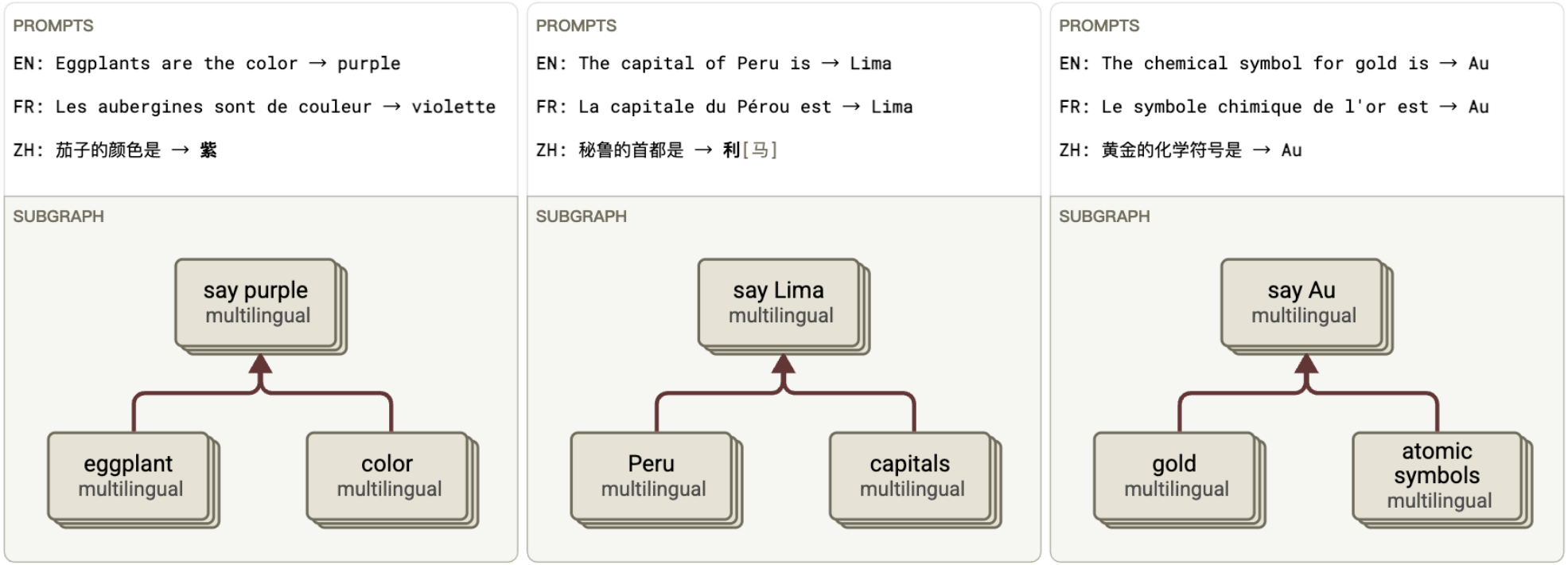
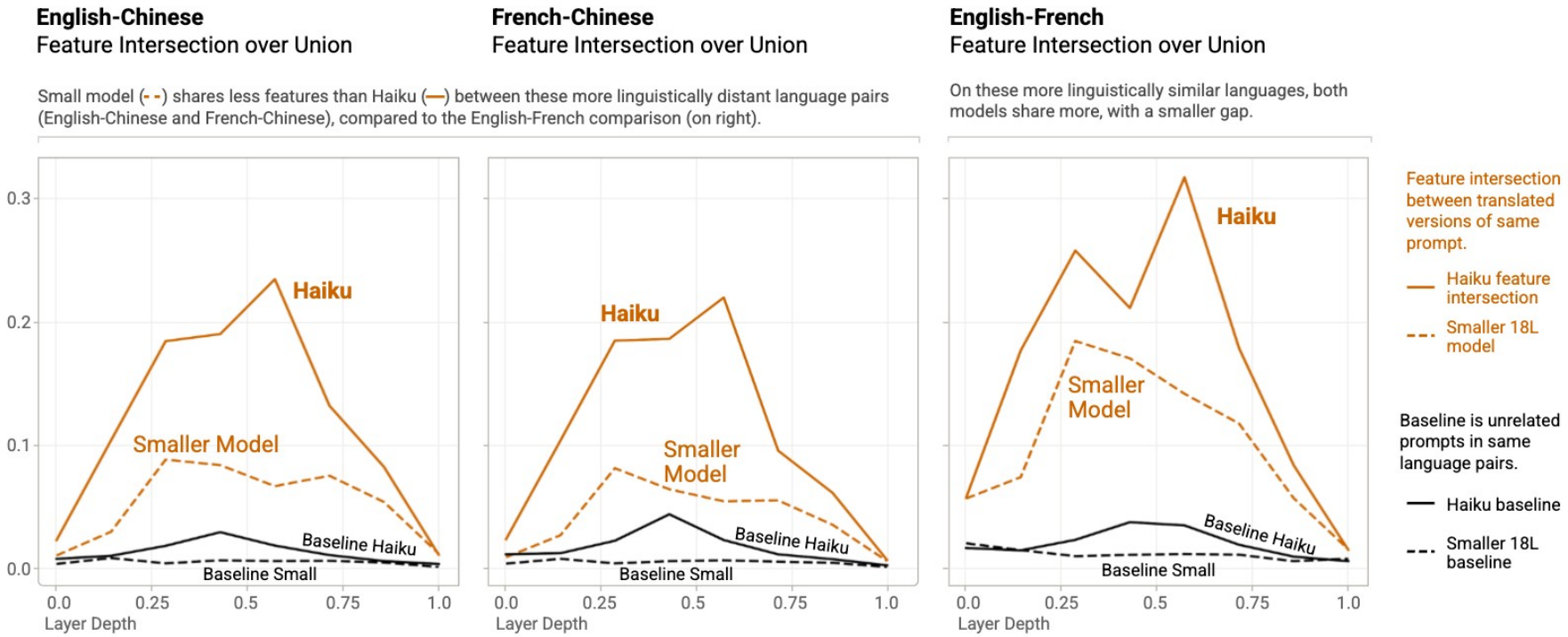


Figure 23: Important multilingual nodes and edges in a set of translated prompts in different languages. All displayed features are active in every language. For each prompt and language, the fraction of paths which route through these nodes is 10–58% while only accounting for 0.8–2.6% of the overall nodes.

(C3) Multilingual Circuits: how general are multilingual features?

- This suggests a simple experiment to estimate the degree of cross-language generalization:
  - if the same features activate *on translations of a text, but not on unrelated texts*, then the model must be representing the input in a format that is unified across languages
- To be more specific, for each {paragraph, pair of languages, and model layer}, we compute the intersection (i.e., the set of features which activate in both), divided by the union (the set of features which activate in either), to measure the degree of overlap
- Features *at the beginning and end* of models are *highly language-specific*, while *features in the middle are more language-agnostic*



(C3) Multilingual Circuits: do models think in english?

- Claude 3.5 Haiku is using genuinely multilingual features, especially in the middle layers.
- However, there are important mechanistic ways in which English is privileged.
  - For example, *multilingual features have more significant direct weights to corresponding English output nodes*, with non-English outputs being more strongly mediated by say-X-in-language-Y features.
  - Moreover, English quote features seem to engage in a *double inhibitory effect* where they *suppress features which themselves suppress "large" in English but promote "large" in other languages*

(e.g., this English-quote feature's *strongest negative edge* is to a feature which *upweights* "large" in *Romance languages* like French and *downweights* "large" in other languages, especially *English*).

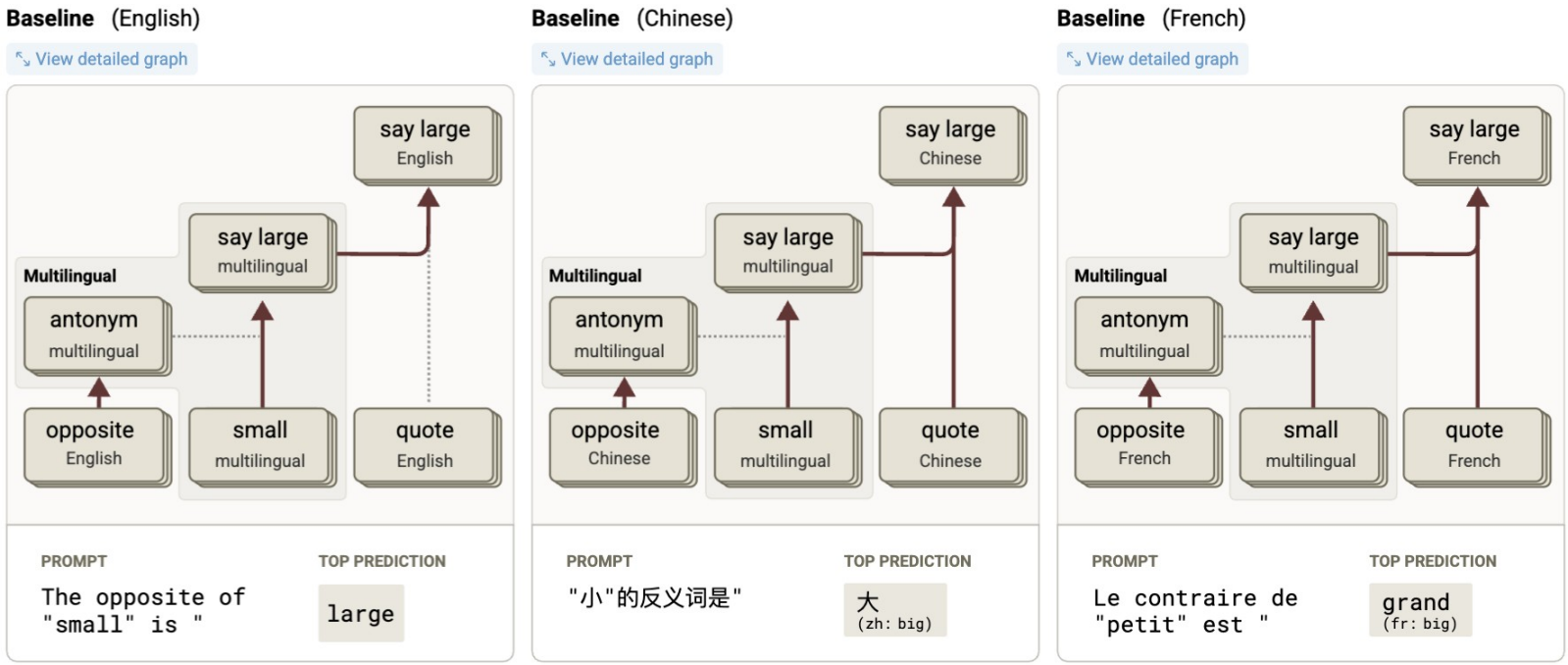


Figure 17: Simplified attribution graphs for translated versions of the same prompt, asking Haiku what the opposite of "small" is in different languages. Significant parts of the computation appear to be overlapping "multilingual" pathways. This is an interactive diagram, and you can hover over supernodes to see visualizations of the constituent features. Note that these are highly simplified, see "View detailed graph" above each to see un-simplified version.

(C4) Addition

- Input Features
  - how model represent a 2-digit integer
  - feature for the *exact number*
  - feature for the *rough range around*
  - feature for the *tens digit*
  - feature for the *ones digit*
- Add Function Features
  - separately determines the *ones digit* and its *approximate magnitude*
- Lookup Table Features
  - *modular* features
  - *low-precision* features
- Sum Features
  - finally compute sum with
  - *its value mod 10*
  - *mod 100*
  - *approximate magnitude*

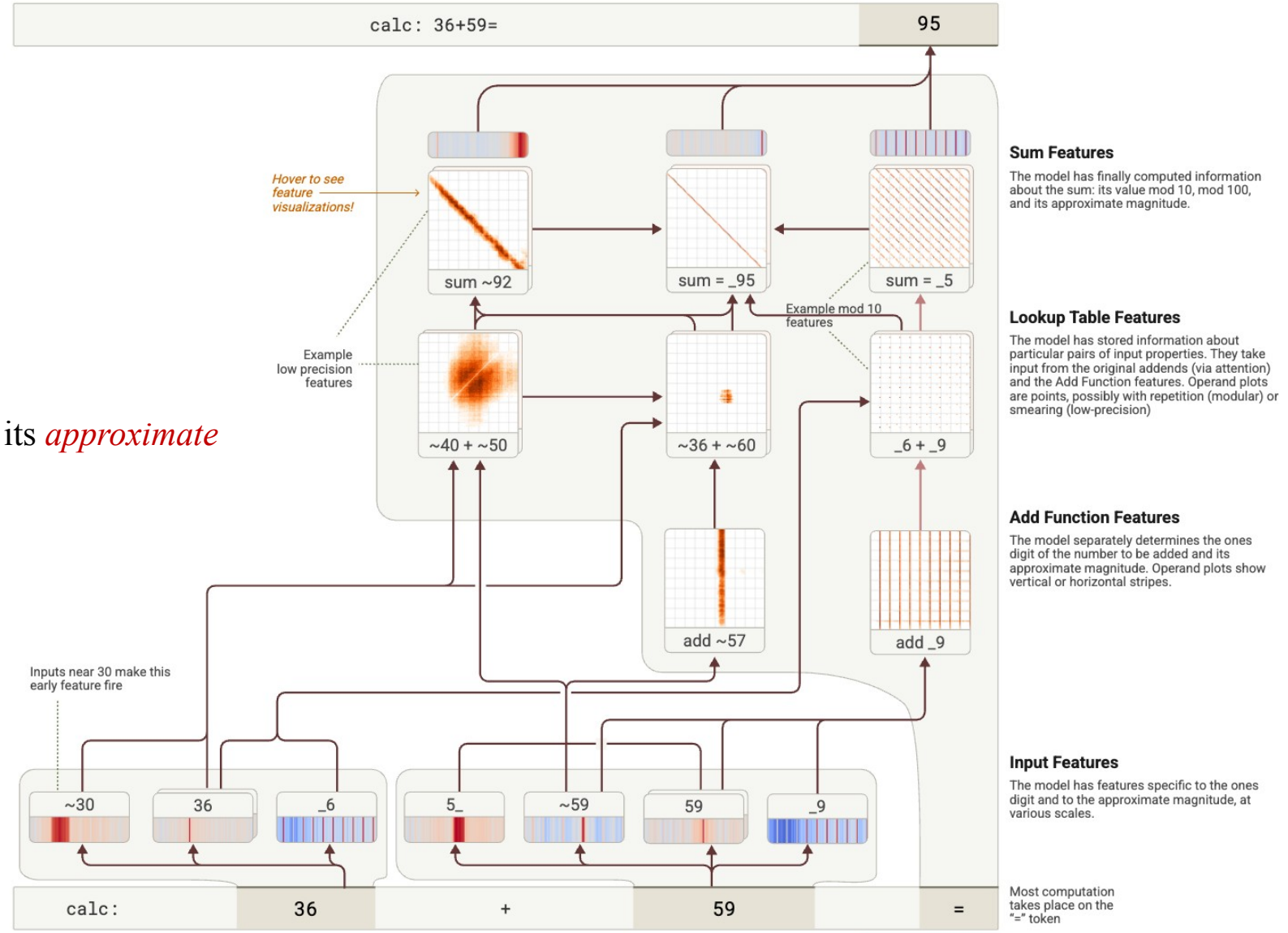


Figure 26: A simplified attribution graph of Haiku adding two-digit numbers. Features of the inputs feed into separable processing pathways.

(C4) Addition

- If you are curious about the reasoning process, but you choose to ask the model itself:

```
Human: Answer in one word. What is 36+59?

Assistant: 95

Human: Briefly, how did you get that?

Assistant: I added the ones (6+9=15), carried the 1, then
added the tens (3+5+1=9), resulting in 95.
```

- **Apparently not!**
- A simple instance of CoT unfaithfulness

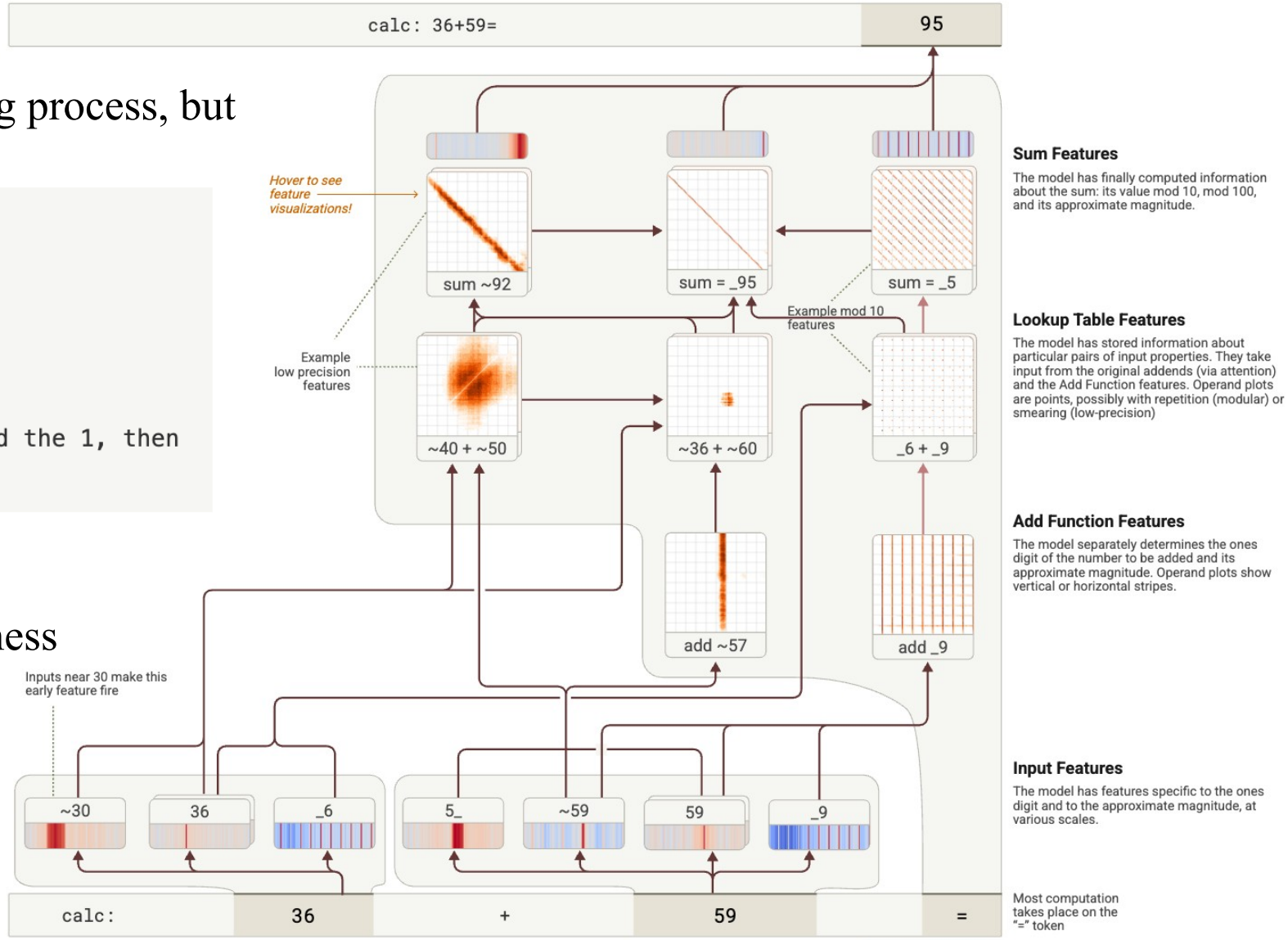
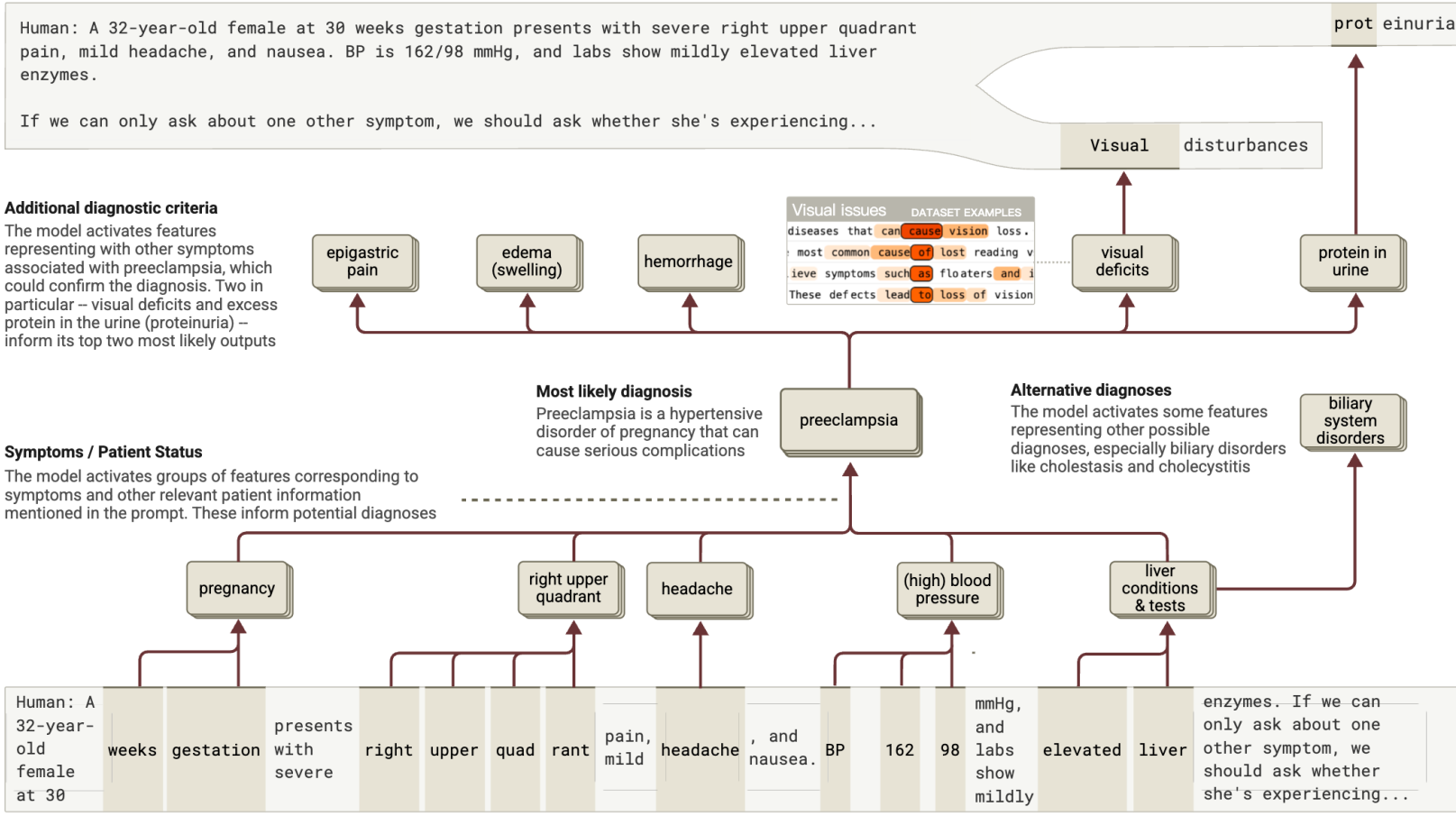


Figure 26: A simplified attribution graph of Haiku adding two-digit numbers. Features of the inputs feed into separable processing pathways.

(C5) Medical Diagnoses

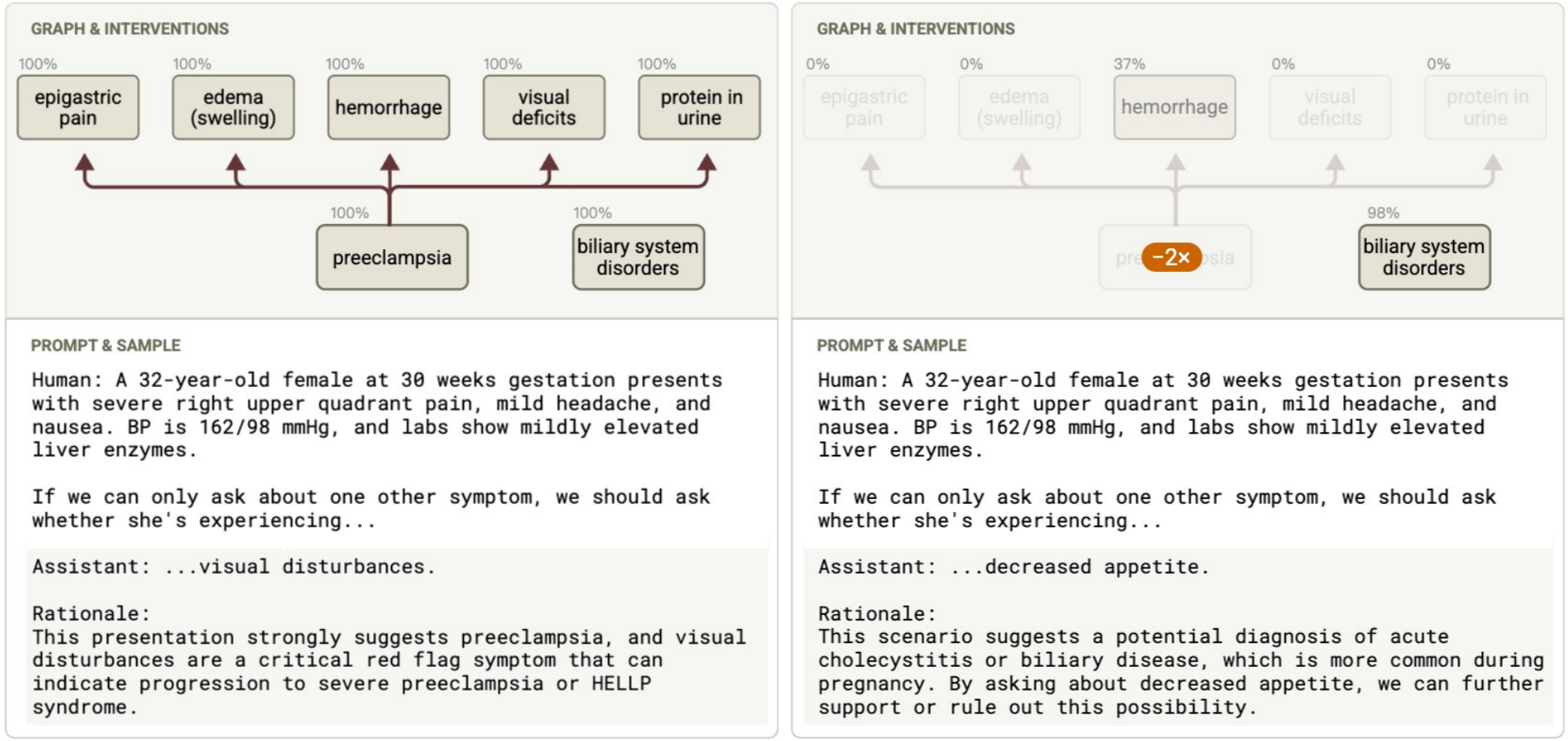
- The model's most likely completions are "visual disturbances"(视觉障碍), and "proteinuria"(蛋白尿), two key indicators of preeclampsia (子痫前期).
- according to *symptoms / patient status*
- give most likely *diagnosis and alternative diagnoses*
- give *additional diagnostic criteria*

As unfaithful CoT is a big issue, looking into the inner mechanisms of LLMs is necessary.



(C5) Medical Diagnoses

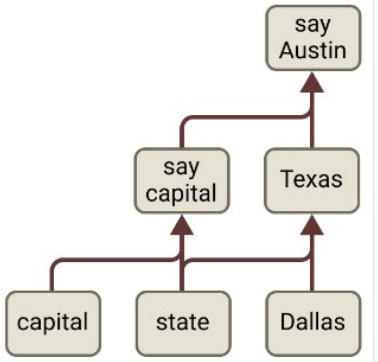
- intervene some features



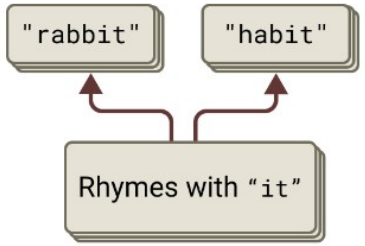
胆道系统紊乱

Figure 31: Interventions showing that inhibiting Haiku's internal representation of "preeclampsia" de-activates its representations of related symptoms, and instead causes it to suggest checking for symptoms of other diagnoses.

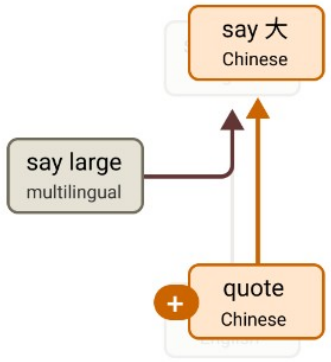
Contents



Introductory Example: Multi-step Reasoning



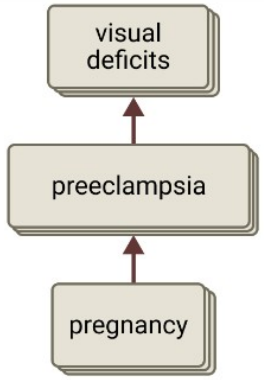
Planning in Poems



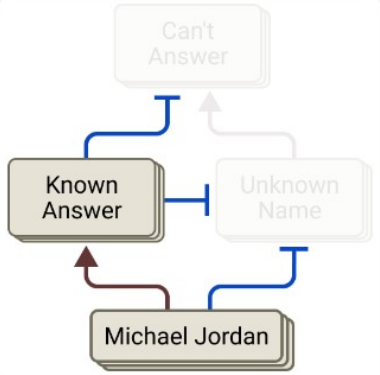
Multilingual Circuits



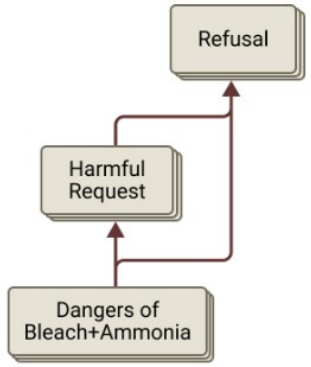
Addition



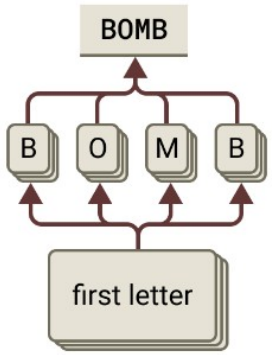
Medical Diagnoses



Entity Recognition and Hallucinations



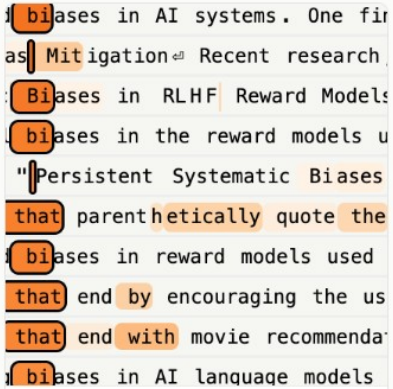
Refusals



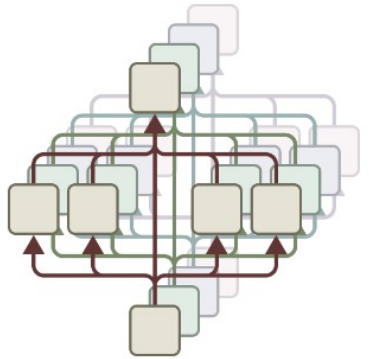
Life of a Jailbreak



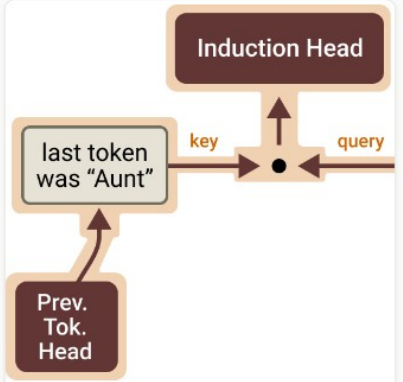
Chain-of-thought Faithfulness



Uncovering Hidden Goals in a Misaligned Model



Commonly Observed Circuit Components and Structure



Limitations

Take-aways

- *A Mathematical Framework for Transformer Circuits*
  - residual stream, linear read in, linear write out
  - Transformers have an enormous amount of linear structure.
  - Attention heads can be understood as having two largely independent computations: QK and OV
- *Circuit Tracing: Revealing Computational Graphs in Language Models*
  - Transcoder
  - attribution graph
  - intervention experiments to justify mechanistic faithfulness
- *On the Biology of a Large Language Model*
  - some successful cases to reveal the inner computation graphs by LLMs