Automatically Identifying Local and Global Circuits with Linear Computation Graphs

Xuyang Ge [1] Fukang Zhu [1] Wentao Shu [1] Junxuan Wang [1] Zhengfu He [1] Xipeng Qiu [1]

Abstract

Circuit analysis of any certain model behavior is a central task in mechanistic interpretability. We introduce our circuit discovery pipeline with sparse autoencoders (SAEs) and a variant called transcoders. With these two modules inserted into the model, the model's computation graph with respect to OV and MLP circuits becomes strictly linear. Our methods do not require linear approximation to compute the causal effect of each node. This fine-grained graph enables identifying both end-to-end and local circuits accounting for either logits or intermediate features. We can scalably apply this pipeline with a technique called Hierarchical Attribution. We analyze three kind of circuits in GPT2-Small, namely bracket, induction and Indirect Object Identification circuits. Our results reveal new findings underlying existing discoveries.

1. Introduction

Recent years have seen the rapid progress of mechanistically reverse engineering Transformer language models (Vaswani et al., 2017). Conventionally, researchers seek to find out how neural networks organize information in its hidden activation space (Olah et al., 2020a; Gurnee et al., 2023; Zou et al., 2023) (i.e. features) and how learnable weight matrices connect and (de)activate them (Olsson et al., 2022; Wang et al., 2023; Conmy et al., 2023) (i.e. circuits). One fundamental problem of studying attention heads and MLP neurons as interpretability primitives is their polysemanticity, which under the assumption of linear representation hypothesis is mostly due to superposition (Elhage et al., 2022; Larson, 2023; LaurenGreenspan & keith_wynroe, 2023). Thus there is no guarantee to explain how these components impact model behavior out of the interested distribution. Additionally, circuit analysis based on attention heads is coarse-grained in that it lacks effective methods to explain the intermediate activations.

Probing (Alain & Bengio, 2017) in the activation for a more fine-grained and monosemantic unit has succeeded in discovering directions indicating a wide range of abstract concepts like truthfulness (Li et al., 2023) and refusal of AI assistants (Zou et al., 2023; Arditi et al., 2024). However, this supervised setting may not capture features we did not expect to present.

Sparse Autoencoders (SAEs) (Bricken et al., 2023; Cunningham et al., 2023) have shown its potential in extracting features from superposition in an unsupervised manner. This opens up a new perspective of understanding model internals by interpreting the activation of SAE features. It also poses a natural research question: **how to gracefully leverage SAEs for circuit analysis?** Compared to prior work along this line (Cunningham et al., 2023; He et al., 2024; Marks et al., 2024), our main contributions are as follows.

- We propose to utilize Transcoders, a variant of Sparse Autoencoders to sparsely approximate the computation of MLP layers. This extends the linear analysis of Transformer circuits (Elhage et al., 2021; He et al., 2024).
- For a given input, OV + Skip SAE (i.e. MLP) circuits strictly form a **Linear Computation Graph** without linear approximation of any non-linear function. This precious linearity enables circuit discovery and evaluation with only one forward and one backward.
- We propose *Hierarchical Attribution* to isolate a subgraph of the aforementioned linear graph in an automatic and scalable manner.
- We present a specific example that our analysis offers more detailed insight into how each single SAE feature contributes to a desired behavior e.g. forms a crucial QK attention, or linearly activate a subsequent node in the computation graph. Such observations are not reported by existing work studying circuits in coarser granularity.

[*]Equal contribution [1]Open-MOSS Team, Fudan University, Shanghai, China. Correspondence to: Xipeng Qiu <xpqiu@fudan.edu.cn>.

Copyright 2024 by the author(s).

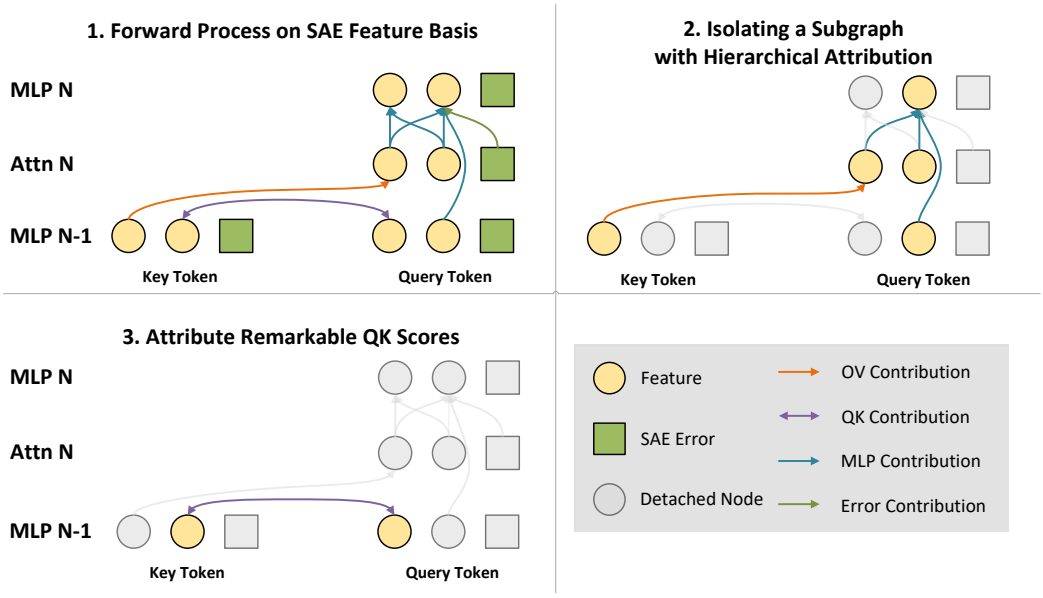


Figure 1. Overview of our method. For a given input, we (1) run forward process once with MLP computation replaced by Transcoders. (2) Then a subgraph is isolated for a given input with *Hierarchical Attribution* in one backward. (3) We then interpret important QK attention involved in the identified circuit.

2. Linear Computation Graphs Connecting SAE Features

2.1. Sparse Autoencoder Features as Analytic Primitives

Sparse Autoencoder (SAE) is a recently emerging method to take features of model activation out of superposition (Elhage et al., 2022). Existing work has suggested empirical success in the interpretability of SAE features concerning both human evaluation (Bricken et al., 2023) and automatic evaluation (Bills et al., 2023).

Concretely, an SAE and its optimization objective can be formalized as follows:

$$\begin{aligned} f &= \mathrm{ReLU}(W_E x + b_E) \\ \hat{x} &= W_D f \\ \mathcal{L} &= \|x - \hat{x}\|_2^2 + \lambda \|f\|_1, \end{aligned} \tag{1}$$

where $W_E \in \mathbb{R}^{d_{\text{SAE}} \times d_{\text{model}}}$ is the SAE encoder weight, $b_E \in \mathbb{R}^{d_{\text{SAE}}}$ encoder bias, $W_D \in \mathbb{R}^{d_{\text{model}} \times d_{\text{SAE}}}$ decoder weight, $x \in \mathbb{R}^{d_{\text{model}}}$ input activation. λ is the coefficient of L1 loss for balance between sparsity and reconstruction. We refer readers to Appendix A for implementation details.

We train Sparse Autoencoders on GPT-2 (Radford et al., 2019) to decompose *all modules that write into the residual stream* (i.e. Word Embedding, Attention output and MLP output). Then, we can derive how a residual stream activation is composed of SAE features:

$$x = \sum_{\mathcal{S} \in \text{Upstream SAEs}} \left(\sum_{i=1}^{d_{\text{SAE}}} f_i^{\mathcal{S}} W_{Di}^{\mathcal{S}} + \varepsilon^{\mathcal{S}} \right) + p, \tag{2}$$

where $f_i^{\mathcal{S}}$ and $\varepsilon^{\mathcal{S}}$ are feature activation and SAE error term of each upstream SAE $\mathcal{S}$. p is the positional embedding of the current token. Since all submodules read and write into the residual stream, such a partition is crucial to connect upstream SAE features to downstream ones.

2.2. Tackling MLP Non-linearity with Transcoders

The denseness and non-linearity of MLP in Transformers make sparse attribution of MLP features difficult. Since MLP activation functions have a privileged basis (Elhage et al., 2023), computation of MLP non-linearity must go through such an orthogonal basis of the MLP hidden space. There is no guarantee of observing sparse and informative correspondence between MLP neurons and learned SAE features. This annoying non-linearity cuts off the connection of upstream SAE features and MLP output (with linear algebraic operations).

To tackle this problem, we develop a new method called Transcoders to get around the MLP non-linearity. Transcoders are generalized forms of SAEs, which decouple the input and output of SAEs and allow for predicting future activations given an earlier model activation. Transcoders take in the pre-MLP activation and yield a sparse decomposition of MLP output. Formally, a Transcoder and its optimization objective can be written as:

$$
\begin{aligned}
f &= \mathrm{ReLU}(W_E x + b_E) \\
\hat{y} &= W_D f \\
\mathcal{L} &= \|y - \hat{y}\|_2^2 + \lambda \|f\|_1,
\end{aligned} \tag{3}
$$

,

which only differs from those of an SAE (Eq. 1) by the label activation $y \in \mathbb{R}^{d_{\text{model}}}$ unbound with input activation x.

Key difference between Transcoders and MLP We may find Transcoders and MLP with similar architecture: both are two fully connected blocks interspersed with an activation function. It's natural to ask why the non-linear activation function in MLP is deemed as an obstacle in circuit analysis but that in Transcoders is allowed. The key difference is that by constraining the sparsity, Transcoders neurons (which are just features) have an *interpretable basis*. When computing how upstream feature $f_i^{\mathcal{S}}$ contributes to activated downstream feature $f_j^{\mathcal{T}}$ of Transcoder $\mathcal{T}$, it holds that $f_j^{\mathcal{T}} = f_i^{\mathcal{S}} \left(W_E^{\mathcal{T}} W_D^{\mathcal{S}}\right)_{ji}$. The $\left(W_E^{\mathcal{T}} W_D^{\mathcal{S}}\right)_{ji}$ part remains constant across inputs, which leads to an **edge invariance** between upstream and downstream features.

Intuitively, this means when a main upstream contributor to a downstream feature has been activated in a different input, we can largely expect this downstream feature to be activated again unless some new resistances (upstream features with negative edges) have also been introduced.

In contrast, we cannot find such invariant edges through MLP. Any connection from upstream to MLP output is indefinite, so we could only find linear approximations to measure these connections under local changes.

2.3. QK and OV Circuits Are Independent Linear Operators on SAE Features

QK and OV circuits account for how tokens attend to one another and how information passes to downstream layers, respectively. The linearity and independence of these two components have been widely discussed in previous work (Elhage et al., 2021; He et al., 2024). Specifically, QK circuits serve as a bilinear operator of any two residual streams w.r.t token i and j:

$$
\begin{aligned}
\mathrm{AttnScore}^h(x)_{ij} &= x_i {W_Q^h}^T W_K^h x_j^T \\
&= \sum_{\mathcal{S},\mathcal{T} \in \text{Upstream SAEs}} \sum_{p=1}^{d_{\text{SAE}}} \sum_{q=1}^{d_{\text{SAE}}} \\
&\quad f_{i,p}^{\mathcal{S}} W_{D\,p}^{\mathcal{S}} {W_Q^h}^T W_K^h {W_{D\,q}^{\mathcal{T}}}^T f_{j,q}^{\mathcal{T}},
\end{aligned} \tag{4}
$$

where $f_{i,p}$ means the activation of the feature p at token i, and W_Q^h, W_K^h are a given head h's the query and key transformation. This decomposition shows how every pair of upstream features contributes to the attention score, making tokens containing critical information get attended.

Once the attention score is determined, we can then move on to the OV circuits, which apply a linear transformation to all past residual streams and take a weighted sum:

$$
\begin{aligned}
\mathrm{Attn}(x)_i &= \sum_h \mathrm{AttnOutput}^h(x)_i \\
&= \sum_h \sum_j \mathrm{AttnPattern}^h(x)_{i,j} W_O^h W_V^h x_j,
\end{aligned} \tag{5}
$$

where W_O^h, W_V^h are a given head h's output and value transformation. With `AttnPattern` determined in the QK circuits, how upstream features affect downstream are successively determined since $W_O^h W_V^h$ is invariant.

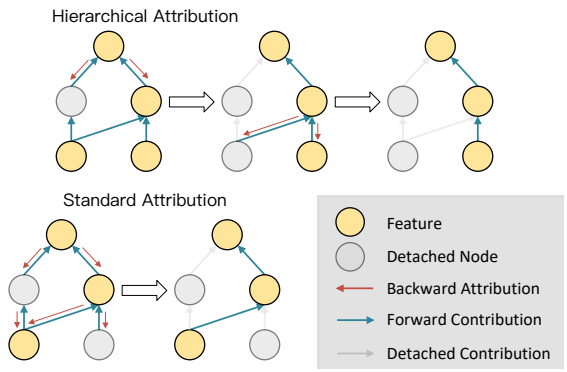
From an input-independent perspective, the quadratic coefficient $W_{D\,p}^{\mathcal{S}} {W_Q^h}^T W_K^h W_{D\,q}^{\mathcal{T}}$ shows how feature pairs co-work for every attention score. Then, $W_{E\,p}^{\mathcal{S}} W_O^h W_V^h W_{D\,q}^{\mathcal{T}}$ (obtained by adding SAE encoder and decoder terms to Eq. 5) determines the edge connecting upstream features and attention output features under a specific attention pattern. This two-step paradigm gives us a simplified and feature-based version of attention functionality and allows a fine-grained analysis through attention in a non-approximated manner.

In real-world applications, we often want to attribute an interested output (e.g., logits) to filter out critical features, which is a backward procedure. For the sake of a linear and exact attribution result, we can reverse the above two-step paradigm and 1) attribute through OV + Transcoder circuits and then 2) select important attention, attribute its attention score through the current QK and once again the upstream OV + Transcoder circuits (showed in Figure. 2(a)). The second step may be repeated several times to attribute attention important to another attention.

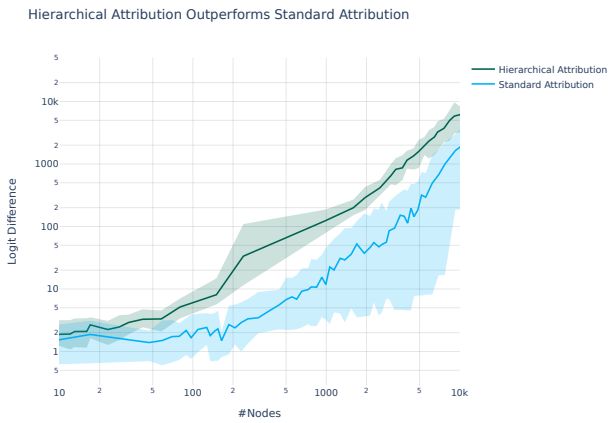
3. Isolating Interpretable Circuits with Hierarchical Attribution

We have now obtained a linear computation graph including all OV and MLP modules, reflecting the model's internal information flow. This section introduces how to isolate and evaluate a subgraph of the key SAE features related to any interested output.

Formulation We are given a linear computation graph $G = (V, E)$, which is a directed acyclic graph. Each node $v \in V$ refers to an activated feature in the model forward pass. The node weight a_v refers to the *activation* of node v. Each edge $v \to u \in E$ represents that a_v linearly affects a_u by the edge weight $k_{v,u}$. For any non-leaf node u, its acti-



(a) Workflow of performing *Hierarchical Attribution* and standard attribution.



(b) Comparison between *Hierarchical Attribution* and standard attribution.

Figure 2. Our *Hierarchical Attribution* detaches unrelated nodes immediately after they receive gradient and stops their backpropagation. While standard attribution detaches nodes after the backward process is completed. (Figure 2(a)). We sweep the number of remaining nodes i.e. sparsity and compare the logit recovery i.e. faithfulness of the identified subgraph. Experiments are conducted on 20 IOI samples (See Section 5) across 30 sparsity thresholds. Results in Figure 2(b) show that *Hierarchical Attribution* consistently outperforms standard attribution.

vation is completely determined by its direct predecessors, i.e., $a_u = \mathrm{ReLU}\left(\sum_{v \to u \in E} k_{v,u} a_v\right)$.

The term linear computation graph means every edge in the graph represents a linear function (under fixed attention scores). This guarantees a one-hop linear effect of activated features. It's not necessary that indirect effects between any two nodes are still linear since we allow a `ReLU` gate inside the nodes, stopping unactivated nodes from forwarding further.

Two Types of Leaf Nodes We denote word embedding SAE features and the position embedding as *interpretable leaf nodes*[1]. SAE errors also have zero in degree, but we cannot establish any explanation for these nodes. Thus, we call them *uninterpretable leaf nodes*.

Isolating a Subgraph with Node Detaching We prune unrelated nodes in the original linear computation graph to identify a subgraph accounting for the desired output.

Definition 3.1 (Detaching a node)**.** The operation of detaching a node v from graph G is to get an induced subgraph $G' = G[V/v]$, which removes v and all edges connecting to v from G.

We first need to detach all SAE errors since they cannot be interpreted, despite their empirically positive correlation to model performance (Gurnee, 2024). In the rest of the graph, with all leaf nodes being *interpretable leaf nodes*, we need to detach nodes unrelated to the task.

Manual Pruning with Direct Contribution For graphs with a small number of nodes, a simple solution is to manually inspect the interpretation of SAE features and their causal relation. This is often useful in understanding local behaviors but may be labor-intensive at scale.

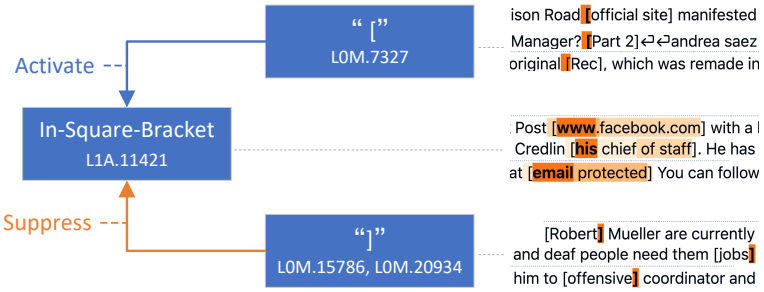
Automatic Circuit Discovery with Hierarchical Attribution We present how to perform scalable circuit discovery on this linear computation graph with gradient-based attribution (Kramár et al., 2024).

Definition 3.2 (Attribution Score)**.** The attribution score of node v w.r.t. an interested output node t is $\mathrm{attr}_{v,t} := a_v \cdot \nabla_{a_t} a_v$.

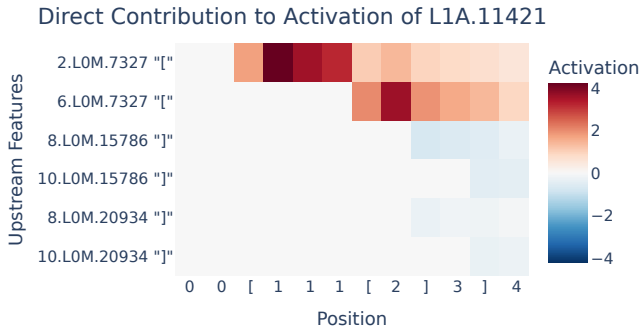
A natural idea would be running backward once and detaching nodes with $\mathrm{attr}_{v,t}$ lower than a given threshold τ, as adopted in most prior work (Conmy et al., 2023; Marks et al., 2024). We propose to operate a breadth-first search style attribution pipeline we call *Hierarchical Attribution*.

Hierarchical Attribution **detaches nodes on backward pass** instead of after backward, as shown in Figure 2(a) and a pseudo-code implementation in Appendix C. When performing model backward, we stop the gradient propagation of any node v that has $\mathrm{attr}_{v,t} < \tau$. This affects the attribution score of all predecessors of v. After we finish the backward propagation, all nodes with gradients make up our desired subgraph. Intuitively, attribution through detached nodes should not be taken into account; otherwise, their effect depends on excluded nodes in the final subgraph.

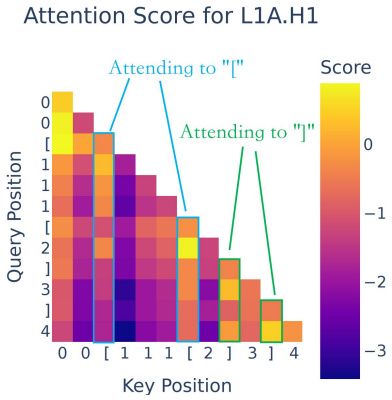
[1]We notice that not all SAE features are interpretable. We adopt a series of methods to improve the interpretability of SAEs further. See Appendix A



(a) Formation of In-Bracket Features



(b) Contribution to a specific *In-Bracket* feature from each token's open or closing bracket features



(c) Attention Score Trends of a Significant *Bracket Head*

Figure 3. (a) *Opening Bracket* features and *Closing Bracket* features have positive and negative contributions to *In-Bracket* features respectively. (b) Closer ” [”s activates the *In-Bracket* feature more prominently. (c) Tokens after ” [”s start with strong attention to ” [”s and become weaker as the sentence continues. This explains the trend in Figure 3(b).

Evaluation We leverage a good property of linear graphs to evaluate identified circuits.

Theorem 3.3. *For any subgraph $G' = G[V/v]$, the node weight of the root node is the sum of the attribution scores of all leaf nodes.*

$$a_t = \sum_{\deg_{in}(v)=0} \text{attr}_{v,t}$$

We refer readers to Appendix D for the proof.

This theorem allows us to instantly obtain how much $G' = G[V/v]$ accounts for the root node activation after we finish the pruning. Besides efficiency, another advantage of such evaluation is that it derives the causal effect of circuits without any intervention in the forward pass. It saves circuit evaluation from backup behaviors (Wang et al., 2023) (also known as hydra effects (McGrath et al., 2023)) due to ablation.

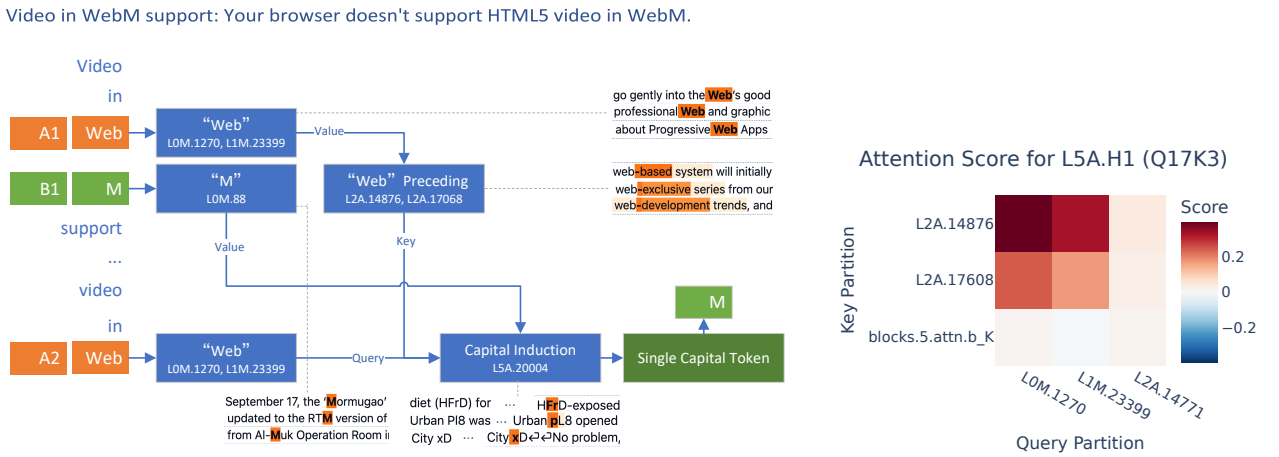
In Figure 2(b), we empirically validate the advantage of *Hierarchical Attribution* over the standard attribution method in Indirect Object Identification circuit discovery (Wang et al., 2023).

4. Attributing Intermediate SAE Features

An exciting application of Sparse Autoencoders is that they serve as *unsupervised feature extractors* in the vast hidden activation space. This opens up opportunities for understanding intermediate activations and local circuit discovery i.e. identifying a subgraph activating a given SAE feature instead of end-to-end circuits.

4.1. How Transformers Implement In-Bracket Features

We start from a series of *In-Bracket* features in attention blocks of early layer, which activate on tokens inside of brackets e.g. *deactivated [activated] deactivated*. These features will demonstrate higher activation in deeper nesting of brackets, imitating the behavior of finite state automata (Bricken et al., 2023) with states of bracket nesting hierarchy. We find an *In-Square-Bracket* feature and an *In-Round-Bracket* feature in SAEs trained on layer 1 attention block output, which we call L1A throughout this paper. Since they are at rather early layers, we leverage our Direct Contribution analysis to see how earlier features produce them.



(a) Information Flow in Induction Circuit

(b) QK Top Contributors to a Significant Induction Head

Figure 4. "Web"(L0M.1270 and L1M.23399) and "Web" *Preceding* features (L2A.14876 and L2A.17608) jointly lead to QK attention of an induction head. The "M" feature is copied to the last token for next token prediction.

Open-bracket features activate in-bracket ones. Figure 3(a) illustrates a simple two-layer bracket circuit in the wild. We inspect contributions to the *In-Square-Bracket* feature in a template e.g. "0 0 [1 1 1 [2] 3] 4", at token "1"s, "2", " 3" and " 4". Experiments show that the activation is mainly promoted by an L0M feature activated by token " [". It takes on 104.1%, 102.6% and 314.2% of the *In-Square-Bracket* feature's activation respectively at token "1", "2", and " 3", respectively. An average of 83.8% of these contribution comes through attention head 1 of L1A i.e. L1A.H1.

Closing-bracket features deactivate in-bracket ones. The activation of the *In-Square-Bracket* feature is mostly suppressed by a "]" feature in L0M (Figure 3(b)). The suppression goes through L1A.H1 as well.

Interpreting QK attention to " [" and "]". We study the QK circuit of L1A.H1, as shown in Figure 3(c). This head attends to " ["s and "]"s regardless of the current token. This is mainly caused by b_Q in L1A.H1 attending to the above " [" and "]" features.

4.2. Revisiting Induction Behavior from the SAE Lens

Induction Heads (Olsson et al., 2022) are an important type of compositional circuit with two attention layers which try to repeat any 2-gram that occured before i.e. [A][B] ... [A] -¿ [B]. These circuits are believed to account for a majority of in-context learning functionality in large transformers. Compared to massive existing literature in understanding the induction mechanism in the granularity of attention heads (Olsson et al., 2022; Hendel et al., 2023; Ren et al., 2024), *inter alia*, we seek to present a finer-grained level interpretation of such behavior.

Induction features form a huge feature family. These features are found to be identified by the logit of tokens they enhance through the logit lens (nostalgebraist, 2020). We first study a *Capital Induction* feature contributing to logits of single capital letters, on a curated input "Video in WebM support: Your browser doesn't support HTML5 video in WebM." (Figure 4(a)). This feature is activated on the second " Web", and amplifies the prediction of "M", copying its previous occurrence.

Upstream Contribution through OV Circuit We notice that a series of "M" features in the residual stream of the first "M" constitute most of the *Capital Induction* feature's activation through OV circuits. L0M.88 takes the lead, which contributes 35.0% of the feature activation. Auxiliary features from L0A, L1M and L3M either directly indicate current token as "M", or indicate current token as a single capital letter. Top 7 of the auxiliary features account for another 33.0% of the feature activation. Most of these contributions come from L5A.H1, which we along with a concurrent research (**?**) identify as an induction head.

Upstream Contribution to QK Attention To study how this induction head attend to the first "M", we attribute the attention score to upstream feature pairs. The commonality of top contributors is a " Web" feature attending to a " Web" *Preceding* feature (i.e. its previous token is " Web"), as shown in Figure 4(b).

Attributing Preceding features We further study how " Web" *Preceding* features indicate previous tokens. These contributions mainly come through L2A.H2, which we think to be a previous token head. The relatively high attention

score for previous token can be attributed to a group of L0A features collecting information from positional embeddings.

5. Revisiting Indirect Object Identification Circuits from the SAE Lens

For end-to-end circuits in GPT2-small, we choose to investigate a task called Indirect Object Identification (IOI) (Wang et al., 2023) with *Hierarchical Attribution*. For instance, GPT2 is able to predict " Mary" following the prompt "When Mary and John went to the store, John gave the bag to". We call this prompt s_{Mary} since it starts with " Mary" and a variant s_{John} with a swap in the first two names i.e. "When John and Mary went to the store, John gave the bag to". The answer to both prompts is " Mary", which GPT2 is able to predict. Existing literature studying this problem does not distinguish between these two templates. Through the lens of SAE circuit, we validate conclusions in previous work and also discover some subtle mechanistic distinction in their corresponding circuits.

5.1. SAE Circuits Closely Agree with Head-Level Ones

We manage to find the end-to-end information flow in the IOI task example s_{Mary} and its variant s_{John} with *Hierarchical Attribution*. Then we identify the pivotal attention heads in the isolated subgraph and attribute their QK scores to earlier SAE features. Discovered SAE feature circuits are of strong consistency with those found on the basis of attention heads: (1) *Name Mover* features correspond to *Name Mover Heads* (L9A.H6, L9A.H9); (2) *Association* features correspond to *S-Inhibition Heads* (L7A.H3, L7A.H6, L8A.H10); (3) *Induction* features correspond to *Induction Heads* (L5A.H5, L6A.H9); (4) *Preceding* features correspond to *Previous Token Heads* (L2A.H2, L3A.H2, L4A.H1).

5.2. Zooming in on SAE Circuits Yields New Discoveries

We present a concrete example in the wild that SAE circuits convey more information than their coarse-grained counterparts. We believe this is a positive signal for us to obtain deeper understanding of language model circuits. Despite the consistency of involved attention heads in s_{John} and s_{Mary}, these two circuits are actually composed of completely different SAE features, as shown in Figure 5(b).

We start with interpreting how GPT2 predicts " Mary" given the prompt "When John and Mary went to the store, John gave the bag to" (s_{John}). Though greatly simplified, the information flow is still somehow complicated. We further pick four pivotal feature clusters, as marked in Figure 5. A non-rigorous interpretation of them is as follows.

A " Mary" is recognized as a Consecutive Entity because it occurs after an " and".

B S2 i.e. the second "John" activates an induction feature. It enhances the logit of "and" though its next token is not.

C " to" is a representative token indicating the next token is some object or entity. It activates an association feature to retrieve possible entities occurring before. It copies information from feature group B and is informed of the existence of an entity going after an " and".

D The Name Mover Head receives this information and easily copies the token " Mary" to its residual stream.

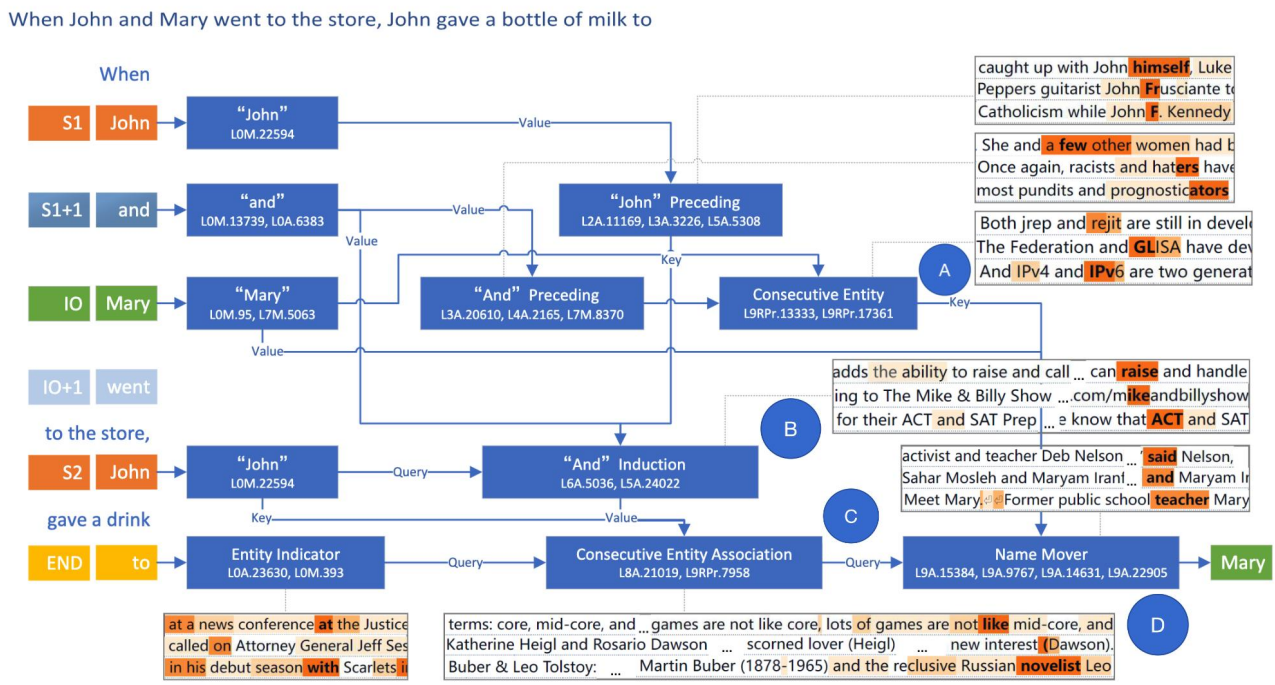
The interpretation above is highly dependent on the fact that the Indirect Object is present after an " and". However, things are quite different in s_{Mary} since it comes before the "and". In fact, token " Mary" first activates a Center Entity feature, whose explanation given by GPT-4 is "People or Objects that is likely to be the main topic of the article". The last token still seeks to associate a previously occurring entity but is *informed* to retrieve the Center Entity instead since the Consecutive Entity Association feature has been suppressed by repeated " John"s.

6. Related Work

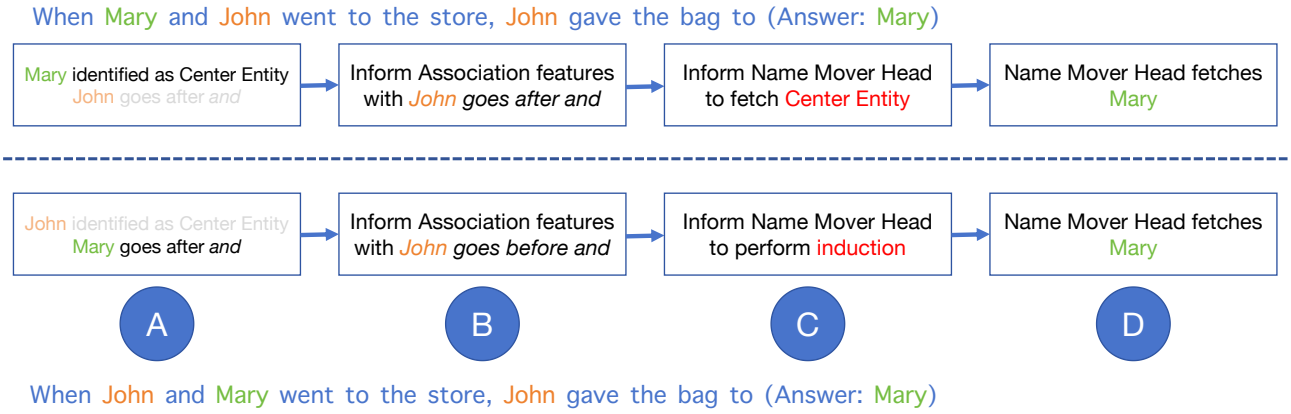
Mechanistic and Representational Interpretability Mechanistic Interpretability (Olah et al., 2020b;a) deems model components e.g. attention heads and MLP neurons as *primitives* and explain how they interact with model input and output. This line of research has succeeded in identifying attention-based circuits implementing various NLP tasks (Olsson et al., 2022; Wang et al., 2023; Stefan Heimersheim, 2023). Efforts are also made to interpret polysemantic MLP neurons (Gurnee et al., 2023) and editing information stored in MLP parameters (Meng et al., 2022; Sharma et al., 2024).

By placing intermediate activations at the center of analysis, Representational Interpretability approaches mostly use linear probes to isolate a targeted behavior in a supervised manner (Kim et al., 2018; Geiger et al., 2023; Zou et al., 2023). However, such methods may fail to capture unanticipated behaviors.

Sparse Autoencoders stand in between these two approaches. SAEs disentangle features in the model's *hidden activation* (Chen et al., 2017; Subramanian et al., 2018; Zhang et al., 2019; Panigrahi et al., 2019; Yun et al., 2021; Bricken et al., 2023; Cunningham et al., 2023) into more interpretable *primitives* than MLP neurons, in an unsupervised manner. Albeit reconstruction errors, Rajamanoharan



(a) Overview of s_{John} circuit



(b) A non-rigorous illustration of the key differences between s_{John} and s_{Mary} circuits

Figure 5. In s_{John}, the consecutive entity feature (denoted as A in Figure 5(a)) serves as the key vector for Name Mover Heads to attend to and copy the answer entity to the last token's residual stream. Such mechanism does not work in s_{Mary} because the correct answer is no longer a consecutive entity (i.e. the entity present after the token *and*). See Appendix E for a detailed interpretation for these two examples.

et al. (2024); Wright & Sharkey (2024) have proposed to improve SAE training with lower loss and more sparsity.

Circuit Discovery with SAE Features Previous work mechanistically interprets circuits connecting attention heads and MLP neurons (Olsson et al., 2022; Wang et al., 2023; Conmy et al., 2023). As for SAE circuits, He et al. (2024) makes a linear approximation of MLP layers by fixing the gate mask of the non-linear activation function; Marks et al. (2024) estimate the indirect effect of each SAE feature with attribution patching (Kramár et al., 2024), which also makes linear assumption of non-linear functions. In contrast, we refactor our computation graph to be completely linear w.r.t. OV and MLP circuits without approximation.

7. Conclusion and Limitation

We frame a pipeline to identify fine-grained circuits in Transformer language models. With Sparse Autoencoders and Transcoders, we refactor the model's computation to linear (with respect to a single input). We also propose an efficient approach to isolate subgraphs (i.e. circuits). We showcase that finer-grained circuit analysis provides more beautiful and detailed structures in Transformers. One limitation of our work is that our analysis is specific to certain inputs and might not generalize to other settings. We deem this as a trade-off between granularity and universality. Some extension can be made to extract more general circuits in regard to more abstract behaviors. We leave this for future work.

References

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HJ4-rAVtl.

Arditi, A., Obeso, O., Aaquib111, wesg, and Nanda, N. Refusal in llms is mediated by a single direction. LessWrong, 2024. URL https://www.lesswrong.com/posts/KicP8fBdHNjZBXxRB/an-ov-coherent-toy-model-of-attention-heads-supports-this.

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.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Chen, Y., Li, G., and Jin, Z. Learning sparse overcomplete word vectors without intermediate dense representations. In Li, G., Ge, Y., Zhang, Z., Jin, Z., and Blumenstein, M. (eds.), *Knowledge Science, Engineering and Management - 10th International Conference, KSEM 2017, Melbourne, VIC, Australia, August 19-20, 2017, Proceedings*, volume 10412 of *Lecture Notes in Computer Science*, pp. 3–15. Springer, 2017. doi: 10.1007/978-3-319-63558-3_1. URL https://doi.org/10.1007/978-3-319-63558-3_1.

Conmy, A., Mavor-Parker, A. N., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability. *CoRR*, abs/2304.14997, 2023. doi: 10.48550/ARXIV.2304.14997. URL https://doi.org/10.48550/arXiv.2304.14997.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. *CoRR*, abs/2309.08600, 2023. doi: 10.48550/ARXIV.2309.08600. URL https://doi.org/10.48550/arXiv.2309.08600.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

Elhage, N., Lasenby, R., and Olah, C. Privileged bases in the transformer residual stream. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/privileged-basis/index.html.

Geiger, A., Potts, C., and Icard, T. Causal abstraction for faithful model interpretation. *CoRR*, abs/2301.04709, 2023. doi: 10.48550/ARXIV.2301.04709. URL https://doi.org/10.48550/arXiv.2301.04709.

Gurnee, W. Sae reconstruction errors are (empirically) pathological. LessWrong, 2024. URL https://www.lesswrong.com/posts/rZPiuFxESMxCDHe4B/sae-reconstruction-errors-are-empirically-patholo

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a haystack: Case studies with sparse probing. *CoRR*, abs/2305.01610, 2023. doi: 10.48550/ARXIV.2305.01610. URL https://doi.org/10.48550/arXiv.2305.01610.

He, Z., Ge, X., Tang, Q., Sun, T., Cheng, Q., and Qiu, X. Dictionary learning improves patch-free circuit discovery in mechanistic interpretability: A case study on othello-gpt. *CoRR*, abs/2402.12201, 2024. doi: 10.48550/ARXIV.2402.12201. URL https://doi.org/10.48550/arXiv.2402.12201.

Hendel, R., Geva, M., and Globerson, A. In-context learning creates task vectors. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pp. 9318–9333. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.624. URL https://doi.org/10.18653/v1/2023.findings-emnlp.624.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C. J., Wexler, J., Viégas, F. B., and Sayres, R. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2673–2682. PMLR, 2018. URL http://proceedings.mlr.press/v80/kim18d.html.

Kramár, J., Lieberum, T., Shah, R., and Nanda, N. Atp*: An efficient and scalable method for localizing LLM behaviour to components. *CoRR*, abs/2403.00745, 2024. doi: 10.48550/ARXIV.2403.00745. URL https://doi.org/10.48550/arXiv.2403.00745.

Larson, D. Expanding the scope of superposition. LessWrong, 2023. URL https://www.lesswrong.com/posts/wHHdJdhKBqoKAMC5d/expanding-the-scope-of-superposition.

LaurenGreenspan and keith_wynroe. An ov-coherent toy model of attention head superposition. LessWrong, 2023. URL https://www.lesswrong.com/posts/KicP8fBdHNjZBXxRB/an-ov-coherent-toy-model-of-attention-head-superposition.

Li, K., Patel, O., Viégas, F. B., Pfister, H., and Wattenberg, M. Inference-time intervention: Eliciting truthful answers from a language model. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/81b8390039b7302c909cb769f8b6cd93-Abstract-Conference.html.

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. *CoRR*, abs/2403.19647, 2024. doi: 10.48550/ARXIV.2403.19647. URL https://doi.org/10.48550/arXiv.2403.19647.

McGrath, T., Rahtz, M., Kramár, J., Mikulik, V., and Legg, S. The hydra effect: Emergent self-repair in language model computations. *CoRR*, abs/2307.15771, 2023. doi: 10.48550/ARXIV.2307.15771. URL https://doi.org/10.48550/arXiv.2307.15771.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in GPT. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Confere
html.

nostalgebraist. interpreting gpt: the logit lens. LessWrong, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. An overview of early vision in inceptionv1. *Distill*, 2020a. doi: 10.23915/distill.00024.002. https://distill.pub/2020/circuits/early-vision.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 2020b. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Panigrahi, A., Simhadri, H. V., and Bhattacharyya, C. Word2sense: Sparse interpretable word embeddings. In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 1: Long Papers*, pp. 5692–5705. Association for Computational

Linguistics, 2019. doi: 10.18653/V1/P19-1570. URL https://doi.org/10.18653/v1/p19-1570.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R., and Nanda, N. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024.

Ren, J., Guo, Q., Yan, H., Liu, D., Qiu, X., and Lin, D. Identifying semantic induction heads to understand in-context learning. *CoRR*, abs/2402.13055, 2024. doi: 10.48550/ARXIV.2402.13055. URL https://doi.org/10.48550/arXiv.2402.13055.

Sharma, A. S., Atkinson, D., and Bau, D. Locating and editing factual associations in mamba. *arXiv preprint arXiv:2404.03646*, 2024.

Stefan Heimersheim, J. J. A circuit for python docstrings in a 4-layer attention-only transformer. 2023. URL https://www.alignmentforum.org/posts/u6KXXmKFbXfWzoAXn/a-circuit-for-python-docstrings-in-a-4-layer-attention-only.

Subramanian, A., Pruthi, D., Jhamtani, H., Berg-Kirkpatrick, T., and Hovy, E. H. SPINE: sparse interpretable neural embeddings. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 4921–4928. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11935. URL https://doi.org/10.1609/aaai.v32i1.11935.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Wang, K. R., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=NpsVSN6o4ul.

Wright, B. and Sharkey, L. Addressing feature suppression in saes. LessWrong, 2024. URL https://www.lesswrong.com/posts/3JuSjTZyMzaSeTxKk/addressing-feature-suppression-in-saes.

Yun, Z., Chen, Y., Olshausen, B. A., and LeCun, Y. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. In Agirre, E., Apidianaki, M., and Vulic, I. (eds.), *Proceedings of Deep Learning Inside Out: The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@NAACL-HLT 2021, Online, June 10 2021*, pp. 1–10. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.DEELIO-1.1. URL https://doi.org/10.18653/v1/2021.deelio-1.1.

Zhang, J., Chen, Y., Cheung, B., and Olshausen, B. A. Word embedding visualization via dictionary learning. *CoRR*, abs/1910.03833, 2019. URL http://arxiv.org/abs/1910.03833.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A., Goel, S., Li, N., Byun, M. J., Wang, Z., Mallen, A., Basart, S., Koyejo, S., Song, D., Fredrikson, M., Kolter, J. Z., and Hendrycks, D. Representation engineering: A top-down approach to AI transparency. *CoRR*, abs/2310.01405, 2023. doi: 10.48550/ARXIV.2310.01405. URL https://doi.org/10.48550/arXiv.2310.01405.

A. Sparse Autoencoder Training

We trained an SAE (Section 2.1) on each of the outputs of the 12 attention layers and 24 residual stream activation (before entering attention layers and MLP layers). We trained a Skip SAE (Section 2.2) through each MLP layer, using residual stream activation before MLP as input and MLP output activation as label. Here's our training settings:

- Each SAE has 24,576 dictionary features, which is 32 times the hidden dimension of GPT-2 Small.
- We use the Adam optimizer with a learning rate of 4e-4 and betas of (0, 0.9999) for 1 billion tokens from the OpenWebText (Radford et al., 2019) corpus. We trained against a reconstruction loss (measured by MSE of input and reconstructed output), a sparsity loss (proximated by the L1 norm of the feature activations, with a coefficient of 8e-5 (1.2e-4 for attention output SAEs)), and a ghost gradient loss. A batch size of 4,096 is used. We use an NVIDIA A100-80GB GPU for training of each SAE, which lasts for 20 hours.
- First 256 tokens of each sequence are used as input, discarding the rest tokens and sequences shorter than 256 tokens. Generated activations are shuffled actively in an activation buffer.
- We normalize the input activations to have a norm of the square root of LM hidden size (i.e. $\sqrt{768}$ for GPT-2 Small). We further normalize the MSE loss by the variance of output along the hidden dimension (a bit like the latter part in LayerNorm, except that we're not taking the mean of output).

$$\mathcal{L}_{\text{MSE}} = (x_{\text{normed}} - \hat{x}_{\text{normed}})/\|\hat{x}_{\text{normed}} - \bar{\hat{x}}_{\text{normed}}\|_2$$

- We use untied weights for the encoder and decoder. Decoder bias (or pre-encoder bias) is removed (for the sake of simplier circuit analysis). Decoder norm are reset to less than or equal to 1 after each training step.
- *We prune the dictionary features with norm less than 0.99, max activation less than 1, and activation frequency less than 1e-6 after training.
- *We finetuned the decoder and a feature activation scaler of the pruned SAEs on the same dataset to deal with feature suppression.

A.1. Feature Pruning

Some of the SAE features obtained from end-to-end training are too sparse (i.e. can hardly be activated) to reflect a certain aspect of the input corpus. These features are more like "local codes" (in neuroscience). They are activated by very specific tokens. These features are trivial and not helpful for understanding an activation pattern from a compositional perspective. Feature pruning aims to remove these trivial features and keep the more meaningful ones.

In practice, a dictionary feature will be pruned if it meets one of the following criteria:

Norm less than 0.99: In SAE training, we use an L1 loss as a differentiable approximation of L0 loss, to encourage sparsity in the feature activations. The side effect is that the L1 loss as well encourages a lower value of the feature activations, and a larger feature norm. Thus, if a feature is really "useful" in reconstructing the input, it should have a norm as large as possible. We prune the features without the tendency to grow.

Max activation less than 1: Given a fixed norm of the feature, a feature with a low max activation value makes little contribution to reconstructing the input. We find this kind of features activated in some non-related situations, and thus non-interpretable. We empirically set the threshold to 1 and prune the features below it.

Activation frequency less than 1e-6: A feature with an ultra-low activation frequency is considered to be too local to be useful. We find this kind of features often correspond to some specific tokens in some specific contexts, which is too trivial to be recognized as a feature. We empirically set the threshold to 1e-6 and prunes the features activated in a frequency below it.

A.2. Finetuning against Feature Suppression

Feature suppression refers to a phenomenon where loss function in SAEs pushes for smaller feature activation values, leading to suppressed features and worse reconstruction quality. Wright and Sharkey deduced that for an L1 coefficient of c and dimension d, instead of having a ground truth feature activation of g, the optimal activation SAEs may learn is $g - \frac{cd}{2}$.

To address this issue, we finetune the decoder and a feature activation scaler of the pruned SAEs on the same dataset. Only the reconstruction loss (i.e. the MSE loss) is applied in this finetuning process. Encoder weights are fixed during this process to keep sparsity of the dictionary. Finetuning may also repair flaws introduced in the pruning process, and improve the overall reconstruction quality.

A.3. Statistics of Sparse Autoencoders

We evaluate the L0 loss, variance explained and reconstruction CE loss of each trained SAEs. The L0 loss computes the average feature activated at each token. Variance explained

computes

$$EV = 1 - \frac{\|\hat{y} - y\|_2^2}{\sigma^2(y)},$$

which is a measurement of the proportion to which an SAE accounts for the activation variation. Reconstruction CE loss is the final cross-entropy loss of the language model where the activation is replaced with the SAE reconstructed one. Reconstruction CE score shows how good the reconstruction CE loss is w.r.t the original CE loss and the ablated CE loss, by computing

$$s = \frac{\mathcal{L}_{\text{recons}} - \mathcal{L}_{\text{ablate}}}{\mathcal{L}_{\text{original}} - \mathcal{L}_{\text{ablate}}},$$

where $\mathcal{L}_{\text{recons}}$, $\mathcal{L}_{\text{original}}$ and $\mathcal{L}_{\text{ablate}}$ refer to the reconstruction CE loss, the original CE loss and the ablated CE loss respectively.

The statistics of each SAE is as shown in Table. 1, Table. 2 and Table. 3.

B. General Direct Contribution Computation

In Sec. 2.3 and Sec. 2.2, we have shown how we compute direct contribution towards attention outputs, attention scores, and SAE feature activation, which is a linear effect of each input partition. However, it may still remain confusing why we can compute a linear contribution in such non-linear functions as attention blocks. For a clarification of how direct contribution works, we introduce our general mathematical formation of direct contribution computation in this section.

The term **direct contribution** refers to how partitions of upstream model activations respectively contribute to the downstream (through only direct ways, e.g. a single model layer), and constitute the downstream model activations. We start from linear functions, which are the simplest case of direct contribution computation. Given a model activation $x \in \mathbb{R}^H$ and its arbitrary n-parted partition $x = \sum_{i=1}^n v_i$, where $v_i \in \mathbb{R}^H$ is the i-th partition of x. For any affine transformation $f : \mathbb{R}^H \to \mathbb{R}^K$ mapping x to a downstream activation $f(x) = Wx + b$, $W \in \mathbb{R}^{K\times H}$, $b \in \mathbb{R}^K$, we have

$$f(x) = W\sum_{i=1}^n v_i + b = \sum_{i=1}^n Wv_i + b, \tag{6}$$

from which we learned that each partition v_i separately contributes to $f(x)$ by Wv_i (since it's the only term related to v_i in the final summation, and the bias b contributes to $f(x)$ by its own value b. This contribution ribution is natural thanks to the additive (w.r.t vector addition) nature of linear mapping.

Nevertheless, computation in practical neural networks is often much complicate than the above affine transformation or its simple nesting. Non-linear transformation (e.g. `LayerNorm`, `Softmax`, `ReLU`) is ubiquitous. We cannot simply ignore these non-linear operators since the powerful fitting capacity of neural networks often just come from the non-linear parts. To deal with these non-linear transformations, we propose a more general direct contribution computing strategy. For any transformation $f : \mathbb{R}^H \to \mathbb{R}^K$ where f has a form of $f(x) = W(x)x + b$, $W : \mathbb{R}^H \to \mathbb{R}^{K\times H}$, $b \in \mathbb{R}^K$, we have

$$f(x) = W(x)\sum_{i=1}^n v_i + b = \sum_{i=1}^n W(x)v_i + b, \tag{7}$$

where we treat $W(x)$ as a constant linear transformation matrix. Then we can claim that i-th partition v_i contributes to the result $f(x)$ by $W(x)v_i$ through the posterior linear transformation with a constant $W(x)$. We must state this contribution computation is nothing but trivial if we don't further interpret how partitions affect $W(x)$ and the relating impact to the following transformation, or further restrict the $W(x)$to make sure it's just close to a constant or its variation is unimportant. Thus, for W that having a similar form as f, e.g. $W(x) = W'(x)x + B$, $W' : \mathbb{R}^H \to \mathbb{R}^{(K\times H)\times H}$, $B \in \mathbb{R}^{K\times H}$, we can iteratively apply the linear decomposition Eq. 7 to W (which we use to interpret attention pattern in Sec. 2.3),

$$W(x) = W'(x)\sum_{i=1}^n v_i + B = \sum_{i=1}^n W'(x)v_i + B \tag{8}$$

The above transformations could be nested to compute direct contribution to further activations. Take $f = f_1 \circ f_2$ as a twofold nesting example, where $f_1(x) = W_1x + b_1$ and $f_2(x) = W_2(x)x + b_2$, it can be easily induced that

$$\begin{aligned} f(x) &= W_1W_2(x)\sum_{i=1}^n v_i + W_1b_2 + b_1 \\ &= \sum_{i=1}^n W_1W_2(x)v_i + W_1b_2 + b_1, \end{aligned} \tag{9}$$

and get the respective contribution of every v_i and b_i. Direct contribution through deeper nested transformations can be computed in similar ways.

As a brief summary, the core idea of direct contribution computation for any non-linear function is to first compute

Table 1. Statistics of Attention Output SAEs

SAE	Var. Explained	L0 Loss	Reconstruction CE Score	Reconstruction CE Loss
L0A	92.25%	29.66	99.24%	3.2327
L1A	82.48%	65.57	97.19%	3.2138
L2A	83.39%	69.85	94.29%	3.2150
L3A	69.23%	53.59	87.00%	3.2173
L4A	74.91%	87.35	89.99%	3.2171
L5A	82.12%	127.18	97.81%	3.2145
L6A	76.63%	100.89	94.31%	3.2158
L7A	78.51%	103.30	91.32%	3.2182
L8A	79.94%	122.46	88.67%	3.2172
L9A	81.62%	107.81	89.55%	3.2187
L10A	83.75%	100.44	87.70%	3.2201
L11A	84.81%	22.69	85.49%	3.2418

Table 2. Statistics of MLP Transcoders

SAE	Var. Explained	L0 Loss	Reconstruction CE Score	Reconstruction CE Loss
L0M	94.16%	19.59	99.65%	3.1924
L1M	82.02%	48.63	86.35%	3.1816
L2M	86.32%	50.90	81.24%	3.1851
L3M	76.55%	56.91	83.43%	3.1867
L4M	73.38%	76.03	80.08%	3.1888
L5M	73.49%	84.11	84.18%	3.1881
L6M	72.79%	90.34	82.85%	3.1912
L7M	73.18%	86.38	81.89%	3.1911
L8M	74.14%	87.29	83.25%	3.1913
L9M	75.89%	90.08	81.89%	3.1930
L10M	79.66%	94.85	81.60%	3.1987
L11M	80.33%	79.12	77.33%	3.2169

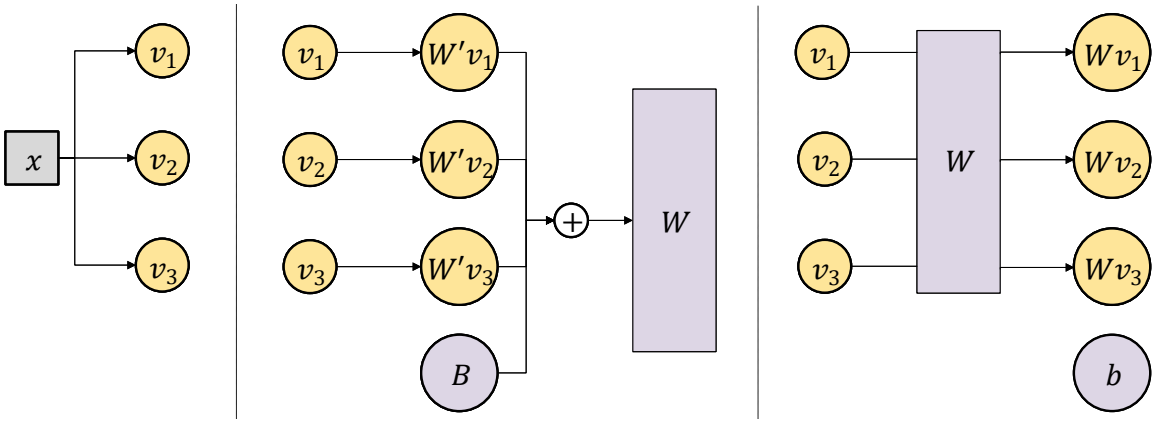


Figure 6. The workflow of interpreting a non-linear transformation where the transformation matrix can be linearly decomposed. We first compute the direct contribution $W'v_i$ to the transformation matrix W of each partition v_i of x so as to reveal the formation of W, and then treat the computed W as constant to compute the final direct contribution Wv_i.

Table 3. Statistics of Residual Stream SAEs

SAE	Var. Explained	L0 Loss	Reconstruction CE Score	Reconstruction CE Loss
L0RPr	98.98%	6.89	99.90%	3.1907
L0RM	95.94%	42.50	99.34%	3.2658
L1RPr	96.98%	21.96	99.62%	3.1935
L1RM	95.53%	34.11	99.77%	3.2133
L2RPr	96.03%	28.18	99.01%	3.2268
L2RM	94.45%	40.17	99.32%	3.2662
L3RPr	94.43%	38.22	98.95%	3.2867
L3RM	93.13%	48.44	99.13%	3.2673
L4RPr	92.08%	49.19	99.31%	3.2782
L4RM	91.00%	61.66	99.26%	3.2771
L5RPr	90.68%	60.34	99.09%	3.2950
L5RM	89.90%	76.22	99.11%	3.2839
L6RPr	90.03%	70.06	98.93%	3.2899
L6RM	89.57%	88.95	98.59%	3.2830
L7RPr	88.86%	79.91	98.88%	3.2943
L7RM	88.28%	98.60	98.94%	3.2828
L8RPr	87.99%	89.37	98.55%	3.2952
L8RM	87.32%	108.72	98.70%	3.2863
L9RPr	87.38%	100.68	99.17%	3.2938
L9RM	86.66%	119.59	98.15%	3.2889
L10RPr	86.72%	115.35	98.59%	3.2984
L10RM	86.07%	126.19	98.14%	3.3036
L11RPr	85.76%	120.86	97.93%	3.3212
L11RM	85.40%	94.20	98.42%	3.3910

how the non-linear part is formed w.r.t each input partition by iteratively applying direct contribution computation, and then consider the non-linear part as determined, regard the function to be linear, and compute a linear contribution to the function output. We usually allow the determined non-linear part to go through a simple extra activation function like `Softmax` or `ReLU`, since this will not undermine the understanding of this non-linear part. This workflow can be applied to non-linear function like bi-linear function and attention.

C. Hierachical Attribution Algorithm

In this section, we introduce the detailed implementation of Hierachical Attribution algorithm, to obtain a subgraph G' from original computational graph G with threshold τ, as shown in Algorithm 1.

Algorithm 1 Hierachical Attribution

Input: $\tau > 0, G, t$ {t for the root node}
Output: Optimized subgraph G'
$N' \leftarrow \varnothing$
for all v in reversed topological sort of G **do**
  if $v = t$ **then**
    $v.grad \leftarrow 1$
  else
    $v.grad \leftarrow 0$
    for all u in direct successors of v in G **do**
      $v.grad \leftarrow v.grad + \nabla_{a_v} a_u \cdot u.grad$ {Do normal back-propagation}
    end for
    if $v.grad \cdot a_v < \tau$ **then**
      $v.grad \leftarrow 0$
      $\text{attr}_v \leftarrow 0$
    else
      $\text{attr}_v \leftarrow v.grad \cdot a_v$
      $N' \leftarrow N' \cup \{v\}$
    end if
  end if
end for
$G' \rightarrow G[N']$

Afterwards, we can compute the contribution of G' by adding up the attribution score of all its leaf nodes.

D. Equality of Output Activation and Leaf Nodes Attribution

We demonstrate the proof for Theorem 3.3 as below, which is quite simple:

Proof. For any activated node u (i.e., $a_u > 0$), it holds that

$$\begin{aligned} a_u &= \text{ReLU}\left(\sum_{v \to u \in E} k_{v,u} a_v\right) \\ &= \sum_{v \to u \in E} k_{v,u} a_v \\ &= \sum_{v \to u \in E, a_t > 0} k_{v,u} a_v \end{aligned} \tag{10}$$

By iteratively applying Eq. 10, we can obtain

$$\begin{aligned} a_t &= \sum_{\deg_{\text{in}}(v)=0, a_v>0} a_v \cdot \nabla_{a_t} a_v \\ &= \sum_{\deg_{\text{in}}(v)=0} a_v \cdot \nabla_{a_t} a_v \\ &= \sum_{\deg_{\text{in}}(v)=0} \text{attr}_{v,t} \end{aligned} \tag{11}$$

□

E. Additional Explanation of IOI Circuit

We further explain the feature circuit we discovered in s_{Mary} and s_{John}, by listing the meaning or functionality of pivotal features in these two exemplars.

The pivotal features in s_{John} (Figure 5(a)):

- ”John”, ”and” and ”Mary” features simply imply current token as ”John”, ”and”, and ”Mary”;
- *Entity Indicator* features are activated on prepositions or transitive verbs, indicating that its next token is likely to be an entity.
- ”John” *Preceding* features collect information from the previous token, and imply its previous token as ”John”;
- ”And” *Preceding* features collect information from the previous token, and imply its previous token as ”and”;
- *Consecutive Entity* features are a mixture of ”Mary” features and ”And” *Preceding* features, and imply the current token as the [B] part of an [A] and [B] pattern, where [A] and [B] serve as entities.
- ”And” *Induction* features attend to ”and” (by matching *S1* and *S2*), and collects the ”and” information from *S1+1*, implying there’s an ”and” goes after ”John”.
- *Consecutive Entity Association* features take advantage of the structural information from ”And” *Induction* features, and decide to retrieve the entity lying after ”and”, by attending to *Consecutive Entity* features in *Name Mover Heads*.

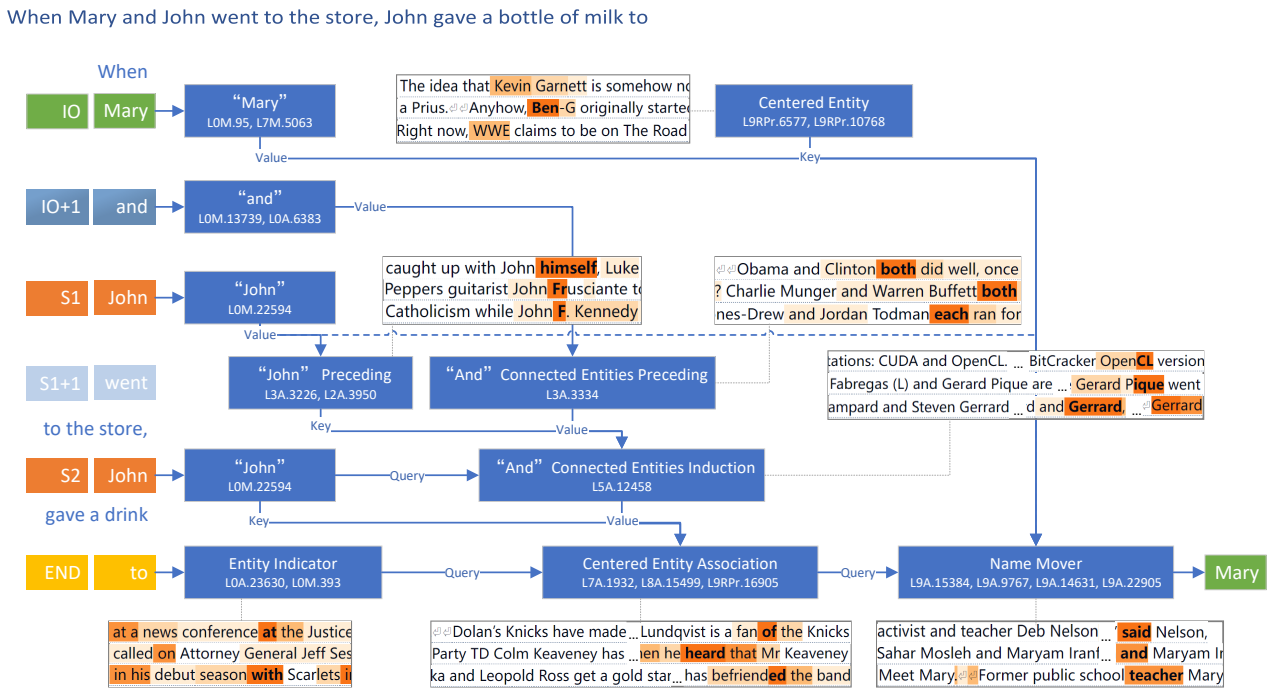


Figure 7. Overview of s_{Mary} circuit.

- *Nave Mover* features conduct the final step to move the "Mary" information from the targeted *Consecutive Entity* token.

The pivotal features in s_{Mary} (Figure 7):

- "John", "and", "Mary", *Entity Indicator* and "John" *Preceding* features play the same role as in s_{John}.
- *Centered Entity* features are activated at the first occurrence of a seemingly important name or object, marking it out for potential future reference.
- "And"-*Connected Entities Preceding* features collect information from several previous tokens (mainly the token "and"), and imply there's an [A] and [B] pattern before this token.
- "And"-*Connected Entities Induction* features collect information from "And"-*Connected Entities Preceding*, again by matching *S1* and *S2*.
- *Centered Entity Association* features take advantage of the structural information from "And"-*Connected Entities Induction* features, and decide to retrieve the entity lying before "and", by attending to *Centered Entity* features in *Name Mover Heads*. This behavior is not completely symmetrical to that with *Consecutive Entity* features, since *Centered Entity* features do not know about the token "and" after it. However, this behavior is still reasonable since if there's another *Centered Entity* before *IO*, then this entity can be another correct answer.
- *Nave Mover* features again conduct the final step to move the "Mary" information from the targeted *Consecutive Entity* token.