Information Flow Routes: Automatically Interpreting Language Models at Scale

Javier Ferrando[1*] **Elena Voita**[2]
[1]Universitat Politècnica de Catalunya
[2]Meta AI
jferrandomonsonis@gmail.com

Abstract

Information flows by routes inside the network via mechanisms implemented in the model. These routes can be represented as graphs where nodes correspond to token representations and edges to computations. We automatically build these graphs in a top-down manner, for each prediction leaving only the most important nodes and edges. In contrast to the existing workflows relying on activation patching, we do this through attribution: this allows us to efficiently uncover existing circuits with just a single forward pass. Unlike with patching, we do not need a human to carefully design prediction templates, and we can extract information flow routes for any prediction (not just the ones among the allowed templates). As a result, we can analyze model behavior in general, for specific types of predictions, or different domains. We experiment with Llama 2 and show that some attention head roles are overall important, e.g. previous token heads and subword merging heads. Next, we find similarities in Llama 2 behavior when handling tokens of the same part of speech. Finally, we show that some model components can be specialized on domains such as coding or multilingual texts.

1 Introduction

Current state-of-the-art language models (LMs) are built on top of the Transformer architecture (Vaswani et al., 2017; Brown et al., 2020; Touvron et al., 2023a,b). Inside the model, each representation evolves from the current input token embedding to the final representation used to predict the next token. This evolution happens through additive updates coming from attention and feed-forward blocks. The resulting stack of same-token representations is usually referred to as

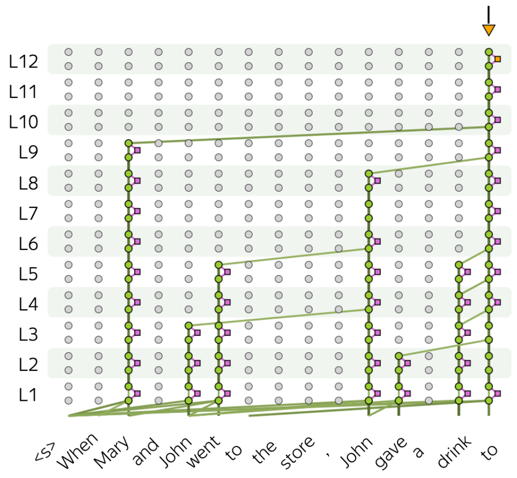


Figure 1: The important information flow routes for a token (Mary) prediction. GPT2-Small, $\tau = 0.04$.

"residual stream" (Elhage et al., 2021), and the overall computation inside the model can be viewed as a sequence of residual streams connected through layer blocks. Formally, we can see it as a graph where nodes correspond to token representations and edges to operations inside the model (attention heads, feed-forward layers, etc.).

While during a forward pass all the edges are present, computations important for each prediction are likely to form a small portion of the original graph (Voita et al., 2019; Wang et al., 2023; Hanna et al., 2023, among others). We extract this important subgraph in a top-down manner by tracing information back through the network and, at each step, leaving only edges that were relevant (Figure 1). To understand which edges are important, we rely on an attribution method (Ferrando et al., 2022) and refuse from activation patching typical for the existing mechanistic interpretability workflows (Wang et al., 2023; Hanna et al., 2023; Conmy et al., 2023; Stolfo et al., 2023). Firstly, patching requires human efforts to create templates

[*]Work done during an internship at Meta AI.

Originally introduced by Vig et al. (2020) to analyze LMs through the lens of causal mediation analysis.

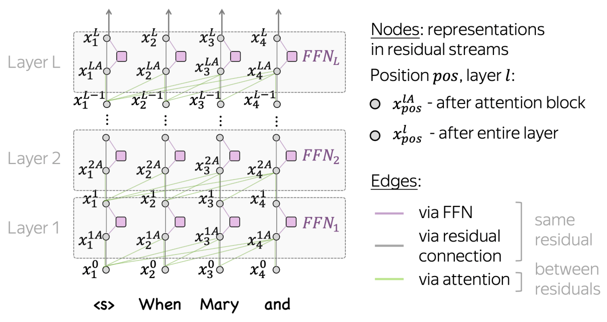

Layer L
Layer 2
Layer 1
FFN_L
FFN_2
FFN_1
<s>
When
Mary
and
Nodes: representations in residual streams
Position pos, layer l:
x^{lA}_{pos} - after attention block
x^{l}_{pos} - after entire layer
Edges:
via FFN
via residual connection
via attention
same residual
between residuals

Figure 2: Full information flow graph.

Algorithm 1: Extracting important information flow subgraph

Input: τ – minimum importance threshold
Output: G – important information flow subgraph

```
G = {}
current_nodes = {$x_t^N$}
while current_nodes is not empty:
    next_nodes = {}
    for $x$ in current_nodes:
        for $y$ in immediate_connections($x$):
            if $importance(y,x) > \tau$:
                $G = G \cup \{(y,x)\}$
                next_nodes.add($y$)
            end
        end
    end
    current_nodes = next_nodes
end
```

Figure 3: General-case algorithm for extracting the important subgraph, the information flow routes, from the full information graph (Figure 2).

and contrastive examples, thus is only applicable to a few pre-defined templates. Secondly, explaining a single prediction demands a substantial number of interventions (patches). Given that each intervention needs a forward pass, studying large models becomes increasingly impractical. In contrast, our method is about 100 times faster.

In the experiments, we first show that our information flow routes rely on the same task-specific attention heads found in patching circuits (Wang et al., 2023; Hanna et al., 2023). However, our method is more versatile and informative than patching: it can evaluate the importance of model components both (i) overall for a prediction, and (ii) compared to a contrastive example (i.e., patching setting). Aditionally, we argue that patching is fragile: its results can vary depending on the choice of the contrastive template (i.e., human preference).

Next, we come to the settings unreachable to patching, i.e. broad set of predictions and overall importance of model components. For Llama 2, we show that some attention heads' functions are overall important, e.g. previous token heads and subword merging heads. Then, we find that information inside Llama 2 flows similarly when handling tokens of the same part of speech. Finally, some model components are specialized on domains such as coding or multilingual texts.

Overall, our contributions are as follows:

- we propose to explain predictions of transformer LMs via information flow routes;
- compared to patching circuits, our method is (i) applicable to any prediction, (ii) more informative, and (iii) 100 times faster;
- we analyze information flow of Llama2 and find model components that are (i) important generally, and (ii) specific to domains.

2 Extracting Information Flow Routes

Figure 2 illustrates computations inside a Transformer LM as a graph.

Information flow graph. In the graph (i) nodes correspond to token representations and (ii) edges to operations inside the network moving information across nodes. Specifically, x^{lA}_{pos} and x^{l}_{pos} are representations of the token at position pos after the attention block in layer l or the entire layer, respectively. Each x^{l-1}_{1}, ..., x^{l-1}_{pos} is connected to x^{lA}_{pos} via attention edges (and via a residual stream edge from x^{l-1}_{pos} to x^{lA}_{pos}), x^{lA}_{pos} is connected to x^{l}_{pos} via two edges: the FFN output and the residual stream.

Extracting the important subgraph. While during a forward pass all edges in Figure 2 are present, computations relevant for each prediction are likely to form a small portion of the original graph (Voita et al., 2019; Wang et al., 2023, etc). We extract this important subgraph, the *information flow routes*, in a top-down manner by tracing information back through the network (Figure 3). We start from a single node – the representation on top of the residual stream. Then, we go over immediately connected lower nodes and, if important at this step, we add them to the subgraph along with the corresponding edges. The algorithm requires setting a threshold τ for the minimum edge importance.

Importance via attribution, not patching. To complete our algorithm in Figure 3, we need to specify how to compute edge importance. While lately it has become typical to use patching (see Appendix A for a more formal description) (Wang et al., 2023; Hanna et al., 2023; Conmy et al., 2023), *instead of patching, we choose to use attribution.*

This choice is crucial for our work and makes our method about 100 times faster than alternatives while (i) being able to recover previously discovered circuits, (ii) doing this in a more versatile manner, and (iii) leading to new observations. Next, we explain the specific attribution method we use.

2.1 Evaluating Edge Importance

For the attribution method, we adopt ideas from ALTI (Aggregation of Layer-Wise Token-to-Token Interactions) (Ferrando et al., 2022). While ALTI propagates attributions throughout the entire model, we only use its definition of contributions between connected nodes. We choose this method due to its simplicity, ease of implementation, and demonstrated effectiveness in practical applications, e.g. detecting hallucinations in neural machine translation (Dale et al., 2023a,b; Guerreiro et al., 2023).

2.1.1 Definition of Importance

In our graph (Figure 2), each node represents a sum of incoming vectors (edges). According to ALTI, the *importance* of each vector (edge) to the overall sum (node) is *proportional to its proximity* to the resulting sum. Formally, if $\boldsymbol{y} = \boldsymbol{z}_1 + \cdots + \boldsymbol{z}_m$,

$$importance(\boldsymbol{z}_j, \boldsymbol{y}) = \frac{proximity(\boldsymbol{z}_j, \boldsymbol{y})}{\sum\limits_k proximity(\boldsymbol{z}_k, \boldsymbol{y})}, \quad (1)$$

$$proximity(\boldsymbol{z}_j, \boldsymbol{y}) = \max(-||\boldsymbol{z}_j - \boldsymbol{y}||_1 + ||\boldsymbol{y}||_1, 0).$$

Here, we use negative distance as a measure of similarity: the smaller the distance between $\boldsymbol{z}_j$ and $\boldsymbol{y}$, the more the information of $\boldsymbol{z}_j$ in $\boldsymbol{y}$. Note also that we ignore contributions of the vectors lying beyond the l_1 length of $\boldsymbol{y}$. For more details, see Ferrando et al. (2022). Next, we define vector updates corresponding to FFN and the attention blocks edges.

2.1.2 Defining FFN Edges

For the FFN blocks, edge vectors are straightforward. Following the notation of Figure 2:

$$\boldsymbol{x}^l_{pos} = x^{lA}_{pos} + \mathrm{FFN}_l(\boldsymbol{x}^{lA}_{pos}),$$

where the terms correspond to the edges of the residual connection and FFN, respectively.

2.1.3 Defining Attention Edges

We follow previous work (Kobayashi et al., 2020; Ferrando et al., 2022) and decompose the output of an attention block into a sum of vectors (edges), each corresponding to a connection between residual streams (Figure 4).

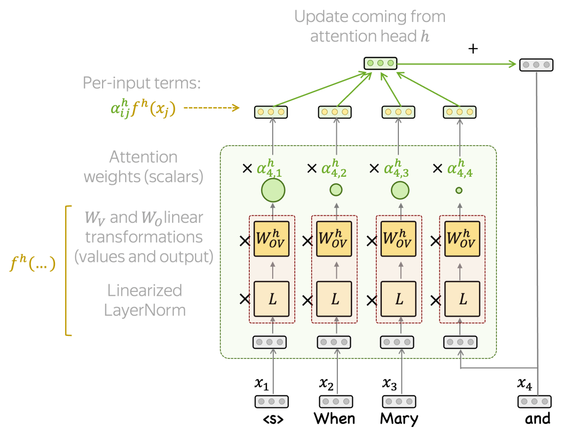


Figure 4: Decomposition of an update coming from an attention head into per-input terms. Layer indices are omitted for readability.

Attention heads. Formally, for attention head h

$$Attn^h(\mathbf{x}_{\le pos}) = \sum_{j \le pos} \alpha^h_{pos,j} f^h(\boldsymbol{x}_j), \quad (2)$$

where $f^h(\boldsymbol{x}_j) = \boldsymbol{x}_j L W^h_{OV}$, $W^h_{OV} = W^h_V W^h_O$ are the values and output combined matrix for head h, $\alpha^h_{pos,j}$ are scalar attention weights, and L is the linearized layer normalization (see Appendix B.1).

In a bit more detail, a typical attention implementation (1) weights corresponding value vectors within each attention head, (2) concatenates head outputs, and (3) further multiplies by the output matrix W_O. We split the output matrix W_O into its head-specific parts W^h_O, drag these parts inside attention heads, and combine them with the head's values matrix: $W^h_{OV} = W^h_V W^h_O$.

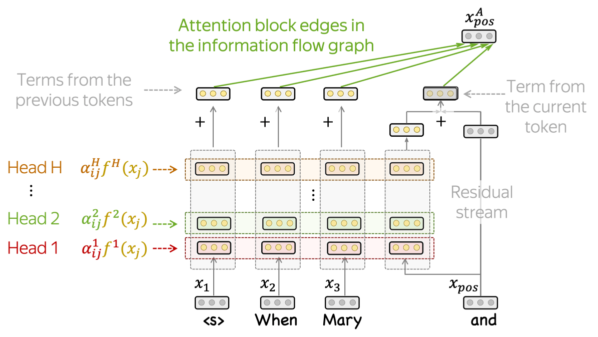
Overall, Figure 4 shows that on the way from attention head input to its output, each representation $\boldsymbol{x}_j$ is transformed linearly into $f^h(\boldsymbol{x}_j) = \boldsymbol{x}_j L W^h_{OV}$ and multiplied by a scalar attention weight $\alpha^h_{pos,j}$. This gives us the "raw output" emitted by each input vector $\boldsymbol{x}_j$ when treating attention weights as prediction-specific constants. In this view, *information flows through attention heads by independent channels* $\alpha^h_{pos,j} f^h(\boldsymbol{x}_j)$ *that converge in the next residual stream state*. We refer to each of this channels as a sub-edge.

Attention block. The information in an attention block flows through all the independent channels (sub-edges) in the H heads (Figure 5):

$$Attn(\mathbf{x}_{\le pos}) = \sum_h^H \sum_{j \le pos} \alpha^h_{pos,j} f^h(\boldsymbol{x}_j). \quad (3)$$

We compute the importance of each of the sub-edges in this sum as described in Section 2.1.1

From here onwards, we omit layer indices for readability.



Attention block edges in the information flow graph
x_{pos}^A
Terms from the previous tokens
Term from the current token
+
Residual stream
Head H $\alpha_{ij}^H f^H(x_j)$
Head 2 $\alpha_{ij}^2 f^2(x_j)$
Head 1 $\alpha_{ij}^1 f^1(x_j)$
x_1
x_2
x_3
x_{pos}
<s>
When
Mary
and

Figure 5: Decomposition of an update coming from an entire attention layer into per-input terms. Layer indices are omitted for readability.

and aggregate across heads the capacities of those sub-edges connecting the same pair of nodes, $\sum_h^H e_{pos,j}^h$. Additionally, we include the importance of the residual connection for the current token, $e_{pos}^{\text{res\_attn}}$ (Figure 5). Formally, attention edge importances are computed as

$$e_{pos,j}^{\text{attn}} = \begin{cases} \sum_h^H e_{pos,j}^h & \text{if } j \neq pos \\ \sum_h^H e_{pos,j}^h + e_{pos}^{\text{res\_attn}} & \text{if } j = pos \end{cases} \tag{4}$$

where $e_{pos,j}^h = importance(\alpha_{pos,j}^h f^h(\boldsymbol{x}_j), \boldsymbol{x}_{pos}^A)$ and $e_{pos}^{\text{res\_attn}} = importance(\boldsymbol{x}_{pos}, \boldsymbol{x}_{pos}^A)$ respectively, as defined in equation (1).

2.2 Extracting the Important Subgraph

Finally, when building the important information flow subgraph (routes), we add only the edges with an importance above the specified threshold τ (Figure 3). See further details in Appendix B.2.

3 Information Flow vs Patching Circuits

First, we compare our information routes to the circuits found in previous work for the Indirect Object Identification (IOI) (Wang et al., 2023) and Greater-than (Hanna et al., 2023) tasks.

Indirect object identification. The IOI task is to predict the next word in sentences like "*When Mary and John went to the store, John gave a drink to* ___". An initial clause features the names of two individuals (Mary and John). Then, the second clause depicts a person exchanging an item with the other person (i.e., Indirect Object) and the goal is to predict this Indirect Object (Mary).

Greater-than. In the Greater-Than task, the model is prompted to predict a number given a sentence following the template: "*The <noun> lasted from the year XXYY to the year XX*___". Here, the task is to predict a number higher than YY.

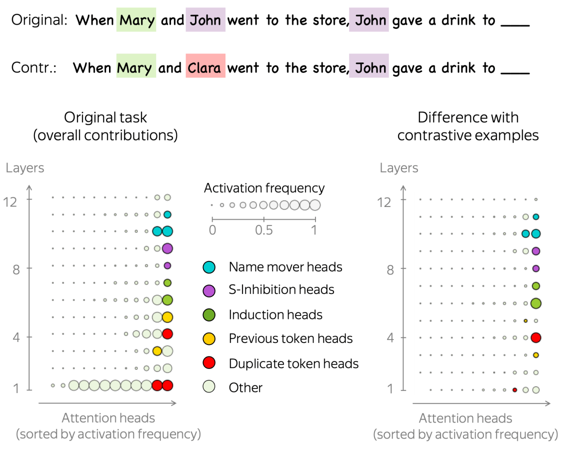


Figure 6: IOI, GPT2-Small. Attention head activation frequency ($\tau = 0.03$).

Information flow vs patching. The circuits for these tasks were previously found using activation patching with contrastive templates. These templates are designed to bring to surface the specific task. For example, for IOI a contrastive template contains three different names instead of repeating one of them (Figure 6). One of the differences between information flow and patching results is that our method finds all the components that contributed to the prediction, while patching finds what is important for the original task but not the contrastive baseline. As we will see, (i) our method can also be used in this manner, and (ii) it gives more reasonable results.

3.1 Indirect Object Identification

For the IOI task, the previously found circuit contains several attention heads with various functions, e.g. Name Mover Heads, Duplicate Token Heads, etc. (Wang et al., 2023). Using our method, we extract information flow routes for both the original task and the contrastive templates, and evaluate the activation frequency of attention heads.

When looking at overall contributions, Figure 6 (left) shows that our routes largely consist of the heads discovered previously. There are, however, several heads which are always part of the routes (have near 1 activation frequency) but are not in the "patching circuit" (shown with pale circles). Interestingly, when looking at the difference with contrastive templates (Figure 6, right), these generic heads disappear. This makes sense: information flow routes contain all the components that were important for prediction, and these contain (i) generic

We consider an attention head activated if it has at least a sub-edge in the information flow routes (important subgraph).

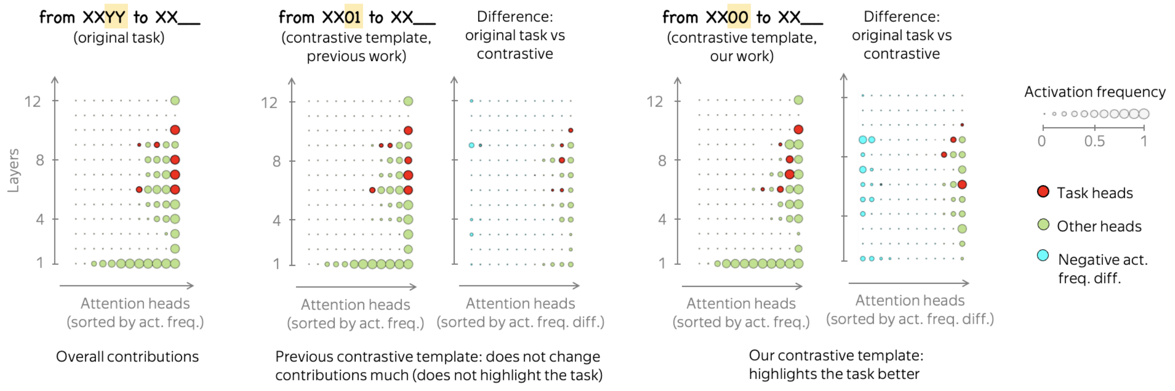


Figure 7: Greater-than, GPT2-Small. Attention head activation frequency ($\tau = 0.03$).

components that are overall important, and (ii) task-specific heads. Note also how our method goes *further than patching* for identifying the difference between the original and the contrastive tasks. Figure 6 shows that e.g. the previous token heads (yellow) are important for both types of predictions and are not specific to the IOI task.

3.2 Greater-than

For the overall contributions in the greater-than task (Figure 7, left), we also see that (i) task-specific heads discovered via patching are among the most important heads in the information flow routes, and (ii) many other heads are important.

Contrastive template matters. Interestingly, when using the same template as Hanna et al. (2023), i.e. "*...from the year XXYY*" with $YY = 01$, instead of $YY > 01$ overall contributions do not change much: all heads important for the original task are also important for the contrastive template (Figure 7, center). While this contradicts the original work by Hanna et al. (2023), this is expected: predicting a number higher than "*01*" still requires greater-than reasoning (not all numbers would fit after "*01*", e.g. "*00*" would not). In contrast, if we consider another template with $YY = 00$, overall contributions change and "greater-than heads" (red in Figure 7) become less important. This difference in the results for "*01*" vs "*00*" in the contrastive template highlights the fragility of patching: not only it requires human-defined contrastive templates, but also *patching results are subjective since they vary depending on the chosen template*.

Overall, we see that, compared to patching, information flow routes can be more versatile and informative. Indeed, they can find the importance of model components both (i) overall for a prediction, and (ii) compared to a contrastive example (i.e., patching setting), and are able to show differences between the contrastive templates.

3.3 Hundred Times Faster than Patching

Obtaining information flow routes involves a two-step process. First, we run a forward pass and cache internal activations. Then, we obtain edge importances to build the subgraph as depicted in Algorithm 3. For comparison purposes, we contrast our approach with the ACDC algorithm (Conmy et al., 2023), an automated circuit discovery algorithm based on patching. According to their findings, on the IOI task with a batch of 50 examples, it requires 8 minutes on a single GPU to discover the circuit. In contrast, our method accomplishes this task in 5 seconds, around 100x in time reduction.

4 General Experiments

Information flows inside the network via mechanisms implemented in the model. In this section, we try to understand whether the important mechanisms depend on the properties of processed by the model tokens and we look at general patterns in component importances.

Setting. Specifically, at each generation step, we evaluate the importance of each attention head and FFN block. In the full information flow graph from Figure 2 we look at individual residual streams and consider all the immediate connections to this stream. We then record importance of all the sub-edges corresponding to individual attention heads (i.e., $e^{h}_{i,j}$), as well as FFN blocks. We do this for each prediction in a subset of 1000 sentences from the C4 dataset (Raffel et al., 2020).

Additionally, we experiment with OPT-125m which has the same number of layers and heads than GPT2-small. We find that the information flow routes for IOI and greater-than tasks are similar for both models (Appendix C).

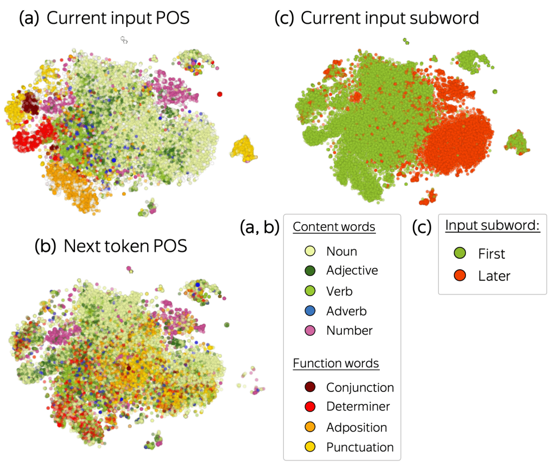


Figure 8: t-SNE of component importance vectors, coloured by: (a) input token POS tag, (b) next token POS tag, (c) input token is the first or a later subword. Llama 2-7B.

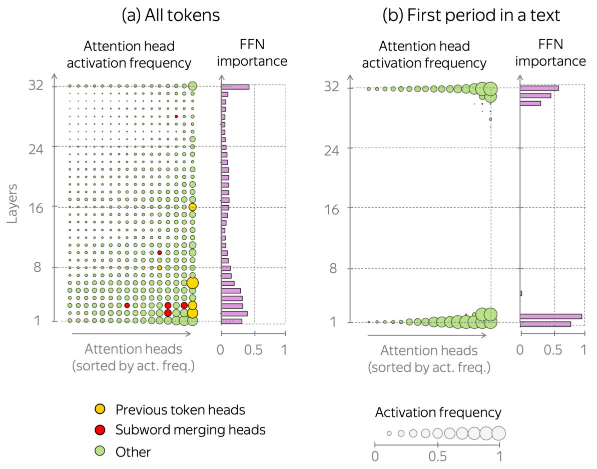


Figure 9: Attention head activation frequency ($\tau = 0.01$) and FFN block importance. We show only top-50% important heads. Llama 2-7B.

4.1 Component Importance for POS

First, let us see how component importance depends on parts of speech (POS). For this, we pack per-prediction component importances into vectors[5] and apply t-SNE. Figure 8 shows the resulting projection with datapoints colored according to either input or next token part of speech.[6]

Content vs function words. Figure 8a shows that for function words as inputs, component contributions are clustered according to their part of speech tag.[7] Roughly speaking, for function words the model has "typical" information flow routes and, by knowing component contributions, we can infer input token POS rather accurately. Interestingly, this is not the case for content words: while we can see verbs being somewhat separated from nouns, overall contribution patterns are mixed together in a large cluster. Apparently, reasoning for these words is more complicated and is defined by a broader context rather than input token POS.

First vs later subwords. Diving deeper, Figure 8c shows that contribution patterns strongly depend on whether the current token is the first or a later subword of a word. Clearly, there are some model components specialized for first or later subwords – we confirm this later in Section 4.3. Note also that Figure 8c explains the two distinct clusters for numbers we saw in Figure 8a (purple): they separate number tokens into first and later digits.

Patterns wrt to current vs next token. Finally, Figure 8b shows the same datapoints colored by the part of speech of the next token. Comparing this to Figure 8a, we see that contribution patterns depend more on input tokens rather than output tokens. This might be because in the lower part of the network, a model processes inputs in a generic manner, while the higher network part (that prepares to generate output) is more fine-grained and depends on more attributes than part of speech.

4.2 Bottom-to-Top Patterns

Since we already started talking about functions of the lower and the higher parts of the network, let us now look at the bottom-up contribution patterns. Figure 9a shows the average activation frequency of attention heads for a small $\tau = 0.01$ – this gives us an estimate of the number times an attention head affects residual stream. Additionally, we show the importance of the FFN block in each layer.

As expected, attention and feed-forward blocks are more active at the bottom part of the network, where the model processes input and extracts general information. Later, when a model performs more fine-grained and specialized operations, the activation frequency of individual attention heads and FFN blocks goes down. Interestingly, the last-layer FFN is highly important: apparently, this last operation between the residual stream and the unembedding matrix (i.e., prediction) changes representation rather significantly.

[5] Vector for to the pos-th position is defined as $\left(\sum_j e^{1,1}_{pos,j}, \sum_j e^{1,2}_{pos,j}, \dots, \sum_j e^{L,H}_{pos,j}, e^{\text{ffn}_1}_{pos}, \dots, e^{\text{ffn}_L}_{pos}\right)$.

[6] Here, we take the next token from the dataset and not the one generated by the model. While this adds some noise to the results, we expect at least parts of speech of the reference and the predicted tokens to be similar.

[7] We obtain POS tags with NLTK (universal tagset, Petrov et al. (2012)) and assign a tag to all subwords of each word.

4.3 Positional and Subword Merging Heads

Our observations above suggest that certain attention head functions might be overall important. First, in Section 3.1 we showed that previous tokens heads are generally important in the IOI task, both for target examples and the contrastive baseline – this attention head function might be important in general. In Section 4.1 we noticed a clear difference between component contributions for first and later subwords; hence, some model components might be responsible for that.

In what follows, we look at two attention head functions found earlier for machine translation, positional and subword merging heads, and check whether they are generally important. Differently from previous work, we take into account that attention weights might not reflect influences properly (Bastings and Filippova, 2020; Kobayashi et al., 2020, among others) and define a head function based on token contributions within the head and not attention weights.

Positional heads. Originally, previous token heads were found to be the most important attention heads in machine translation encoders (Voita et al., 2019). Now, let us check their importance for LLMs. We refer to a head as the previous token head if in at least 70% of the cases it puts more than half of its influence on the previous token. Figure 9a shows previous token heads in yellow. As in the earlier work for machine translation (Voita et al., 2019), we also see that for LLMs, (i) there are several previous tokens heads in the model, and (ii) almost all of them are by far the most important attention heads in the corresponding layers.

Subword merging heads. Putting together our first-vs-later subword observations in Section 4.1 and previous work, we might expect our model to have subword merging heads found for machine translation encoders (Correia et al., 2019). We refer to a head as a subword merging head if later subwords take information from the previous subwords of the same word but the head is *not* previous token head.

Figure 9a shows subword merging heads in red. We see that the model has several such heads in the bottom part of the network, and these heads are among the most important heads in the corresponding layers. This is interesting: previous work that noticed such head functions did not study their general importance (Correia et al., 2019). Note that these subword merging heads are important for later subwords and not important otherwise – this explains the clusters we saw in Figure 8c.

Formally, for a subword merging head (i) in at least 70% of the cases, later subwords put more than half of their influence on previous subwords of the same word, (ii) in at least 70% of the cases, for the first subword this head's overall influence is no more than 0.005%.

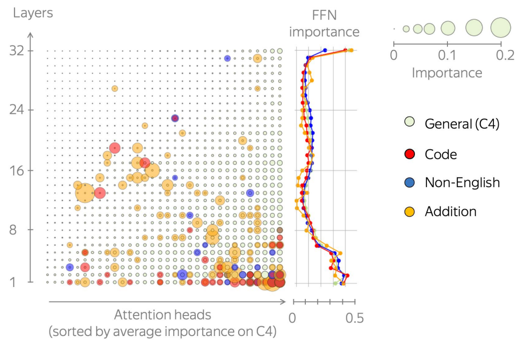


Figure 10: Average importance of attention heads and FFNs of Llama 2-7B across datasets. Heads with importance above 0.015 are shown for non-general domains.

We would like to highlight that for LMs, we are the first to talk about *general importance of attention heads* and to find that some of the most important heads are previous token and subword merging heads. Future work might explore the functions of other important heads in the model.

5 Domain-Specific Model Components

We saw that some attention heads' roles are generally important across predictions. Now, let us see whether not universally important components might be specialized for specific domains.

Setting. We evaluate the average importance of Llama 2-7B's components on 1000 sentences from different domains. Specifically, we consider (i) C4 (Raffel et al., 2020), (ii) different languages: English, Russian, Italian, Spanish FLORES-200 devtest sets (NLLB et al., 2022; Goyal et al., 2022), (iii) code data from CodeParrot, (iv) addition/subtraction data (Stolfo et al., 2023).

5.1 Components are Specialized

Figure 10 shows the importance of attention heads and FFN blocks in general and for specific datasets. We see that generally unimportant attention heads become highly relevant for specific tasks. For example, some important addition heads have low scores on C4 (yellow blobs to the left). Furthermore, domain-specific heads are different across

`https://huggingface.co/datasets/codeparrot/github-code`

🔴 Code: L 19, H17		🔴 Code: L 17, H 25				🔵 Non-En: L23, H6		🔵 Non-En: L 13, H17	
4	6	1	2	3	6	1	2	2	3
_return	_function	userId	_iOS	java	_android	ры	_immagini	erde	_Austral
return	_functions	groupId	_Xcode	_java	android	号	_techni	_Philipp	_Queensland
_Return	'\u200c'	filename	_ios	_Java	_Android	会	_tout	ulas	_Хронологија
_Returns	{}	_className	_Apple	Java	Android	ды	_ou	_Indones	_Sydney
_returning	iesa	artifactId	_iPhone	_JVM	_Xcode	лы	_pour	igin	Austral
Return	junit	_filename	_Mac	_instanceof	_tap	э	_cho	£	_kwiet

Figure 11: Top 6 tokens after projecting singular values of W_{OV}^h (ids in grey) onto the unembedding matrix.

domains: important heads for addition, code, and non-English are not the same. Overall, we can see that model components are largely specialized.

In Appendix E, we show more fine-grained results looking at tasks within the same narrow domain: addition vs subtraction. Regarding FFNs, we observe the last layer is much less relevant for non-English than other domains (Figure 10), and the importance of FFN blocks for addition and subtraction falls to zero in some layers (Figure 16, right). Future work might conduct a more fine-grained analysis of the importance of FFNs by looking at individual neurons.

5.2 Specialized Heads Output Topic-Related Concepts

In this section, we show that some of our domain-specific heads write into the residual stream highly interpretable and topic-related concepts.

Weight matrices analysis with SVD. As we illustrated in Figure 4, W_{OV}^h transforms representations from each of the residual streams into vectors that are added to the current residual stream. To understand what kind of information is embedded in this transformation, we use Singular Value Decomposition (SVD). To get an intuitive explanation of the W_{OV}^h impact, we can factorize it via the "thin" SVD (Millidge and Black, 2022) as $W_{OV}^h = U\Sigma V^T$. Projecting $\boldsymbol{x} \in \mathbb{R}^{1\times d}$ through W_{OV}^h is expressed as

$$\boldsymbol{x}W_{OV}^h = (\boldsymbol{x}U\Sigma)V^T = \sum_{i=1}^{r}(\boldsymbol{x}\boldsymbol{u}_i\sigma_i)\boldsymbol{v}_i^T. \quad (5)$$

Here, each $\boldsymbol{u}_i\sigma_i \in \mathbb{R}^{d\times 1}$ can be interpreted as a key that is compared to the query ($\boldsymbol{x}$) via dot product (Molina, 2023). Each query-key dot-product weights the right singular vector $\boldsymbol{v}_i^T$. If we project these right singular vectors to the unembedding matrix ($\boldsymbol{v}_i^T W_U$), we get an interpretation of the attention head's influence in terms of concepts (i.e., tokens) it promotes in the residual stream.

After applying layer normalization first.

$U \in \mathbb{R}^{d\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$, $V^T \in \mathbb{R}^{r\times d}$; d is vector dimensionality in the residual stream, r is the rank of W_{OV}^h.

Top singular values. For some of the heads specific to code and non-English inputs we saw in Figure 10, in Figure 11 we show the top 6 tokens that come from the described above projection. We see that code-specific heads promote tokens related to coding and technology (e.g., "Apple", "iOS", "Xcode", "iPhone", etc.). Heads most active for non-English promote tokens in multiple languages, avoiding English ones. Also, we see tokens related to locations and currencies ("Sydney", "£").

Overall, in this work we looked at the functions of attention heads from two perspectives: (i) when a head is active, and (ii) how it updates the residual stream, and found they are consistent. While a similar kind of analysis was done before for neurons (Voita et al., 2023), our method made it possible to talk about entire model components being active/non-active for a prediction.

6 Additional Related Work

Earlier works evaluating the importance of model components include Bau et al. (2019) who identified relevant neurons in NMT and Voita et al. (2019) who looked at the relevance of entire attention heads in the Transformer. Instead of overall importance, recent research largely focuses on specific tasks and aims to find the important LM subparts via activation patching (Wang et al., 2023; Hanna et al., 2023). Patching has been also used to locate factual knowledge in LMs (Meng et al., 2022) and to discover task vectors behind in-context learning capabilities (Hendel et al., 2023; Todd et al., 2023). Our work can be related to the concurrent line of research developing methods to approximate activation patching (Syed et al., 2023; Kramár et al., 2024; Hanna et al., 2024).

7 Conclusions

We view computations inside the Transformer as information flowing between token representations through model components. Using this view, we propose to interpret language model predictions by extracting the important part of the overall information flow. Our method for extracting these important information flow routes is automatic, highly efficient, applicable to any prediction, more versatile and informative compared to existing pipelines.

8 Limitations

Although the proposed method should work in most Transformer-based language models, the experiments in this paper are limited to models from the GPT-2, OPT and Llama 2 families.

9 Acknowledgements

We would like to thank Yihong Chen, Christoforos Nalmpantis, Igor Tufanov, Nicola Cancedda and Eduardo Sánchez for the insightful discussions.

References

Jasmijn Bastings and Katja Filippova. 2020. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–155, Online. Association for Computational Linguistics.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

David Dale, Elena Voita, Loic Barrault, and Marta R. Costa-jussà. 2023a. Detecting and mitigating hallucinations in machine translation: Model internal workings alone do well, sentence similarity Even better. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 36–50, Toronto, Canada. Association for Computational Linguistics.

David Dale, Elena Voita, Janice Lam, Prangthip Hansanti, Christophe Ropers, Elahe Kalbassi, Cynthia Gao, Loic Barrault, and Marta R. Costa-jussà. 2023b. Halomi: A manually annotated benchmark for multilingual hallucination and omission detection in machine translation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore. Association for Computational Linguistics.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Javier Ferrando, Gerard I. Gállego, and Marta R. Costa-jussà. 2022. Measuring the mixing of contextual information in the transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8698–8714, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Atticus Geiger, Kyle Richardson, and Christopher Potts. 2020. Neural natural language inference models partially embed theories of lexical entailment and negation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 163–173, Online. Association for Computational Linguistics.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

Nuno M. Guerreiro, Duarte Alves, Jonas Waldendorf, Barry Haddow, Alexandra Birch, Pierre Colombo, and André F. T. Martins. 2023. Hallucinations in large multilingual translation models. *Preprint*, arXiv:2303.16104.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Advances in Neural Information Processing Systems*, volume 36, pages 76033–76060. Curran Associates, Inc.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *Preprint*, arXiv:2403.17806.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. *Preprint*, arXiv:2310.15916.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. 2024. Atp*: An efficient and scalable method for localizing llm behaviour to components. *Preprint*, arXiv:2403.00745.

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36.

Beren Millidge and Sid Black. 2022. The singular value decompositions of transformer weight matrices are highly interpretable.

Raul Molina. 2023. Traveling words: A geometric interpretation of transformers. *Preprint*, arXiv:2309.07315.

Team NLLB, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Judea Pearl. 2009. *Causality*, 2 edition. Cambridge University Press.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Cody Rushing and Neel Nanda. 2024. Explorations of self-repair in language models. *Preprint*, arXiv:2402.15390.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. Understanding arithmetic reasoning in language models using causal mediation analysis. *Preprint*, arXiv:2305.15054.

Aaquib Syed, Can Rager, and Arthur Conmy. 2023. Attribution patching outperforms automated circuit discovery. *Preprint*, arXiv:2310.10348.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. 2023. Function vectors in large language models. *Preprint*, arXiv:2310.15213.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In

Advances in Neural Information Processing Systems, volume 33, pages 12388–12401. Curran Associates, Inc.

Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. 2023. Neurons in large language models: Dead, n-gram, positional. *Preprint*, arXiv:2309.04827.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

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

A Background on Activation Patching

Activation patching (Vig et al., 2020; Meng et al., 2022; Geiger et al., 2020; Wang et al., 2023) refers to intervening some internal activation (intermediate representation) computed by a model component c (attention head, feedforward network) in the forward pass ($f^c(x)$) with 'base' input x. The patched activation is taken from a forward pass $f^c(\tilde{x})$ on a 'source' input $\tilde{x}$. We can express this intervention using the do-operator (Pearl, 2009) as $f(x|\text{do}(f^c(x) = f^c(\tilde{x})))$. Upon intervention, the forward pass continues and the model output is compared with the prediction with the 'base' input, e.g. by measuring $f(x) - f(x|\text{do}(f^c(x) = f^c(\tilde{x})))$.

Identifying subnetworks (circuits) through activation patching has several shortcomings:

- It requires large human efforts to create the input base templates (x) and contrastive source examples ($\tilde{x}$) for the specific task to study, thus results vary depending on the choice of the contrastive template. Additionally, analyses are constrained to those templates, preventing from studying models on more general types of predictions.
- For each prediction, one needs to patch every edge (or node) in the computational graph, which becomes impractical when studying large language models.
- It has been shown that downstream components can compensate for the ablation as a form of self-repair (McGrath et al., 2023; Rushing and Nanda, 2024), which interferes with the analysis.

In contrast, our method doesn't require specific templates with contrastive examples which allows us to study specific tasks and more general behaviors, computes the information flow routes graph in a single forward pass, and it's not affected by self-repair issues, since we make no interventions.

B Details about the Information Flow Routes

B.1 Linearizing the Layer Normalization

Given an input representation $\boldsymbol{x}$, the layernorm computes

$$\text{LN}(\boldsymbol{x}) = \frac{\boldsymbol{x} - \mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})} \odot \gamma + \beta \tag{6}$$

with μ and σ obtaining the mean and standard deviation, and $\gamma \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ refer to learned element-wise transformation and bias respectively. Considering $\sigma(\boldsymbol{x})$ as a constant, LN can be treated as a constant affine transformation:

$$\mathrm{LN}(\boldsymbol{x}) = \boldsymbol{x}L + \beta \tag{7}$$

where $L \in \mathbb{R}^{d \times d}$ represents a matrix that combines centering, normalizing, and scaling operations together.

$$\mathbf{L} := \frac{1}{\sigma(\boldsymbol{x})} \begin{bmatrix} \gamma_1 & 0 & \cdots & 0 \\ 0 & \gamma_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \gamma_n \end{bmatrix} \begin{bmatrix} \frac{n-1}{n} & -\frac{1}{n} & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & \frac{n-1}{n} & \cdots & -\frac{1}{n} \\ \cdots & \cdots & \cdots & \cdots \\ -\frac{1}{n} & -\frac{1}{n} & \cdots & \frac{n-1}{n} \end{bmatrix}$$

The linear map on the right subtracts the mean to the input vector, $\boldsymbol{x}' = \boldsymbol{x} - \mu(\boldsymbol{x})$. The left matrix performs the hadamard product with the layer normalization weights ($\boldsymbol{x}' \odot \gamma$).

B.2 Further details about the implementation of Information Flow Routes

Although we didn't notice significant differences, in our experiments we first remove the sub-edges with importances $e^{h}_{pos,j}$ and $e^{\text{res\_attn}}_{pos,j}$ below τ and renormalize the rest before aggregating across heads.

Generally, edges from FFN updates have higher importance than those from attention heads. This is expected, since attention heads are only a small part of the attention block, but the FFN is not decomposed further. Therefore, one can set different thresholds for retaining attention and FFN edges, although we did not experiment with this.

B.3 Folding the Layernorm

Any Transformer block reads from the residual stream by normalizing before applying a linear layer (with weights W and $\boldsymbol{b}$) to the resulting vector:

$$\mathrm{LN}(\boldsymbol{x}_j)W + \boldsymbol{b} \tag{8}$$

Following the reformulation of the layernorm shown in Eq. 7, we can fold the weights of the layernorm into those of the subsequent linear layer as follows:

$$\begin{aligned} \mathrm{LN}(\boldsymbol{x}_j)W + \boldsymbol{b} &= \left(\frac{1}{\sigma(\boldsymbol{x}_j)}\boldsymbol{x}_j L + \beta\right) W + \boldsymbol{b} \\ &= \frac{1}{\sigma(\boldsymbol{x}_j)}\boldsymbol{x}_j LW + \beta W + \boldsymbol{b} \\ &= \frac{1}{\sigma(\boldsymbol{x}_j)}\boldsymbol{x}_j W^* + \boldsymbol{b}^* \end{aligned} \tag{9}$$

where $W^* = LW$ and $\boldsymbol{b}^* = \beta W + \boldsymbol{b}$.

C Examples of Routes

Figures 12 and 13 show the information flow routes for IOI and greater-than tasks extracted for GPT2-small and OPT-125m.

D Peculiar Information Flow Patterns, or Periods Acting as BOS

In Section 4.1 we talked about general contribution patterns and saw visible clusters corresponding to the input tokens' part of speech. Now, let us go deeper and look in detail at one of the clusters. We choose the outlier punctuation cluster shown in yellow in Figure 8 (to the right) – this cluster corresponds to the first period in a text.

Figure 9b shows the average importance of model components for examples in this cluster. We see that for these examples, the residual stream ignores all attention and FFN blocks in all the layers except for the first and last few: for most of the layers, contributions of all model components are near zero. When we look at the information flow graphs for these examples, we see that, even for a rather small threshold $\tau = 0.01$, bottom-to-top processing happens largely via "untouched" residual connection (Figure 14). In Appendix D we show that up to the last three layers, this residual stream *takes the role of the BOS token* and future tokens treat this residual stream as such. While for some examples this might be reasonable, this also happens in cases where the period does not have an end-of-sentence role. For example, Figure 14 (right): while in a sentence `For the second game in a row, St. Thomas ...` the first period does not have the end-of-sentence meaning, the residual stream is still acting in the same manner. In future work, it might be valuable to explore whether this behavior might cause incorrect generation behavior.

Figure 15 shows an example of an attention map for one of the heads in Llama 2-7b. We see that after the first period, attention is spread between the BOS token and this period.

E Specialization

While the important heads for addition and subtraction largely intersect (Figure 16 right), we see several heads that are active only for one task and not the other (bright blue and yellow blobs). This could mean that this fine-grained specialization might be

823 out of 826 datapoints.

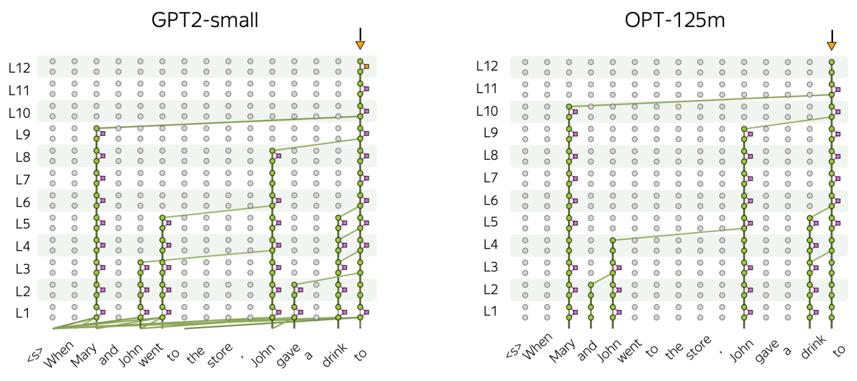


Figure 12: The important information flow routes, IOI task, $\tau = 0.04$.

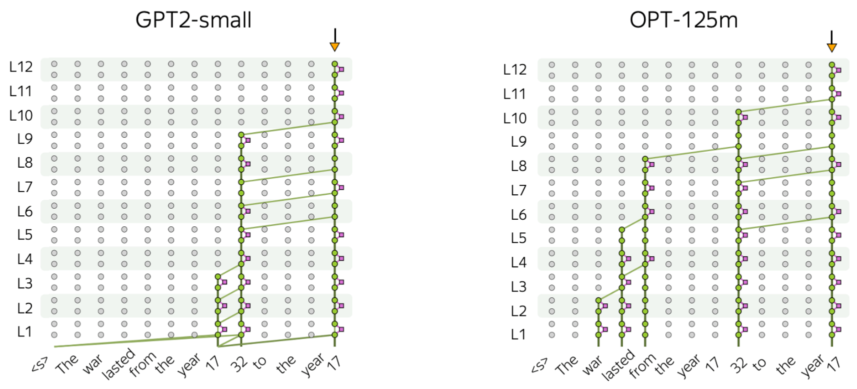
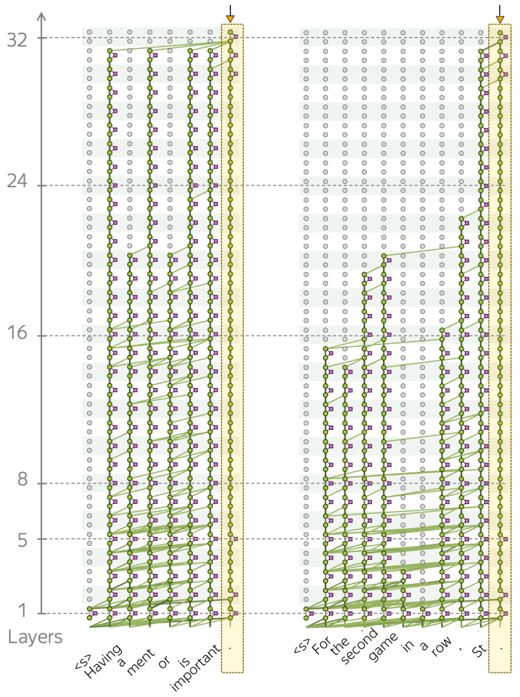


Figure 13: The important information flow routes, greater-than task, $\tau = 0.04$.



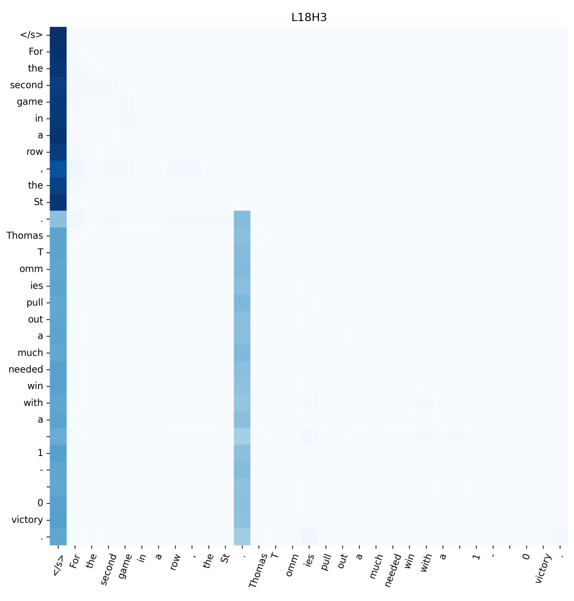


Figure 14: Examples of important information flow subgraphs ($\tau = 0.01$). Llama 2-7B.

Figure 15: Llama 2 7B attention weights matrix L18H3.

responsible for "reasoning" inside the model and not just domain-specific processing; future work may validate this further.

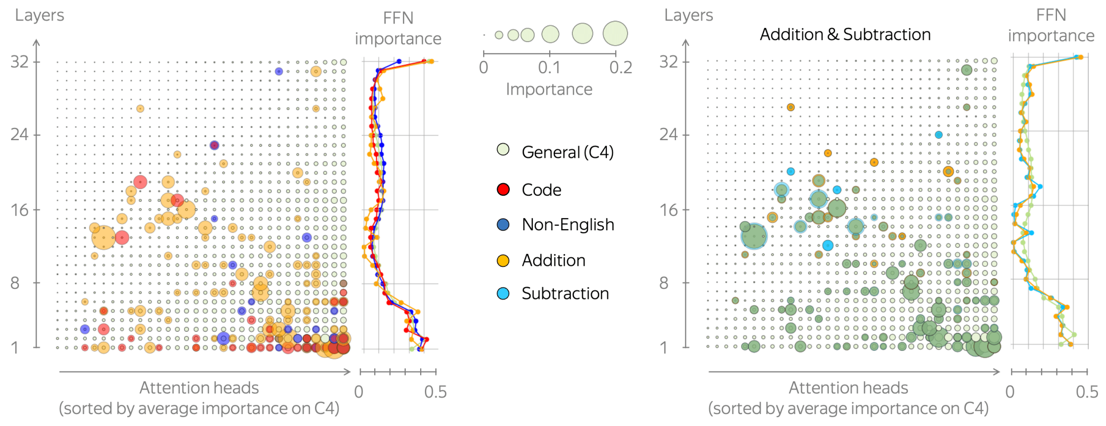


Figure 16: Average importance of attention heads and FFN blocks for different datasets. For non-general domains, we show only heads with importance higher than 0.015. Llama 2-7B.