Mechanistic Interpretability of Grammatical Structures

Chirag Adwani* cadwani@umd.edu

Marvyn Bailly* mbailly@umd.edu

Kejia Zhang* zkj15@umd.edu

1 Problem statement

Large Language Models (LLMs) have demonstrated remarkable versatility, achieving high performance not only across diverse natural languages but also in formal languages such as code. Despite this success, the internal computational mechanisms enabling this transfer remain largely opaque. While recent work in mechanistic interpretability has identified specific circuits for linguistic tasks, such as the Indirect Object Identification (IOI) circuit in English (Wang et al., 2023), it remains an open question whether these mechanisms are language-specific instantiations or reusable, universal algorithms.

The primary goal of this project is to investigate the extent to which LLMs reuse computational primitives across significant distribution shifts. Specifically, we aim to determine if the attention heads responsible for reference resolution (e.g., "Name Mover" heads) in English are conserved when the model is tasked with:

1. **Multilingual Transfer:** Processing the same semantic task in a grammatically distinct natural language (Chinese).
2. **Cross-Domain Transfer:** Processing an analogous logical task in a formal programming language (Python).

Understanding whether models learn abstract, language-agnostic algorithms or rely on distinct, surface-level heuristics for different domains is critical. If models reuse the same circuits for English, Chinese, and Python, it suggests they have internalized a generalized understanding of the task. Our work seeks to quantify this overlap using path patching and activation frequency correlations, thereby contributing to a deeper understanding of modularity and mechanism reuse in Transformer models.

* Equal contribution.

2 What we proposed vs. what we accomplished

We have provided a short list of tasks that we accomplished during this project and explanations for missing items that we presented in the project proposal:

- ~~Recreate some of the results from (Zhang et al., 2025)~~
- ~~Recreate some of the results from (Wang et al., 2023)~~
- *Extend the work of (Zhang et al., 2025) to remove different grammatical structures*: We overestimated the difficulty of designing sentences to study that were solvable by simple models.
- *Study the effects of fuzzy patching*: We failed to do this because we decided it wasn't within the scope of our project anymore.
- ~~Extend the work of (Wang et al., 2023)~~

3 Related Work

Early work on multilingual language processing focused on developing shared representations that enable transfer across languages. Large multilingual language models such as XLM-R (Conneau et al., 2020) and BLOOM (Workshop et al., 2023) demonstrated that a single parameterized model

could support dozens of languages without explicit parallel supervision, achieving strong zero-shot performance. These results established multilingual pretraining as a powerful paradigm, but largely treated models as black boxes, leaving open questions about how cross-lingual sharing is realized internally.

Subsequent research began to investigate whether multilingual models rely on common latent representations across languages. Representational analyses showed that hidden states for different languages often align in intermediate layers, suggesting the emergence of shared semantic spaces. For example, lens-based methods such as the tuned lens (Belrose et al., 2023) enable decoding of intermediate predictions and have been used to analyze how representations evolve across layers. More recently, Wendler et al. (2024) provide evidence that multilingual models may internally operate in a dominant "pivot" language (typically English), implying that non-English inputs are projected into an English-aligned latent space before generation. While these findings suggest representational sharing, they do not establish whether the underlying *computation* is shared or whether similar representations arise from distinct mechanisms.

A complementary line of work investigates language-specific specialization within multilingual models. Rather than focusing on representations, these studies examine whether parameters are differentially responsible for individual languages. Choenni et al. (2023) analyze language-specialized subnetworks and show that modular structure can arise naturally, but that sparse fine-tuning intended to enforce modularity may instead increase cross-lingual sharing. At a finer granularity, Tang et al. (2024) identify a small subset of neurons that are highly predictive of language identity and demonstrate that manipulating these neurons can steer the model' s output language. These results indicate that multilingual models are neither fully shared nor fully partitioned, but they do not explain how language specialization manifests in concrete linguistic operations.

Mechanistic interpretability provides tools for bridging this gap by enabling causal analysis of internal computation. The transformer circuits framework (Nanda et al., 2021) formalizes attention heads as compositional units. A canonical example is the Indirect Object Identification (IOI) circuit identified by Wang et al. (2023), which showed that specific heads (e.g., "Name Movers") causally implement algorithms to resolve references. While originally discovered in English, recent work has questioned if these algorithmic primitives are universal. Merullo et al. (2024) suggest that circuit components are reused across related tasks, raising the hypothesis that transformers implement a library of reusable primitives rather than task-specific solutions.

Extending this to a multilingual setting, Zhang et al. (2025) compared circuits for similar linguistic constructions across languages, finding that structural alignment often leads to circuit reuse. Our work builds on this but introduces a distinct dimension: domain transfer. In addition to analyzing cross-lingual consistency (English vs. Chinese), we investigate whether algorithmic reuse extends to formal languages (Python code), where the logical task (reference resolution) remains the same but the syntactic structure differs fundamentally.

Methodologically, we draw on recent advances in automated circuit analysis. While earlier methods relied on manual path patching, techniques like Information Flow Routes (Ferrando and Voita, 2024) allow for more scalable analysis of computation graphs. Adopting a simplified ablation-based approach inspired by these methods, along with the standard path-patching framework, we systematically compare the activation frequencies and causal roles of attention heads across English, Chinese, and Python to determine the extent of mechanism reuse.

4 Dataset

An IOI sentence begins with an initial dependent clause (e.g., "When Mary and John went to the store,") and ends with a main clause (e.g., "John gave a drink to Mary"). The two names that are introduced in the dependent clause are referred to as the subject (`S`) and the indirect object (`IO`). In the main clause, the subject will preform an action onto the indirect object in such a way that the indirect

object is the last word in the sentence. For example, "When Mary and John went to the store, John gave a drink to Mary". If we remove the last word from this example and call it x_{IOI}, then we say that the model has correctly completed the IOI task if it outputs the indirect object (e.g., "Mary").

To understand how the model solves the IOI task, we require a corresponding corrupt pair x_{corrupt} that has some of the relevant information required to complete the IOI task removed. Here, we will maintain the grammatical structure while removing the S name reoccurrence. This prevents the model from being able to solve the IOI task due to the ambiguity of x_{corrupt}. An example pair from the data set:

- x_{IOI} = "When Mary and John went to the store, John gave a drink to"
- x_{corrupt} = "When Alice and Bob went to the store, Julia gave a drink to"

4.1 Data generation

To generate a dataset, we design a small set of IOI sentence templates that have interchangeable words and a small set of possible words to draw from. In our case, we use a set of names and objects to fill in templates such as the following

- "When `IO` and `S1` went to the store, `S2` gave the `object` to"
- `IO`,`S1`, `S2`$\in$ {"Alice","Mary","Bob",...}
- `object`$\in$ {"book","pen","phone",...}

We manually designed 5 templates, 30 names, and 12 objects. To generate a dataset pair $(x_{\text{IOI}}, x_{\text{corrupt}})$, we randomly pick 1 template, 5 names, and 1 object. We fill in the template with S1 and S2 as the same name to get x_{IOI} and use the remaining three names (now with S1 and S2 not equal) to get x_{corrupt}. We will also save the S and IO name in the dataset as these will give us a way to measure if the model is completing the IOI task correctly. Such a set up allows us to generate 522000 unique x_{IOI} giving a dataset of $(x_{\text{IOI}}, x_{\text{corrupt}}, \texttt{IO}, \texttt{S})$. Throughout the paper, we use three sized datasets, small with 100, medium with 1000, and large with 5000 examples.

4.2 Data preprocessing

Since we desire a language model to succeed the IOI task when given x_{IOI}, we need to design the template sentences with care such that the model outputs `IO` with high probability. Thus we need to verify templates before using them. For example, a template such as "While `IO` and `S1` were working, `S2` passed the object to" often predicts "the" or "a" over `IO`. To automatically verify that a template produces the correct solution, we can use the template to generate $(x_{\text{IOI}}, x_{\text{corrupt}}, \texttt{IO}, \texttt{S})$ and verify that model returns `IO`.

4.3 Extending data generation

To generate the Chinese IOI dataset, we translated the templates, objects, and names into Chinese using a native speaker. Here, only slight modifications had to be made to the objects list to remove objects that did not make sense in the templates. The following is an example entry in the dataset generated from the template "当 `IO` 和 `S1` 去商店时，`S2` 把 `object` 给了":

```
{
  "clean":
      "当小娟和小龙去商店时，小龙把书给了",
  "corrupt":
      "当小燕和小强去商店时，小芳把书给了",
  "io_name": "小娟",
  "s_name": "小龙",
  "io_token": " 小娟"
},
```

Similarly, we design a Python IOI dataset which is meant to be understood as the coding language analogue of the IOI task. In this case, we use a single template of

```
def {function name}({var1}, {var2}):\n return
    {var1} {operator}
```

and using the same process, we generate pairs such as:

```
{
"clean": "def merge(b, a):\n return b +",
"corrupt": "def merge(e, i):\n return c
    +",
"s_arg": "b",
"io_arg": "a",
},
```

The datasets, generators and explanations of how to use them can be found under the data_generation folder on the Github page.

5 Baselines

The baseline for our work are the results shown in (Wang et al., 2023) where the authors use mechanistic interpretability on English IOI sentences. They propose an algorithm to solve IOI tasks and argue that a small model, GPT-2 small with 12 layers and 12 heads, follows this algorithm. We use this paper as a baseline since they pioneered path patching, a method of mechanistic interpretability, and so presents a clear and straightforward task. If we are able to recreate some of the results shown using GPT-2, we will have a strong baseline for our work.

To understand the proposed algorithm, let's take the example "When Mary and John went to the store, John gave a drink to". A simple approach to solve this IOI problem is to

- Identify all previous names in the sentence (Mary, John, John)
- Remove the duplicate names (John)
- Output the last remaining name (Mary)

The paper claims that GPT-2 small implements a similar algorithm through a classes of heads known as duplicate token heads, S-inhibition heads, and name mover heads. The duplicate token heads identify tokens that have already appeared in the sentence. The S-inhibition head can be understood as removing attention from the `S` tokens. Finally the name mover head is responsible for picking the correct name in the final position. One can see how the algorithm can be created using these heads to remove attention from the duplicate name and thus have the name mover give the `IO` token over the `S` token. For our baseline, we aim to verify that the significant heads of GPT-2 small applied to our English dataset correspond to those found in (Wang et al., 2023). Moreover, we will identify the name mover heads.

6 Mechanistic Interpretability

All of our implementations were preformed using Python and can be found on the Github page. To preform mechanistic interpretability, we use a Python library TransformerLens created by Neel Nanda. The library allows us to cache and overwrite the outputs of heads on any layer in GPT-2 using existing functions calls. All code was executed locally on a RTX-4070 GPU using CUDA.

6.1 Path Patching

The area of mechanistic interpretability, seeks to understand why a model produces an output y for an input x. Patch patching, is a method that aims to determine the direct effect of each head on the final logits using the original x_{IOI} and a corrupted input x_{corrupt}. To identify the attention heads that play a significant role in correctly solving x_{IOI}, we follow a simplified path patching approach focused on the direct effect of each head on the logits. For every layer l and head h, we measure the recovery in the *logit difference* when the corrupt head's output in x_{corrupt} is restored to its state from the clean input x_{IOI}. The logit difference is defined as the difference between the logit of the correct answer token and the logit of the incorrect answer token (e.g., the indirect object vs. the subject in IOI):

$$\Delta = \text{logit}(IO) - \text{logit}(S). \tag{1}$$

The algorithm proceeds as follows for each head h:

1. Run the model on the **clean** input and cache the activation output of head h.
2. Run the model on the **corrupt** input and calculate the baseline logit difference ($\mathcal{L}_{corrupt}$).
3. Run the model on the **corrupt** input again, but intervene by **patching** (replacing) the output of head h with the cached clean activation.
4. Calculate the new logit difference ($\mathcal{L}_{patched}$) and compute the effect: $\Delta = \mathcal{L}_{patched} - \mathcal{L}_{corrupt}$.

The intuition is that we isolate the specific contribution of head h to the final prediction. By running the model on corrupt inputs but restoring head h's clean output, we ask: "If this specific head sees the correct context while the rest of the model sees noise, how much of the model's performance is restored?" A positive Δ indicates the head directly contributes

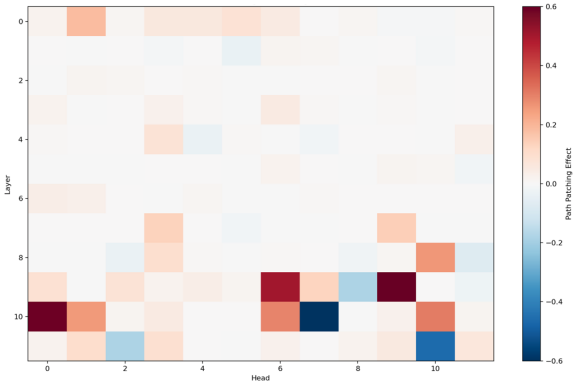

Figure 1: Logit difference of IO and S logit between corrupt and patched feed forward result for the English IOI Task.

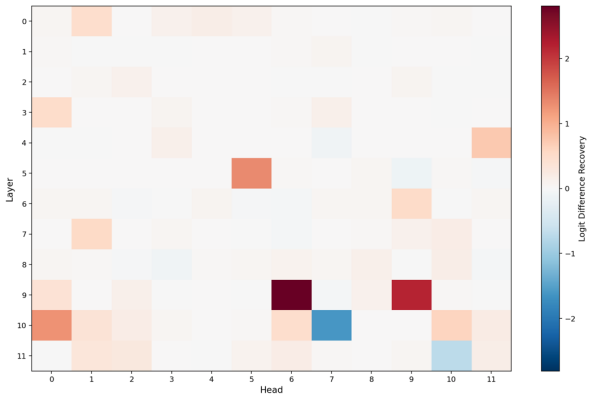

Figure 2: Logit difference of IO logit between corrupt and patched feed foward result for the Python IOI Task.

to the correct prediction, while a value near zero implies the head does not leverage the clean context for this task directly. A negative Δ shows that the head's contribution leads to the wrong name being returned.

6.2 English name mover head analysis

The results of running the path patching algorithm on the English IOI dataset yields Figure 1. The heatmap corresponds to the 12 layers and 12 heads architecture of GPT-2 small. Let's use the notation LnHm, to denote the m'th head of the n'th layer. The tile colors show the logit difference corresponding to patching the corresponding in LhHm where red shows a positive logit difference, white shows near zero logit difference, and blue shows a negative logit difference. Thus heads colored in red can be understood to positively influence the model to output the correct name (IO) while heads in blue influence the model to output the incorrect name (S). As suggested by (Wang et al., 2023), these heads correspond to name mover and negative name mover heads. To reiterate, a name mover head plays the role of picking the final logit to be the IO or S name where negative name movers influence the incorrect solution of S over IO. We see that heads L9H9, L10H0, and L9H6 have strong positive effects in the logic difference while L10H7 and L11H10 have strong negative effects. These heads correspond to the name mover and negative name movers found in our baseline (Wang et al., 2023).

To further verify what effect the name mover heads are having on tokens, we can extract the state of the residual stream at the position of each name token after the first layer. We can now directly multiply this by the value and output matrix (OV matrix) of a name mover head to simulate the head perfectly attending to this token. Next, we multiply the result with the unembedding matrix to get the logit probabilities. Finally we compute the proportion of samples that contain the input name token in the top 5 logits can call this value score. The results are summarized in Figure 3. We can see that identified name mover heads from before also have a high copy score such as L9H9 and L10H0. (Wang et al., 2023) argue that this provides sufficient evidence to conclude that these heads are indeed name mover heads playing a role in the model's production of a the correct solution. Repeating this process but focusing on the negative name movers, we can see how often the heads return S in its top logits. The results are summarized in Figure 4 where we see that heads such as L10H7 and L11H10 have a high negative copy score verifying their role as negative name movers in the IOI task.

Our baseline continues to use a similar method to identify and verify the role of each head of their suspected algorithm. Rather than repeating this work, we following (Zhang et al., 2025) and look into the IOI task in Chinese to determine the importance of grammatical structures such as past tense in the IOI task. While we aimed to recreate the name mover head analysis on a CPM-distilled

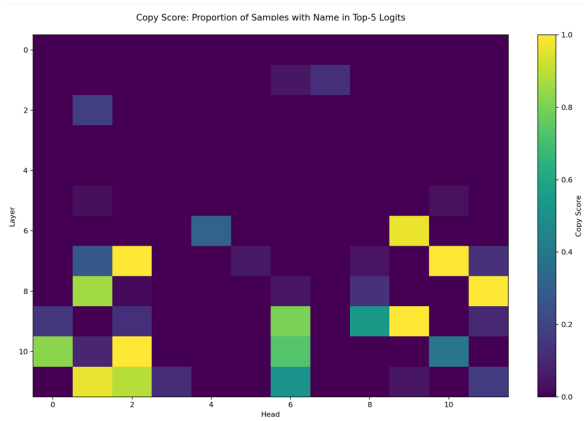

Figure 3: Logit difference of IO logit between corrupt and patched feed forward result for the English IOI Task.

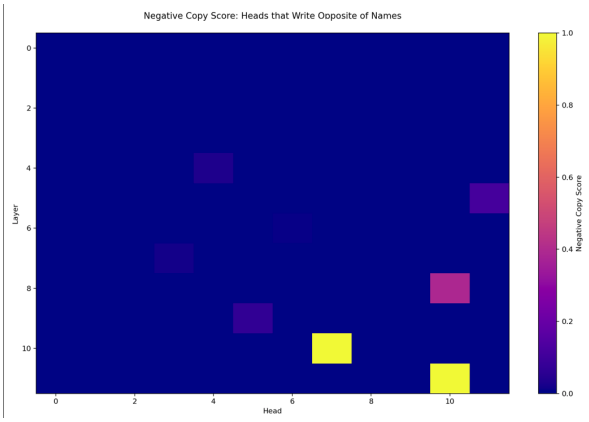

Figure 4: Logit difference of IO logit between corrupt and patched feed forward result for the English IOI Task.

model, a Chinese trained model with the same architecture as GPT2-small, we were unable to get the model working correctly. The suspected issue is within how the tokenizer handled the input which lead to the model outputting none-meaningful results. This motivated us to move on to the next form of mechanistic interpretability presented in the following section using a multilingual model.

6.3 Activation frequencies and Ablations

The aforementioned method of Path Patching, though considered to be an industry standard now, suffers the pitfall of being able to carefully construct minimal pair (corrupt and non-corrupt) templates. This carefulness was demonstrated in the previous section, where we had to select names (both IO and S) that are exactly one token, and that there is one particular token that repeats and hence must be discarded by the LM and then also design the corresponding corrupt task. The circuit that was originally discovered and proposed by (Wang et al., 2023), and henceforth verified by this project, relies heavily on these carefully designed minimal pairs.

Thus, given these downsides, we looked at the method of Information Flow Routes in (Ferrando and Voita, 2024; Wang et al., 2023), and employed a simplified version of the method. Following, is a brief description of what we did. In each iteration of the code file `activation_frequencies_ifr.py` on our Github page, we compute the contribution of each head to the residual stream. We achieve this by running the model normally for a specific task and then ablating each head, one at a time, and remove its contribution from the residual stream by manually zeroing out the contribution of the per-head vector in the residual stream. After this, we compare the difference of logits before and after ablation, and if they exceed a certain threshold, we mark those heads as "active". These head-level ablations have been described in (Heimersheim and Nanda, 2024; Li and Janson, 2024; Petrovic, 2025). For this task, the corrupt IO pairs are not required, and we ask the LM to only read the non-corrupt IO task and complete the sentence.

Following are the mathematical details of the same.

For instance, if we are looking at the i^{th} IOI example, we have the logit difference (defined in Eq.1), Δ_i, where the logits are, as usual, the pre-softmax outputs of the language model. We first run the model unablated to compute our baseline logit difference $\Delta_i^{\text{baseline}}$ for this example, across all layers and heads. Then, we remove the contribution of the attention head at (l, h) at the l^{th} Layer, h^{th} head, and compute the corresponding logit difference $\Delta_{i,l,h}^{\text{ablated}}$.

Then, we quantify the contribution of the head at (l, h) by computing its *causal effect* $\delta_{i,l,h}$:

$$\delta_{i,l,h} := \left|\Delta_i^{\text{baseline}} - \Delta_{i,l,h}^{\text{ablated}}\right|.$$

Quantifying the contribution this way, al-

lows us to conclude that

$$\begin{cases} \delta > 0 & \text{the head has significant contribution,} \\ \delta \approx 0 & \text{the head is irrelevant.} \end{cases}$$

Then, we decide on an arbitrary cutoff τ, which we choose as $\tau = 0.03$ for our computations, and produce the Activation matrix with elements

$$A_{i,l,h} = 1[\delta_{i,l,h} > \tau].$$

The above notation denotes the fact that the $(i, l, h)^{th}$ entry of A is 1 if the condition $\delta_{i,l,h} > \tau$ holds true, and is 0 when its false. And then finally, we aggregate these activations across examples and compute the *activation frequency* of each head by defining

$$f_{l,h} := \frac{1}{N} \sum_{i=1}^{N} A_{i,l,h}.$$

This $f_{l,h}$ is the average number of activations (activation frequency) of the head at (l, h). Note that $f_{l,h} \in [0, 1]$. In simpler words, this is the fraction of examples where this particular head had a significant contribution to the given IOI task, i.e., this particular head causally mattered.

Finally, we compute the activation frequency for the IOI task on our English database and on our Chinese one and then correlate them using some simple statistical methods.

To give more details about the ablation, we have to go back to the details of the transformer architecture. We know that a transformer starts with token embeddings of the prefix, adds the positional embeddings to the token embeddings, and performs a masked self attention task to generate the first hidden layer output. This output is what is called the residual stream. After masked self attention is performed again on the next layer, each head's contribution (output of the self-attention, taking previous layer's output as this layer's input) is simply added to this residual stream. For this activation frequency task, we ablate a given head at a given layer by simply removing this contribution that it would have added to the stream and proceeding through the rest of the generation as we normally would.

6.4 Correlations between activation frequencies

The activation frequency task described in the previous section was performed on the generated English and Chinese datasets using the model BLOOM-560M(Workshop et al., 2023). The following two activation frequency matrices were generated

$$f_{l,h}^{English} \quad \text{and} \quad f_{l,h}^{Chinese},$$

each of size `[n_layers, n_heads]`. In the case of BLOOM-560M, `n_layers = 24` and `n_heads = 16`. This model was chosen for this task because BLOOM is a multilingual model, trained on both English and Chinese data and hence is more suited for such a task. This is in fact what (Zhang et al., 2025) also use for computing their correlations.

`crosslingual_head_corr.py` takes in these activation frequency matrices $f_{l,h}^{English}$ and $f_{l,h}^{Chinese}$ as input and flattens them to a 1D vector of length $M =$`n_heads*n_layers` which is $24*16 = 384$ in the case of BLOOM-560M. So, we have

$$\mathbf{x} = \text{vec}\left(f_{l,h}^{English}\right), \mathbf{y} = \text{vec}\left(f_{l,h}^{Chinese}\right) \in R^M.$$

We then compute the Pearson correlation coefficient

$$r_{Pearson} := \frac{\sum_{j=1}^{M}(x_j - \bar{x})(y_j - \bar{y})}{\sqrt{\sum_{j=1}^{M}(x_j - \bar{x})^2}\sqrt{\sum_{j=1}^{M}(y_j - \bar{y})^2}},$$

where $\bar{x} = \frac{1}{M}\sum_{j=1}^{M} x_j$ and $\bar{y} = \frac{1}{M}\sum_{j=1}^{M} y_j$. The values we obtained for this coefficient were $r_{Pearson} = 0.6854$ for the small data sets, and $r_{Pearson} = 0.7096$ for the medium data sets. This is in great agreement with the (Zhang et al., 2025), where they obtained a correlation of 0.72. We argue that the differences in the coefficients are because of the dataset we generated being different from the one that the authors (Zhang et al., 2025) use in their work.

Additionally, we also compute the Spearman correlation coefficient, which is just the Pearson correlation coefficient computed between the ranked versions of $\mathbf{x}, \mathbf{y}$. Basically, we rearrange both of these vectors in a descending order and define the following vector

$$\tilde{x_j} = \text{rank}(x_j),$$

where $\text{rank}(x_j)$ is the location where x_j is in the descending order, and similarly

$$\tilde{y_j} = \text{rank}(y_j),$$

and then compute

$$r_{Spearman} = r_{Pearson}(\mathbf{x}, \mathbf{y}).$$

We argue that for this task, correlating the ranks produces a particularly good correlation metric as it potentially omits any arbitrary differences that might exist on the actual values of activation levels. Basically, this is the quantification of whether the heads that are important in English, are the same heads that are also important in Chinese, even if their activation frequencies are not exactly the same.

The Spearman coefficient turned out to be $r_{Spearman} = 0.6216$ for the small data sets and $r_{Spearman} = 0.6588$ for the medium ones. This demonstrates a significant correlation between what heads are important for both tasks.

6.5 Extending the IOI task

We desired to repeat the mechanistic interpretability process, using path patching and correlations, on our own novel task. Initially, we aimed to derive a new type of sentence structure that had a simple algorithmic solution as the IOI task, but this proved to be a nontrivial task. The difficulty arose from balancing the required simplicity of the sentence for the model to output the desired result while having maintaining sufficient complexity to generate a dataset and draw meaningful results. Since GPT2-mini was trained on code, we moved on to trying to design a Python analog to the IOI task and concluded with the previously shown template. The motivation being that GPT-2 could deploy a similar algorithm of identify and removing duplicate names but now with parameter names.

The results from repeating the name mover head analysis on the Python dataset are summarized in Figure 2. We see a strong similarity of the positive and negative logit difference as seen in the English IOI task. For example, L10H0 and L9H6 in both cases are predominate positive heads while L10H7 and L12H10 are negative. Notable, the presence of a significant name mover head L5H5 is present in the Python but not English IOI task. Carrying out the same Correlation computation between English and Python IOI, we found a spearman coefficient on the medium dataset of $r_{Spearman} = 0.3295$.

The relatively modest Spearman correlation ($r_{Spearman} = 0.3295$) suggests a nuance in how these tasks are processed. While the most dominant heads align perfectly, the broader distribution of head importance differs, implying that GPT-2 is not simply executing an identical algorithm for both English and Python IOI problems. Instead, the model appears to be repurposing the same fundamental "building blocks," specifically the Name Mover and S-Inhibition heads, to construct different, domain-specific algorithms. This reuse of components suggests these heads act as specialized functional primitives (e.g., "copy this token" or "suppress this repetition") that are recruited by the model regardless of whether the context is natural language or code. This can be seen by the presence of a name mover head in one task but not the other. Thus, while the global circuit wiring adapts to the distinct syntax of Python, the local reliance on these specific heads for reference resolution remains constant, explaining their consistent appearance in the heatmaps but low correlation coefficient.

7 Error analysis

This method of mechanistic interpretability is only as good as the language model is at the task. That is, we can only test tasks that a small language model can consistently preform well on. In addition, the tasks has to be designed in such a way that a dataset can be generated from them. A few iterations of IOI templates had to be tried before finding a form that was suitable. To increase the complexity of the task, a larger model is required but this increases the complexity and computation time of the path patch analysis.

For the correlation coefficients, a statistical significance test was run on the obtained Pearson and Spearman coefficients, with the Null hypothesis being $r = 0$, i.e., no correlation. A t-statistic was used, and the results turned

out to be statistically significant with a p-value $< \mathcal{O}(10^{-50})$.

8 Contributions of group members

- Chirag Adwani: Wrote code for the Head-level ablations, activation frequencies and correlations. Ran preliminary versions of those codes on my (less powerful) device[1]. Assisted in Write-ups for those sections.
- Marvyn Bailly: Wrote code to generate the datasets. Came up with templates for the English and coding datasets that passed accuracy testing. Generalized code to preform general path patching algorithm. Ran computations on personal GPU. Assisted in write ups.
- Kejia Zhang: Idea design. Assisted in write ups and analysis. Designed Chinese template and proof read dataset.

9 Conclusion

From figure 1, and a similar figure generated for the Chinese IOI task, we already had anecdotal correlation between the heads that are important for the IOI tasks in both languages. The images looked largely similar, even for a model that is not trained on Chinese data, i.e., GPT-2 Small. However, as we saw in subsections 6.3 and 6.4, there was, in fact, a mathematical correlation between the activation frequencies of each head inside the model BLOOM-560M. This consistency in the internal circuitry for the same task, but for different languages is perhaps a conclusive evidence that our language models have a language-independent understanding of certain tasks and are not simple next word generators. They are very much capable of performing tasks that require a higher level of understanding.

We believe that grasping what levels of understanding our models are capable of, may prove to be of paramount importance to the improvement of said models and also an improvement to how and what we use them for.

Furthermore, the application of Path Patching was decisive in moving our analysis from correlation to causation. While the activation frequency analysis provided strong statistical evidence for cross-lingual similarities between English and Chinese, it was the granular intervention of path patching that allowed us to mechanically verify the behavior of the Python IOI task. By isolating the direct effect of individual heads on the logits, we confirmed that heads L9H9 and L10H0 act as domain-agnostic "functional primitives," executing the same copy-and-paste operation regardless of whether the input is natural language or code. This distinction is crucial: while the global correlation between English and Python was moderate ($r_{Spearman} \approx 0.33$), the path patching results demonstrated that the core machinery—the Name Mover heads—remains invariant. This suggests that future interpretability work should not only look for shared global circuits but also for these specialized, reusable modular components that the model recruits across vastly different modalities.

Therefore, one of the things that we would like to explore, if we were to continue this project, is how do the correlations of activation frequency look like between the English IOI task and the Python IOI task. An argument to be made here is that if the Language model truly understands the deeper concept of an IOI task, so much so that similar heads are able to perform the task in Chinese too, can this logic be extended to a Python IOI task? An obvious shortcoming for us is the complete dissimilarity between the human-level meanings of our data sets of English IOI and Python IOI. In the case of English and Chinese, we at least had similar/same sentences that were mere translations of each other. Therefore, if we generate an English IOI dataset that is a "translation" of our existing Coding IOI dataset, will we see a similar correlation that we saw for English and Chinese IOI?

One of the biggest things we learnt from this project was simply how vast the field of Mechanistic Interpretability actually is. The blog post cited above (Petrovic, 2025), describes many methods of the particular question of tracing LLM activations for a given task. Our entire project, more or less, lies within this realm of activation logging as described in the blog. (Rai et al., 2025) gives an

[1]When compared to the computational resources available with group member Marvyn Bailly

amazing overview of the current state of the field of mechanistic interpretability.

Furthermore, looking at adjacent fields inside Mechanistic interpretability, the tools that help us diagnose the understanding that a model has also seem very important. This is different from identifying the circuits or parts of model that are important for its working, which is what Path Patching and Head-level Ablations achieve. Tools like Probing and Visualization (Rai et al., 2025), on the other hand, help us investigate the presence of various linguistic features in the model activations.

10 AI Disclosure

- Did you use any AI assistance to complete this project (report and/or code)? If so, please also specify what AI(s) you used.
  - Marvyn: I used Gemini and Claude Sonnet throughout the project.
  - Chirag: I used ChatGPT for my tasks.
  - Kejia: I used Gemini and ChatGPT throughout the project.

If you answered yes to the above question, please complete the following as well:

- Marvyn: I used Gemini to generate the Python code needed to build the database. I used Gemini to generate the code needed to preform path patching. I used Claude Sonnent to connect the projects and produce the final results.

- Chirag: I used ChatGPT to help me write the codes for doing head-level ablations task and computing the correlations between the English and Chinese activation frequencies. I also used it for brainstorming the exact method of head-level ablation that we ended up presenting. The method presented in (Ferrando and Voita, 2024) turned out to be a bit too complicated, so, while asking ChatGPT to help me understand the paper, it suggested that I use this simplified method of finding contributions of each head to the final IOI task. It was also able to provide citations for this method when I asked for them.

- Kejia: I used chatGPT 5.2 to find the related papers. I used the Gemini to check if my logic makes sense. Overall experience is good, but chatGPT still holds strong hallucination with paper finding. It will provide the false doi that cannot match with the paper name. Generally I feel chatGPT cannot deal with the detailed and precise doi related work. Even you send the correct paper/doi/link to it, but it still insists with the other unrelated one. You have to upload the pdf to force it focus on the correct one.

References

Belrose, N., Ostrovsky, I., McKinney, L., Furman, Z., Smith, L., Halawi, D., Biderman, S., and Steinhardt, J. (2023). Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Choenni, R., Shütova, E., and Garrette, D. (2023). Examining modularity in multilingual lms via language-specialized subnetworks. *arXiv preprint arXiv:2311.08273*.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of ACL*.

Ferrando, J. and Voita, E. (2024). Information flow routes: Automatically interpreting language models at scale. *arXiv preprint arXiv:2403.00824*.

Heimersheim, S. and Nanda, N. (2024). How to use and interpret activation patching.

Li, M. and Janson, L. (2024). Optimal ablation for interpretability.

Merullo, J., Eickhoff, C., and Pavlick, E. (2024). Circuit component reuse across tasks in transformer language models. *arXiv preprint arXiv:2310.08744*.

Nanda, N. et al. (2021). A mathematical framework for transformer circuits.

Petrovic, D. (2025). Advanced interpretability techniques for tracing llm activations. https://dejan.ai/blog/advanced-interpretability-techniques-for-tracing-llm-activa

Rai, D., Zhou, Y., Feng, S., Saparov, A., and Yao, Z. (2025). A practical review of mechanistic interpretability for transformer-based language models.

Tang, T., Luo, W., Huang, H., Zhang, D., Wang, X., Zhao, X., Wei, F., and Wen, J.-R. (2024). Language-specific neurons: The key to multilingual capabilities in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Wang, K. R., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. (2023). Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

Wendler, C. et al. (2024). Do llamas work in english? on the latent language of multilingual transformers. In *Proceedings of ACL*.

Workshop, B., :, Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P. O., Sanh, V., Laurençon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Soroa, A., Aji, A. F., Alfassy, A., Rogers, A., Nitzav, A. K., Xu, C., Mou, C., Emezue, C., Klamm, C., Leong, C., van Strien, D., Adelani, D. I., Radev, D., Ponferrada, E. G., Levkovizh, E., Kim, E., Natan, E. B., Toni, F. D., Dupont, G., Kruszewski, G., Pistilli, G., Elsahar, H., Benyamina, H., Tran, H., Yu, I., Abdulmumin, I., Johnson, I., Gonzalez-Dios, I., de la Rosa, J., Chim, J., Dodge, J., Zhu, J., Chang, J., Frohberg, J., Tobing, J., Bhattacharjee, J., Almubarak, K., Chen, K., Lo, K., Werra, L. V., Weber, L., Phan, L., allal, L. B., Tanguy, L., Dey, M., Muñoz, M. R., Masoud, M., Grandury, M., Šaško, M., Huang, M., Coavoux, M., Singh, M., Jiang, M. T.-J., Vu, M. C., Jauhar, M. A., Ghaleb, M., Subramani, N., Kassner, N., Khamis, N., Nguyen, O., Espejel, O., de Gibert, O., Villegas, P., Henderson, P., Colombo, P., Amuok, P., Lhoest, Q., Harliman, R., Bommasani, R., López, R. L., Ribeiro, R., Osei, S., Pyysalo, S., Nagel, S., Bose, S., Muhammad, S. H., Sharma, S., Longpre, S., Nikpoor, S., Silberberg, S., Pai, S., Zink, S., Torrent, T. T., Schick, T., Thrush, T., Danchev, V., Nikoulina, V., Laippala, V., Lepercq, V., Prabhu, V., Alyafeai, Z., Talat, Z., Raja, A., Heinzerling, B., Si, C., Taşar, D. E., Salesky, E., Mielke, S. J., Lee, W. Y., Sharma, A., Santilli, A., Chaffin, A., Stiegler, A., Datta, D., Szczechla, E., Chhablani, G., Wang, H., Pandey, H., Strobelt, H., Fries, J. A., Rozen, J., Gao, L., Sutawika, L., Bari, M. S., Al-shaibani, M. S., Manica, M., Nayak, N., Teehan, R., Albanie, S., Shen, S., Ben-David, S., Bach, S. H., Kim, T., Bers, T., Fevry, T., Neeraj, T., Thakker, U., Raunak, V., Tang, X., Yong, Z.-X., Sun, Z., Brody, S., Uri, Y., Tojarieh, H., Roberts, A., Chung, H. W., Tae, J., Phang, J., Press, O., Li, C., Narayanan, D., Bourfoune, H., Casper, J., Rasley, J., Ryabinin, M., Mishra, M., Zhang, M., Shoeybi, M., Peyrounette, M., Patry, N., Tazi, N., Sanseviero, O., von Platen, P., Cornette, P., Lavallée, P. F., Lacroix, R., Rajbhandari, S., Gandhi, S., Smith, S., Requena, S., Patil, S., Dettmers, T., Baruwa, A., Singh, A., Cheveleva, A., Ligozat, A.-L., Subramonian, A., Névéol, A., Lovering, C., Garrette, D., Tunuguntla, D., Reiter, E., Taktasheva, E., Voloshina, E., Bogdanov, E., Winata, G. I., Schoelkopf, H., Kalo, J.-C., Novikova, J., Forde, J. Z., Clive, J., Kasai, J., Kawamura, K., Hazan, L., Carpuat, M., Clinciu, M., Kim, N., Cheng, N., Serikov, O., Antverg, O., van der Wal, O., Zhang, R., Zhang, R., Gehrmann, S., Mirkin, S., Pais, S., Shavrina, T., Scialom, T., Yun, T., Limisiewicz, T., Rieser, V., Protasov, V., Mikhailov, V., Pruksachatkun, Y., Belinkov, Y., Bamberger, Z., Kasner, Z., Rueda, A., Pestana, A., Feizpour, A., Khan, A., Faranak, A., Santos, A., Hevia, A., Unldreaj, A., Aghagol, A., Abdollahi, A., Tammour, A., HajiHosseini, A., Behroozi, B., Ajibade, B., Saxena, B., Ferrandis, C. M., McDuff, D., Contractor, D., Lansky, D., David, D., Kiela, D., Nguyen, D. A., Tan, E., Baylor, E., Ozoani, E., Mirza, F., Ononiwu, F., Rezanejad, H., Jones, H., Bhattacharya, I., Solaiman, I., Sedenko, I., Nejadgholi, I., Passmore, J., Seltzer, J., Sanz, J. B., Dutra, L., Samagaio, M., Elbadri, M., Mieskes, M., Gerchick, M., Akinlolu, M., McKenna, M., Qiu, M., Ghauri, M., Burynok, M., Abrar, N., Rajani, N., Elkott, N., Fahmy, N., Samuel, O., An, R., Kromann, R., Hao, R., Alizadeh, S., Shubber, S., Wang, S., Roy, S., Viguier, S., Le, T., Oyebade, T., Le, T., Yang, Y., Nguyen, Z., Kashyap, A. R., Palasciano, A., Callahan, A., Shukla, A., Miranda-Escalada, A., Singh, A., Beilharz, B., Wang, B., Brito, C., Zhou, C., Jain, C., Xu, C., Fourrier, C., Periñán, D. L., Molano, D., Yu, D., Manjavacas, E., Barth, F., Fuhrimann, F., Altay, G., Bayrak, G., Burns, G., Vrabec, H. U., Bello, I., Dash, I., Kang, J., Giorgi, J., Golde, J., Posada, J. D., Sivaraman, K. R., Bulchandani, L., Liu, L., Shinzato, L., de Bykhovetz, M. H., Takeuchi, M., Pàmies, M., Castillo, M. A., Nezhurina, M., Sänger, M., Samwald, M., Cullan, M., Weinberg, M., Wolf, M. D., Mihaljcic, M., Liu, M., Freidank, M., Kang, M., Seelam, N., Dahlberg, N., Broad, N. M., Muellner, N., Fung, P., Haller, P., Chandrasekhar, R., Eisenberg, R., Martin, R., Canalli, R., Su, R., Su, R., Cahyawijaya, S., Garda, S., Deshmukh, S. S., Mishra, S., Kiblawi, S., Ott, S., Sang-aroonsiri, S., Kumar, S., Schweter, S., Bharati, S., Laud, T., Gigant, T., Kainuma, T., Kusa, W., Labrak, Y., Bajaj, Y. S., Venkatraman, Y., Xu, Y., Xu, Y., Xu, Y., Tan, Z., Xie, Z., Ye, Z., Bras, M., Belkada, Y., and Wolf, T. (2023). Bloom: A 176b-parameter open-access multilingual language model.

Zhang, R., Yu, Q., Zang, M., Eickhoff, C., and Pavlick, E. (2025). The same but different: Structural similarities and differences in multilingual language modeling. In *The Thirteenth International Conference on Learning Representations*.