

Interpreting Key Mechanisms of Factual Recall in Transformer-Based Language Models

Ang Lv^{1*} Kaiyi Zhang^{1*} Yuhan Chen^{1*}
 Yulong Wang² Lifeng Liu² Ji-Rong Wen¹ Jian Xie^{2†} Rui Yan^{1†}
¹ Gaoling School of Artificial Intelligence, Renmin University of China
² Baichuan Inc.

{anglv, kaiyizhang, yuhanchen, jrwen, ruiyan}@ruc.edu.cn
 {wangyulong, liulifeng, richard}@baichuan-inc.com

<https://github.com/trestad/Factual-Recall-Mechanism>

Abstract

In this paper, we deeply explore the mechanisms employed by Transformer-based language models in factual recall tasks. In zero-shot scenarios, given a prompt like “The capital of France is,” task-specific attention heads extract the topic entity, such as “France,” from the context and pass it to subsequent MLPs to recall the required answer such as “Paris.” We introduce a novel analysis method aimed at decomposing the outputs of the MLP into components understandable by humans. Through this method, we quantify the function of the MLP layer following these task-specific heads. In the residual stream, it either erases or amplifies the information originating from individual heads. Moreover, it generates a component that redirects the residual stream towards the direction of its expected answer. These zero-shot mechanisms are also employed in few-shot scenarios. Additionally, we observed a widely existent anti-overconfidence mechanism in the final layer of models, which suppresses correct predictions. We mitigate this suppression by leveraging our interpretation to improve factual recall confidence. Our interpretations have been evaluated across various language models, from the GPT-2 families to 1.3B OPT, and across tasks covering different domains of factual knowledge.

1 Introduction

Mechanistic interpretability research aims to reverse engineer deep neural networks into human-understandable algorithms or mechanisms [41, 28]. This endeavor holds significant value for various applications, including providing insights into internal reasoning [5, 16, 42], enriching knowledge [22, 14], improving controllability [44, 2], guiding future architecture and training paradigm designs [6, 19, 35], and detecting safety issues [36]. Despite the remarkable achievements made by language models [1, 29, 30, 37, 38] based on Transformer [39] in natural language understanding and generation, they remain mainly opaque to human, with many of their internal mechanisms understudied. In this paper, we delve into the mechanisms in factual recall tasks, which represent one of the cornerstone abilities of contemporary language models [48, 47].

Recent observations made by Merullo et al. [23] have delineated the behavior of language models when tasked with factual recall into two major phases as layers deepen: (1) *Argument Formation*: When presented with a query such as “The capital of France is”, the decoded top tokens from the residual stream gradually yield the name of the queried country, in this case, “France.” The authors

*Equal Contribution.

†Corresponding authors.

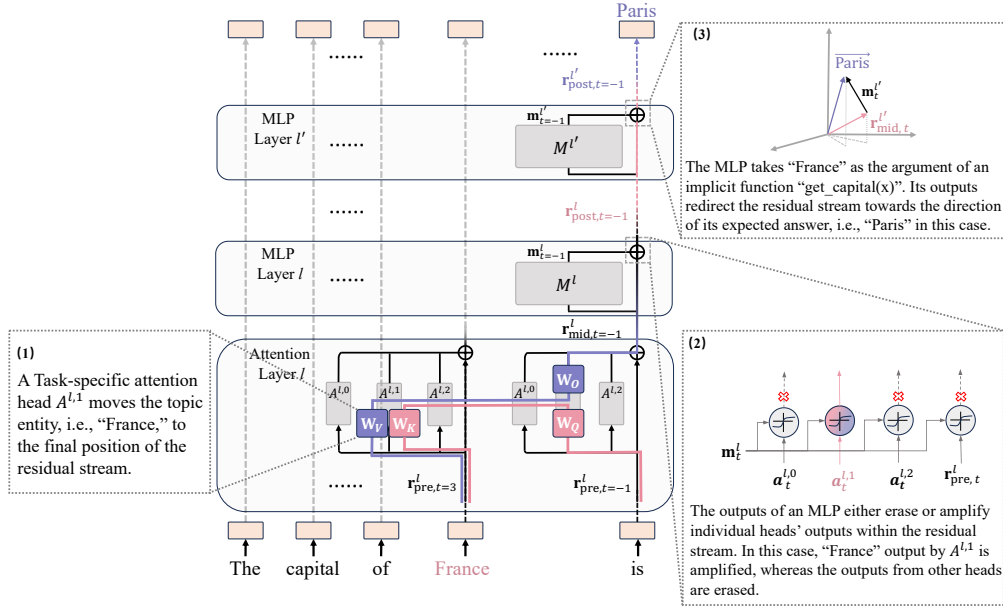


Figure 1: The key mechanisms of factual recall employed by Transformer-based language models. Please refer to §2 for detailed notations.

likened this process to the model forming an implicit function akin to “get_capital(x),” where “France” is the argument for the implicit function. (2) *Function Application*: As layers deepen, the top tokens decoded from the residual stream transition from the country name to the capital name, such as shifting from “France” to “Paris.” The authors described this transition as the application of the implicit function and highlighted the crucial role of MLPs in this phase.

Although these observations lay a valuable groundwork, several unresolved research questions remain, which are essential for achieving a deep understanding of the factual recall mechanism in language models. For instance, how does the model extract the argument from the context and then pass it to the so-called “function?” What exactly constitutes “function application,” and how does it correlate with MLPs? Moreover, [23] primarily focused on a one-shot setting. What about the factual recall mechanism of language models in basic zero-shot settings or when provided with more shots?

In this paper, we delve into the mechanistic interpretability of these crucial questions and offer several innovative insights. We investigated the scalability of Transformer-based language models, ranging from GPT-2 small, medium, and large [34], to OPT-1.3B [46], across various factual recall tasks encompassing diverse domains of knowledge. Based on our analysis, we outline the following steps by which a language model achieves factual recall in zero-shot. For readers unfamiliar with mechanistic interpretability research, we suggest referring to Section 2 for necessary background information beforehand.

(1) First, task semantics formed in shallow layers activate some task-specific attention heads in the mid-to-deep layers. These attention heads possess a QK matrix inherently sensitive to tokens related to specific topics, such as the name of a country. They attend to these topic tokens and move them to the final position of the residual stream. This mechanism enables the model to extract the “argument” from the context and pass it for further processing, e.g., the function application. Additionally, we observed that some attention heads’ OV matrix can directly map the argument to the desired outputs without further processing via MLPs. This mapping can be viewed as a partial function application.

(2) Given that all attention heads’ outputs are aggregated equally before being added to the residual stream, the subsequent MLP serves as an “activation” for each head’s output. It can erase or amplify the outputs of individual heads by generating vectors that either align with or oppose the direction of heads’ outputs. Through this process, the passed argument by task-specific heads “stands out” within the residual stream.

(3) In addition to activating attention heads, the MLP’s output also incorporates a task-aware component that directs the residual stream towards the direction of the target token’s unembedding vector. We demonstrated that the component’s addition into the residual stream accomplishes the “function application.”

We illustrate these three steps in Figure 1. Furthermore, we observed that various models suppresses correct predictions in the final layer. This is achieved by integrating frequent tokens into the residual stream via attention heads and employing the MLP to steer the residual stream towards a “mean” token over the training corpus. This phenomenon appears to be a universal anti-overconfidence mechanism adopted by various models to evade a significant training loss, and it persists irrespective of the task, model, and the number of in-context demonstrations.

To sum up, this paper has the following contributions:

1. In zero-shot scenarios, we offer a detailed interpretation of the “argument passing” and “function application” mechanisms employed in Transformer-based language models for factual recall tasks. The mechanisms we detected still work in few-shot scenarios.
2. We propose a novel analysis method to understand certain deep MLPs. Our approach reveals that some MLPs behave akin to the activation of attention heads while also generating a task-aware component responsible for “function application.” The efficacy of this analysis method is supported by ample empirical evidence.
3. We observed a universal anti-overconfidence mechanism in the final layer of models. Building on our interpretation, we devise strategies to mitigate this anti-overconfidence mechanism, enhancing factual recall confidence.

2 Background

2.1 A Residual-Centric Perspective of the Transformer

Elhage et al. [8] presented a mathematical framework for comprehending the Transformer architecture, which we follow in this paper. In Transformer models, the residual connections [12] serve as the data bus for information flow. Each Transformer layer comprises multiple attention heads and an MLP layer, each of which reads from and writes to this data bus.

Within a Transformer layer indexed as l , comprising H attention heads, we label each attention head as L/Hh , where $h \in [0, 1, \dots, H-1]$. Each attention head’s specific operation is denoted by $A^{l,h}(\mathbf{r})$, where \mathbf{r} is the input residual stream. Similarly, we denote the MLP as M^l , with its operation as $M^l(\mathbf{r})$. Token positions in the sequence dimension are denoted by the subscript t . For clarity, we will omit these subscripts or/and superscripts when there’s no ambiguity.

There are three pivotal nodes in the residual stream within a Transformer layer: The first node corresponds to the initial input of the layer, denoted as $\mathbf{r}_{\text{pre}}^l$. The subsequent two nodes are:

$$\begin{aligned} \mathbf{r}_{\text{mid},t}^l &= \mathbf{r}_{\text{pre},t}^l + \sum_{h=0}^{H-1} \mathbf{a}_t^{l,h}, \quad \mathbf{r}_{\text{post}}^l = \mathbf{r}_{\text{pre},t}^l + \sum_{h=0}^{H-1} \mathbf{a}_t^{l,h} + \mathbf{m}_t^l, \\ \text{where } \mathbf{a}_t^{l,h} &= A^{l,h}(\mathbf{r}_{\text{pre},\leq t}^l) \quad \text{and} \quad \mathbf{m}_t^l = M^l(\mathbf{r}_{\text{mid},t}^l). \end{aligned} \quad (1)$$

Figure 1 illustrates the Transformer layer l from this perspective. Elhage et al. [8] reformulated attention heads for a better theoretical understanding of models, which is equivalent to that presented in [39]. This reformulation can be represented as follows:

$$A(\mathbf{r}) = \left(\text{softmax}(\mathbf{r}^\top \mathbf{W}_Q^\top \mathbf{W}_K \mathbf{r}) \otimes \mathbf{W}_O \mathbf{W}_V \right) \cdot \mathbf{r}. \quad (2)$$

Let’s denote the OV matrix as $\mathbf{W}_{OV} = \mathbf{W}_O \mathbf{W}_V$ and the QK matrix as $\mathbf{W}_{QK} = \mathbf{W}_Q^\top \mathbf{W}_K$, for clarity. The parameters represented by \mathbf{W}_{OV} generate outputs corresponding to inputs; The parameters represented by \mathbf{W}_{QK} generate attention patterns, determining which token’s information is moved from and to.

2.2 Circuit Discovery

Discovering Circuit by Ablation Let us conceptualize the neural network as a causal graph, where each node represents either neurons [6, 7, 20] or module activations [8]. When there is a direct connection between two nodes in the network, they are linked by an edge in the graph. The graph is considered causal because intervening in one node’s output can causally influence

the subsequent nodes that rely on its output as input. A circuit is a subgraph of the neural network [27]. Research indicates that only a few nodes are working given a particular task [41, 24]. The objective of circuit discovery is to uncover the circuit comprising these activated nodes.

The prevailing approach for circuit discovery is grounded in causal mediation analysis [32, 40]. Figure 2 illustrates a toy example to elucidate this process. Suppose we want to discern the impact of node g on node f . When we intervene in g 's output (e.g., by knocking out or corrupting it), f is affected by both the change from g and g 's indirect intervention via z , as shown in Figure 2(b). In a more nuanced approach, we could maintain z 's outputs consistent to measure g 's "direct effect" to f , as shown in Figure 2(c). By conducting such ablation studies, we can first identify nodes that significantly affect the neural network's final outputs. Then, we can pinpoint which nodes influence these influential nodes through iterative backward tracing. This iterative process forms the fundamental concept behind many circuit discovery methods [41, 18, 3].

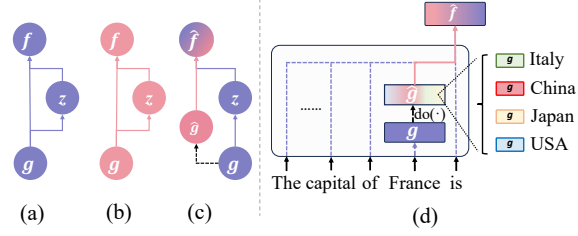


Figure 2: Subfigures (a), (b), and (c) showcase a toy diagram of causal mediation analysis in discovering important circuits in a neural network. Colors distinguish different node values. Subfigure (d) illustrates an activation patching example in studying the node affecting the correct capital city prediction, as detailed in §2.2.

Activation Patching and Impact Metric “Activation patching” is a widely employed technique for implementing the interventions in module activations [41, 22, 40]. Suppose we have an original prompt, denoted as p_{original} , such as “The capital of X is,” with the expected answer being Y . The activations of node f in the neural network given this prompt is represented as $f(p_{\text{original}})$. Suppose we aim to pinpoint the node responsible for extracting country identity, which consequently influences the final node f , i.e., $\mathbf{r}_{\text{post}, t=-1}$ in the final layer. We might intervene on g , a preceding node of f in the causal graph, at the country name position. Let us say we have N intervention prompts sharing the template with p_{original} but differing in keywords. For instance, each intervention prompt $p_{\text{intervention}, n}$ takes the form “The capital of X_n is,” where X_n is sampled from a set of country names distinct from X . By substituting $g(p_{\text{original}})$ with $\frac{1}{N} \sum_{n=1}^N g(p_{\text{intervention}, n})$, we corrupt the country identity in $g(p_{\text{original}})$. This process can be formalized as:

$$\hat{g}(p_{\text{original}}) = g \left(p_{\text{original}} \mid \text{do}(g(p_{\text{original}}) = \frac{1}{N} \sum_{n=1}^N g(p_{\text{intervention}, n})) \right). \quad (3)$$

The operation $\text{do}(\cdot)$, a standard notation in causality [33], indicates the replacement of one value with another. When the node f takes $\hat{g}(p_{\text{original}})$ as inputs, we denote its current value as \hat{f} . If certain metrics indicating the probability of Y decoded from \hat{f} are lower compared to those from f , we can deduce that the edge from g to f has a positive impact on correct predictions, and vice versa. The patching procedure is illustrated in Figure 2(d). This technique used to assess the impact of patching the path from g to f based on decoded probabilities or logit values is referred to as “logit lens” [26]. To apply this technique, we introduce a function named “Unembed,” defined as follows:

$$\text{Unembed}(\cdot) = \mathbf{W}_U \text{LN}(\cdot). \quad (4)$$

$\text{LN}(\cdot)$ represents the normalization in the final layer. $\mathbf{W}_U \in \mathbb{R}^{V \times d_{\text{model}}}$, where V is the vocabulary size, denotes the unembedding matrix (i.e., the weight of the language model head). We obtain $\pi(Y, f)$, the logit value of Y decoded from f , as follows:

$$\pi(Y, f)_{t=-1} = \left(\text{Unembed}(f)[Y] - \frac{1}{V} \sum_v \text{Unembed}(f)[v] \right)_{t=-1}. \quad (5)$$

Here, V represents the vocabulary size, and $[Y]$ follows the Python syntax, selecting the logit of the token Y . We center the logits in order to measure the net effect of patching. The subscript $t = -1$ indicates our focus solely on the logit change at the last position where the current prediction occurs. Now, we can measure the impact of patching by evaluating the degree of change in logits:

$$\Delta\pi(Y, f)_{t=-1} = \left(\frac{\pi(Y, \hat{f})_{t=-1}}{\pi(Y, f)_{t=-1}} - 1 \right) \cdot 100\%. \quad (6)$$

3 How Does the Language Model Do Factual Recall Tasks in Zero-Shot

Recall that Merullo et al. [23] observed that during factual recall, the language model first forms an “argument”. Subsequently, it passes the argument to an implicit function for deriving the final answer. In this section, we explore the following unresolved research questions within a zero-shot scenario:

1. How does the model discern the “argument” from context and pass it to the “implicit function?”
2. What exactly is the “function application”, and how does it relate to MLPs?

3.1 Experiment Setup

We study Transformer-based language models, ranging in scale from the 117M GPT-2 small [34], GPT-2 medium and large, up to the OPT-1.3B [46]. Our study involved two factual recall tasks concerning country-capital and product-developer associations, requiring different domain knowledge. The main text details our experiments conducted with GPT-2 small, which has 12 layers and 12 heads in each layer, using the country-capital task. Other experiments are presented in the appendix. Unless otherwise specified, the interpretations below are generally validated across models and tasks.

For the country-capital task, we prompt the language model, asking it to recall the capital of a given country. For instance, a prompt would be framed as “As we all know, the capital of X is.” Here, X denotes the country in question, and we denote the corresponding capital of X by Y . To ensure the robustness of our experiments and to mitigate any bias stemming from specific prompt templates, we devise a set of 15 distinct prompts. We construct a set containing 15 country-capital pairs. To generate a test dataset, we fill each template with various country names, resulting in a dataset comprising $N = 225$ samples. Data details can be found in appendix C.

3.2 Attention Heads Pass Arguments to the Implicit Function

L9H8 and L10H0 are Argument Passers We replicated experiments conducted by Merullo et al. [23] using GPT-2 small. Given an input p_{original} , we utilize the $\text{Unembed}(\cdot)$ function to unembed each layer’s outputs $\mathbf{r}_{\text{post}, t=-1}^l$. Then, we apply softmax to the early-unembedded logits to assess the probabilities of X and Y , respectively. Figure 3(a) illustrates the probability dynamics across layers. A significant change occurs at layer 9, where X becomes noticeable in the residual stream. This marks the argument formation phase. By the end of layer 10, the probability of Y surpasses that of X for the first time, indicating the function application. Meanwhile, we implement the “path patching” algorithm, an activation patching algorithm introduced by Wang et al. [41] to identify influential nodes. For patching, p_{original} is sampled from the test dataset, while the set of interventions $\{p_{\text{intervention}, n \leq N}\}$ is the entire test dataset. We employ Eq.6 as our impact metric. Path patching results illustrated in Figure 3(b) indicate that paths from outputs of L9H8 and L10H0, i.e., $\mathbf{a}_{t=-1}^{9,8}$ and $\mathbf{a}_{t=-1}^{10,0}$, to $\mathbf{r}_{\text{post}, t=-1}^{11}$ have the most substantial positive impact on correct prediction. The alignment between these experiments suggests that L9H8 and L10H0 play pivotal roles in the two phases.

We identified that L9H8 and L10H0 are “mover heads”, which have garnered attention in many studies [41, 18, 24]. These heads are characterized by their tendency to move the information from any position they focus on to the final position (i.e., $t = -1$). As illustrated in Figure 3(c), both L9H8 and L10H0 predominantly attend to the token X ; In Figure 3(d), we present a scatter plot illustrating the attention score from the final token to X , plotted against the inner product $\langle \mathbf{a}_{t=-1}^{l,h}, \mathbf{W}_U[X] \rangle$. The significant correlation between the attention score and the inner product (with Pearson correlation coefficient 0.77 for L9H8 and 0.90 for L10H0) strongly suggests that these two heads are responsible for passing the country name, the “argument” in the country-capital task, to the final position.

These pieces of evidence imply that the behavior exhibited by L9H8 and L10H0 can be likened to passing arguments to an implicit function, for example, “get_capital(x).” Moreover, beyond the mere act of argument passing, we found that L9H8 demonstrates some capacity to directly produce the correct answer, as discussed below.

L9H8 Can Map Country Names to Capital Cities To examine L9H8 closely, we employ $\text{Unembed}(\cdot)$ to early-decode ($\mathbf{W}_{OV}^{\text{L9H8}} \cdot \mathbf{r}_{\text{pre}, t=X}^9$) to obtain top-5 tokens. We noticed that correct capital cities emerged among these top tokens. This indicates that partial function application happens

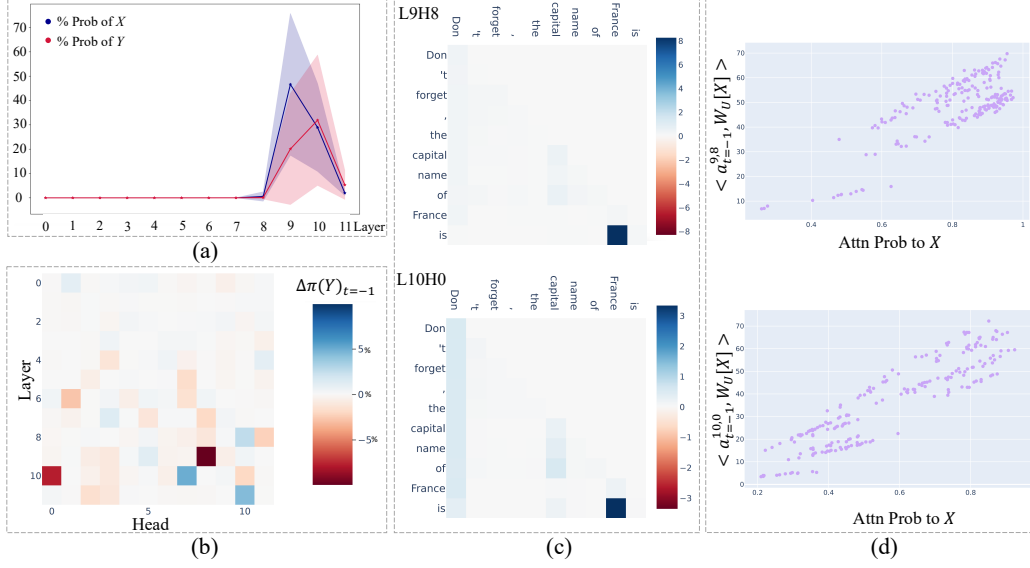


Figure 3: (a) The probability of X and Y when early unembed each layer’s outputs. The shaded regions indicate variances. (b) The effect of patching $\mathbf{a}_{t=-1}^{l,h} \rightarrow \mathbf{r}_{\text{post}}^{11}$. (c) Value weighted attention pattern of L9H8 and L10H0. (d) L9H8 and L10H0’s attention to X is proportional to the projection value of their output along $\mathbf{W}_U[X]$, indicating they are moving country names to the final position.

within this attention head. Mapping countries to capitals is an inherent ability of L9H8: Despite receiving the embedding vector of a country rather than a representation from deeper layers at position X , its OV matrix still yields some capital cities. Our investigation reveals that L9H8 is a task-specific head mainly activated given country-related prompts, as evidenced by its inactivation in other tasks (see appendix F). In contrast, although L10H0 lacks an inherent ability to map countries to capital cities within its OV matrix, it is activated across a variety of tasks, even those solely reliant on in-context learning [41, 24]. It functions more like an amplifier for L9H8 or, more generally, an amplifier for the confident answer in the residual stream.

3.3 MLP Acts as the “Activation” of Attention Heads

In §3.2, we have demonstrated that only a few attention heads have a positively direct effect to the correct prediction. This naturally leads to a question: Given that Transformer models aggregate the outputs of all attention heads with equal weight (as shown in Eq. 1), how does the model discern and extract information written by specific heads within the residual stream? We first compare the norm of each attention head’s outputs and employ principal component analysis (PCA) to detect anomalous heads’ outputs. Unfortunately, these experiments failed to yield any informative results. Then, we hypothesize the existence of an “activation” mechanism after these attention heads that either amplifies or suppresses the outputs of certain attention heads. Now, let us turn to the subsequent module following the attention heads—the MLP layer.

Recall that a Transformer layer’s output $\mathbf{r}_{\text{post}}^l$ is the linear combination of $\mathbf{r}_{\text{pre}}^l$, the collective output of attention heads $\sum_{h=0}^{H-1} \mathbf{a}^{l,h}$, and the MLP output \mathbf{m}^l (See Eq.1). This prompts us to consider whether MLP outputs can be decomposed into a linear combination of $\mathbf{r}_{\text{pre}}^l$ and $\sum_{h=0}^{H-1} \mathbf{a}^{l,h}$, where each term’s coefficient indicates the degree of amplification or erasure imposed by the MLP? Intrigued by this notion, we tackle the following multivariable linear regression problem:

$$\tilde{\mathbf{m}}^l = \mathbf{b}^l + \sum_{h=0}^{H-1} w^{l,h} \mathbf{a}^{l,h} + w^{l,r} \mathbf{r}_{\text{pre}}^l. \quad (7)$$

Here, $w^{l,\cdot}$ are scalars, and the intercept $\mathbf{b}^l \in \mathbb{R}^{d_{\text{model}}}$ is expected to encapsulate higher-order transformations of head outputs, along with abstract information such as knowledge introduced by the MLP. We employ the gradient descent method to solve this linear regression problem, with the objective of minimizing the Mean Squared Error (MSE) loss between $\tilde{\mathbf{m}}^l$ and \mathbf{m}^l . We set the learning rate to 0.005 and performed 60,000 optimization steps using the SGD optimizer with a momentum of 0.99. Note

that Eq.7 is position-specific, and our primary focus is on the final position in this section. Unless otherwise specified, the position subscripts default to $t = -1$ throughout the subsequent paper.

We employed 4-fold cross-validation to validate the regression solution. We found that, for each layer, the optimization consistently converges to similar solutions with minimal loss, irrespective of the initial values assigned to $w^{l,h}$, $w^{l,r}$, and \mathbf{b}^l . Please refer to appendix H for detailed training information. Additional experiments were conducted to validate the faithfulness of the regression solution further. First, we replaced the MLP outputs \mathbf{m}^l in each layer with the $\tilde{\mathbf{m}}^l$, the regression value following Eq 7. We computed the Kullback-Leibler (KL) divergence [17] between the output distributions of the modified model and those of the original model. The resulting average KL divergence is 0.21, indicating a close match between the distributions. Moreover, we analyzed the average final logit of Y produced by both the original and modified models. The logit values are close, measuring 13.08 and 13.15, respectively.

Our primary focus lies on the regression coefficients for layer 9 because MLP9 directly follows the L9H8, the most important argument passer in the country-capital task. The coefficients learned from all test data for layer 9 are reported in Table 1. These coefficients reveal an interesting pattern. Among all the terms, only three exhibit positive values, with $w^{9,8}$ being the highest. This suggests that MLP9 enhances the information sourced from L9H8. Consequently, the arguments passed by L9H8, along with some capital names it generates, “stand out” in the residual stream.

Table 1: Learned coefficients for layer 9.

$w^{9,0}$	$w^{9,1}$	$w^{9,2}$	$w^{9,3}$	$w^{9,4}$	$w^{9,5}$	$w^{9,6}$	$w^{9,7}$	$w^{9,8}$	$w^{9,9}$	$w^{9,10}$	$w^{9,11}$	$w^{9,r}$
-0.05	-0.24	0.04	-0.01	-0.26	-0.13	-0.17	-0.04	0.08	-0.09	-0.08	-0.64	0.05

To summarize our progress thus far, we have demonstrated that task-specific heads pass the arguments while other heads also individually write to the residual stream. A subsequent MLP serves as an “activation” for the attention heads’ outputs, effectively highlighting the expected arguments in the residual stream. These findings collectively address the first research question. In the following subsection, we move forward and analyze the layer 10, illustrating how MLP10 applies the “implicit function” via the intercept \mathbf{b}^l .

3.4 Function Application is “ $\mathbf{b} + \mathbf{r}_{\text{mid}}$ ”

While partial function application has been identified within the OV circuit of L9H8, the probability of capital cities Y in the decoded residual stream by the end of layer 9 remains lower compared to that of countries, as illustrated in Figure 3(a). It is not until layer 10 that the probability of capital cities Y surpasses that of countries X for the first time, indicating the primary occurrence of function application. In this section, we delve into what happens within layer 10.

We apply the $\text{Unembed}(\cdot)$ function to the $\mathbf{r}_{\text{pre}}^{10}$, $\mathbf{r}_{\text{mid}}^{10}$, and $\mathbf{r}_{\text{post}}^{10}$, sequentially, yielding the probabilities of X and Y at the final positions of each node. Results are shown in Figure 4(a). Notably, up to $\mathbf{r}_{\text{mid}}^{10}$, the probability of capitals remains lower than that of countries, with this trend reversing at $\mathbf{r}_{\text{post}}^{10}$. Therefore, MLP10 must write information related to the correct answers into the residual stream. However, upon decoding \mathbf{m}^{10} , the outputs of MLP10, we observe that the probabilities of countries and cities are both 0, indicating that \mathbf{m}^{10} is almost orthogonal to both X and Y ’s unembedding vectors. How does this contribute to enhancing the correct prediction?

We analyzed the cosine similarities among several pairs of vectors, and the results are reported in Figure 4(b). In Figure 4(c), we offer a 3D toy example to grasp the relationships between these vectors. Based on these two figures, we hypothesize how MLP10 accomplishes “function application.” We can see that $\mathbf{r}_{\text{mid}}^{10}$ shows a higher degree of similarity with $\mathbf{W}_U[X]$, a column vector of dimension d_{model} , indicating a stronger tendency towards predicting X . Despite \mathbf{m}^{10} being perpendicular to both $\mathbf{W}_U[X]$ and $\mathbf{W}_U[Y]$, its addition to $\mathbf{r}_{\text{mid}}^{10}$ could divert the residual stream away from $\mathbf{W}_U[X]$ and towards $\mathbf{W}_U[Y]$, resulting in a higher probability of Y in $\mathbf{r}_{\text{post}}^{10}$.

To validate this hypothesis, we project \mathbf{m}^{10} onto the hyperplane spanned by $\mathbf{W}_U[Y]$ and $\mathbf{W}_U[X]$. Specifically, we stack $\mathbf{W}_U[Y]$ and $\mathbf{W}_U[X]$ as columns, constructing the corresponding projection

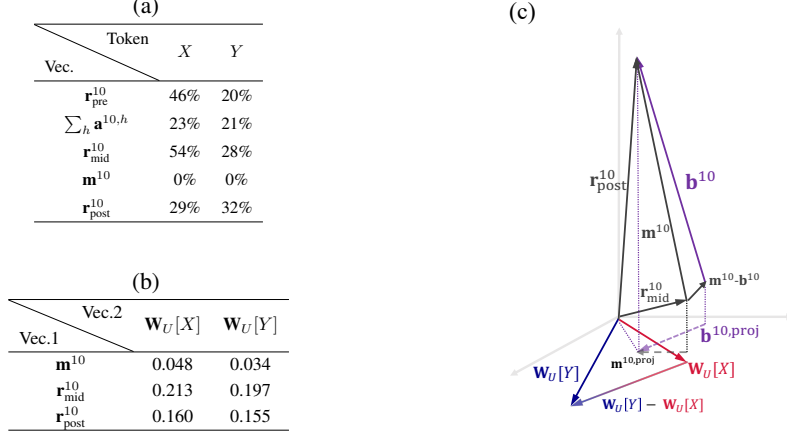


Figure 4: (a) The probabilities of X and Y decoded from various vectors. (b) Cosine similarity between the vectors. Note that although the cosine similarity between \mathbf{r}_{post}^{10} and $\mathbf{W}_U[X]$ remains higher than that with $\mathbf{W}_U[Y]$, when taking into account the norm, the logit value of Y is higher. (c) A simplified example demonstrating the relationships among vectors in 3D space.

matrix \mathbf{P} . The projection of \mathbf{m}^{10} onto the hyperplane is achieved by its multiplication with \mathbf{P} :

$$\begin{aligned}
 \mathbf{M} &= [\mathbf{W}_U[X], \mathbf{W}_U[Y]], \\
 \mathbf{P} &= \mathbf{M}(\mathbf{M}^\top \mathbf{M})^{-1} \mathbf{M}^\top, \\
 \mathbf{m}^{10,proj} &= \mathbf{P} \mathbf{m}^{10}.
 \end{aligned} \tag{8}$$

If $\mathbf{m}^{10,proj}$ exhibits a significant cosine similarity with the vector $(\mathbf{W}_U[Y] - \mathbf{W}_U[X])$, then our hypothesis is verified. We observed that this cosine similarity is 0.361, which, while not exceptionally high, still provides insight. Considering that \mathbf{m}^l can be decomposed into the activations of attention heads and an intercept \mathbf{b}^l designed to capture newly added influences imposed by the MLP, we further project the intercept \mathbf{b}^{10} to the hyperplane. It is noteworthy that $\mathbf{b}^{10,proj}$ exhibits a cosine similarity of 0.946 with $(\mathbf{W}_U[Y] - \mathbf{W}_U[X])$. We also analyze the GPT-2 large model and present its probability dynamics in Figure 9. It becomes apparent that beginning from layer 27, there is a significant increase in the probability gap between X and Y , and we measured a cosine similarity of 0.821 between $\mathbf{b}_{large}^{27,proj}$ and $(\mathbf{W}_U[Y] - \mathbf{W}_U[X])$.

These results highlight several key points: (1) MLPs apply “implicit functions” by adding a task-aware component into the residual stream, guiding the residual stream towards its expected direction. (2) The validity of the linear regression method we propose for analyzing the MLPs has once again been affirmed empirically.

We should clarify that we do not assert that the intercept invariably directs the residual stream from “ X ” towards “ Y ” during the function application phase. Understanding the underlying mechanism of specific MLP layers calls for individual scrutiny in each case, considering the potentially intricate behaviors that influence the probability of specific tokens. We will discuss this in appendix G.

Additionally, given the false negative results when directly applying the logit lens technique to \mathbf{m}^{10} , we make two recommendations for future works. Firstly, it is important to be cautious when applying the logit lens to MLPs or attention heads’ outputs. Secondly, when examining the outputs of modules within middle layers, it is essential to consider cosine similarity because subtle nuances may undergo significant amplification in probability after softmax.

To sum up, we have addressed the research questions listed earlier.

4 The Universal Anti-Overconfidence Mechanisms at the Final Layer

Readers may notice another phase termed “saturation” in the *one-shot* scenario explored by Merullo et al. [23], following “function application,” wherein Y maintains its top rank in the final layers. In contrast, as depicted in Figure 3(a) in our zero-shot scenario, the probabilities of both X and Y

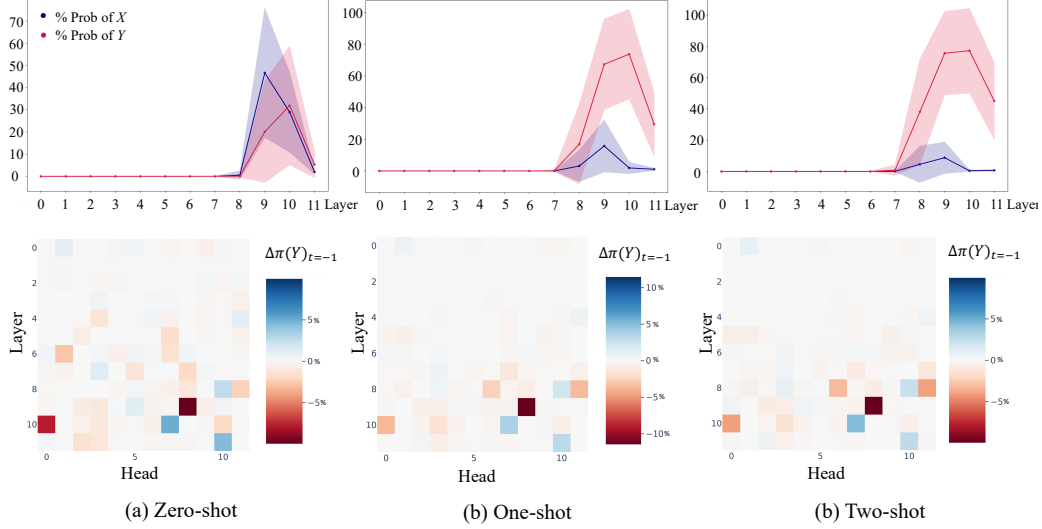


Figure 5: The sub-figures illustrate the probability dynamics of X and Y , alongside influential heads impacting final logits across zero, one, and two-shot settings. The fundamental mechanisms detected in the zero-shot scenario still work in few-shot settings.

drastically drop in the last two layers. We then extend our investigation from the zero-shot to few-shot settings to delve deeper.

4.1 Comparing Zero-shot with Few-shot Scenarios

Zero-shot Mechanisms Still Work in Few-shot Scenarios We construct two new datasets: one with a single in-context demonstration (one-shot) and another with two (two-shot). We achieve this by appending one or two prefixes in the form of “The capital of X_{ICL} is Y_{ICL} ,” respectively, to the original prompts. each $(X_{\text{ICL}}, Y_{\text{ICL}})$ is randomly selected and distinct from (X, Y) .

Using these datasets, we conduct “patching $\mathbf{a}_{t=-1}^{l,h} \rightarrow \mathbf{r}_{\text{post}, t=-1}^{11}$ ” experiments as described in §3.2. We also plot the probability dynamics of X and Y across the layers. The results are shown in Figure 5. Upon comparing the results of 0-shot and few-shot experiments, we observe that the mechanisms studied in the zero-shot scenario persist in the few-shot scenarios. This persistence manifests in two aspects: Firstly, the consistency of probability dynamics and influential heads across varying numbers of ICL demonstrations; Secondly, the consistent decline in the probability of both X and Y as they reach the final layers across all scenarios. Now, it becomes evident that the identification of the “saturation” phase by Merullo et al. [23] is not due to in-context demonstrations but rather stems from the metric they utilize, the token rank rather than detailed probability. Consequently, the final declines are overlooked. The following subsections will delve into the reasons behind this universal decline at the final layer.

ICL Improves Model Confidence in Generating Capitals’ Name Before delving into the universal decline, there is an extra crucial finding when comparing the zero-shot results to the few-shot results. One noticeable distinction lies in the emergence of probabilities for X and Y at layer 8 in few-shot settings, compared to their emergence until layer 9 in 0-shot settings. This distinction can be attributed to L8H11 and L8H6, where L8H11 is identified as a “mover” head. As the number of ICL demonstrations increases, these heads become more influential.

The ICL demonstrations enhance the model’s confidence in generating the *name* of capitals from two key aspects: Firstly, an additional mover head, L8H11, moves the country names to the final position, implying that with more ICL demonstrations, the model gains insights into the “argument” earlier, allowing for deeper processing of the answer across more layers. Secondly, in the zero-shot scenario, the model tends to produce “safe” tokens such as “not” to follow the prompt, resulting in sentences like “The capital of X is not...” which, although coherent, lack useful information. This might stem from a lack of confidence in the output domain or format. However, in few-shot settings, L8H6 pays considerable attention from the final token to Y_{ICL} , i.e., the in-context capital names. It seems prompt the model to recognize that the expected answer is the *name* of a capital city. We leave

Table 2: Learned coefficients for layer 11.

$w^{11,0}$	$w^{11,1}$	$w^{11,2}$	$w^{11,3}$	$w^{11,4}$	$w^{11,5}$	$w^{11,6}$	$w^{11,7}$	$w^{11,8}$	$w^{11,9}$	$w^{11,10}$	$w^{11,11}$	$w^{11,r}$
-0.22	0.74	-0.11	-0.31	-0.05	1.55	0.54	0.42	0.13	0.98	0.13	0.22	0.01

the exploration of the following questions to future research, as the answer is non-trivial: How do ICL demonstrations trigger the earlier activation of mover heads? How does the model determine that it should output city names?

4.2 Negative heads Suppress X

As shown in Figure 5, we identify two negative heads, L10H7 and L11H10, which suppress the correct prediction, irrespective of the number of shots. Take the zero-shot scenario as an example. We investigate the impact of these heads. We patch their outputs with their average activations given the set $\{p_{\text{intervention}, n \leq N}\}$ at position X , thereby eliminating the country identity in their outputs and thus blocking them out. We can gain insight into their suppression effect by examining the probabilities of X and Y after blocking out these heads. The zero-shot probability dynamics when patching these heads are illustrated in Figure 6. Even though blocking these negative heads mitigates the suppression of X in layer 10, it remains strong in layer 11. The probability of Y experiences a slight increase but remains relatively stable and low overall. These findings suggest that both negative heads primarily suppress the probability of X . This observation is intriguing, especially considering that, despite L11H10 and L10H7 being the only two negative heads in the last layers, they do not appear to suppress Y directly. What mechanism, then, is at play in decreasing the probability of Y at the final layer?

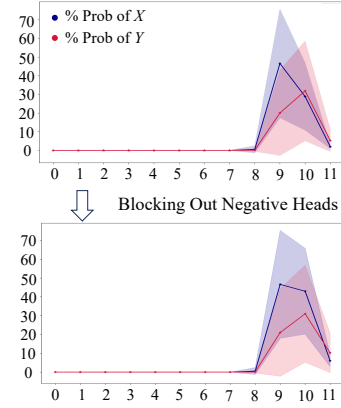


Figure 6: When blocking out the negative heads in zero-shot settings, only suppression of X is slightly alleviated.

4.3 The Final Layer Avoids Overconfidence

Within layer 11, as depicted in Figure 14(a.1), (b.1), and (c.1), we observe a consecutive decline in the probability of Y at $\mathbf{r}_{\text{mid}}^{11}$ and $\mathbf{r}_{\text{post}}^{11}$, regardless of the number of demonstrations. This suggests that certain attention heads in this layer, along with MLP11, contribute to the suppression of Y . This phenomenon is commonly observed across various models and tasks. We suggest that the behavior observed could be interpreted as an anti-overconfidence strategy employed within the final layer. This strategy could mitigate the risk of large losses stemming from an incorrect yet confident prediction.

Attention Heads Incorporate Frequent Tokens into the Residual Stream Despite our efforts to block out each attention head, the suppression effect persists, suggesting that a large portion of all heads may contribute together to the suppression of Y . Visualizing the attention patterns of each head (see Figure 13) provides more insights. It becomes evident from the figure that most heads “park” their attention on the initial token. This pattern is consistent across various tasks and models, indicating its generality. Previous works [25, 43] suggest that this attention pattern indicates that attention heads are inactive and indifferent to the context. We analyzed the output of the OV circuit in these heads at the initial position and observed that some of them output frequent English tokens such as “the,” “,” “.”, “a,” and “and,” while others produce letters and some subwords irrelevant to the task. We hypothesize that these inactive heads move information in the initial position to the final position, diluting the content of Y in the residual stream. To test our hypothesis, we applied attention masks to restrict these heads to attend solely to the final position (i.e., only the diagonal is unmasked). The result, as depicted in Figure 14(a.2), (b.2), and (c.2), shows a mitigation of suppression at $\mathbf{r}_{\text{mid}}^{11}$. For example, with zero-shot, We improved the probability of Y at $\mathbf{r}_{\text{mid}}^{11}$ from 13.41% to 31.68%.

The Final MLP Prefers “Safe” Prediction We examined the regression coefficients of Eq. 7 for the layer 11, as detailed in Table 2. It is observed that most attention heads are assigned with positive weights, some even surpassing 1. Meanwhile, $\mathbf{r}_{\text{pre}}^{11}$, which contains much information of Y , has a relatively modest weight. These findings suggest that MLP11 further amplifies the suppression of Y imposed by attention heads.

Further, we delve into examining the learned intercept of the final layer to gain deeper insights into the workings of the final MLP. We hypothesize that the final MLP aligns the residual stream with $\mathbb{E}[\mathbf{W}_U]$, the expectation of output logits over the training corpus, thereby effectively mitigating significant training losses. $\mathbb{E}[\mathbf{W}_U]$ is computed as the average of $w_v \cdot \mathbf{W}_U[v]$ over V tokens. Here, w_v represents the relative word frequency derived from OpenWebText [11], the training corpus of GPT-2. We utilize the word frequency data provided by [15].

In the case of GPT-2 medium and large models, which consist of 24 and 36 layers, respectively, when we project the individual intercepts $\mathbf{b}_{\text{medium}}^{23}$ and $\mathbf{b}_{\text{large}}^{35}$ onto the hyperplane spanned by $\mathbb{E}[\mathbf{W}_U]$ and $\mathbf{W}_U[Y]$, we observe that the projected intercepts point from $\mathbf{W}_U[Y]$ towards $\mathbb{E}[\mathbf{W}_U]$. This directional alignment is supported by their respective cosine similarities, measuring 0.976 and 0.661 with respect to the vector $(\mathbb{E}[\mathbf{W}_U] - \mathbf{W}_U[Y])$. In GPT-2 Small, a notable correlation is evident between \mathbf{b}^{11} and $\mathbb{E}[\mathbf{W}_U]$, indicated by a cosine similarity of 0.933, while the similarity between $\mathbf{b}^{11, \text{proj}}$ and $(\mathbb{E}[\mathbf{W}_U] - \mathbf{W}_U[Y])$ remains relatively small.

We acknowledge that the hypothesis regarding the final MLP layer remains anecdotal since we have only tested it within the GPT-2 family due to a lack of training corpus data for other models. Nevertheless, subtracting the intercept from the residual stream in the final layer empirically proves to be an effective approach in mitigating the suppression of correct predictions. Results on GPT-2 small are depicted in Figure 14(a.3), (b.3), and (c.3). Through this approach, for example, we were able to enhance the zero-shot probability of Y at $\mathbf{r}_{\text{post}}^{11}$ in GPT-2 small from 15.51% to 25.93%. The efficacy of two anti-suppression techniques—namely, applying attention masks and subtracting intercepts—has been validated across various models and tasks. Please refer to the appendix F and G for further details.

5 Conclusion

In this paper, we conduct a thorough investigation into the factual recall mechanisms utilized by language models. We studied the mechanisms behind “argument passing” and “function application”. Furthermore, we uncovered the prevalence of anti-overconfidence mechanisms in language models. Our proposed analysis method, based on linear regression, effectively decomposes MLP outputs into components that are easily understandable to humans. This method has been substantiated through numerous empirical experiments and lays a valuable foundation for our interpretations. Finally, we study the anti-overconfidence mechanisms in the final layer, which is a universal mechanism regardless of models, tasks, and the number of in-context demonstrations.

Acknowledgments and Disclosure of Funding

We appreciate the valuable discussions with Zhuocheng Gong (Peking University) and Wei Yao (Renmin University of China), assistance in figure presentation from Tao Tan (Renmin University of China), and writing suggestions from Shuqi Li (Renmin University of China). This study is sponsored by CCF-BaiChuan-Eblech Foundation Model Fund.

References

- [1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
- [2] Yuhan Chen, Ang Lv, Ting-En Lin, Changyu Chen, Yuchuan Wu, Fei Huang, Yongbin Li, and Rui Yan. Fortify the shortest stave in attention: Enhancing context awareness of large language models for effective tool use, 2024.
- [3] Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

- [4] James Dao, Yeu-Tong Lau, Can Rager, and Jett Janiak. An adversarial example for direct logit attribution: Memory management in gelu-4l, 2023.
- [5] Subhabrata Dutta, Joykirat Singh, Soumen Chakrabarti, and Tanmoy Chakraborty. How to think step-by-step: A mechanistic understanding of chain-of-thought reasoning, 2024.
- [6] Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. Softmax linear units. *Transformer Circuits Thread*, 2022. <https://transformer-circuits.pub/2022/solu/index.html>.
- [7] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022.
- [8] 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. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. <https://transformer-circuits.pub/2021/framework/index.html>.
- [9] Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore, December 2023. Association for Computational Linguistics.
- [10] Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space, 2022.
- [11] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. <http://Skylion007.github.io/OpenWebTextCorpus>, 2019.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [13] Benjamin Heinzerling and Kentaro Inui. Monotonic representation of numeric properties in language models, 2024.
- [14] Evan Hernandez, Belinda Z. Li, and Jacob Andreas. Inspecting and editing knowledge representations in language models, 2023.
- [15] Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Transformer language models handle word frequency in prediction head. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4523–4535, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [16] Jorrit Kruthoff. Carrying over algorithm in transformers, 2024.
- [17] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
- [18] Tom Lieberum, Matthew Rahtz, János Kramár, Neel Nanda, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla, 2023.
- [19] Ang Lv, Kaiyi Zhang, Shufang Xie, Quan Tu, Yuhang Chen, Ji-Rong Wen, and Rui Yan. Are we falling in a middle-intelligence trap? an analysis and mitigation of the reversal curse, 2023.
- [20] Aleksandar Makelov, Georg Lange, Atticus Geiger, and Neel Nanda. Is this the subspace you are looking for? an interpretability illusion for subspace activation patching. In *The Twelfth International Conference on Learning Representations*, 2024.
- [21] Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations, 2023.

- [22] Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- [23] Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. A mechanism for solving relational tasks in transformer language models, 2023.
- [24] Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models. In *The Twelfth International Conference on Learning Representations*, 2024.
- [25] Evan Miller. Attention is off by one, 2023.
- [26] nostalgebraist. Interpreting gpt: the logit lens., 2020.
- [27] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. <https://distill.pub/2020/circuits/zoom-in>.
- [28] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. <https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html>.
- [29] OpenAI. OpenAI: Introducing ChatGPT, 2022.
- [30] OpenAI. Gpt-4 technical report, 2023.
- [31] Francesco Ortu, Zhijing Jin, Diego Doimo, Mrinmaya Sachan, Alberto Cazzaniga, and Bernhard Schölkopf. Competition of mechanisms: Tracing how language models handle facts and counterfactuals, 2024.
- [32] Judea Pearl. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI’01, page 411–420, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
- [33] Judea Pearl. *Causality*. Cambridge University Press, 2 edition, 2009.
- [34] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *arxiv*, 2019.
- [35] Gautam Reddy. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *The Twelfth International Conference on Learning Representations*, 2024.
- [36] Toby Shevlane, Sebastian Farquhar, Ben Garfinkel, Mary Phuong, Jess Whittlestone, Jade Leung, Daniel Kokotajlo, Nahema Marchal, Markus Anderljung, Noam Kolt, Lewis Ho, Divya Siddarth, Shahar Avin, Will Hawkins, Been Kim, Iason Gabriel, Vijay Bolina, Jack Clark, Yoshua Bengio, Paul Christiano, and Allan Dafoe. Model evaluation for extreme risks, 2023.
- [37] 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. Llama: Open and efficient foundation language models, 2023.
- [38] 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. Llama 2: Open foundation and fine-tuned chat models, 2023.

- [39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [40] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc., 2020.
- [41] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023.
- [42] Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Do llamas work in english? on the latent language of multilingual transformers, 2024.
- [43] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2023.
- [44] Qinan Yu, Jack Merullo, and Ellie Pavlick. Characterizing mechanisms for factual recall in language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- [45] Zeping Yu and Sophia Ananiadou. Locating factual knowledge in large language models: Exploring the residual stream and analyzing subvalues in vocabulary space, 2024.
- [46] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.
- [47] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren’s song in the ai ocean: A survey on hallucination in large language models, 2023.
- [48] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2023.

A Related Works

In investigating the factual recall of language models, Geva et al. [9, 10] traced the information change in factual recall tasks. Yu et al. [44] and Ortu et al. [31] studied important attention heads in a counterfactual setting. [13] is a contemporary work that finds the specific direction in activation space encodes the semantics of model final outputs. None of these works provide an in-depth analysis of the mechanisms behind each module’s function. Merullo et al. [23] described the factual recall as a process of “passing an argument and then applying the function.” Based on their analogy, we take a further step to provide the details of how “arguments” are identified from the context and passed to the “function.” Similar to our paper, Geva et al. [9] also found that the OV matrix of attention heads contains knowledge. Other related works focus on factual recall and mechanistic interpretability. Our findings do not overlap with theirs but complement each other. Lieberum et al. [18] conducted a multiple-choice task involving factual knowledge within large language models. Yu et al. [45] pinpointed the location of knowledge within neurons in MLPs and interpreted how the residual stream activates knowledge in MLPs.

Elhage et al. [8] proposed that Transformers operate using a “memory management” approach, where modules are reading from and writing to the residual stream. Dao et al. [4] trained a 4-layer model and found that certain attention heads could erase the outputs of previous attention heads. McGrath et al. [21] observed the erasure of tokens by MLPs. More than observations, we dissected the role of MLPs by *quantifying* their impact on individual heads, and interpreting their role in “function application.”

B Limitations and Future Works

This section aims to outline several research questions that remain unexplored. We present hypotheses for each and leave them for future research endeavours.

Firstly, in Section 3.2, we directly start with that task-specific heads are activated by task semantics. However, we omit to explore how the model constructs this task semantics within its shallow layers. We attempted to trace the information flow from the query and key vectors of the mover heads, only to find that the heads influencing these vectors are scattered, and the individual impact of a head is minimal. We posit that the formation of the task semantic occurs through collaborative efforts across multiple circuit paths. Thus, attempting to patch a single pathway might be futile, as another pathway could compensate. Collaborative circuit paths have long posed a challenge for circuit discovery methods [3]. It might be related to the Hydra effect in neural networks [21]. Additionally, the mechanism of in-context learning boosting model confidence also needs future investigation.

Secondly, in this paper, we did not study the origins of these mechanisms. Take, for instance, the behavior of an MLP that redirects the residual stream to its expected answer. This phenomenon seems to correlate with the residual connections within the Transformer architecture. When a module handles an input, denoted as x , to attain a target, y , the most effective strategy entails minimizing loss by structuring the output as $y - x$ instead of aligning the output with the direction of y . This manner is more effective despite the potential for outputs to seem orthogonal or display a negative cosine similarity with y .

We recognize that our proposed analysis method, aimed at decomposing MLP outputs, requires preliminary human reasoning. Exploring more automated interpretation techniques for MLPs would be a promising research topic.

C Details of Datasets

The Country-Capital Task Table 3 shows prompt templates we used. Here are country-capital (X, Y) pairs we used: (China, Beijing), (USA, Washington. D.C.), (Russia, Moscow), (England, London), (France, Paris), (Japan, Tokyo), (Italy, Rome), (Canada, Ottawa), (Australia, Canberra), (Spain, Madrid), (Egypt, Cairo), (Portugal, Lisbon), (Austria, Vienna), (Greece, Athens), (Thailand, Bangkok).

The Product-Developer Task In addition to the country-capital task, we also conduct experiments on a product-developer task. Table 4 shows prompt templates we used. Here are Product-Developer

Table 3: Prompt templates used in the country-capital task.

Index	Prompt Template
0	It's crucial to know that the capital of X is
1	You are right to say that the capital of X is
2	Therefore, it's correct to state that the capital of X is
3	When asked, always remember that the capital of X is
4	We confirm that the capital of X is
5	Don't forget, the capital of X is
6	Bear in mind, the capital of X is
7	Keep in mind, the capital of X is
8	Just a reminder, the capital of X is
9	As we all know, the capital of X is
10	According to the textbook, the capital of X is
11	I am sure that the capital of X is
12	Without a doubt, the capital of X is
13	In case you didn't know, the capital of X is
14	To emphasize, the capital of X is

Table 4: Prompt templates used in the product-developer task.

Index	Prompt Template
0	It's crucial to know that X is developed by
1	You are right to say that X is developed by
2	Therefore, it's correct to state that X is developed by
3	When asked, always remember that X is developed by
4	We confirm that X is developed by
5	Don't forget, X is developed by
6	Bear in mind, X is developed by
7	Keep in mind, X is developed by
8	Just a reminder, X is developed by
9	As we all know, X is developed by
10	According to the textbook, X is developed by
11	I am sure that X is developed by
12	Without a doubt, X is developed by
13	In case you didn't know, X is developed by
14	To emphasize, X is developed by

(X , Y) pairs we used: (iPhone, Apple), (Windows7, Microsoft), (GTX1060, Nvidia), (YouTube, Google), (Firefox, Mozilla), (VirtualBox, Oracle), (Instagram, Facebook), (Pentium, Intel), (Steam, Valve), (Radeon, AMD), (Photoshop, Adobe), (PlayStation, Sony), (Kindle, Amazon), (GameBoy, Nintendo).

D The Influence of Prompt Templates

In experiments in the main text, the task word “capital” always precedes the country name X , as shown in Table 3. We now investigate whether altering the order of these keywords affects the factual recall. To explore this, we construct a new set of prompts, detailed in Table 5. We have the following key findings:

(1) **Consistent Influential Heads:** Comparing Figure 7(a) and Figure 7(b), we can see that despite template differences, crucial task-specific heads, such as L9H8 and L10H0, maintain significance. L10H7 and L11H10 continue to suppress the correct prediction. Notably, some heads become more influential when using new templates such as L11H2 and L7H8. L11H2 is a mover head, and L7H8 attends to “capital,” appearing to impact the formation of task semantics, which falls out of the scope of this paper.

(2) **Consistent Probability Dynamics’ Patterns:** The probability dynamics pattern remains akin to the original when using the new templates. Specifically, at layer 9, X and Y become evident in the residual stream, while at layer 10, the probability of Y surpasses that of X for the first time. The cosine similarity between current $\mathbf{b}^{10, \text{proj}}$ and $(\mathbf{W}_U[Y] - \mathbf{W}_U[X])$ is 0.944. Recall that this similarity stands at 0.946 when employing the original prompt templates. The consistency across various templates underscores our previous findings regarding MLP10’s function in utilizing the intercept to steer the direction of the residual stream.

Moreover, at layer 11, the model exhibits the anti-overconfidence mechanism, with the cosine similarity between current \mathbf{b}^{11} and $\mathbb{E}[\mathbf{W}_U]$ measuring at 0.89. This value is also notably close to 0.933 observed when employing the original templates. As shown in Figure 8(a), the anti-overconfidence mechanism can be suppressed by the two techniques we proposed.

Table 5: Prompt templates in the country-capital task where X precedes the task word “capital.”

Index	Prompt Template
0	It's crucial to know that X 's capital is
1	You are right to say that X 's capital is
2	Therefore, it's correct to state that X 's capital is
3	When asked, always remember that X 's capital is
4	We confirm that X 's capital is
5	Don't forget, X 's capital is
6	Bear in mind, X 's capital is
7	Keep in mind, X 's capital is
8	Just a reminder, X 's capital is
9	As we all know, X 's capital is
10	According to the textbook, X 's capital is
11	I am sure that X 's capital is
12	Without a doubt, X 's capital is
13	In case you didn't know, X 's capital is
14	To emphasize, X 's capital is

(3) **Prompt Templates Impact Zero-Shot Prediction Confidence:** The primary distinction observed when utilizing two sets of prompts resides in the probability value. For prompts where X precedes “capital,” the probabilities of both X and Y are lower. As depicted in Figure 7, this divergence originates primarily from layer 9. At layer 9, with original prompts, the probability of X could reach 50%, whereas with the new prompts, it decreases to a mere 5%.

Upon scrutinizing the top decoded tokens given the new prompts, we found that “located” emerges as the predominant choice for output at layer 9 across almost all test samples. However, adding prefixes “ X_{ICL} ’s capital is Y_{ICL} ” before our new prompts, to create few-shot test samples, eliminates the disparity of probability dynamics between given the new prompts and the original ones. The results are shown in Figure 8(b) and (c).

It seems that our new prompts, where we place “ X ” before “capital,” exacerbate the model’s uncertainty regarding the output domain, resulting in the model favoring the word “located” as a safe output. This uncertainty is effectively eliminated by in-context learning, aligning with the findings in Section 4. Understanding how in-context learning boosts the model’s confidence remains a complex question, which we defer to future research endeavors (appendix B).

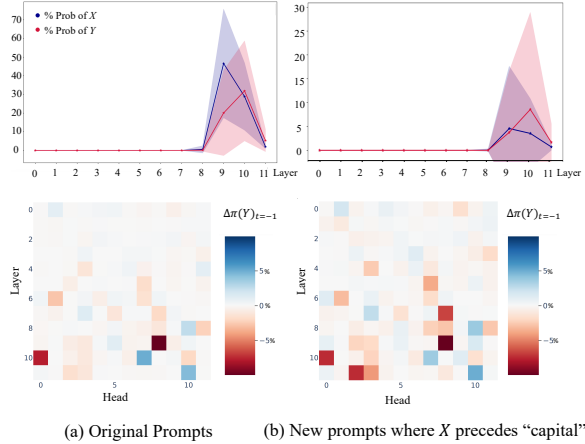


Figure 7: Probability dynamics and detected influential heads across varied prompt templates.

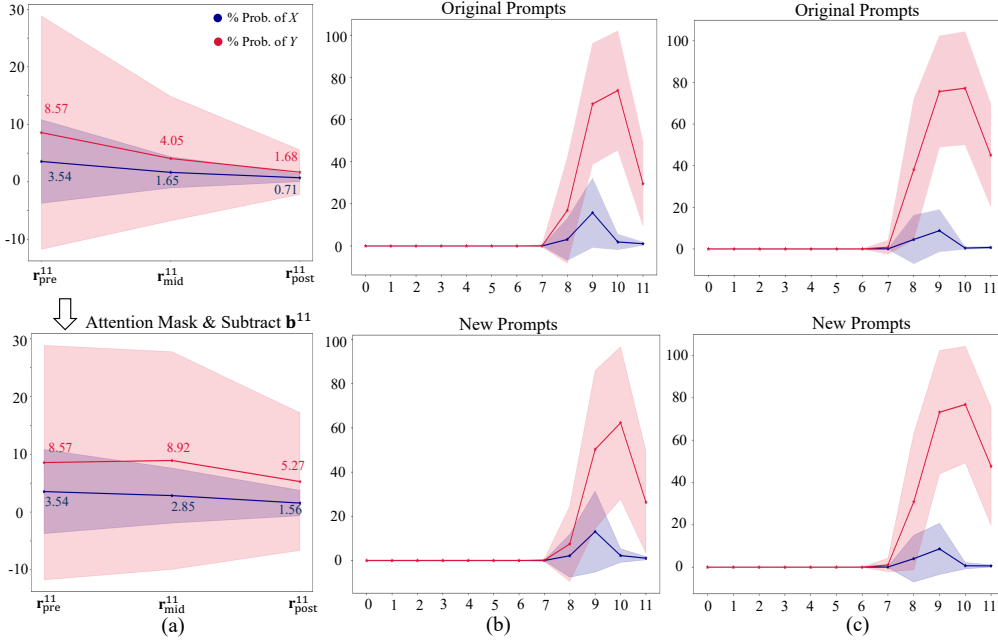


Figure 8: (a) Our proposed techniques, including applying attention masks and subtracting the intercept from the output, can suppress the anti-overconfidence mechanism in the last layer. (b) One-shot probability dynamics given original/new prompts. (c) Two-shot probability dynamics given original/new prompts.

E Identifying the MLP Layer for “Function Application”

In our main text, for simplicity’s sake, we followed Merullo et al. [23] to identify the layer where the probability of Y surpasses that of X as the point of function application. However, this principle is not always accurate. A more general principle would involve examining the layers “around” the point where the probabilities of X and Y begin to diverge significantly. To illustrate this, Figure 9 showcases the zero-shot probabilities of X and Y across the layers in GPT-2 medium and GPT-2 large models.

The behavior of GPT-2 medium mirrors that of GPT-2 small. The probability of Y exceeds that of X for the first time at layer 15. Upon analyzing the intercepts at each layer of GPT-2 medium, we found that at layer 15, $\mathbf{b}^{15, \text{proj}}$ indeed demonstrates the highest alignment with $\mathbf{W}_U[Y] - \mathbf{W}_U[X]$, boasting a cosine similarity of 0.661.

However, in GPT-2 large, the scenario is slightly different. Here, it is at layer 22 where the probability of Y marginally surpasses that of X . Following this, the probabilities of X and Y remain comparable until layer 27, where the probability of X experiences a significant decrease, coinciding with the layer where the projected intercept aligns most closely with $\mathbf{W}_U[Y] - \mathbf{W}_U[X]$, with the cosine probability of 0.821. Hence, it is in layer 27, rather than layer 22, where we recognize the occurrence of function application.

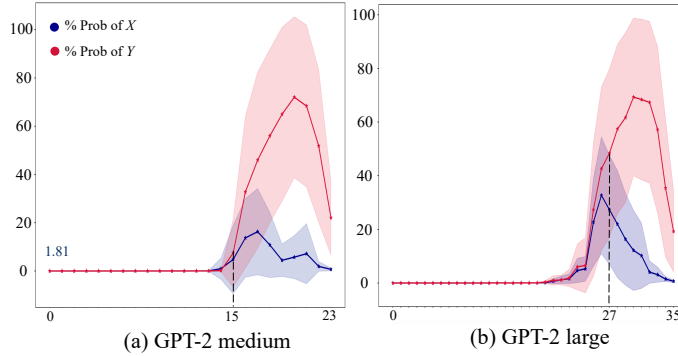


Figure 9: The probability dynamics across layers of GPT-2 medium and GPT-2 large in the zero-shot country-capital task.

F GPT-2 small on the “Product-Developer” task

We validated our findings using GPT-2 small on the product-developer task. We illustrate the probability dynamics of X and Y in Figure 10. While L10H0 remains activated, L9H8 is not, thus confirming the assertion in the main text that L9H8 is task-specific, whereas L10H0 exhibits a more general activation pattern. In layer 8, the emergence of probability for Y is evident, indicating the function application. Our analysis reveals a cosine similarity of 0.635 between $\mathbf{b}^{8, \text{proj}}$ and $(\mathbf{W}_U[Y] - \mathbf{W}_U[X])$. Furthermore, we conducted few-shot experiments on this dataset, depicted in Figure 11. Consistent with the observations outlined in the main text, our findings indicate that the mechanisms examined in the few-shot scenarios are still applicable in the zero-shot scenarios. In Figure 15, we observe the anti-overconfidence phenomenon in the final layer. This phenomenon can be effectively addressed through our proposed anti-suppression techniques, including applying attention masks and subtracting the intercept from the residual stream.

G Experiments on OPT-1.3B

OPT-1.3B [46] consists of 24 layers with 32 heads per layer. In Figure 12, we present the probability dynamics across layers, along with path patching experiments. From Figure 12(a), the argument passing is observed in layer 19. Figure 12(b) reveals that L19H15 in the zero-shot settings acts as the first influential argument passer. While L17H8 and L18H7 also serve as argument passers, their impact on increasing the probability of X is diminished within the layer. Their activation is enhanced

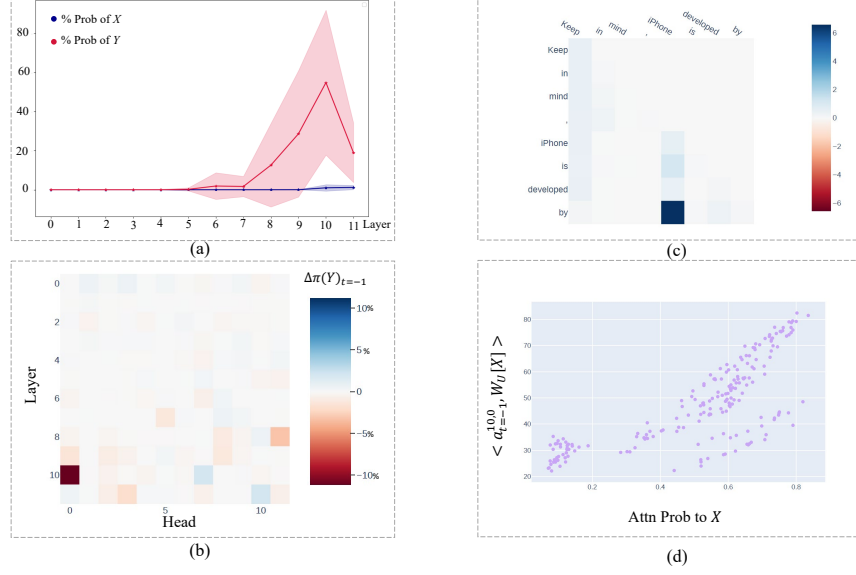


Figure 10: We test GPT-2 small on the product-developer task. (a) The probability of X and Y when early unembed each layer’s outputs. (b) The effect of patching path $\mathbf{a}_{t=-1}^{l,h} \rightarrow \mathbf{r}_{\text{post}}^{11}$. (c) Value weighted attention pattern of L10H0. (d) L10H0 is a mover head.

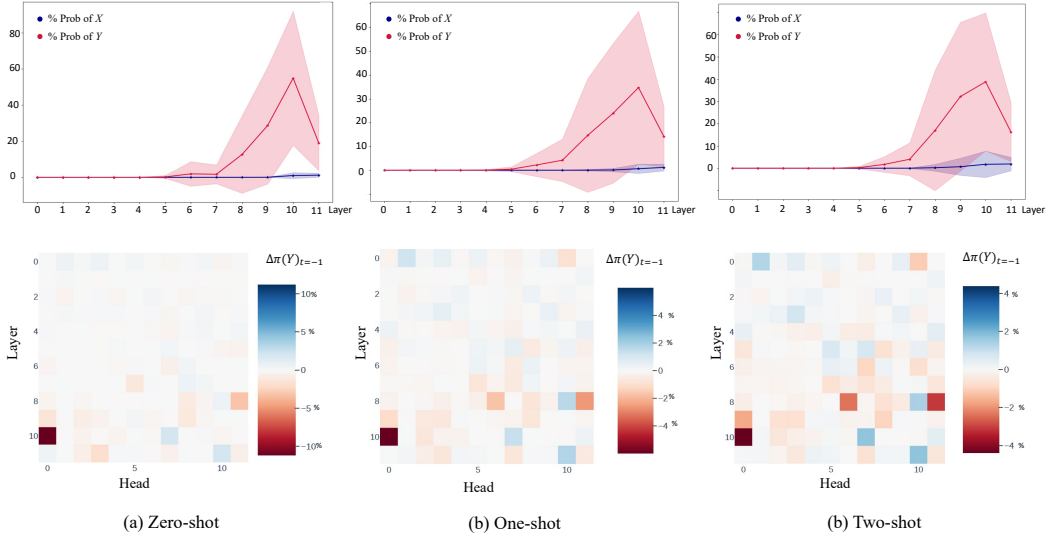


Figure 11: When testing GPT-2 small on the product-developer task, we present probability dynamics of X and Y , alongside influential heads impacting final logits across zero (a), one (b), and two-shot (c) settings. The underlying mechanisms utilized by the models remain consistent with what we observed in the country-capital task.

in few-shots scenarios, making the probability of Y emerge in the residual stream early, as shown in Figure 17(b) and (c), which is similar to the behavior of L8H11 in GPT-2 small.

Recall that in Section 3.4, two key factors prompted us to realize that \mathbf{b}^{10} directs from X to Y within the MLP10 of GPT-2 small: Firstly, X and Y are equally important tokens that occupy most of the probability distribution. Secondly, after MLP10, the probability of X decreases, but that of Y increases. We have emphasized that these two factors do not always occur, and interpreting the mechanisms of MLPs requires a case-by-case analysis, depending on the model, task, or even prompts.

As a case study, given the prompt “Therefore, it is correct to state that the capital of England is,” the MLP20 in OPT-1.3B increases the probability of “London” by replacing the most likely incorrect answer “England” with a safe token “not,” rather than directly steering the residual stream towards “London” from “England.” The cosine similarity between $\mathbf{b}^{20, \text{proj}}$ and $(\mathbf{W}_U[\text{not}] - \mathbf{W}_U[\text{England}])$ is 0.778, while the cosine similarity between $\mathbf{b}^{20, \text{proj}}$ and $(\mathbf{W}_U[\text{London}] - \mathbf{W}_U[\text{England}])$ is only 0.1. As shown in Table 6, in the transition from $\mathbf{r}_{\text{mid}}^{20}$ to $\mathbf{r}_{\text{post}}^{20}$, the probabilities of “London” and “not” are increased by MLP20, while the probability of “England” decreases. This increase in “London,” as demonstrated, is merely a by-product of the decline in “England.”

Token \ Node	$\mathbf{r}_{\text{mid}}^{20}$	$\mathbf{r}_{\text{post}}^{20}$
not	1.32%	4.97%
London	4.17%	3.21%
England	67.40%	74.40%

Table 6: Probability of several tokens decoded from certain nodes in OPT-1.3B.

Regarding the final layer, as illustrated in Figure 17, OPT-1.3B also demonstrates anti-overconfidence tendencies, suppressing accurate predictions across zero-shot, one-shot, and two-shot scenarios. By employing the anti-suppression techniques outlined in Section 4, we effectively alleviate the anti-overconfidence mechanism employed by OPT.

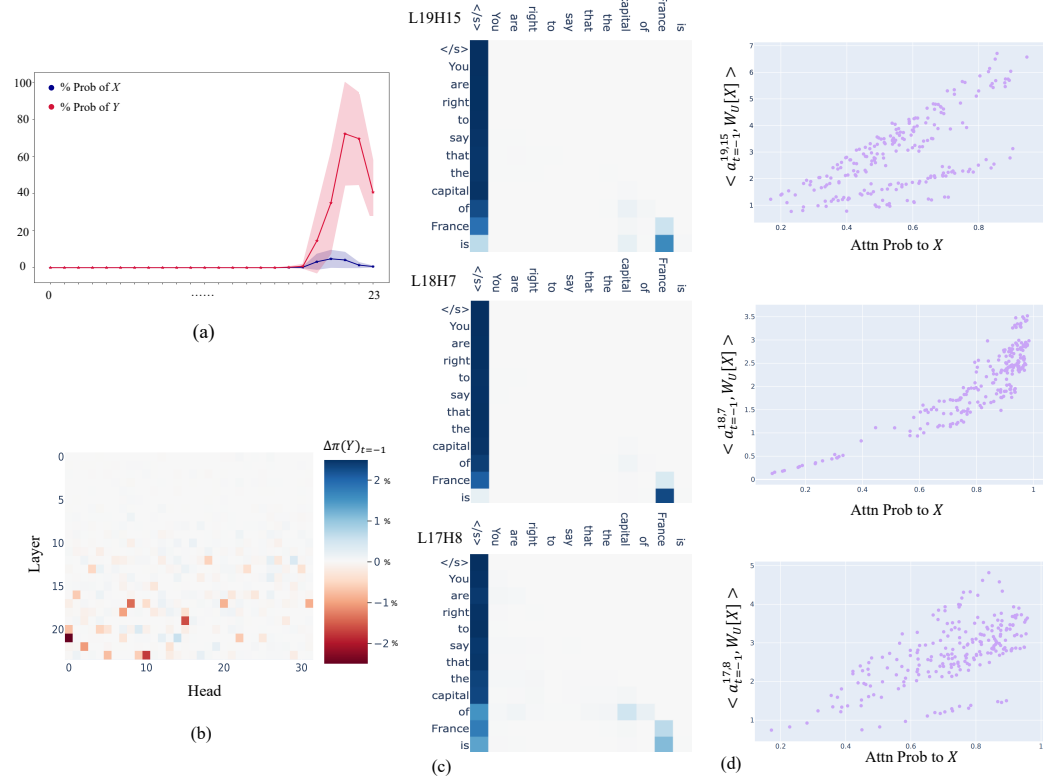


Figure 12: Experiments on OPT-1.3B. The argument passing mechanism and the anti-overconfidence phenomenon are observed.

H Details on Solving the Linear Regression

We present the regression solutions for Eq.7 in GPT-2 small on the country-capital task. The coefficients for other tasks and models will be available in our code repository. Table 7 presents the learned coefficients for each layer, along with the validation set loss in our 4-fold cross-validation. The optimization consistently converges to similar solutions for each layer with minimal loss. Although a few weights in one of the four experiments might diverge from the other three a little, the overall results remain stable enough to support qualitative analysis. Furthermore, it is natural for the validation MSE loss to rise as the layer depth increases, given that the vectors in deeper layers possess larger norms. As MSE is a quadratic loss whose penalty is proportional to the square of the error,

to alleviate any concerns regarding the convergence, we also present the Absolute Percentage Error (APE) in Table 8, expressed as $\frac{\sqrt{\text{MSE}(\tilde{\mathbf{m}}^l, \mathbf{m}^l)}}{\|\mathbf{m}^l\|} \cdot 100\%$. The APE values for each layer hover around a mere 1%, indicating minimal deviation.

Table 7: The learned coefficients for individual layers of GPT-2 small on the country-capital task. Utilizing a 4-fold cross-validation, our results present four rows per layer, accompanied by the corresponding validation loss.

Layer i \ $w^l \cdot$	0	1	2	3	4	5	6	7	8	9	10	11	r	val. loss
0	-0.3686	0.9726	-0.2864	0.2247	-0.5162	1.4244	-0.3301	-0.2689	0.3259	-0.0227	-0.1424	-0.4358	-0.0054	0.0138
	-0.3800	1.0004	-0.2849	0.1797	-0.5336	1.4090	-0.3203	-0.2905	0.3299	-0.0173	-0.1103	-0.4587	-0.0026	0.0126
	-0.3864	0.9796	-0.2923	0.2084	-0.5123	1.4315	-0.3312	-0.2953	0.2912	-0.0206	-0.1128	-0.4189	-0.0094	0.0127
	-0.3914	0.9988	-0.2914	0.2020	-0.5298	1.4186	-0.3192	-0.2759	0.3044	-0.0027	-0.1215	-0.4616	-0.0094	0.0132
1	-0.2393	-0.3280	-0.3935	-0.4470	-0.3156	-0.7519	-0.6252	-0.4797	-0.4534	-0.7626	-0.3916	-0.4750	-0.1573	0.0107
	-0.2429	-0.3327	-0.4280	-0.4585	-0.3394	-0.7344	-0.6290	-0.4874	-0.4745	-0.7808	-0.4168	-0.4807	-0.1531	0.0117
	-0.2415	-0.3313	-0.3565	-0.4324	-0.3067	-0.7680	-0.6353	-0.4822	-0.4353	-0.7687	-0.3753	-0.4521	-0.1587	0.0106
	-0.2372	-0.3275	-0.4528	-0.4507	-0.3318	-0.7301	-0.6256	-0.4961	-0.4737	-0.7519	-0.4076	-0.4665	-0.1493	0.0124
2	-0.2887	-0.4553	-0.2495	-0.2351	-0.2574	-0.1066	-0.1911	-0.1704	-0.2661	-0.2948	-0.2853	-0.2810	-0.0932	0.0153
	-0.3077	-0.4516	-0.2314	-0.2335	-0.2223	-0.0894	-0.2429	-0.1626	-0.2793	-0.3056	-0.2923	-0.2700	-0.0947	0.0134
	-0.2853	-0.4341	-0.2446	-0.2390	-0.2337	-0.0997	-0.1758	-0.1613	-0.2627	-0.2987	-0.2708	-0.2735	-0.0932	0.0141
	-0.2856	-0.4360	-0.2471	-0.2180	-0.2246	-0.0876	-0.2141	-0.1622	-0.2542	-0.2926	-0.2864	-0.2580	-0.0961	0.0147
3	-0.8012	-0.1225	-0.1426	0.0427	-0.6382	-0.3052	-0.0909	-0.1436	-0.1325	-0.3933	-0.0073	-0.2214	-0.0493	0.0280
	-0.6559	-0.1240	-0.1452	0.0240	-0.5122	-0.3043	-0.1142	-0.1450	-0.1045	-0.3816	-0.0126	-0.2123	-0.0543	0.0279
	-0.7190	-0.1228	-0.1308	-0.0014	-0.5717	-0.2948	-0.1150	-0.1526	-0.1014	-0.3853	0.0165	-0.2102	-0.0517	0.0267
	-0.6730	-0.1243	-0.1428	0.0183	-0.5318	-0.3005	-0.1112	-0.1356	-0.1200	-0.3980	-0.0059	-0.2139	-0.0518	0.0275
4	0.1505	-0.3024	-0.2439	-0.0427	-0.2223	-0.3436	-0.1989	-0.1019	-0.4813	-0.2967	-0.1689	-0.0462	-0.0488	0.0396
	0.1756	-0.3114	-0.2359	-0.0214	-0.2308	-0.3454	-0.1885	-0.1140	-0.4238	-0.2932	-0.2134	-0.0459	-0.0535	0.0385
	0.1720	-0.3159	-0.2396	-0.0286	-0.2486	-0.3203	-0.2100	-0.1187	-0.4806	-0.2973	-0.2572	-0.0410	-0.0516	0.0408
	0.1771	-0.3127	-0.2385	-0.0200	-0.2402	-0.3337	-0.2011	-0.1045	-0.4873	-0.2937	-0.2231	-0.0462	-0.0523	0.0423
5	-0.4228	0.0504	-0.0660	-0.1955	-0.2287	-0.6634	-0.2867	-0.1622	-0.0024	-0.2363	-0.1871	0.0556	-0.1288	0.0560
	-0.5210	-0.0116	-0.0694	-0.1978	-0.2577	-0.7679	-0.2767	-0.1730	-0.0364	-0.3313	-0.1937	0.0697	-0.1248	0.0531
	-0.5281	0.0226	-0.0544	-0.1991	-0.2338	-0.7687	-0.2494	-0.1651	-0.0209	-0.3130	-0.1769	0.0788	-0.1309	0.0505
	-0.5418	0.0120	-0.0534	-0.1911	-0.2512	-0.8036	-0.2755	-0.1677	-0.0075	-0.3269	-0.1885	0.0743	-0.1289	0.0476
6	-0.3502	-0.3408	-0.1792	-0.2084	-0.2060	-0.2533	-0.1816	-0.2186	-0.2031	-0.6458	-0.0613	-0.2533	-0.0860	0.0919
	-0.3403	-0.3274	-0.1782	-0.2105	-0.1974	-0.2533	-0.1714	-0.2152	-0.2454	-0.6167	-0.1515	-0.2605	-0.0842	0.1000
	-0.3792	-0.3282	-0.1688	-0.2234	-0.1978	-0.2596	-0.1536	-0.2075	-0.1982	-0.6181	-0.0502	-0.2456	-0.0829	0.0931
	-0.3911	-0.3466	-0.1774	-0.2028	-0.2056	-0.2538	-0.1571	-0.2176	-0.2213	-0.6132	0.0202	-0.2514	-0.0892	0.0948
7	0.3063	-0.1772	0.7709	-0.0521	-0.1195	-0.1144	0.0212	-0.2405	0.0831	-0.1142	-0.2198	-1.4439	-0.0679	0.1322
	0.1489	-0.2222	0.9735	-0.0497	-0.1247	-0.1020	0.0019	-0.3483	0.1054	-0.1258	-0.1997	-1.4224	-0.0637	0.1356
	0.2591	-0.2214	0.8430	-0.0717	-0.1161	-0.1049	0.0296	-0.1952	0.0712	-0.1193	-0.1429	-1.5894	-0.0678	0.1313
	0.4062	-0.1981	0.8108	-0.0660	-0.1272	-0.1004	0.0159	-0.2716	0.0753	-0.1178	-0.2205	-1.5409	-0.0726	0.1333
8	0.0325	-0.6455	-0.0394	-0.0859	-0.0266	-0.0056	-0.3653	0.1872	0.1150	-0.1155	-0.1474	-0.1198	-0.0203	0.2312
	-0.0340	-0.9104	-0.0605	-0.0866	-0.0279	-0.0186	-0.3973	0.1453	0.0788	-0.1179	-0.1669	-0.1223	-0.0222	0.2055
	-0.0393	-0.9561	-0.0617	-0.0975	-0.0263	-0.0183	-0.4420	0.1786	0.0936	-0.1270	-0.1670	-0.1288	-0.0213	0.2093
	-0.0215	-1.0882	-0.0729	-0.0792	-0.0441	-0.0045	-0.4936	0.1692	0.0913	-0.1387	-0.1671	-0.1256	-0.0225	0.2219
9	-0.0357	-0.2265	0.0362	-0.0383	-0.2627	-0.1390	-0.1492	-0.0289	0.0788	-0.0914	-0.0783	-0.7471	0.0572	0.4518
	-0.0456	-0.2074	0.0455	0.0299	-0.2481	-0.1098	-0.1293	-0.0514	0.0820	-0.1197	-0.0763	-0.6468	0.0548	0.4441
	-0.0291	-0.2880	0.0192	-0.0146	-0.3020	-0.1289	-0.1951	-0.0381	0.0843	-0.0906	-0.0755	-0.6017	0.0576	0.4773
	-0.0767	-0.2336	0.0256	-0.0015	-0.3060	-0.1643	-0.2030	-0.0439	0.0727	-0.0755	-0.0675	-0.5727	0.0554	0.4763
10	-0.0693	0.11717	-0.0888	0.1557	-0.9873	-0.1603	-0.1341	-0.0967	-0.8532	-0.5571	0.0060	-0.3232	0.0940	1.0827
	-0.0754	0.0791	-0.0852	0.1605	-1.036	-0.2174	-0.0937	-0.1049	-0.3328	-0.5843	0.0003	-0.4004	0.0978	1.1474
	-0.0780	0.0764	-0.1289	0.1524	-0.9804	-0.1736	-0.1194	-0.0984	-0.8746	-0.5403	0.0075	-0.4186	0.1031	1.1478
	-0.0683	0.0347	-0.0666	0.1490	-0.9140	-0.1260	-0.1350	-0.0868	-1.4332	-0.45792	0.0050	-0.3183	0.0954	1.3073
11	-0.2254	0.6708	-0.0932	-0.3486	-0.0158	1.7178	0.6173	0.4640	0.1175	1.1539	0.1363	0.1677	0.0078	1.4323
	-0.1900	0.8313	-0.0928	-0.3666	0.0761	1.4899	0.5619	0.4020	0.1213	0.9887	0.1271	0.2024	0.0090	1.5322
	-0.2594	0.8378	-0.0944	-0.3074	-0.0770	1.3838	0.6542	0.4058	0.1273	0.9250	0.1597	0.2791	0.0071	1.4179
	-0.2315	0.6442	-0.0931	-0.2749	-0.0269	2.0470	0.5061	0.4638	0.1149	1.1552	0.1235	0.2852	0.0076	1.4370

Table 8: The Absolute Percentage Error (APE) of Eq.7 within each layer.

Layer	0	1	2	3	4	5	6	7	8	9	10	11
APE	0.37%	1.53%	1.71%	1.68%	1.58%	1.63%	1.70%	1.41%	1.49%	1.56%	0.98%	1.65%

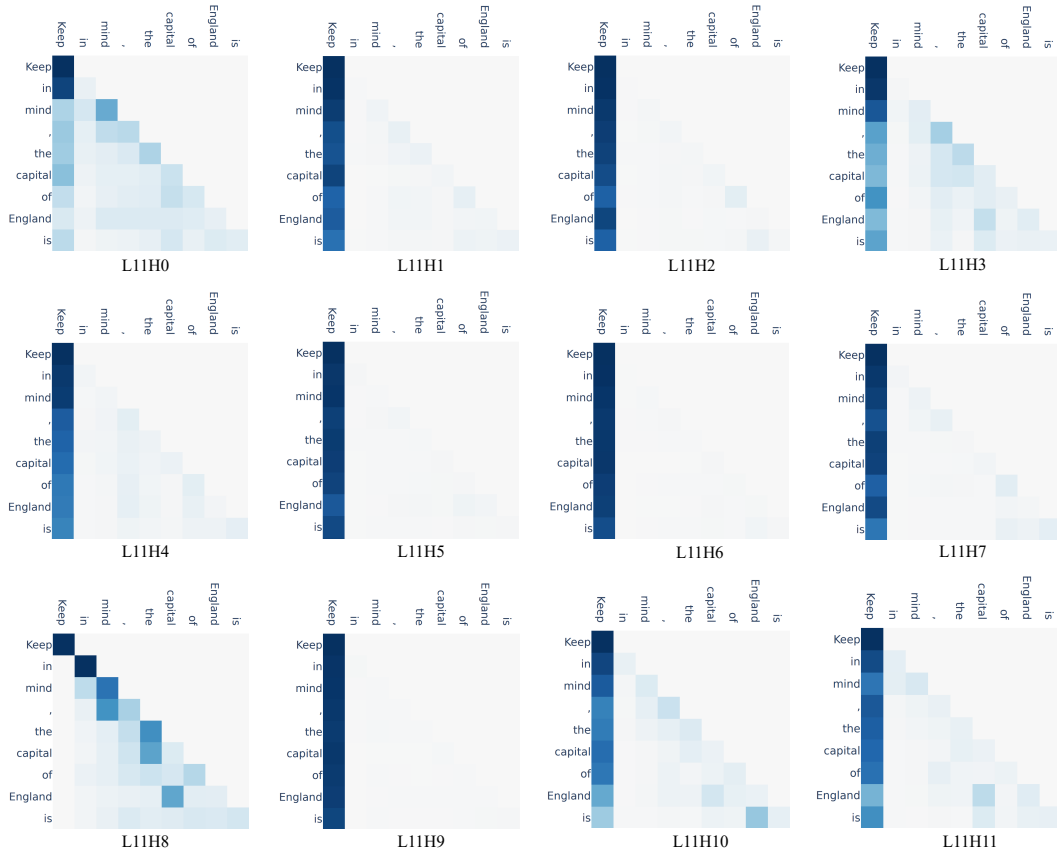


Figure 13: The attention patterns in layer 11 of GPT-2 small in the country-capital task.

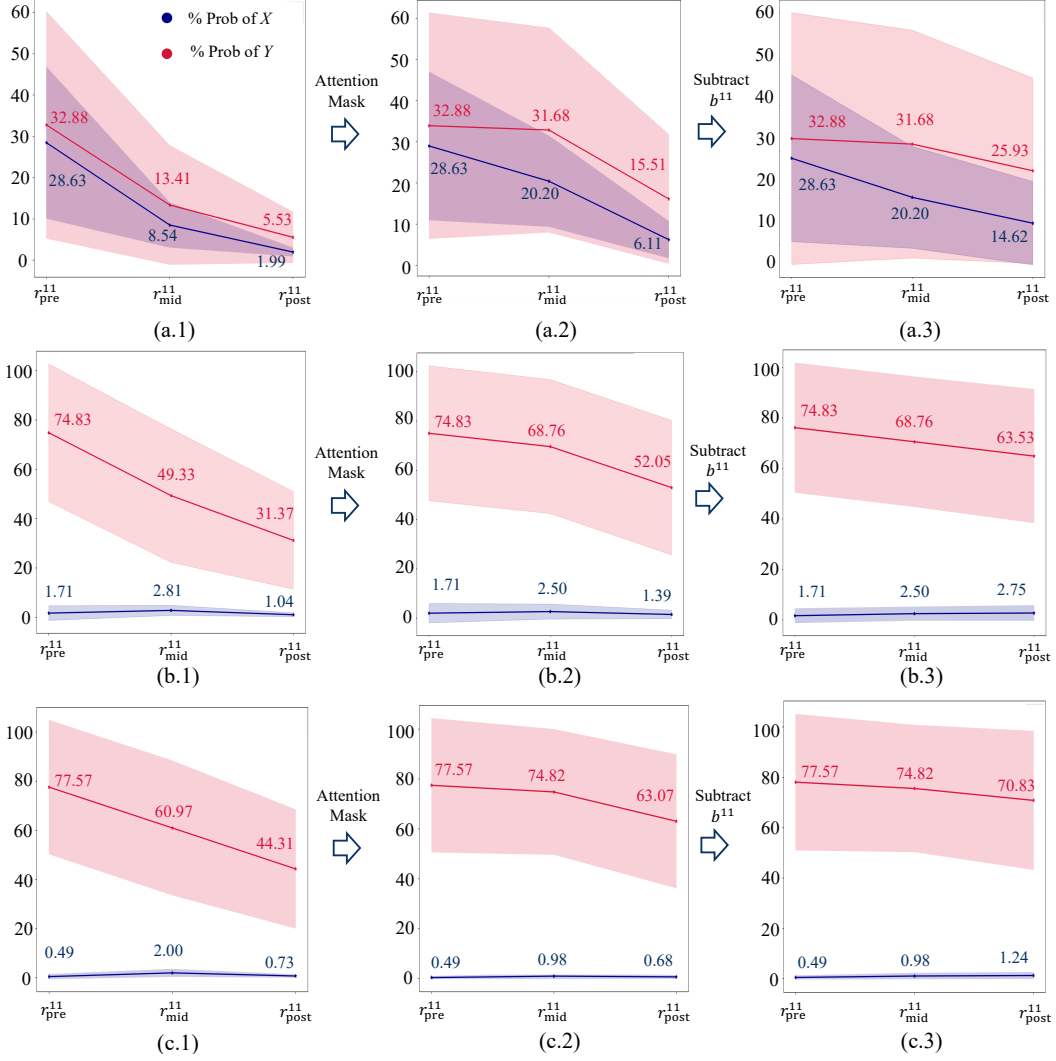


Figure 14: Given the country-capital task, the probability of X and Y in the final layer of GPT-2 small in (a) zero-shot, (b) one-shot, and (c) two-shot scenarios. The suppression of Y by the attention heads in layer 11 can be eliminated by reallocating their attention. The suppression of Y by MLP11 can be mitigated by subtracting b^{11} from the residual stream.

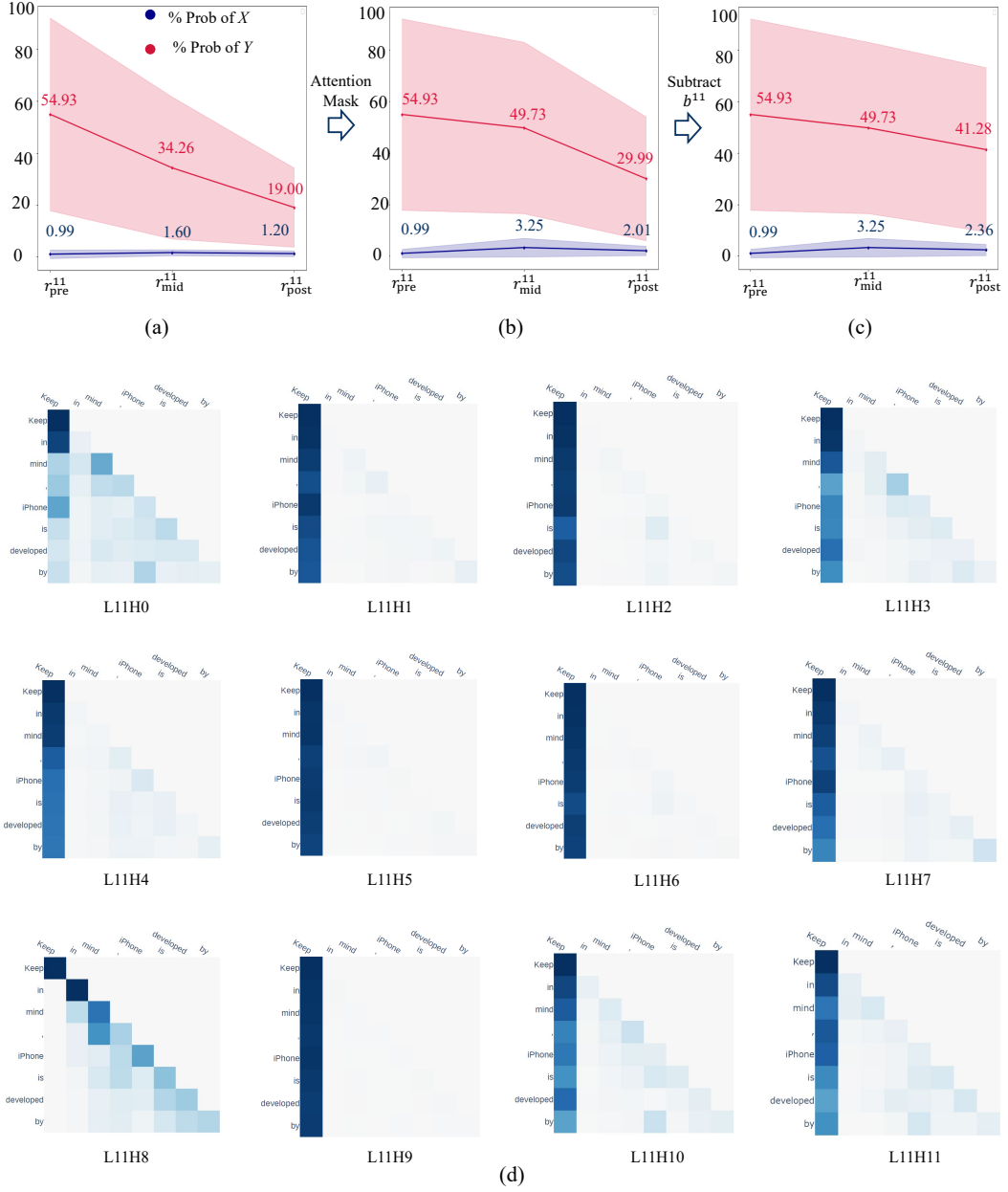


Figure 15: Experiments on GPT-2 small in the product-developer task. From (a) to (b): The suppression of Y by the attention heads in layer 11 can be eliminated by reallocating their attention. From (b) to (c): The suppression of Y by MLP11 can be mitigated by subtracting \mathbf{b}^{11} from the residual stream. The results are obtained in zero-shot settings, and these findings hold true in few-shot settings. (d) The attention pattern of heads in layer 11.

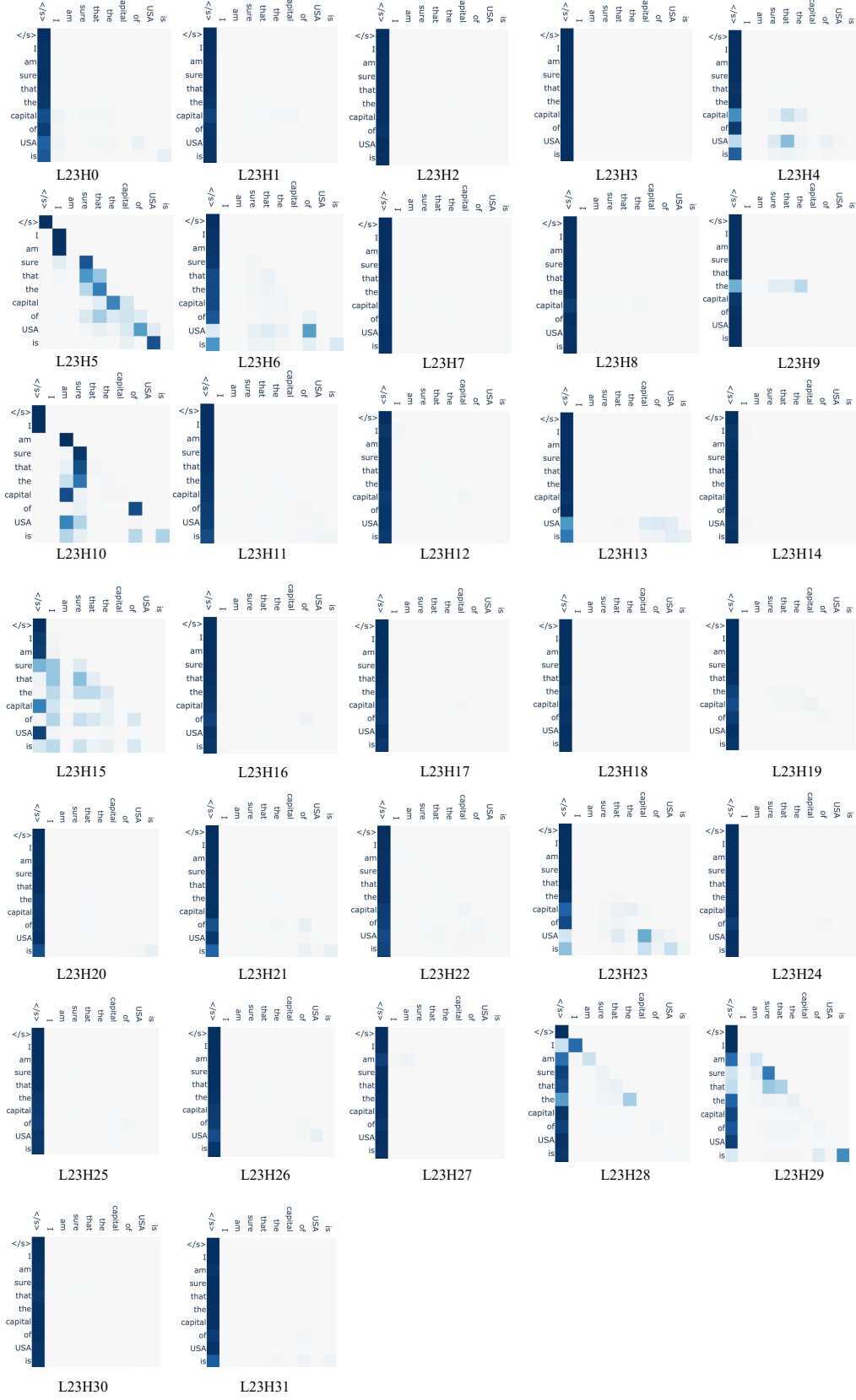


Figure 16: The attention patterns in layer 23 of OPT-1.3B in the country-capital task.

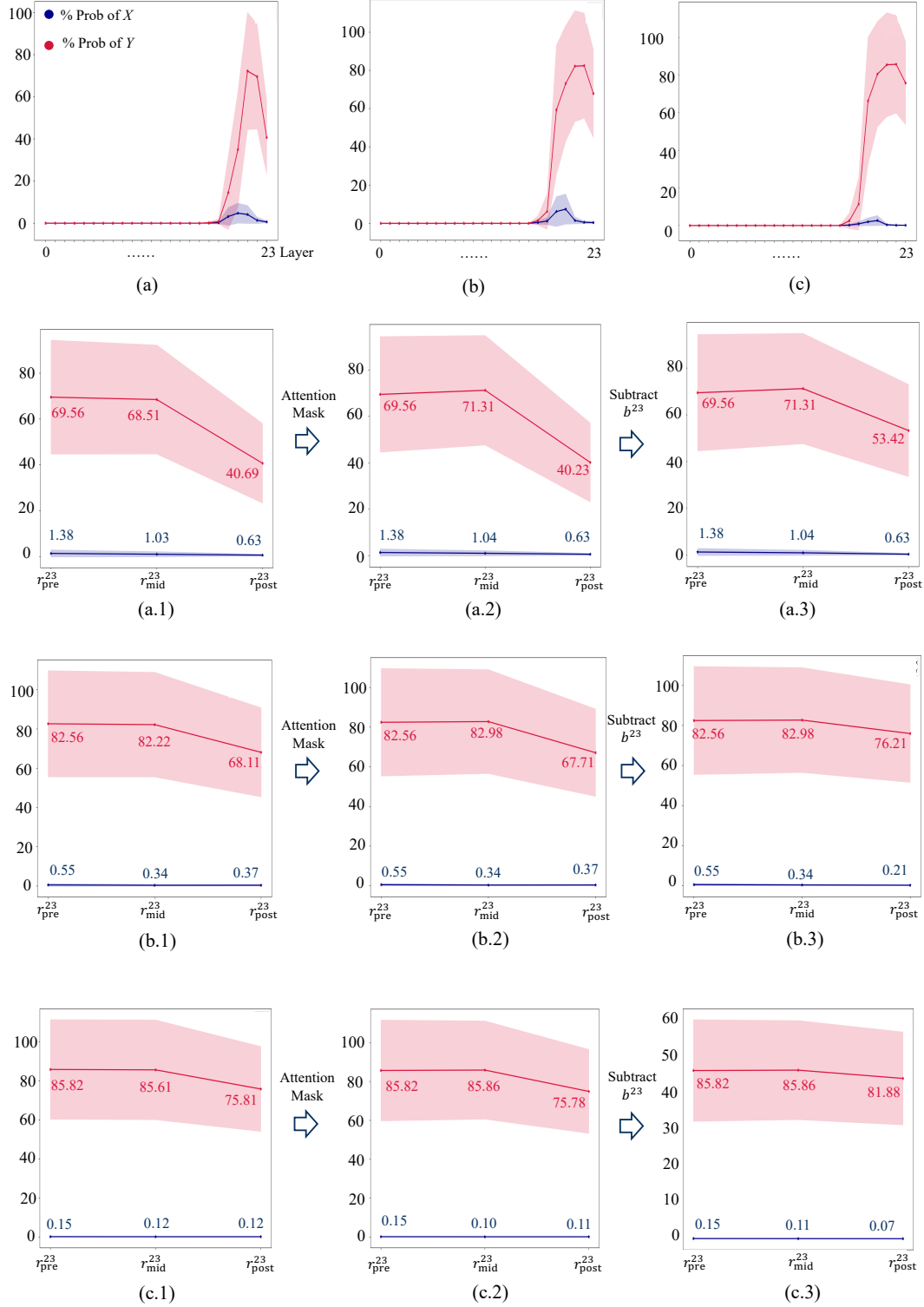


Figure 17: The final layer of OPT-1.3B exhibits suppression of Y across (a) zero-shot, (b) one-shot, and (c) two-shot scenarios. Our anti-suppression techniques are applicable for OPT-1.3B, effectively enhancing the probability of Y .