Factual Retrieval in LLMs Is a Redundant, Distributed and Non-Contiguous Process

Hail Hochman
Bar-Ilan University
hailzanbar1@gmail.com

Natalie Shapira
Northeastern University
nd1234@gmail.com

Yoav Goldberg
Bar-Ilan University
yoav.goldberg@gmail.com

Abstract

Large language models (LLMs) store and recall factual knowledge, yet the precise mechanism of how entity representations are transformed to enable specific attribute retrieval remains underexplored. In this work, we investigate this mechanism through the lens of an "attribute-computation path"—a sequence of computational steps over the entity representation required to elicit a target attribute. We then propose an iterative patching protocol to identify a minimal subset of layers necessary for this computation. Applying our method to LLaMA 3.1 8B and Qwen3 8B, we find that these paths are non-contiguous, often skipping layers, and that models possess multiple, functionally-equivalent paths for the same entity and fact, highlighting a high degree of redundancy in attribute computation. This implies that knowledge computation is highly distributed, potentially explaining the localization-editing mismatch and suggesting that knowledge storage and retrieval in LLMs is far from being well understood.

1 Introduction

Large language models (LLMs) have been shown to store and recall factual knowledge expressed as entities and their relations (Petroni et al., 2019; Jiang et al., 2020; Cohen et al., 2023). For example, when prompted with *"The mother tongue of Angela Merkel is"*, an LLM will predict *"German"*, the correct answer. While prior studies attempted to identify components of the model where factual knowledge is stored (Geva et al., 2021; Dai et al., 2021; Geva et al., 2022; Meng et al., 2022; Gurnee and Tegmark, 2023; Katz et al., 2024; Yu and Ananiadou, 2024) and to map the general information flow during recall (Geva et al., 2023; Nanda et al., 2023; Chughtai et al., 2024; Wang and Xu, 2025; Yao et al., 2024; Yu et al., 2025), the precise mechanics of how facts are retrieved from model parameters remain unclear. Existing work posits that this process occurs at the last entity token position; however, the term "retrieval" itself is not well defined. It remains ambiguous whether retrieval is a continuous process of information accumulation across a range of layers, or if it is defined by a discrete state—a specific point where the representation becomes rich enough to elicit the specific correct attribute. Furthermore, the transformations the entity representation undergoes *before* it becomes capable of eliciting the target attribute are largely unmapped. To address these complexities, we propose investigating the factual recall mechanism through a new lens: mapping the *minimal computation path* the entity representation must undergo in order to achieve attribute recall, in a given factual-recall prompt.

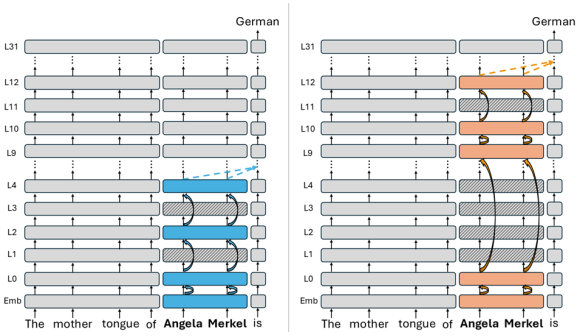


Figure 1: Two functionally equivalent minimal attribute-computation paths for the entity "Angela Merkel". Both paths are non-contiguous, skipping intermediate layers. The blue path represents a compact, early computation, while the orange path illustrates a deeper, more distributed alternative.

Given a prompt with an entity with a relation (e.g., *"The mother tongue of Angela Merkel is"*), and a target attribute (e.g., *"German"*), we define an *attribute-computation path* as a sequence of computations over the entity representation, that together enable the LLM to answer the prompt correctly. Each computation is implemented as a transformer layer, and we seek a subset of layers

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 46747–46768

which is both sufficient and necessary for computing the desired output. We call such a subset of layers, in which no layer can be removed without hurting the computation, *a minimal attribute computation path*, and observe that each layer along such a minimal path plays an active role in the factual recall process for the attribute.

We present a method to identify minimal computation paths for a given factual recall prompt, and use it to extract computation paths for a diverse set of such prompts, in two open-weight LLMs (LLaMA 3.1 8B and Qwen3 8B). Our experiments reveal that for a majority of target attributes:

- *The minimal path naturally ends in the early-mid layers*, after which the attribute value is fully available and no further processing is required. This is consistent with observations in previous works (Geva et al., 2023; Nanda et al., 2023; Meng et al., 2022).
- *Minimal paths consist of more than a single layer*: there isn't a single layer responsible for knowledge retrieval for a given entity-relation-attribute triplet, but rather the entity must be processed by multiple layers before the information is fully available.
- *Minimal paths are often sparse and non-contiguous*: only a subset of layers participate in each factual recall process, while others may be skipped (but the same layer may participate in multiple factual recall processes).

Taken together, the last two items indicate that factual knowledge is stored in a distributed and non-localized manner: there isn't a single layer or range of layers we can say is responsible for a given fact, rather the factual information is distributed across the parameters of multiple layers. Our experiments further reveal "backup" paths (Wang et al., 2022; McGrath et al., 2023):

- *Minimal paths are not unique*: for most facts we explored, there is at least one alternative set of layers that is sufficient for retrieving the factual information. These alternate paths are deeper and longer than the primary paths the model uses in natural, non-intervened runs (Figure 1).

We find that the same factual knowledge can be retrieved through multiple distinct computational paths within the model. This suggests an even more complex and elaborate form of knowledge representation, where the storage and retrieval mechanism is not only distributed but also highly redundant. These findings provide a new perspective on the mechanism of factual recall (Geva et al., 2023; Meng et al., 2022; Yu et al., 2025). We detail these, together with additional findings, in Section 6. To locate the minimal paths, we use a novel iterative activation-patching protocol, described in Section 5.

In summary, our work makes three main contributions.

- We offer a new perspective on the sub-process of entity enrichment (Geva et al., 2023; Yu et al., 2025), showing that the computation of the entity representation for specific attribute recall unfolds as a *multi-step process* involving several necessary intermediate transformations, while at the same time not requiring most model layers.
- We reveal that *multiple minimal and functionally equivalent attribute computation paths exist* within the model with different lengths and in different regions, highlighting the distributed and redundant nature of factual knowledge representation.
- We propose a novel activation patching protocol to identify a minimal subset of layers required for computation of an attribute over the entity activations, providing a precise view of how factual knowledge enriches the entity representations.

Based on our findings, we propose an explanation for the discrepancy between where the facts are located and where they are most effectively edited (Hase et al., 2023). We suggest that because attribute computation requires multiple transformations of the entity representation, editing succeeds by intervening at a necessary transformation stage, not necessarily where the knowledge is localized. Our code is available at `https://github.com/hhochman/llm-factual-retrieval`.

2 Related Work

Entity Representation & Factual Recall. Prior work frame factual recall in large language model as a three-stage process (Geva et al., 2023; Yu et al., 2025; Chughtai et al., 2024; Nanda et al., 2023). In their framing, early attention layers first *consolidate* the representation of the entities initial tokens

into its final token (Nanda et al., 2023). Then, the final entity token undergoes an *entity enrichment phase* in which various entity attributes are loaded from the model parameters and encoded on the (final) entity tokens. This stage is primarily driven by early feed-forward (MLP) sublayers (Meng et al., 2022; Geva et al., 2023; Yu et al., 2025) in which the knowledge resides. Finally, once entity enrichment concludes and the knowledge is retrieved and loaded into the entity representation, information from the relation tokens propagates forward in the sequence, reaching the final token of the prompt. Then, the representation at the final token of the prompt *queries* the enriched entity representation to extract the target attribute (Geva et al., 2023).

It is also established that the resulting representation at the entity's final token plays a central role in knowledge storage and retrieval (Geva et al., 2023; Yu et al., 2025; Hernandez et al., 2023; Ghandeharioun et al., 2024). Recent work demonstrates that, for many relations, attributes can be decoded from the entity's final token using a simple, approximately linear transformation (Hernandez et al., 2023). Furthermore, probing the hidden states of this token in late–mid layers can reveal the extent of the model's stored knowledge about the entity, without relying on generated outputs (Gottesman and Geva, 2024). When the relation follows the entity in the prompt, enrichment and attribute extraction can occur not only at the entity position but also at the relation and final positions (Yu et al., 2025; Chughtai et al., 2024). In these cases, deeper attention and feed-forward components store factual knowledge linking the entity and relation to the attribute.

While Meng et al., 2022 identified stages where representations become sufficient for attribute restoration and hypothesized accumulation across MLPs, the precise causal computations of this process remain under-explored. We zoom in on the entity enrichment stage and trace the complete causal paths by which individual facts are loaded into the entity representation. Our findings reveal a complex and elaborate system in which each individual fact requires processing by multiple LLM layers before it can be accessed: there is no single layer in which a specific fact is retrieved.

Circuit Redundancy. Early work in mechanistic interpretability of LLMs and other models focused on identifying circuits: subgraphs of the model's computational graph responsible for specific behaviors (Olah et al., 2020; Elhage et al., 2021; Wang et al., 2022; Goldowsky-Dill et al., 2023; Ameisen et al., 2025). However, the definition of a "circuit" is complicated by the distributed and redundant nature of LLMs.

McGrath et al. (2023) revealed the "hydra effects", where the model self-repairs by activating redundant components when primary ones are ablated. Wang et al. (2022) similarly identified "backup heads" that perform the same function as primary heads but are only active when the primary path is disrupted. This complexity extends to model editing; Hase et al. (2023) demonstrated that while facts can be successfully edited at specific model locations, the most effective editing targets often diverge from the locations identified by standard localization methods. These findings suggest that models do not rely on a single, unique circuit for a given task, but rather possess multiple functionally equivalent mechanisms.

While Wang et al. (2022) revealed backup "Name-Mover Heads" in the IOI task, and the hydra effect (McGrath et al., 2023) has been investigated primarily regarding the information flow to the final token, we demonstrate redundancy within a different factual recall mechanism: the layers performing the computation process over the entity tokens that is used to retrieve the attribute.

3 Preliminaries and Notations

Layers and Activations. Let $I^{(i,\ell)}$ denote the input activations to layer ℓ at position i.[1] A layer function L_ℓ applied to $I^{(i,\ell)}$ results in the layer's output $O^{(i,\ell)}$, which serves as the input to the next layer, $I^{(i,\ell+1)}$:

$$L_\ell(I^{(i,\ell)}) = O^{(i,\ell)} = I^{(i,\ell+1)}$$

We refer to the same activations as either $O^{(i,\ell)}$ or $I^{(i,\ell+1)}$, depending on the context. The layer function L_ℓ computes attention ($Attn$) and MLP with residual connections:

$$\begin{aligned} L_\ell(I^{(i,\ell)}) &= I^{(i,\ell)} + a_{(\leq i,\ell)} + m_{(i,\ell)} \\ a_{(\leq i,\ell)} &= Attn_\ell(I^{(1,\ell)}, ..., I^{(i,\ell)}) \\ m_{(i,\ell)} &= MLP_\ell(I^{(i,\ell)} + a_{(\leq i,\ell)}) \end{aligned}$$

Runs and Activation Patching. We denote the prediction of a model M given prompt P as M(P).

[1] With slight abuse of notation, we use a single index to refer to either a single token position or a range of tokens (corresponding to an entity).

In this work we restrict ourselves to single word predictions using greedy decoding.

Given an n tokens input prompt, the transformer decoder step with $|L|$ layers computes the values $O^{(i,\ell)}$ for $1 \leq \ell \leq |L|; 1 \leq i \leq n$, in topological order where a node $O^{(i,\ell)}$ is computed after nodes with positions $\leq i$ and layers $< \ell$. A *patch operation* Φ specifies a list of locations (i, ℓ) and a corresponding patch vector for each. A *patched run* $M(P \mid \Phi)$ on an input prompt P computes the activation vectors according to their topological order, but, whenever (i, ℓ) corresponds to a patching location, $I^{(i,\ell)}$ is replaced with the corresponding value instead of its computed value, and the computation continues.

4 Setup and Goal

To understand how factual knowledge is stored and retrieved in LLMs, we are interested in tracing the computations performed by a transformer-based LLM when completing *factual recall prompts* such as *"The mother tongue of Angela Merkel is"*. We focus our attention on the process in which the *entity representation* evolves through the layers until it is sufficient for the model to produce the correct answer ("German"). Considering each layer as performing a computation over the entity representation, we look for what we call a *minimal attribute computation path*: a subset of layers that are *required* for producing the correct attributes.

Factual Recall Prompts. We study prompts of the form $(e, r) \rightarrow a$, where e is a subject entity, r is a relation, and a is the target attribute to be predicted (not part of the prompt). For example, in the above prompt the subject entity e is "Angela Merkel", the relation r is "mother's tongue", and the target attribute a is "German". The relation may appear either before the entity or after it. Both entities and relations are drawn from a variety of domains.

Entity Representation is the activation values at the entity position for a given prompt. It evolves through the layers, with the representation at layer ℓ denoted as $O^{(e,\ell)}$.

Computation Paths. A *computation path* is a series of computations (a subset of layers) applied to an entity representation. An *attribute computation path* is a computation path after which the model correctly predicts the target attribute. A *minimal attribute-sufficient computation path* (or *minimal attribute path* for short) is a path in which every included layer is essential; removing any layer from this set disrupts the computation of the attribute over the entity activations.[2]

Relations to Knowledge Retrieval. Each layer belonging to a minimal attribute path plays an essential role in the process of *knowledge retrieval* for that attribute: it either extracts (parts of) the attribute value from the model's parameters; transforms the entity representation to facilitate the knowledge extraction; or transforms extracted values into a form from which the attribute can be decoded. Thus, we consider the study of minimal attribute paths as an essential milestone for understanding factual knowledge retrieval in transformer-based LLMs.

5 Method

In an $|L|$-layers transformer M for which $M(P_{(e,r)}) = a$, the path $[L_1, ..., L_{|L|}]$ is attribute sufficient for a. But are all the layers necessary?

To identify a minimal attribute path, we start by determining the termination point of the path: the earliest layer $\ell_{attr} \leq |L|$ after which no more computations on the entity representation are needed for M to produce the attribute a at the last token (subsection 5.1). Second, using layer ℓ_{attr} as an upper bound, we iteratively identify a minimal sequence of layers that progressively transform the entity embedding for recalling attribute a (subsection 5.2). Finally, we show that by replacing the upper bound ℓ_{attr} with $|L|$ and using the same iterative procedure, we can identify alternative minimal attribute paths (subsection 5.3).

5.1 Locating ℓ_{attr}

To bound the minimal attribute path from above, we seek a layer ℓ whose output does not need to be transformed by subsequent layers to produce attribute a. Removing a layer from the end of the path amounts to overriding its output with that of the previous layer. This gives rise to an operation we call *lock*:

$$lock(O^{(e,\ell)}) := O^{(e,\ell+k)} \underset{patch}{\leftarrow} O^{(e,\ell)} \quad \forall k > 0$$

This is an instance of activation patching (Vig et al., 2020; Wang et al., 2022; Meng et al., 2022) which

[2]Note that "minimal" refers to the irreducibility of the path rather than its length: multiple distinct minimal paths of different lengths may exist. Indeed, we show that this is often the case in practice.

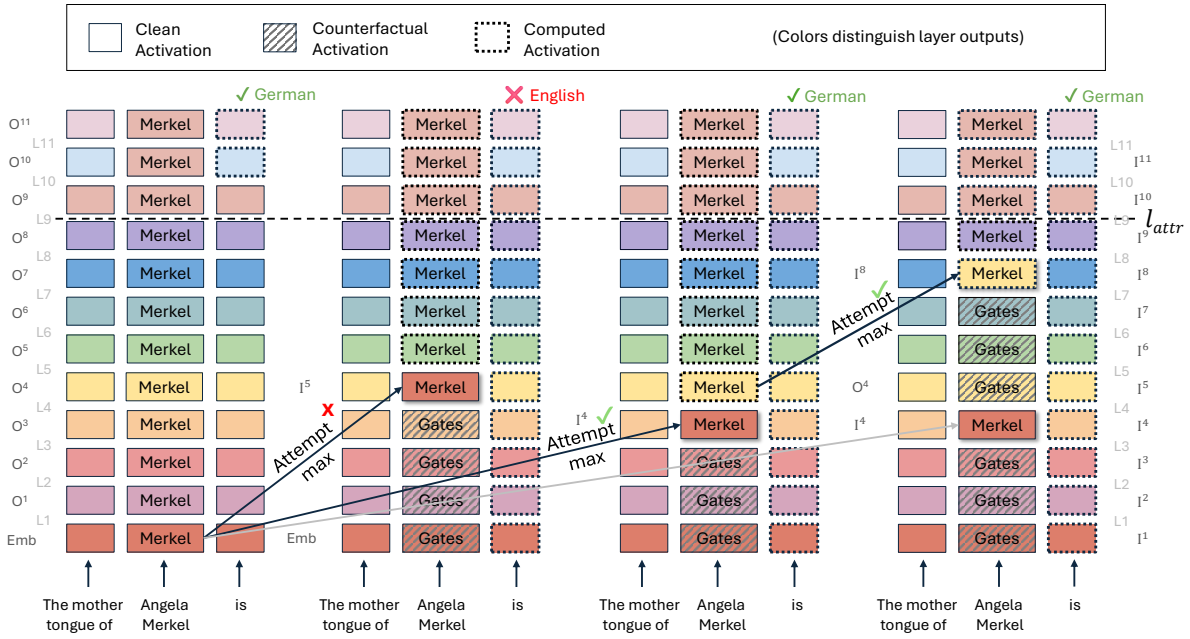


Figure 2: **Iterative Greedy Search for Minimal Computation Paths.** We illustrate the process using the original prompt *"The mother tongue of Angela Merkel is"* (Target: *"German"*) and the counterfactual prompt *"The mother tongue of Bill Gates is"* (Target: *"English"*). First, L_9 is established as the attribute sufficient layer using the *lock* operation. Then, **Failed Attempt (Left Arrow):** The algorithm attempts a maximal jump from the embeddings to layer 5, but the model predicts the counterfactual target, marking this skip as invalid. **Successful Attempt (Middle Arrow):** The algorithm identifies layer 4 as the highest layer into which a jump recovers the correct target. **Next Iteration (Right Arrow):** With layer 4 established as a necessary step, treating its output as the new source, the search repeats to identify the next maximal jump (to layer 8).

we call *activation locking*: patching the activations from an entity representation at layer ℓ to all higher layers of the same entity.

For a given prompt P, we seek the earliest layer on which we can apply lock operation and retain the same correct attribute a:

$$\ell_{attr} = \min_{\ell} \text{s.t. } M\big(P \mid lock(O^{(e,\ell)})\big) = M(P)$$

Fig 4 (Appendix A) illustrates this process.

5.2 Minimal Path Identification

Having identified ℓ_{attr}, we seek to identify a path (a sequence of layers) $[L_{\alpha_1}, \ldots, L_{\alpha_k}]$, $\alpha_i < \alpha_{i+1}$; $\alpha_k \leq \ell_{attr}$, which forms a subset of the model's layers constrained by the upper bound ℓ_{attr}. This sequence represents a minimal set of layers such that, when the entity embedding is processed only through these layers, the model M still produces the desired output a.

To identify non-essential layers, we use counterfactual prompts: prompts of the form $(\hat{e}, r) \rightarrow \hat{a}$ where $\hat{e}$ shares the same semantic type, number of tokens and sequence positions with e, but for which M produces an answer $\hat{a} \neq a$. We propose a heuristic: if a layer can process counterfactual entity data without disrupting the final attribute prediction, this layer is not necessary for the computation. Thus, given counterfactual entity activations $O^{(e,\ell)}_{\text{counter}}$ from running M on the counterfactual prompt, and a candidate path, we define an activation patching operations called *isolate* in which: (a) the first path layer L_{α_1} receives the clean entity embedding $E^{(e)}_{\text{clean}}$; (b) each subsequent path layer L_{α_i} receives the output of the previous path layer $L_{\alpha_{i-1}}$; and (c) all layers between path layers receive the corresponding activations from the counterfactual run. Formally:

$$isolate(layers = [\alpha_1, ..., \alpha_k]) :=$$

$$:= \begin{cases} I^{(e,\ell=\alpha_1)} \underset{patch}{\leftarrow} E^{(e)}_{\text{clean}} & \ell = \alpha_1 \\ I^{(e,\ell=\alpha_i)} \underset{patch}{\leftarrow} O^{(e,\alpha_{i-1})}_{\text{computed}} & \ell = \alpha_i \in \text{layers} \\ I^{(e,\ell)} \underset{patch}{\leftarrow} I^{(e,\ell)}_{\text{counter}} & \ell \notin \text{layers}, \ell < \alpha_k \end{cases}$$

where O_{computed} are the non-patched values computed in the current run. Steps (a) and (b) ensure that only the layers within the candidate path con-

tribute to the attribute computation over the entity tokens. Meanwhile, step (c) ensures that any information propagating to subsequent positions from the bypassed layers is counterfactual, demonstrating that these intervening layers are not necessary for the factual information flow. The rightmost column in Figure 2 visualizes this operation for *isolate[4,8]*: the path is isolated by patching the output of each path layer to the input of the next path layer (e.g., patching $E^{(e)}_{\text{clean}}$ into $I^{(e,4)}$ and $O^{(e,4)}_{\text{computed}}$ into $I^{(e,8)}$), while intermediate layers are patched with counterfactual values.

Given a prompt $P_{(e,r)\to a}$ and ℓ_{attr} we say that a path prefix $[\alpha_1, ..., \alpha_k]$, $\alpha_k \leq \ell_{attr}$ is a *valid* prefix of a *computation path* iff:

$$M\Big(P \mid isolate([\alpha_1, ..., \alpha_k])\Big) = M\Big(P \mid isolate([\alpha_1, ..., \alpha_k]), lock(O^{(e,\ell_{attr})})\Big) = M(P) = a$$

We construct a minimal computational path via an iterative greedy search starting from the entity embedding[3]. If the embedding layer itself is sufficient (i.e., locking $E^{(e)}$ yields a), the path is empty. Otherwise, We start with an empty prefix (feeding the clean embedding through all the layers) and at each step, we extend the prefix by skipping as many layers as possible after its last layer while remaining a valid prefix. This strategy maximizes the number of skipped layers at every iteration, ensuring the final sequence is minimal. Formally, we define the extension operation:

$$\begin{aligned} &\text{extend}(\text{path}) := \text{path} \oplus \text{next}(\text{path}) \\ &\text{next}([\alpha_1, \ldots, \alpha_{k-1}]) := \\ &\max_{\alpha_{k-1}<\alpha_k\leq\ell_{attr}} \text{s.t. } ([\alpha_1, \ldots, \alpha_{k-1}, \alpha_k] \text{ is valid}) \end{aligned}$$

and repeatedly apply *extend*, starting from an empty path, until either the last path layer $\alpha_k = \ell_{attr}$ or $M(P|lock(O^{(e,\alpha_k)})) = MP(P) = a$.[4] The process is illustrated in Figure 2.

[3]In our analysis (e.g., when calculating path length), we consider the entity embedding E as the implicit start of all paths.

[4]We allow early stopping when $\alpha_k < \ell_{attr}$ as skipping intermediate layers may allow the path to reach sufficiency earlier than the initial upper bound. In such cases, we redefine $\ell_{attr} = \alpha_k$ for later analyses.

5.3 Alternative Minimal Path

The procedure in subsection 5.2 searches for a minimal path that reaches the sufficient representation in ℓ_{attr}. What if we relax this restriction, and allow the search procedure to construct paths that end after layer ℓ_{attr}? Under this definition a path prefix is valid if $M(P \mid isolate([\alpha_1, ..., \alpha_k])) = M(P) = a$ and we stop the prefix extension process when locking the last prefix layer yields the correct attribute.

This procedure reveals layers capable of executing the attribute computation even when the information integration occurs at a different depth than in the original forward pass.

6 Experiments and Results

Models. we experiment with two decoder-only transformer models: LLaMA 3.1 8B (32 layers) (Dubey et al., 2024) and Qwen3 8B (36 layers) (Yang et al., 2025).

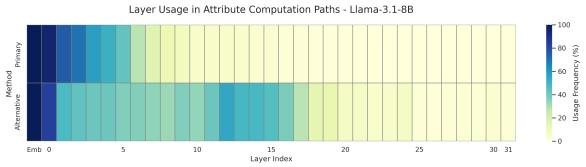
Data We use the CounterFact dataset (Meng et al., 2022), which provides prompts in the format described above, each paired with the correct target attribute. For each model, we select 2,000 prompts for which the target attribute is a single token and this token is correctly predicted by the model.

6.1 Main Experiments

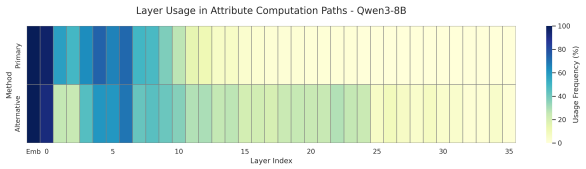
We applied our path identification method to the dataset to extract the causal computation paths for all entity–attribute pairs. We implemented the interventions using the `NNsight` package (Fiotto-Kaufman et al., 2025).

When is attribute knowledge available? For both models, using the *lock* operation, we find that the attribute computations complete in the early-to-mid layers, and further computation at the entity tokens is not required to produce the correct answer. The average ℓ_{attr} is 4.61 for LLaMA and 7.97 for Qwen, although for some cases we reach layer 15 and even 20 for both models (See Appendix B Figure 5(a) for the complete distribution). This aligns with previous studies on factual recall (Geva et al., 2023; Yu et al., 2025) that also place the conclusion of entity enrichment (for all attributes) at these early-to-mid layers.

How is attribute knowledge constructed? We now turn to examine the paths through which we arrive at the final representations ℓ_{attr}. The overwhelming majority of paths lengths (the number



(a) LLaMA 3.1 8B



(b) Qwen3 8B

Figure 3: **Layer Usage Patterns.** Aggregated layer utilization frequency for Primary (top row in each panel) vs. Alternative (bottom row) computation paths.

of layers in a path, including the embedding layer) are > 2, with an average of 5.91 (LLaMA) and 7.97 (Qwen). However, many paths (33.1% of the LLaMA cases and 78.6% of Qwen's) skip at least one layer, with an average skip size of 0.7 for LLaMA and 2.0 for Qwen (Appendix B Figure 6). For both models, the entities need to be processed by several (but not all) model layers in order for the knowledge to be available to use. The overall number of needed layers is similar for both models, but the Qwen paths are longer and sparser on average.

Existence of alternate paths. The alternative path search confirms that models contain multiple, functionally equivalent circuits for factual computation, identifying non-identical alternative paths in 80.1% (LLaMA) and 82.7% (Qwen) of cases. Quantitative analysis (Appendix B, Figure 7) reveals that alternative paths are universally longer (mean length: 9.47 for LLaMA, 10.47 for Qwen), relying on massive non-sequential jumps (mean skip size: 6.46 layers for LLaMA, 10.4 for Qwen).

In addition, ℓ_{attr} shifts significantly deeper compared to the primary path—averaging 13.93 for LLaMA and 18.06 for Qwen (Appendix B, Figure 5). Layer usage heatmaps aggregated across paths (Figure 3) further confirm that these paths recruit a distinct, deeper set of modules, effectively bypassing the primary processing centers. Thus, while redundant paths exist, the primary path consistently offers the most compact route to attribute sufficiency.

Are the constructed representations sufficient? To characterize the functional role of ℓ_{attr} output ($O^{(e,\ell_{attr})}$), we performed a suite of four targeted patching interventions:

1. **Representation Knockout:** We execute the minimal path up to ℓ_{attr}, then immediately overwrite the entity representation with a counterfactual one. This tests if the constructed representation in layers above ℓ_{attr} is necessary for prediction.

2. **Downstream Injection:** We inject the representation from ℓ_{attr} into all subsequent layers inputs ($l > \ell_{attr}$) while making all previous layers counterfactual. This tests if the final state alone, without the path's history, is sufficient to drive the upper model layers.

3. **Path + Continuation:** We inject the representation from ℓ_{attr} into all path layers and to all higher layers ($l > \ell_{attr}$), leaving only off-path intermediate layers as counterfactual.

4. **Global Broadcast:** We inject the representation from ℓ_{attr} into *all* model layers.

The results of these experiments, detailed in Table 1, yield two insights. First, the representation at ℓ_{attr} is strictly necessary for factual recall; in the Representation Knockout setting, overriding it and following layers with a counterfactual state disrupted the correct prediction in $> 94\%$ of cases across both models. Second, we find that the representation constructed at ℓ_{attr} on its own is often not sufficient: while Downstream Injection restores the desired attribute in almost 56% of primary paths in LLaMA, it fails to do so in the remaining 44%, for which activations at intermediate path layers are also required. This indicates that the computation path often acts as a coherent functional unit, where earlier entity representations are used for the final prediction.

The Roles of Path Layers. To further investigate the functional role of intermediate representations, we explicitly consider the two ways in which path layers can be utilized. A path layer can either act as an intermediate transformation—producing representations that will be fed to subsequent layers at the same token position—or it can enable inter-token transfer, moving information about the entity from the layer's input to subsequent sequence

	LLaMA 3.1 8B		Qwen3 8B	
Intervention Type	**Primary**	**Alternative**	**Primary**	**Alternative**
Representation Knockout	99.50%	94.95%	99.40%	94.35%
Downstream Inject	44.50%	76.60%	56.05%	76.85%
Path + Cont.	34.45%	48.25%	53.20%	64.55%
Global Broadcast	37.00%	61.95%	62.65%	78.65%

Table 1: **Prediction failure rates** across functional analysis experiments. All values indicate the percentage of cases where the model failed to output the correct attribute.

positions via the attention mechanism. To determine which function a layer serves, we test each identified path (both primary and alternative) using an extended *isolate* operation. Iteratively, we feed each path layer counterfactual *inputs* while restoring its entity-position *outputs* to their clean-path states. If the final prediction remains correct, the layer is solely performing same-token representation transformation, rather than propagating necessary information to later sequence positions. For these layers, we leave this transformation-only restriction in place while testing higher layers. We find that reliance on this propagation correlates strongly with path length: primary paths require at least one layer to act as an inter-token propagator in 50.1% of cases (average path length 7.4, vs. 4.4 when not required), while alternative paths require it in 80.6% of cases (average length 10.8, vs. 3.9). Similar patterns are observed in Qwen3 8B. For primary paths, 63.8% require information propagation (average path length 8.64, vs. 6.78 when not required). In alternative paths, this dependency increases to 82.4% (average path length 11.11, vs. 7.49). This indicates that longer, non-standard computation paths depend more heavily on information propagation across the sequence. Notably, approximately 15% of all paths across both Llama and Qwen require clean information propagation as early as Layer 0. This is unexpected under our hypothesis that the specific attribute would not yet be computed at this initial stage. Detailed layer-wise usage for information propagation is visualized in the heatmaps in Appendix C.

Constructing additional paths via recombination. We test the viability of "hybrid" paths composed of early steps from a primary path and later steps from an alternative path, by concatenating a prefix from a primary path with a valid suffix from the corresponding alternative path, enforcing that the resulting path is distinct from the originals and non-trivial (the chosen prefix from the primary does not contain the corresponding initial segment of the alternative). Each hybrid is evaluated using our counterfactual patching method (subsection 5.2); success is defined as restoring the target prediction and achieving attribute sufficiency at the final layer.

The results reveal significant modular flexibility. In LLaMA 3.1 8B, 35.4% of prompts admit at least one successful hybrid path (avg. 2.75 successful combinations per prompt), while Qwen3 8B shows a 32.8% success rate with higher redundancy (avg. 4.13 combinations). This proves the existence of additional computation paths beyond those identified in previous experiments, demonstrating that different layer sequences can perform functionally equivalent transformations. It also indicates that layer roles in the primary and alternate paths do not necessarily map one-to-one: in some cases a single layer in the primary path is accounted for by several layers in the alternate path, or vice versa.

6.2 Additional Experiments and Analyses

Effect of order. While in most prompts the relation appears before the entity (*"The mother tongue of X is"*), in few of them the order is reversed (*"X's mother tongue is"*). Does this affect the retrieval process? We could not identify a strong trend, beyond the paths concluding on later layers in LLaMa (but not in Qwen). However, this could be due to small sample size, and could be the topic of a future study. Further details are available in Appendix D.

Paths in Individual Relations. We analyzed relation types with sufficient coverage ($N \geq 50$). While not a universal trend, we observe anecdotal examples suggesting a potential link between semantic specificity and computational cost (Appendix E). In certain instances, broad categorical relations (e.g., *Continent*) require fewer layers than more specific factual retrievals (e.g., *Headquarters*), hinting that the model might occasionally recruit longer circuits to resolve highly specific

attributes. Additionally, per-relation layer-usage heatmaps in Appendix E illustrate that distinct relations recruit specific sets of layers beyond simple path length differences. However, this length hierarchy is highly inconsistent: we observe significant rank discordance between models and between primary and alternative paths. The fact that relative difficulty shifts in the alternative setting implies that "backup" mechanisms utilize distinct processing logic rather than simply mirroring primary circuits.

Relation to Entity Resolution. Entity resolution (ER) refers to the stage in which a model forms an internal entity representation sufficient to explicitly reconstruct the entity's name within a suitable target context (Ghandeharioun et al., 2024). To examine the relationship between ER and attribute computation, we apply the Patchscopes method (Ghandeharioun et al., 2024) to *multi-token* entities along our identified paths ($N_{LLaMA} = 1,972$, $N_{Qwen} = 1,945$). We use the original constant prompt from this method, of the form "$e_1 : d_1$,..., $e_k : d_k$, x" where each description d_i begins with the entity name e_i. We extract the entity's final token representation from every layer along its computation path (for both primary and alternative paths) and inject it into the x position across all target layers. We then identify the first source–target layer pair, if any, that decodes the full multi-token entity name.

The results (Appendix F) reveal a strong dissociation between ER and attribute sufficiency. Explicit ER steps are detected in a minority of primary paths (12.9% for LLaMA, 9.1% for Qwen) and even fewer alternative paths (9.1% and 6.6%). This implies that detectable ER is not a universal prerequisite for factual recall; while primary paths retain slightly stronger identity traces, attribute retrieval often occurs implicitly without a distinguishable "identity state" accessible to Patchscopes.

Impact of Counterfactual Noise on Attribute Computation. To test whether counterfactual noise in the *isolate* operation artificially alters identified paths—due to incorrect information propagating through non-path layers—we additionally experiment with a modified algorithm: instead of injecting counterfactual activations into excluded layers, we supply them with the most recent preceding path layer's input (effectively "locking" the representation). Under this setting, average path lengths remain highly consistent with the original operation across both LLaMA and Qwen (noise-free vs. original — LLaMA primary: 5.88 vs. 5.91, alternative: 8.99 vs. 9.47; Qwen primary: 7.92 vs. 7.97, alternative: 10.69 vs. 10.47). Primary path termination depth (mean ℓ_{attr}) is also virtually identical for both models (LLaMA: 4.65 vs. 4.61; Qwen: 8.02 vs. 7.97)—an expected outcome since ℓ_{attr} serves as the explicit search bound. However, while alternative path lengths remain stable, their computation terminates deeper in this locked setting (LLaMA: mean layer 17.73 vs. 13.93; Qwen: 20.87 vs. 18.06). Layer usage heatmaps (Appendix G) confirm this shift is driven by increased utilization of the deepest layers in both models; LLaMA's usage of layers 27–30 jumps from 1–3% to 9–15%, and Qwen exhibits a similar late-stage spike, peaking at 22% in its penultimate layer (layer 34). We hypothesize that counterfactual noise "forces" the model to compute the attribute earlier to override conflicting signals; without this noise, our greedy search naturally defers computation. Overall, while the *amount* of required computation (path length) is an intrinsic property of the task, the *depth* of this extraction is highly sensitive to counterfactual noise.

7 Discussion and Conclusion

We introduced the concept of an Attribute-Computation Path and a novel patching protocol to identify such paths in LLMs. Our analysis confirmed that the computation of an attribute is a multi-step, non-contiguous process that frequently skips layers and requires multiple necessary steps. Furthermore, we show that models possess multiple, functionally equivalent computation paths for the same fact, emphasizing a high degree of computational redundancy.

Our findings suggests that the dominant folk view of factual storage and retrieval, in which facts are stored in specific MLP layers, is simplistic and misleading: the reality appears to be significantly more complex, with knowledge being stored in a distributed way across multiple layers, that needs to be processed together for an effective recall of an individual fact.

Limitations

While we showed redundancy in minimal attribute computation paths exists, we did not fully quantify it. Our estimation of model redundancy is constrained by our search strategy in two primary

ways. First, our reliance on an iterative greedy search algorithm—rather than an exhaustive combinatorial search—means that we naturally may miss many valid computational paths that might arise from different layer combinations. Second, we explicitly targeted two distinct path types by defining specific upper bounds for this search (ℓ_{attr} and the final model layer). Consequently, our findings likely represent only *a lower bound* on the true degree of redundancy; it is plausible that a granular sweep of search bounds across all intermediate layers would reveal a much larger spectrum of functionally equivalent paths that our current analysis did not explore.

Second, while our interventions demonstrate that these redundant paths can be mechanistically accessed in an intervention, it remains unclear whether the model utilizes these deeper layers for entity enrichment during a standard, unperturbed forward pass, or if they represent degenerate remnants of the training process.

References

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, and 8 others. 2025. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*.

Bilal Chughtai, Alan Cooney, and Neel Nanda. 2024. Summing up the facts: Additive mechanisms behind factual recall in llms. *arXiv preprint arXiv:2402.07321*.

Roi Cohen, Mor Geva, Jonathan Berant, and Amir Globerson. 2023. Crawling the internal knowledge-base of language models. *arXiv preprint arXiv:2301.12810*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

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, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Jaden Fried Fiotto-Kaufman, Alexander Russell Loftus, Eric Todd, Jannik Brinkmann, Koyena Pal, Dmitrii Troitskii, Michael Ripa, Adam Belfki, Can Rager, Caden Juang, Aaron Mueller, Samuel Marks, Arnab Sen Sharma, Francesca Lucchetti, Nikhil Prakash, Carla E. Brodley, Arjun Guha, Jonathan Bell, Byron C Wallace, and David Bau. 2025. NNsight and NDIF: Democratizing access to foundation model internals. In *The Thirteenth International Conference on Learning Representations*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 30–45.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. *arXiv preprint arXiv:2401.06102*.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. 2023. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*.

Daniela Gottesman and Mor Geva. 2024. Estimating knowledge in large language models without generating a single token. *arXiv preprint arXiv:2406.12673*.

Wes Gurnee and Max Tegmark. 2023. Language models represent space and time. *arXiv preprint arXiv:2310.02207*.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36:17643–17668.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2023. Linearity of relation decoding in transformer language models. *arXiv preprint arXiv:2308.09124*.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Shahar Katz, Yonatan Belinkov, Mor Geva, and Lior Wolf. 2024. Backward lens: Projecting language model gradients into the vocabulary space. *arXiv preprint arXiv:2402.12865*.

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Neel Nanda, Senthooran Rajamanoharan, János Kramár, and Rohin Shah. 2023. Fact finding: Attempting to reverse-engineer factual recall on the neuron level. AI Alignment Forum.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*. Https://distill.pub/2020/circuits/zoom-in.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

Zijian Wang and Chang Xu. 2025. Functional abstraction of knowledge recall in large language models. *arXiv preprint arXiv:2504.14496*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024. Knowledge circuits in pretrained transformers. *Advances in Neural Information Processing Systems*, 37:118571–118602.

Zeping Yu and Sophia Ananiadou. 2024. Neuron-level knowledge attribution in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3267–3280.

Zeping Yu, Yonatan Belinkov, and Sophia Ananiadou. 2025. Back attention: Understanding and enhancing multi-hop reasoning in large language models. *arXiv preprint arXiv:2502.10835*.

A Visualization of ℓ_{attr} Identification

Figure 4 provides a detailed visualization of the ℓ_{attr} identification process, using the prompt *"The mother tongue of Angela Merkel is"* (Target: *"German"*). The figure illustrates this step-by-step procedure in which we lock the entity representation at each layer—starting from the embeddings and moving upwards—until the operation successfully reproduces the correct target token.

B Extended Quantitative Analysis

In this section, we provide the detailed statistical breakdown of the identified paths. Figure 5 illustrates the distribution of the ℓ_{attr} for both the primary and alternative strategies, as well as the layers skipping ratio in the paths. Figure 7 offers a direct comparison between the primary and alternative paths, quantifying the increase in depth and path length when the model is forced to utilize later layers for attribute computation.

C Necessary Information Propagation

In this section, we provide a detailed breakdown of layer usage across the computation paths identified in our experiments. We compare the Original Path distribution (where all layers are active) with the Necessary Attention distribution. The latter represents only the layers where intra-layer information propagation was strictly required to maintain the correct prediction.

The heatmaps in Figure 8 illustrate the frequency (as a percentage of total paths) with which each layer is utilized. We highlight two key observations:

Correlation with Depth: Consistent with our main findings, longer alternative paths show a higher density of necessary attention layers in the middle and late stages of the model.

The Layer 0 Surprise: Across both Llama and Qwen, approximately 15% of paths require clean information propagation at Layer 0. This contradicts the initial hypothesis that attribute-specific information is only computed in later blocks.

D Analysis by Prompt Structure (Entity Position)

Here, we provide detailed results of the experiments. First, we classified each prompt as one of the following types:

- **entity_first**: The entity appears before the relation (e.g., "Albert Einstein was born in...").
- **relation_first**: The relation appears before the entity (e.g., "The birthplace of Albert Einstein is...").

To perform this classification, we used GPT-4o. We provided the model with the formatted prompt (e.g., "The official religion of Edwin of Northumbria is") and requested a classification using the few-shot prompt structure in Figure 9.

Figure 10 shows the comparison of ℓ_{attr} and path length between these two prompt types for the primary and the alternative paths. We restricted our comparison to relations that had at least 50 samples for both structure types together across both models. The number above each bar indicates the sample size used to calculate the average.

E Relation-Specific Path Analysis

In Section 6.2, we discussed how path length varies by relation. This section provides a comprehensive breakdown of these variations. First, Figure 11 illustrates the mean path length across all relations, sorted by the primary path length in LLaMA 3.1 8B. Following this overview, Figures 12 and 13 present the granular layer-usage distributions for these relations, comparing LLaMA 3.1 8B and Qwen3 8B.

F Entity Resolution Detection Results

In Section 6.2, we analyzed the relationship between the attribute computation path and the explicit resolution of the entity's identity.

To perform this analysis, we employed the Patchscopes method using the identical constant prompt configuration from Ghandeharioun et al., 2024. The specific few-shot prompt is:

> "Syria: Syria is a country in the Middle East, Leonardo DiCaprio: Leonardo DiCaprio is an American actor, Samsung: Samsung is a South Korean multinational corporation, x"

where x serves as the placeholder for the injected representation.

Figure 14 presents the quantitative results of this analysis. It details the success rate of detecting Entity Resolution (ER) along both the primary and alternative computation paths for LLaMA 3.1 8B and Qwen3 8B.

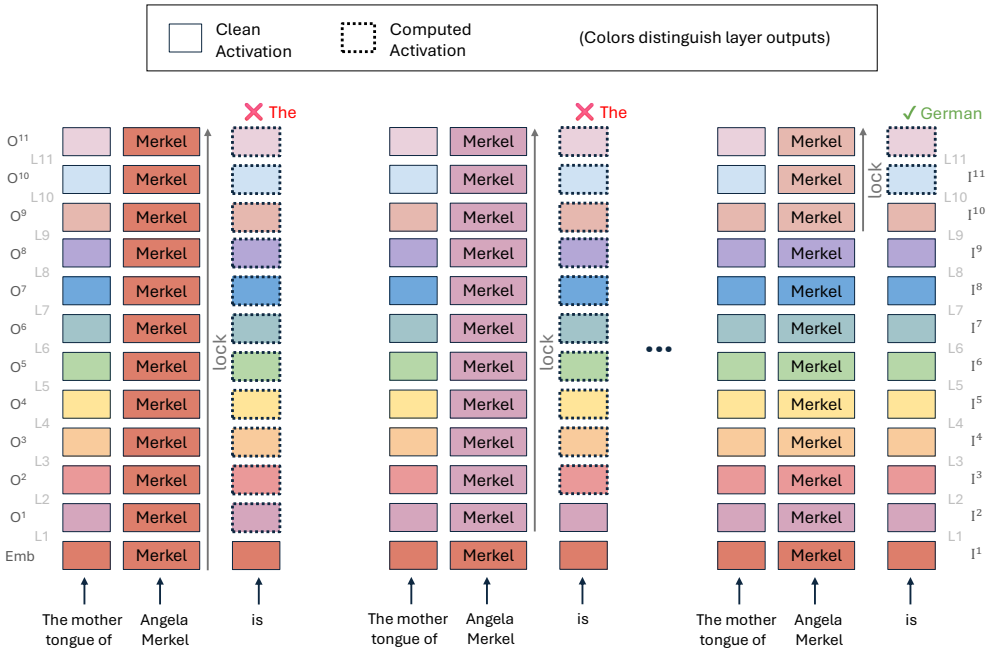
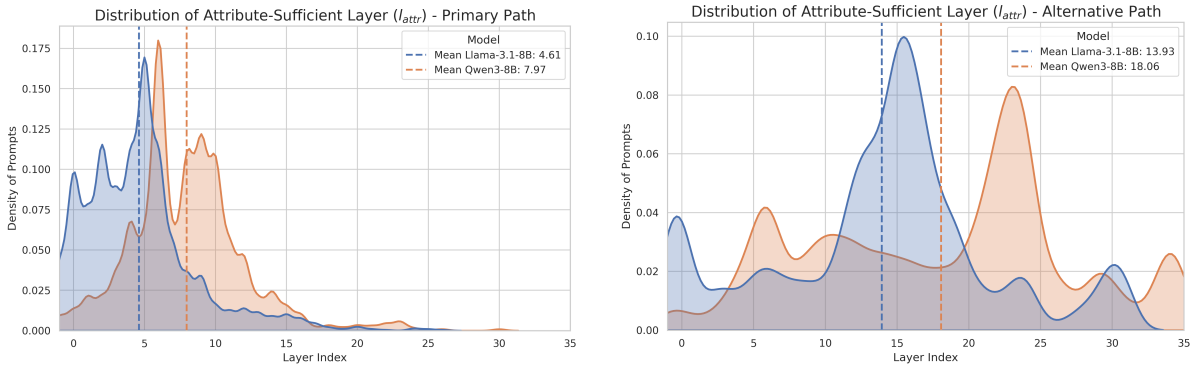


Figure 4: ℓ_{attr} **Identification.** This process identifies ℓ_{attr}, the first layer where the entity representation is robust enough to elicit the target attribute without further processing. The labeled squares represent the entity representation at different layers. **Left & Middle (Testing an Insufficient Layer):** We lock the "Angela Merkel" representation at an early layer. The model fails to retrieve the correct attribute, predicting a generic or nonsensical token (e.g., *"The"*), indicating the representation is not yet sufficient. **Right (Testing a Sufficient Layer):** We lock the representation at a deeper layer. The model successfully predicts *"German"*, identifying this layer as ℓ_{attr}.



(a) Distribution of ℓ_{attr} (primary method).

(b) Distribution of ℓ_{attr} (Alternative Path)

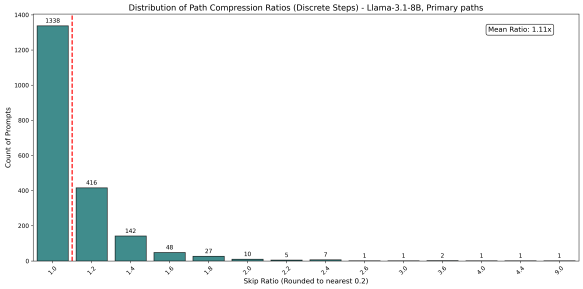
Figure 5: Analysis of minimal computation paths for the primary method. (a) The distribution of ℓ_{attr}, showing both models complete attribute computation in the early-to-mid layers. (b) The distribution of the final ℓ_{attr} for the alternative path, which is significantly deeper and wider than the primary path's distribution.

G Impact of Counterfactual Noise

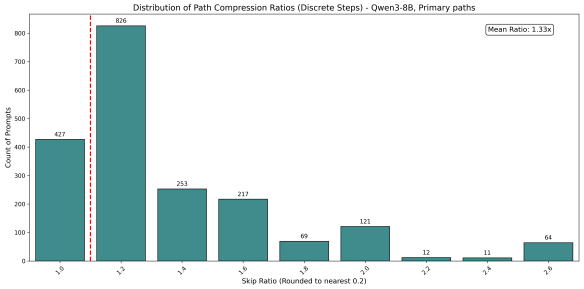
In this section, we provide the detailed layer-wise usage heatmaps for both LLaMA 3.1 8B and Qwen3 8B under the modified *isolate* operation, where excluded layers are supplied with the output of the most recent preceding path layer rather than counterfactual noise.

The heatmaps in Figure 15 illustrate the frequency (as a percentage) at which each specific layer is incorporated into the identified primary and alternative computation paths across the dataset.

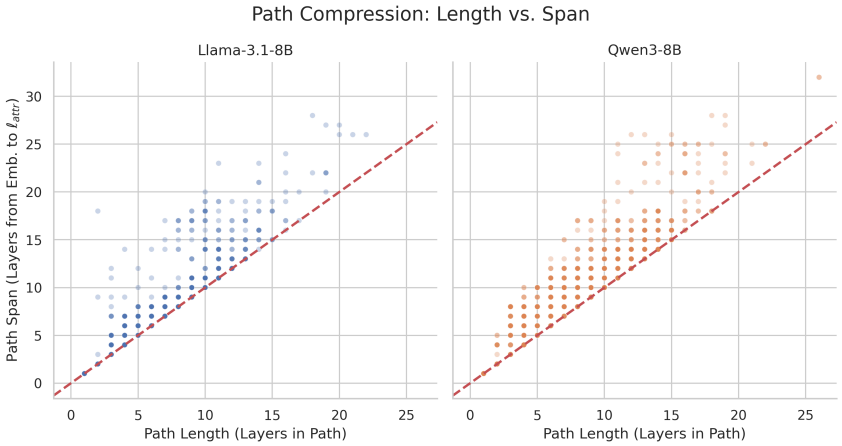
Consistent with the findings discussed in Section 6.2, these visualizations highlight a distinct structural shift in the alternative paths when counterfactual noise is removed. While primary path layer distributions remain relatively stable, the alternative paths in both models exhibit a distinct increase



(a) Distribution of Path Compression Ratios (LLaMA).



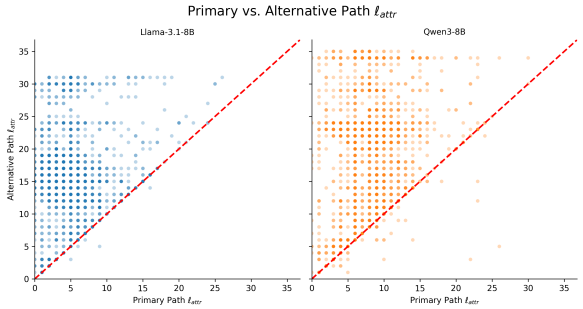
(b) Distribution of Path Compression Ratios (Qwen)



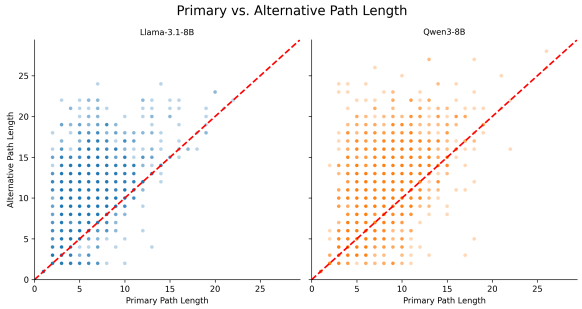
(c) Path length vs. path span (primary method).

Figure 6: Analysis of minimal computation paths for the primary method. (a) Distribution of Path Compression Ratios for LLaMA 3.1 8B. The ratio is computed as the path depth divided by the number of steps. The first bar (ratio=1.0) represents purely sequential paths. The red dashed line acts as a boundary, separating these sequential paths from the compressed paths (ratio $>$ 1.0) to the right. Ratios $>$ 1.0 are rounded to the nearest 0.2 for visualization. (b) The corresponding distribution for Qwen3 8B. (c) A comparison of actual path length vs. path span for the primary paths. Dots below the $y = x$ line represent "compressed" paths that skip layers.

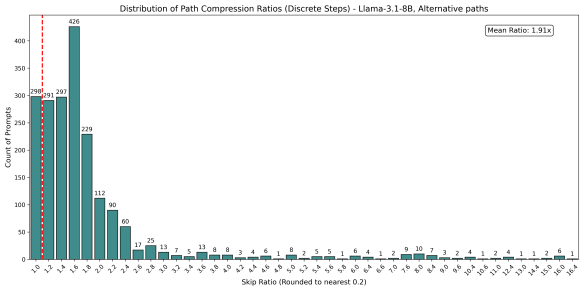
in the utilization of deeper layers. Specifically, LLaMA demonstrates a concentrated increase in utilization across layers 27–30, while Qwen exhibits a sharp spike at 22% in its penultimate layer (layer 34). These visual patterns corroborate the hypothesis that without the forced intervention of counterfactual noise, our greedy search naturally defers the attribute computation to much deeper layers.



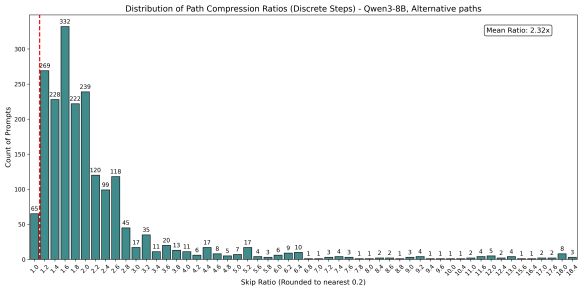
(a) ℓ_{attr}: Primary vs. Alternative Path



(b) Path Length: Primary vs. Alternative Path

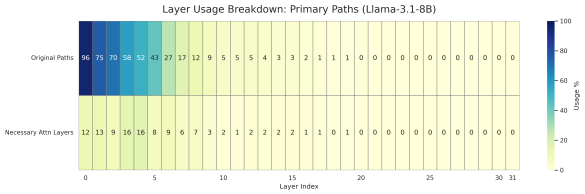


(c) Distribution of Path Compression Ratios (LLaMA).

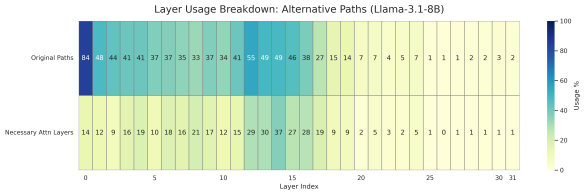


(d) Distribution of Path Compression Ratios (Qwen)

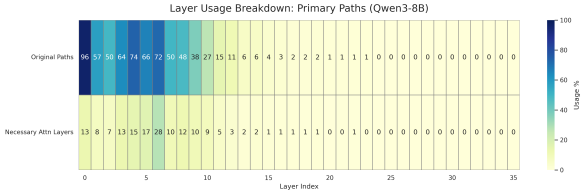
Figure 7: Analysis of the alternative computation path. (a) Comparison of ℓ_{attr} for the primary path (x-axis) vs. the alternative path (y-axis). Most points lie above the $y = x$ line, showing the alternative path is deeper. (b) Comparison of path length. Points are again mostly above the $y = x$ line, indicating the alternative path is also longer (more computational steps). (c) Distribution of Path Compression Ratios for LLaMA 3.1 8B. The ratio is computed as the path depth divided by the number of steps. The first bar (ratio=1.0) represents purely sequential paths. Ratios > 1.0 are rounded to the nearest 0.2 for visualization. (d) The corresponding distribution for Qwen3 8B.



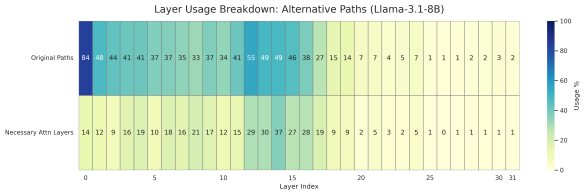
(a) LLaMA 3.1 8B - Primary Paths



(b) LLaMA 3.1 8B - Alternative Paths



(c) Qwen3 8B - Primary Paths

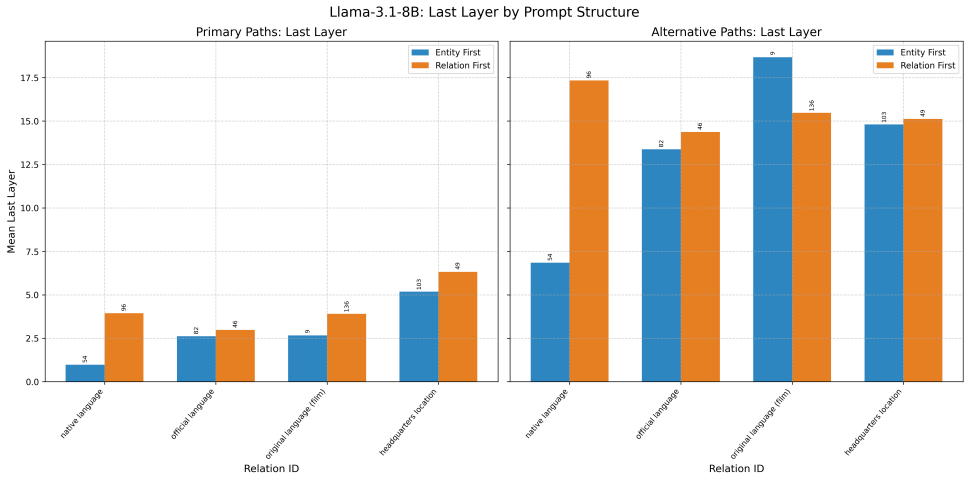


(d) Qwen3 8B - Alternative Paths

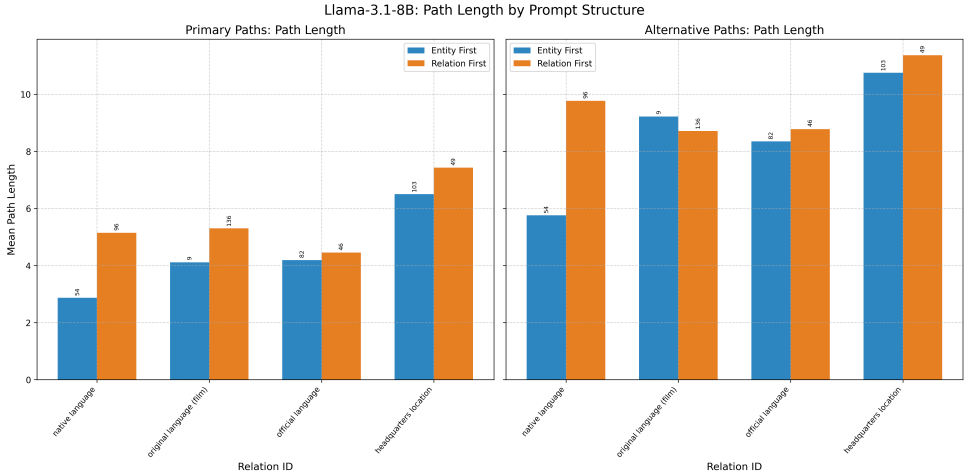
Figure 8: **Information Propagation Heatmaps.** Each plot compares the percentage of paths utilizing a specific layer in the original identified path (Top Row) versus the subset of layers where clean information propagation was found to be necessary (Bottom Row). Note the non-zero usage at Layer 0 across all configurations.

```
Example 1:
Sentence: The official religion of Edwin of Northumbria is
Answer: relation

Example 2:
Sentence: In Nykarleby, the language spoken is
Answer: entity

Now, analyze the following sentence.
What appears first: the relation or the entity?

Sentence: {prompt}
Answer with only 'relation' or 'entity'.
```

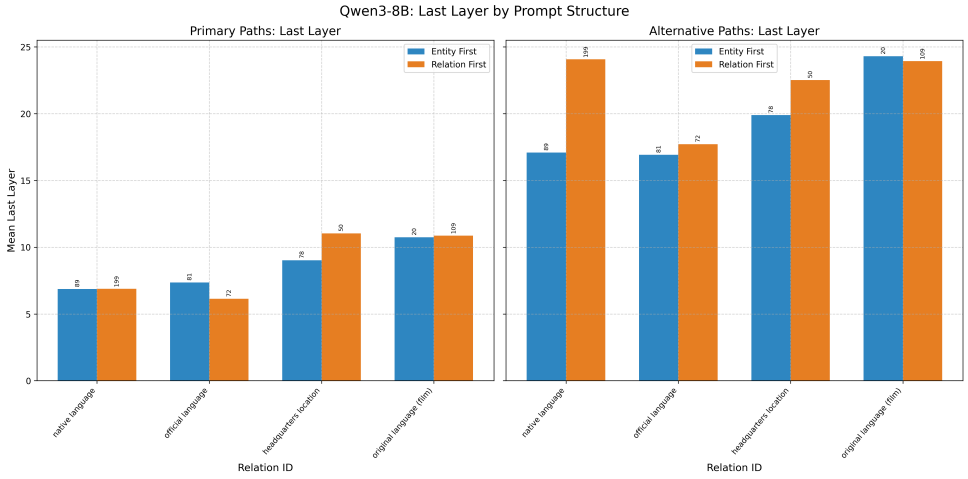
Figure 9: The few-shot prompt used to classify the ordering of entity and relation in the query.



(a) ℓ_{attr} vs. Entity Position - LLaMA 3.1 8B.

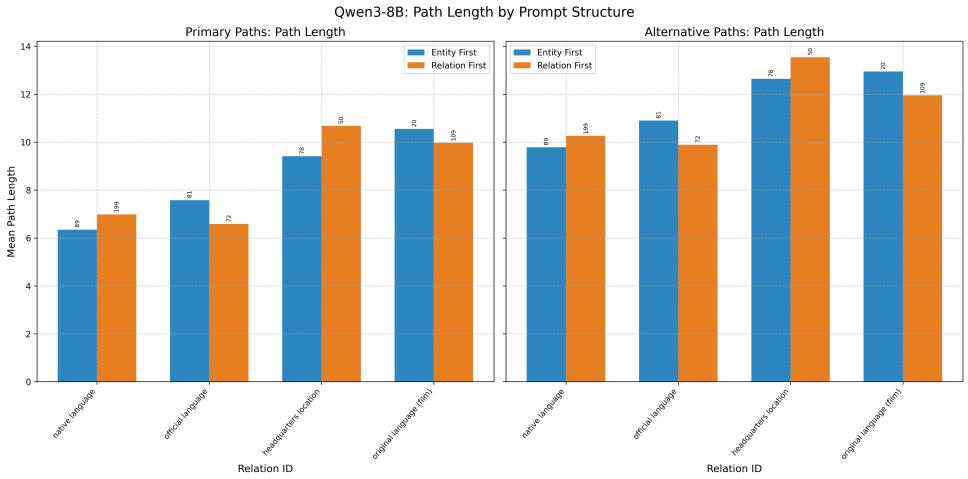


(b) Path Length vs. Entity Position - LLaMA 3.1 8B.



(c) ℓ_{attr} vs. Entity Position - Qwen3 8B.

Figure 10: Analysis of ℓ_{attr} and Path Lengths by Prompt Structure.



(a) Path Length vs. Entity Position - Qwen3 8B.

Figure 10: Analysis of ℓ_{attr} and Path Lengths by Prompt Structure. (continued)

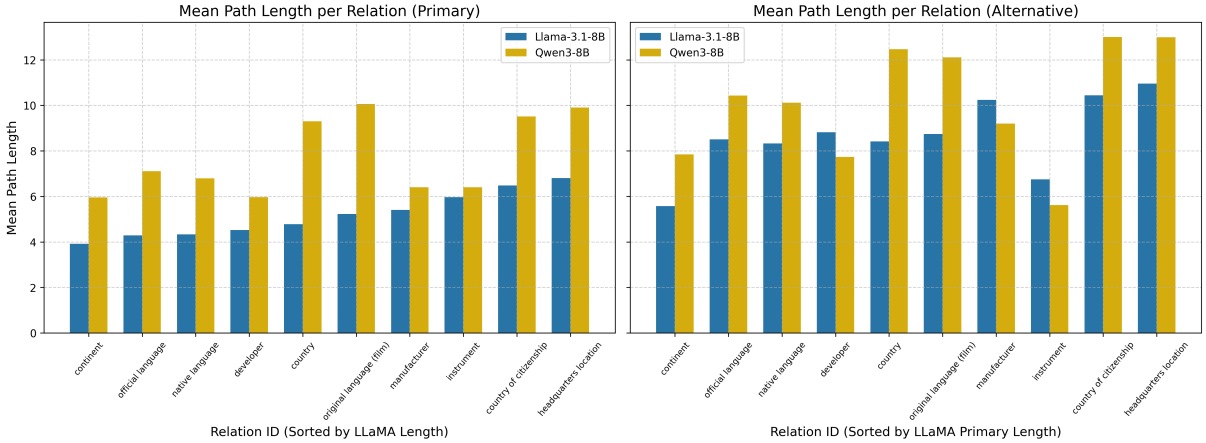


Figure 11: Mean path length per relation. Relations are sorted by LLaMA primary path length.

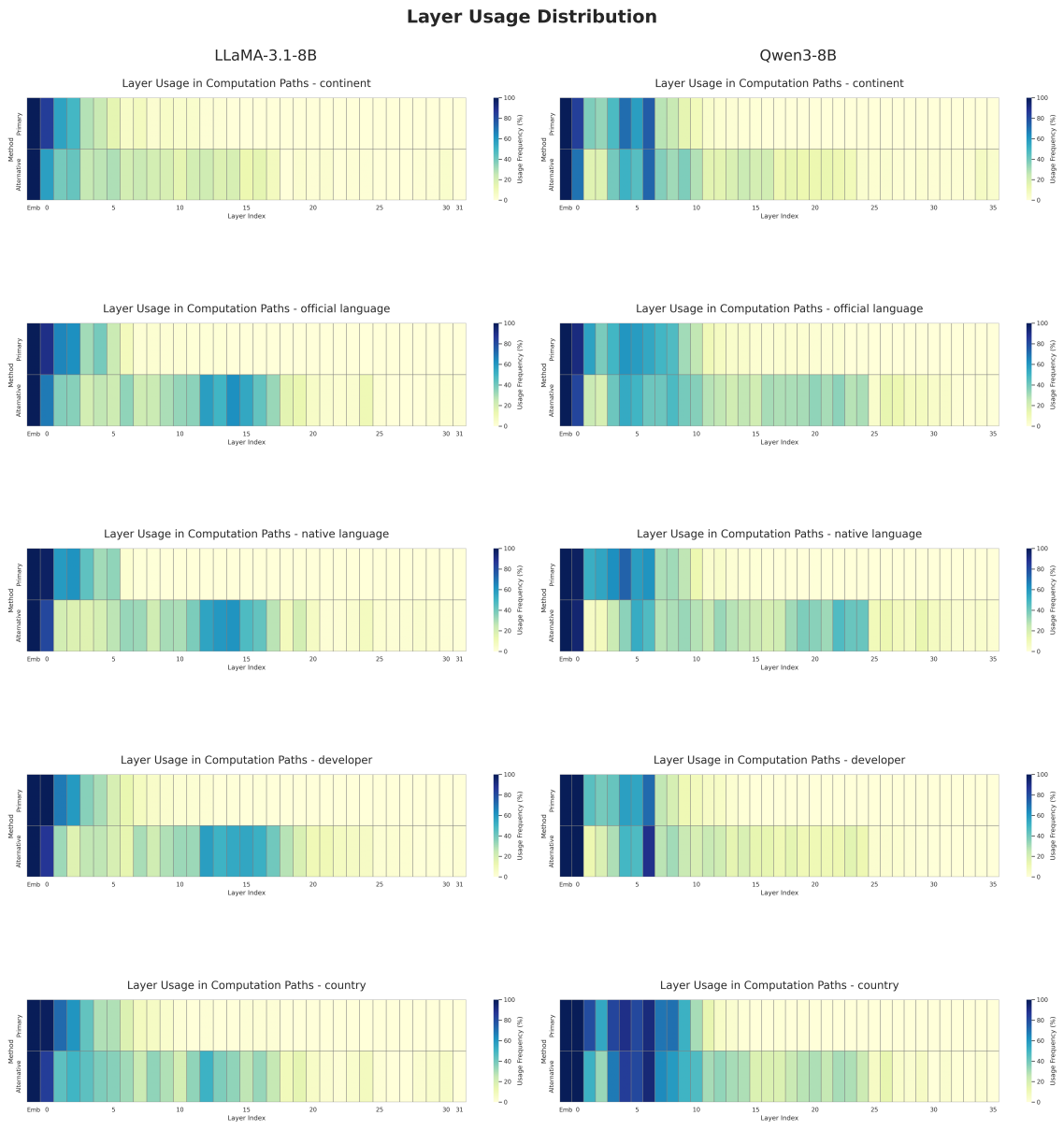


Figure 12: **Layer Usage per Relation (Part 1).** Comparison of Relations 1–5. LLaMA 3.1 8B (Left) vs. Qwen3 8B (Right). Warmer colors indicate higher usage probability.

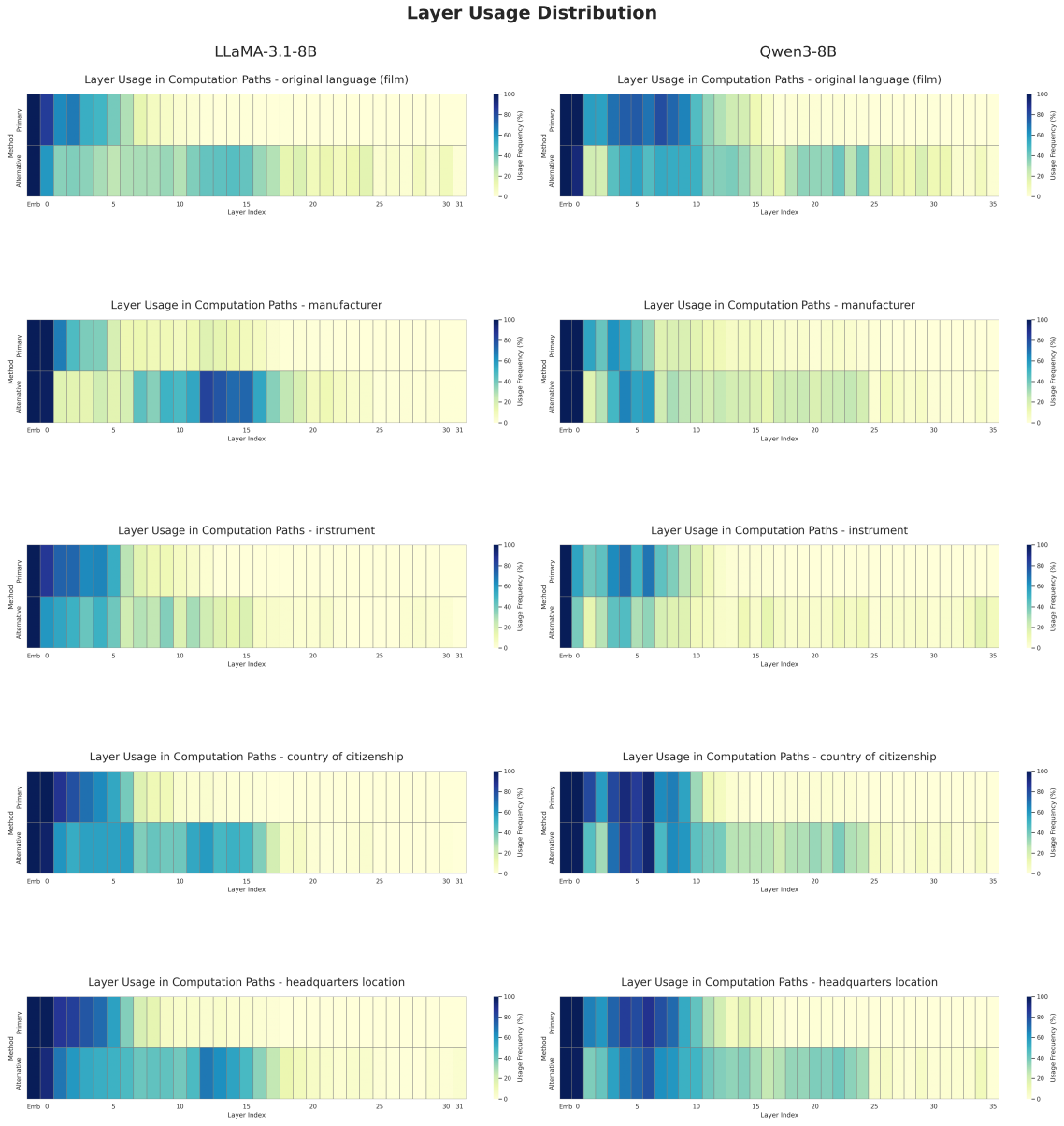


Figure 13: **Layer Usage per Relation (Part 2).** Comparison of Relations 6–10.

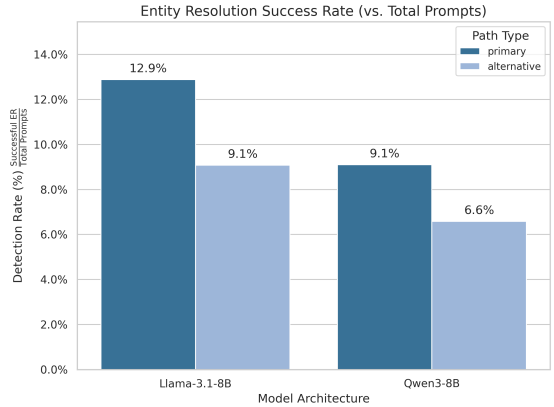
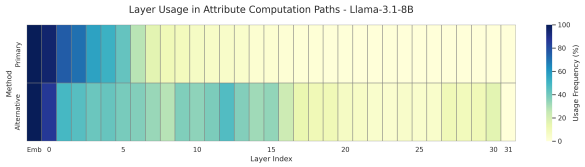
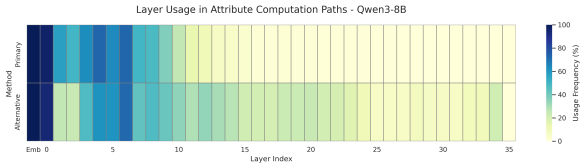


Figure 14: **Entity Resolution (ER) Detection Success Rate.** The figure shows the percentage of prompts where entity resolution was successfully detected along the minimal computation path (relative to the prompts analyzed for each method and model).



(a) LLaMA 3.1 8B



(b) Qwen3 8B

Figure 15: Layer usage heatmaps for primary and alternative paths in LLaMA 3.1 8B and Qwen3 8B under the noise-free setting.