LLMs Process Lists With General Filter Heads

Anonymous authors
Paper under double-blind review

Abstract

We investigate the mechanisms underlying a range of list-processing tasks in LLMs, and we find that they have learned to encode a compact, causal representation of a general filtering operation that mirrors the generic "filter" function of functional programming. Using causal mediation analysis on a diverse set of list-processing tasks, we find that a small number of attention heads, which we dub *filter heads*, encode a compact representation of the filtering predicate in their query states at certain tokens. We demonstrate that this predicate representation is general and portable: it can be extracted and reapplied to execute the same filtering operation on different collections, presented in different formats, languages, or even in tasks. However, we also identify situations where LMs can exploit a different strategy for filtering: eagerly evaluating if an item satisfies the predicate and storing this intermediate result as a flag directly in the item representations. Our results reveal that transformer LMs can develop human-interpretable implementations of abstract computational operations that generalize in ways that are surprisingly similar to strategies used in traditional functional programming patterns.

1 Introduction

When asked to *find the fruit* in a list, language models reveal a surprisingly systematic mechanism: they don't solve each filtering task anew, but instead encode predicates into portable representations. This neural representation of *"is this a fruit?"* can be extracted from one context and applied to a different list, presented in a different format, in a different language, and to some extent to a different task. These abstract, reusable operations suggest that transformers develop modular computational primitives rather than task-specific heuristics.

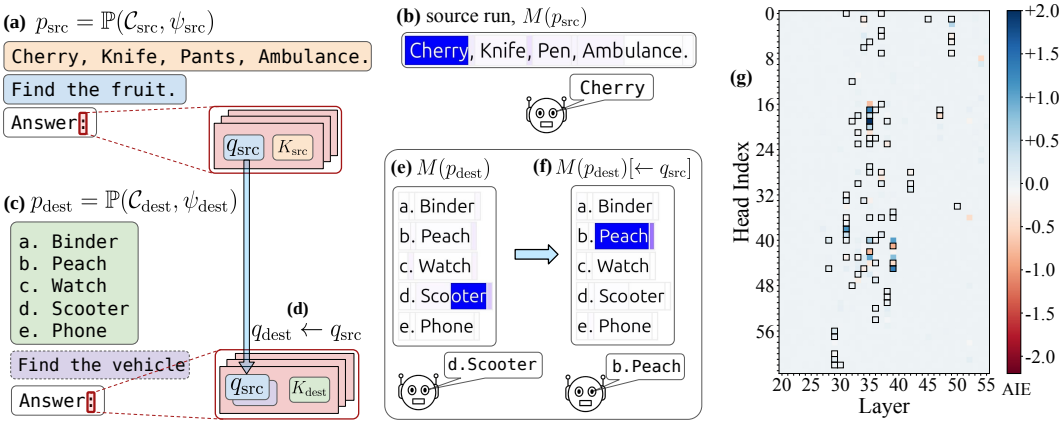


Figure 1: A filter head [35, 19] in Llama-70B encodes a compact representation of the predicate "is this fruit?" **(a)** Within a prompt p_{src} to find a fruit in a list, we examine the attention head's behavior at the last token ":" **(b)** The head focuses its attention on the one fruit in the list. **(c)** We examine the same attention head's behavior in a second prompt p_{dest} searching a different list for a vehicle **(d)** and we also examine the behavior of the head when patching its query state to use the q_{src} vector from the source context. **(e)** The head attends to the vehicle but then **(f)** redirects its attention to the fruit in the new list after the query vector is patched. **(g)** A sparse set of attention heads work together to conduct filtering over a wide range of predicates; these filter heads are concentrated in the middle layers (out of 80 layers in Llama-70B).

To understand this phenomenon systematically, we turn to Marr's three levels of analysis (Marr, 1982). At the *computational* level, we identify what is being computed: the selection of elements satisfying a predicate. At the *algorithmic* level, we reveal how this is achieved: through a three phase computation corresponding to a *map*, *filter*, and *reduce*, occurring in that order. The *map* step is equivalent to populating the latents of the items in a list with the right associations or semantic information, a step that is documented in prior literature (Geva et al., 2023; Meng et al., 2022). In this work we focus on the non-trivial computation step, *filter*, that follows after map. At the *implementation* level, we reveal how filtering is implemented in LMs: through specialized attention heads, which we dub *filter heads*, that encode predicates as geometric directions in query space. We find that these heads, concentrated in the middle layers of the LM, remain largely shared even as the specific predicate varies. This framework allows us to move beyond simply observing that models can filter, to understanding the explicit mechanisms through which list-processing operations emerge from the transformer architecture.

Our analysis yields three key insights:

Localized Mechanism. The list processing algorithm is implemented in a consistent set of localized components: a set of attention heads that we call filter heads. These heads encode a "compiled" representation of the predicate as query states at specific tokens — typically where the LM is required to produce its answer. These query states interact with the key states that carry semantic information of the list items, producing attention patterns that select the items satisfying the predicate.

Generalization. These filter heads are not specific to a single predicate, but can encode a distribution of predicates. And this encoding is sufficiently abstract that it can be extracted from one context and transported to another context to trigger the same filtering operation on a different collection of items, presented in a different format, in a different language, even in a different *reduce* task that follows after the filtering step.

Computational Redundancy. Additionally, our investigations reveal that LMs can perform filtering in two complementary ways: lazy evaluation via filter heads vs eager evaluation by storing *is_match* flags directly in the item latents. This dual implementation strategy mirrors the fundamental lazy/eager evaluation strategies in functional programming (Henderson & Morris Jr, 1976; Friedman et al., 1976). This second route reveals a broader principle in neural computations: transformer LMs can maintain multiple pathways for the same operation (McGrath et al., 2023; Wang et al., 2022) and can dynamically select between them based on what information is available.

We validate these findings through experiments across six different filter-reduce tasks of varying complexity, each requiring the LM to filter based on different information before performing a reduce step to provide a specific answer. We test the portability of the "compiled" predicate across different presentation format, language, and tasks. We conduct ablation studies to confirm the necessity of filter heads while performing filter operations. Finally, we demonstrate that the learned predicate representations can serve as zero-shot probes for concept detection, offering a training-free alternative to traditional linear probing methods.

2 METHOD

2.1 BACKGROUNDS AND NOTATIONS

Language Model. An autoregressive transformer language model, $M : \mathcal{X} \to \mathcal{Y}$ over a vocabulary $\mathcal{V}$, maps a sequence of tokens $x = \{x_1, x_2, \ldots, x_n \mid x \in \mathcal{V}\}$ to $y \in \mathbb{R}^{|\mathcal{V}|}$, which is a probability distribution over the next token continuation of x. Internally, M has L layers, where the output of the ℓ^{th} layer is computed as, $h^\ell = h^{\ell-1} + m^\ell + \sum_{j \le J} a^{\ell j}$. Here, m^ℓ is the output of the MLP , and $a^{\ell j}$ is the contribution of j^{th} attention head. For an individual head, its contribution to h^ℓ at token position t is computed as:

$$a_t^{\ell j} = W_{OV}^{\ell j} \cdot \text{Attn}(q_t, K) \tag{1}$$

$$\text{where} \quad q_t = h_t^{\ell-1} W_Q^{\ell j}, \quad K = h_{\le t}^{\ell-1} W_K^{\ell j}, \quad \text{and} \quad \text{Attn}(q_t, K) = \text{softmax}\left(\frac{q_t K^T}{\sqrt{d_k}}\right)$$

Here, $\le t$ denotes all tokens up to the current token t. Following Elhage et al. (2021), we combine the value projection $W_V^{\ell j}$ and out projection $W_O^{\ell j}$ in a single $W_{OV}^{\ell j}$. From here onward we will denote the j^{th} attention head at layer ℓ as $[\ell,\ j]$.

Filter Tasks. In functional programming, the `filter` operation is used to select items from a collection that satisfy specific criteria. `filter` takes two arguments: the collection, and a *predicate* function that returns a boolean value indicating whether an item meets the criteria. Formally:

$$\begin{aligned} \texttt{filter}(\mathcal{C}, \psi) &= \{c \in \mathcal{C} \mid \psi(c) \text{ is True}\} \\ \text{where} \quad & \mathcal{C} = \{c_1, c_2, \ldots, c_n\} \text{ is a collection of items} \\ \text{and} \quad & \psi : X \to \{\text{True}, \text{False}\} \text{ is the predicate function} \end{aligned} \tag{2}$$

To study how language models implement filtering, we design a suite of filter-reduce tasks $\mathcal{T}$. For each task $\tau \in \mathcal{T}$, we construct a dataset $\mathcal{D}_\tau$ containing prompts $\{p_1, p_2, \ldots, p_m\}$. Each prompt $p_i = \mathbb{P}(\mathcal{C}, \psi)$ represents a natural language expression of a specific filter-reduce operation, where $\mathbb{P}$ denotes the verbalization function that converts the formal specification into natural language. Figure 1 shows a concrete example, and we include additional examples from each task in Appendix A.

2.2 Filter Heads

We observe that, for a range of filtering tasks, specific attention heads in the middle layers of Llama-70B consistently focus their attention on the items satisfying a given predicate, ψ. See Figure 1 (more in Appendix K) where we show the attention distribution for these filter heads from the last token position. From Equation (1), we know that this selective attention pattern emerges from the interaction between the query state at the last token (q_{-1}) and the key states from all preceding tokens $\big(K_{\leq t} = \{k_1, k_2, \ldots, k_t\}\big)$. We employ activation patching (Meng et al., 2022; Zhang & Nanda, 2023) to understand the distinct causal roles of these states.

To perform activation patching, we sample two prompts from $\mathcal{D}_\tau$: the source prompt, $p_{\text{src}} = \mathbb{P}(\mathcal{C}_{\text{src}},\ \psi_{\text{src}})$ and the destination prompt, $p_{\text{dest}} = \mathbb{P}(\mathcal{C}_{\text{dest}},\ \psi_{\text{dest}})$, such that the predicates are different ($\psi_{\text{src}} \neq \psi_{\text{dest}}$), and the collections are mutually exclusive ($\mathcal{C}_{\text{src}} \cap \mathcal{C}_{\text{dest}} = \emptyset$). We ensure that there is at least one item $c_{\text{targ}} \in \mathcal{C}_{\text{dest}}$, that satisfies ψ_{src}.

Figure 1 illustrates our activation patching setup with an example. For a filter head $[\ell,\ j]$ we analyze its attention pattern on three different forward passes.

source run $M(p_{\text{src}})$**:** We run the LM on the source prompt p_{src} and cache the query state for $[\ell,\ j]$ at the last token position, $q_{-1}^{\ell j}$, hereafter denoted as q_{src} for brevity.

destination run $M(p_{\text{dest}})$**:** The LM is run with p_{dest}.

patched run $M(p_{\text{dest}})[\leftarrow q_{\text{src}}]$**:** We run the LM with p_{dest} again, but we replace the query state at the last token position for head $[\ell,\ j]$, $q_{-1}^{\ell j}$ with q_{src} cached from the source run.

The attention patterns for the head $[\ell, j]$ from the three forward passes for an example prompt pair are depicted in Figure 1(b), (e), and (f) respectively. In the source and destination runs, the head attends to the items that satisfy the respective predicates. But in the patched run, the filter head $[\ell, j]$ shifts its attention to the item in $\mathcal{C}_{\text{dest}}$ that satisfies ψ_{src}. Patching q_{src} is enough to trigger the execution of ψ_{src} for this head in a different context, validating that q_{src} encodes a compact representation of ψ_{src}.

Notably, we cache the query states before the positional embedding (Su et al., 2024) is applied, while $\text{Attn}(q_t, K)$ in Equation (1) is calculated after the position encoding is added. This indicates that filter heads are a category of semantic heads (Barbero et al., 2024) with minimal sensitivity to the positional information.

2.3 Locating Filter Heads

Now we introduce the methodology to systematically locate these filter heads within a LM.

Activation Patching with DCM. While analyzing attention patterns can provide valuable insights, attention patterns can sometimes be deceptive (Jain & Wallace, 2019) as they may not give insights into the underlying *causal* mechanisms of the LM (Grimsley et al., 2020). To address this issue, we perform causal mediation analysis with the activation patching setup discussed in Section 2.2 to isolate the heads carrying the predicate representation. We want to find a set of heads that *causes* the score (logit or probability) of the target item c_{targ} to increase in the patched run.

We begin by patching the attention heads *individually* and selecting the heads that maximize the logit difference of c_{targ} in the patched run vs the destination run, $\text{logit}[\leftarrow q_{\text{src}}](c_{\text{targ}}) - \text{logit}(c_{\text{targ}})$. We

use logits instead of probabilities as logits have a more direct linear relationship with the influence caused by the intervention (Zhang & Nanda, 2023).

However, we find that patching a single filter head is often not a strong enough intervention to exert influence over the final LM behavior because other filter heads, in addition to backup mechanisms (Wang et al., 2022; McGrath et al., 2023), may work against the intervention and rectify its effects. To address this issue, we learn a sparse binary mask over all the attention heads, similar to in De Cao et al. (2020) and Davies et al. (2023). We cache the query states for the source run $M(p_{\text{src}})$ and destination run $M(p_{\text{dest}})$, and then perform the following interchange intervention over the query states of all the attention heads in the patched run:

$$q_{-1}^{\ell j} \leftarrow \text{mask}^{\ell j} * q_{\text{src}}^{\ell j} + (1 - \text{mask}^{\ell j}) * q_{\text{dest}}^{\ell j} \tag{3}$$

Here, $q_{-1}^{\ell j}$ denotes the query state of head $[\ell, j]$ at the last token position; $q_{\text{src}}^{\ell j}$ and $q_{\text{dest}}^{\ell j}$ are the query states of the same head at last token from $M(p_{\text{src}})$ and $M(p_{\text{dest}})$, respectively. The mask $\text{mask}^{\ell j}$ is a binary value learned with an objective to maximize the logit of c_{targ} in the patched run. We use a sparsity regularizer to ensure that the mask is sparse (i.e., only a few heads are selected). In Figure 1(g) we mark the filter heads selected for one of our filtering tasks, *SelectOne — Obj*, with their individual average indirect effect (AIE) of promoting the logit of c_{targ}.

Causality. If the filter heads we have identified fully capture a compact representation of the predicate ψ in their query states that the LM uses to perform the filtering operation, then transferring q_{src} from $M(p_{\text{src}})$ to $M(p_{\text{dest}})$ should be *causally* influential: it should cause the LM to select c_{targ}, the item in $\mathcal{C}_{\text{dest}}$ that satisfies ψ_{src}. We introduce a *causality* score to quantify the collective causal influence of the selected filter heads.

$$c^* = \underset{c \in \mathcal{C}_{\text{dest}}}{\text{argmax}} \left(M(p_{\text{dest}}) \left[q_{-1}^{\ell j} \leftarrow q_{\text{src}}^{\ell j} \mid \forall [\ell, j] \in \mathcal{H} \right] \right)_t$$

$$\text{Causality}(\mathcal{H}, p_{\text{src}}, p_{\text{dest}}) = \mathbb{1}[c^* \stackrel{?}{=} c_{\text{targ}}] \tag{4}$$

$$\text{where} \quad \mathcal{H} \text{ is the set of all selected filter heads}$$

We run the LM on p_{dest} and patch the query state of only the selected heads at the last token position with their corresponding query states cached from $M(p_{\text{src}})$. We then check if the LM predicts c_{targ} as the most probable item in the LM's output distribution among all items in $\mathcal{C}_{\text{dest}}$.

Notably, while finding the heads we do not care if the heads exhibit the attention behavior illustrated in Figure 1. But, we notice that the aggregated attention pattern of the identified heads consistently align with the selective attention pattern for filtering (see Appendix K for some examples).

3 Experiments

We now empirically test the role of filter heads in different settings to validate our claims.

Models. We study autoregressive transformer LMs in our experiments. Unless stated otherwise, all the reported results are for Llama-70B (Touvron et al., 2023). We include additional results for Gemma-27B (Team et al., 2024) in Appendix H.

Datasets. To support our evaluation, we curate a dataset consisting of six different tasks that all require the LM to perform filtering, followed by a reduce step to provide a specific answer. Each task-specific dataset $\mathcal{D}_\tau$ contains a collection of items categorized in different categories (e.g. *fruits*, *vehicles*, ... in *Obj type*), as well as different prompt templates for questions specifying the predicate and the reduction task (e.g. *How many [category]s are in this list?*). When we curate a prompt for the task, we sample the collection from the items in $\mathcal{D}_\tau$ and fill in the template with the target predicate. The tasks are listed in Figure 3, and see Appendix A for example prompts from each task.

Implementation Details. For each task we locate the filter heads using the method detailed in Section 2.3 on 1024 examples. During localization we perform the interchange operation (Equation (3)) only at the last token, but for evaluation we consider last 2 tokens ({"`\Answer`", "`:`" }) to reduce information leakage. We also calculate q_{src} as a mean of n source prompts achieved from a single p_{src} by changing the index of c_{src} in $\mathcal{C}_{\text{src}}$[1]. While sampling the counterfactual prompts, we ensure that the

[1]This slightly increases the causality by removing the order information. See Appendix F.

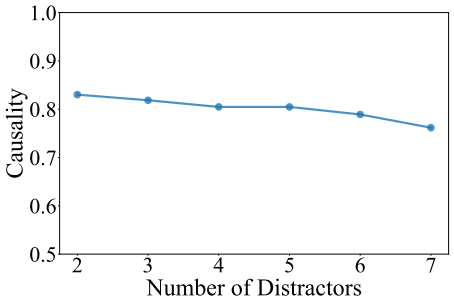


Figure 2: Filter heads retain a causality close to 0.8 even with 7 distractors in the collections of the destination prompt.

Table 1: Causality of filter heads on *SelectOne* tasks. Heads identified using object-type filtering (e.g., *find the fruit*) generalize to semantically distinct predicates like profession identification (*find the actor*).

Filtering Task	Causality	Δlogit
Object Type	0.863	+9.03
Person Profession	0.836	+7.33
Person Nationality	0.504	+5.04
Landmark in Country	0.576	+7.02
Word rhymes with	0.041	+0.65

answer for the source prompt, destination prompt, and the target answer for the patched prompt are all different from each other. All the reported scores are evaluated on a draw of 512 examples where the LM was able to correctly predict the answer. In some cases we include Δlogit, the logit difference of c_{targ} in the patched run versus the destination run, as a softer metric to causality from Equation (4).

3.1 Portability/Generalizability Within Task

Following the approach detailed in Section 2.3, we identify the filter heads on the *SelectOne* task for object categorization. While localizing these heads we use English prompts that follow a specific format: the items are presented in a single line and the question specifying the predicate is presented *after* the items. We test whether the filter heads identified with this format generalize to various linguistic perturbations and *SelectOne* tasks that require reasoning with information of different semantic type. We evaluate generalization using the causality score (Equation (4)).

Information Types. Table 1 shows that filter heads identified on object categorization maintain high causality even in entirely different semantic domains — notably, identifying people by profession shows comparable causality despite the semantic shift. The filter heads also retain non-trivial causality for person-nationality and landmark-country associations, with causality improving by approximately 10 points when we include prefixes which prime the LM to recall relevant information in the item representations (see Appendix G).

However, the predicates captured by these filter heads show poor causality in situations that require reasoning with non-semantic information, such as identifying rhyming words. This indicates that the filter heads play a causal role specifically in situations that require filtering based on semantic information rather than non-semantic properties like phonological similarity or letter counting.

Size of the collection, $\mathcal{C}$ In Figure 2 we plot the causality of filter heads by varying the number of distractor items in the list. The figure shows that the heads are not very sensitive to the size of the collection, retaining high causality even with 7 distractors.

Table 2: Portability of predicate representations across linguistic variations. The predicate vector q_{src} is extracted from a source prompt and patched to destination prompts in **(a)** different languages, **(b)** different presentation formats for the items, and **(c)** placing the question before or after presenting the collection.

	To				
From	English	Spanish	French	Hindi	Thai
English	**0.863**	0.893	0.779	0.928	0.951
Spanish	0.857	**0.877**	0.775	0.875	0.891
French	0.938	0.932	**0.793**	0.931	0.9473
Hindi	0.920	0.920	0.885	**0.918**	0.957
Thai	0.897	0.928	0.887	0.940	**0.943**

(a) Cross-lingual transfer

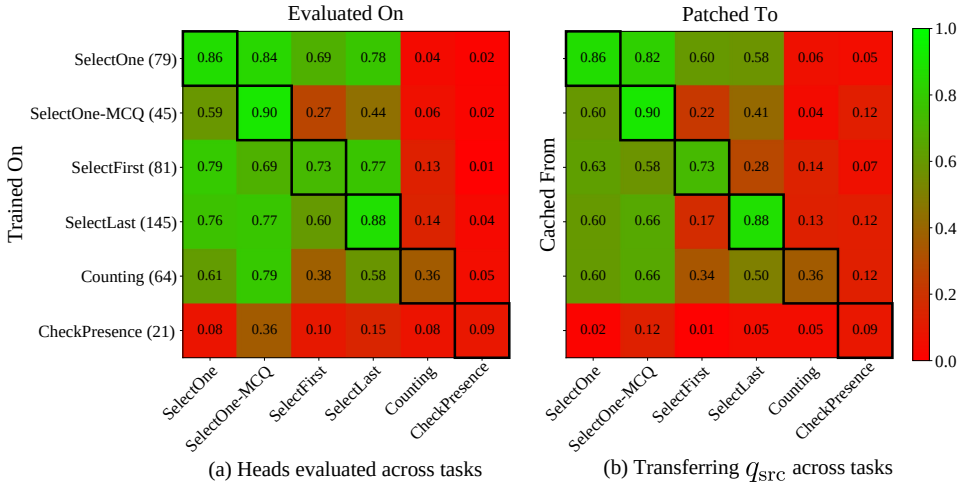
	To	
From	single line	bulleted
single line	**0.863**	0.842
bulletted	0.840	**0.848**

(b) Across option presentation style

	To	
From	after	before
after	**0.863**	0.580
before	0.398	**0.020**

(c) Placement of the question

Linguistic Variations. We test whether predicate representations remain causal under linguistic perturbations by extracting q_{src} from one prompt format and applying it to destination prompts with



(a) Heads evaluated across tasks (b) Transferring q_{src} across tasks

Figure 3: Generalization across different tasks. **(a)** shows whether the heads identified with one task (rows) maintain causal influence in another task (columns). **(b)** shows how portable are the predicate representation is across tasks. The predicate rep q_{src} is cached from one source task example (e.g., *find the fruit* in SelectOne task) was patched to an example from another destination task (e.g., *count the vehicles* in Counting task). The heatmap shows causality scores — whether the LM correctly performs the destination task with the transferred predicate (e.g., *count the fruits*). For both **(a)** and **(b)** the values in the diagonal grid show within task scores.

different presentation styles, or even languages. Table 2(a) and (b) demonstrate remarkable robustness: the same filter heads maintain high causality across different item presentation formats, and even cross-lingual transfer. This invariance to surface-level variation confirms that filter heads encode abstract semantic predicates rather than pattern-matching on specific linguistic forms. However, we also observe that when the question is presented *before* the items, the filter heads show poor causality (see Table 2(c)). We find that this is because in the question-before case the LM relies more on a complementary implementation of filtering, which we discuss in Section 5 and in Appendix B. All the other results presented in this section are calculated on prompts following the question-after format.

3.2 Portability/Generalizability Across Filter-Reduce Operations

To understand the scope of filter head usage, we examine their participation across six filter-reduce tasks of different complexity. Each of these tasks require the LM to perform a separate reduce step to produce an answer in a specific format. We measure whether the heads identified from one task maintain their causality when tested on another task.

Figure 3(a) reveals two distinct patterns. First, transferring the heads across the four tasks — *SelectOne*, *SelectOne(MCQ)*, *SelectFirst*, and *SelectLast* — show high causality scores ($\geq 70\%$[2]) among them, which indicate a high overlap of the same filter heads. In contrast, *Counting* shows an interesting asymmetric pattern: while *Select** heads fail on the *Counting* task, *Counting* heads show partial generalization to the *Select** tasks — suggesting that *Counting* does share some common sub-circuit with *Select** tasks, while having a more complex mechanism, likely involving additional circuits for specialized aggregation, that we have not yet identified. *CheckPresence* heads show poor causality even within the task, indicating that the LM possibly performs this task in an alternate way that can bypass the filtering sub-circuit.

We also test the portability of the predicate information encoded in q_{src} by transferring it across tasks, Figure 3(b). *SelectFirst* and *SelectLast* tasks show notably poor cross-task transfer of the predicate, even though the filter heads retain high within-task causality. This suggests that, in q_{src} the predicate information is possibly entangled with task-specific information. Otherwise, predicate transfer scores (Figure 3b) mirror the head transfer scores (Figure 3a) with slightly lower values.

Our findings suggest that filter heads form a foundational layer for a range of reduce operations, with simpler selection tasks relying primarily on this mechanism while more complex aggregation tasks build additional computation on top of it. This insight aligns with Merullo et al. (2023) that transformer LMs use common sub-circuits (filter heads) across different (filter-reduce) tasks.

[2]Except the heads from *SelectOne-MCQ*, possibly because selecting MCQ-options is computationally simpler than to output the filtered item.

Table 3: LM performance on filtering tasks drops significantly when filter heads are ablated. These heads constitute $< 2\%$ of the heads in the LM. Evaluated on 512 samples that the LM predicts correctly without any ablation (baseline 100%).

Task(#Heads)	LM Acc (Heads Abl)	
	Filter	Random
SelectOne (79)	22.5%	100%
SelectOne(MCQ) (45)	0.4%	100%
SelectFirst (81)	13.1%	97.3%
SelectLast (145)	9.22%	99.4%
Count (64)	89.80%	99.19%
CheckExistence (21)	98.61%	99.2%

Table 4: Filter heads play a distinct causal role during filtering tasks. Table shows the causality of filter heads with other type of heads documented in literature. None of the other head types match the causality of filter heads in the *SelectOne* task. To keep our comparisons fair we keep the number of heads equal (79) for every head type.

Head Type	Causality	Δlogit
Filter	0.863	+9.03
Function Vector	0.00	−3.20
Concept	0.00	−1.37
Induction	0.00	−3.23
Random	0.00	−0.96

3.3 Necessity of Filter Heads

We seek to understand to what extent the LM relies on filter heads during these filter-reduce tasks. To assess their importance, we perform ablation studies.

We ablate an attention head $[\ell, j]$ by modifying its attention pattern during the forward pass so that the last token can only bring information from the <BOS> token[3]. Previous works (Geva et al., 2023; Sharma et al., 2024) have investigated if critical information flows through a certain attention edge with similar attention knock-out experiments. The results in Table 3 reveal a dramatic performance drop for the *Select** tasks when filter heads are ablated, despite these heads comprising less than 2% of the model's total attention heads — confirming their critical importance for the *Select** tasks. In contrast the performance for *Counting* and *CheckExistence* do not drop significantly due to this ablation, again indicating that these tasks do not fully rely on the filter heads.

To determine whether filter heads represent a novel discovery or merely overlap with existing attention head categories previously documented in the literature, we compared their functionality against such head categories. Specifically, we measure the causality of Function Vector heads (Todd et al., 2023), Concept heads (Feucht et al., 2025), and Induction heads (Olsson et al., 2022). As shown in Table 4, none of these previously identified head types exhibit the distinctive causal role of filter heads, confirming that filter heads are a unique and previously unrecognized component of transformer LMs.

4 Key States Carry Item Semantics for Filtering

To understand *how* the predicate-encoding query states in the filter heads implement filtering via interacting with the key states from previous tokens, we design another activation patching experiment.

Approach. To isolate the contribution of key states, we designed a two-part intervention that combines query patching with key swapping. We select two items $(c_{\text{targ}}, c_{\text{other}})$ from $\mathcal{C}_{\text{dest}}$ such that c_{targ} satisfies ψ_{src}, but not ψ_{dest}; and c_{other} doesn't satisfy either of the predicates.

The intervention proceeds as follows. For a filter head $[\ell, j]$, we patch the q_{src} from the source prompt (as before). Then we swap the key states between c_{targ} and c_{other} within the same forward pass. Figure 4 illustrates this key swapping. After this key-swapping intervention, the filter head $[\ell, j]$ redirects its focus from c_{targ} (Figure 1(f)) to c_{other} (Figure 4-right).

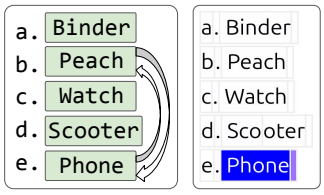


Figure 4: Swapping the key states of the items in addition causes the filter head to redirect its focus to an unrelated item.

Result. We consider this 2-step intervention to be causally effective if the LM assigns highest probability to c_{other} among the items. On the *SelectOne* object categorization task, we achieve a causality score of 0.783 (432/552 examples) with $\Delta\text{logit} = 8.2591 \pm 3.352$, confirming that key states indeed encode the semantic properties that predicates evaluate. For experimental simplicity, we restrict this analysis to only single-token items.

[3] for the LM tokenizers that do not add a <BOS> token by default, we prepend <BOS> manually.

This result confirms our mechanistic hypothesis: filter heads implement filtering through a key-query interaction where queries encode "what to look for" (the predicate) and keys bring "what is there" (item properties) from the corresponding item latents ($h^{\ell-1}$).

5 What happens if the question comes *before* the options?

In Table 2(c) we see that if we simply reverse the order of the question and the collection to ask the question before presenting the items, the causality scores drop to almost zero. Our investigations reveal that this seemingly innocent ordering change fundamentally alters the computational strategy available to the LM. When the question comes first, the transformer can perform *eager evaluation*: as each item is processed, the model can immediately evaluate whether it satisfies the predicate and store this information as an *is_match* flag in the item's latents. And, at the final token, rather than performing the predicate matching operation via filter heads, the LM can simply retrieve items based on pre-computed flags. If this hypothesized flagging mechanism is true, then manipulating this flag should result in predictable outcomes in the LM's behavior. We find evidence for this alternative mechanism through a series of carefully designed activation patching experiments. We illustrate the core experiment setup in Figure 5, while we leave the detailed analysis to Appendix B.

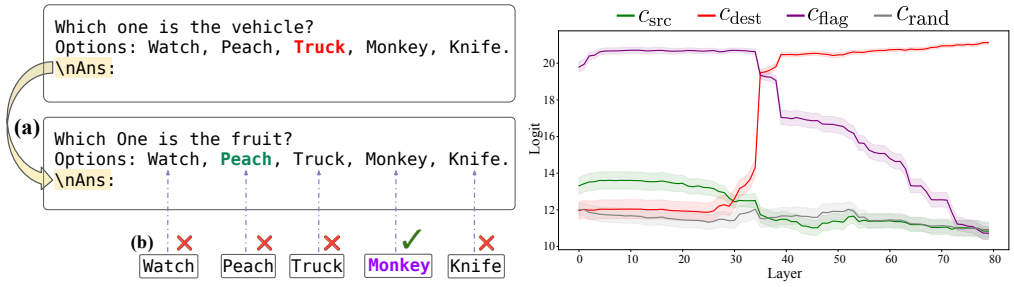


Figure 5: Testing for answer flags in question-first prompts. We perform a two-part intervention to determine whether the model stores filtering decisions as flags in item representations. **(a)** We patch the residual states at the final two token positions (`"\nAns"` and `":"`) from a source prompt p_{src} to a destination prompt p_{dest} at a single layer. **(b)** Additionally, for an item $c \in \mathcal{C}_{\text{dest}}$, we replace its hidden representations across *all* layers with those from a prompt p_{diff} containing a different predicate $\psi_{\text{diff}} \notin \{\psi_{\text{src}}, \psi_{\text{dest}}\}$. We use a different p_{diff} (with different ψ_{diff}) per item in $\mathcal{C}_{\text{dest}}$. Crucially, we ensure that exactly one item c_{flag} (distinct from both the source and destination answers) carries the *is_match* flag from its corresponding p_{diff}. Right panel shows results for the *SelectOne-Obj* task with **(a)** applied per layer in conjunction with **(b)** for all layers. After this 2-step intervention the LM consistently selects c_{flag} in early layers, confirming the LM's reliance on pre-computed answer flags stored in the item representations. And, in later layers **(a)** simply brings over the decision from the source run.

However, filter heads remain partially active even in question-first examples. Table 2(c) also shows that caching q_{src} from question-first prompts and patching to question-after prompts gives non-trivial causality scores, though lower than our original setup. This suggests filter heads can still partially encode the predicate information in question-first settings, but this "filter head"-based mechanism competes with, and is typically overshadowed by, the flag-based mechanism.

The co-existence of these two filtering strategies: on-demand filtering through filter heads and precomputing the flags and storing them in the item latents, echoes the lazy versus eager evaluation strategies in functional programming from Henderson & Morris Jr (1976). This also shows how transformer LMs can maintain multiple redundant pathways for the same operation (McGrath et al., 2023; Wang et al., 2022) and can dynamically select between them based on task demands and information availability.

6 Application: A light-weight Probe Without Training

The predicate information encoded by the filter heads can be leveraged for a practical use-case: zero-shot concept detection through training-free probes.

Since filter heads encode predicates as query states that interact with key-projected item representations to perform filtering, we can repurpose this mechanism for classification. To detect whether a representation h belongs to a particular concept class (e.g., animal, vehicle), we create filter prompts for each class: $p_{\text{cls}} = \mathbb{P}(\mathcal{C}, \psi_{\text{is\_cls}})$ and collect the query states q_{cls} from a filter head $[\ell, j]$. Then we classify h by finding the class whose q_{cls} has the maximum affinity.

$$\hat{y} = \arg\max_{\text{cls}} \left(q_{\text{cls}} \cdot W^K_{\ell,j} h \right) \tag{5}$$

Where $W^K_{\ell,j}$ is the key projection from the head $[\ell, j]$. Figure 6 demonstrates that this approach achieves strong classification performance without any training, validating that filter heads learn generalizable concept representations that can be extracted and applied as probes. See Appendix K.3 where we illustrate how filter heads can utilized to detect other concepts such as the presence of false information or certain sentiment in free-form text.

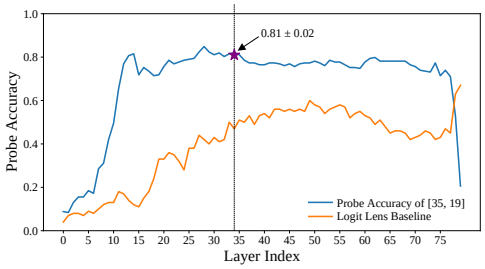


Figure 6: Training-free probe using filter head [35, 19]. Accuracy across layers on 238 objects spanning 16 classes from the SelectOne-Object dataset. Final token is used for multi-token items. Compared against using the embedding vectors of LM decoder as class probes.

7 Related Works

Attention Head Studies. Previous works have identified specialized attention heads that serve distinct computational roles. Olsson et al. (2022) discovered induction heads that implement pattern matching and copying, while Feucht et al. (2025) have identified heads that copy concepts instead of individual tokens. Todd et al. (2023) have found function vector heads that encode task representations that are transportable across contexts. Filter heads are an addition to this class of attention heads that show distinct functional specialization.

LM Selection Mechanisms. A few empirical studies have explored the selection mechanism in LMs, primarily in MCQA settings. Tulchinskii et al. (2024) identifies "select-and-copy" heads based on their attention pattern that focus on "\n" after a correct item in a question-first MCQ format. Lieberum et al. (2023) also identify attention heads that attend to the correct MCQ label/letter and show that these "correct label" heads encode the ordering ID of the presented options. Wiegreffe et al. (2024) showed that attention modules in the middle layers promote the answer symbols in a MCQA task. Unlike these works focused on MCQA settings, in this paper we investigate list-processing in general and find a set of filter heads implement predicate evaluation that generalize across formats, languages, and even different reduction operations.

Symbolic Reasoning in Neural Networks. Recently researchers have been increasingly interested in the question of whether transformer LMs can develop structed symbolic-style algorithmic behavior. Yang et al. (2025) discuss how LMs can implement an abstract symbolic-like reasoning through three computational stages: with early layers converting tokens to abstract variables, middle layers performing sequence operations over these variables, and then later layers accessing specific values of these variables. Meng et al. (2022) and Geva et al. (2023) also notice similar stages while the LM recalls a factual association. Several works have documented mechanisms/representations specialized for mathematical reasoning (Nanda et al., 2023; Hanna et al., 2023; Kantamneni & Tegmark, 2025) and variable binding (Feng & Steinhardt, 2023; Prakash et al., 2025).

Our paper continues this tradition of validating Smolensky (1991)'s assertion that distributed representations in connectionist systems can have "sub-symbolic" structures, with symbolic structures emerging over the interaction between many units. In this work we study a specific symbolic abstraction — filtering in list processing — which is a fundamental abstraction for both symbolic computation and human reasoning (Treisman, 1964; Johnson-Laird, 1983).

8 Discussion

In this work, we have identified and characterized filter heads — specialized attention heads that implement filtering operations in autoregressive transformer LMs. These heads encode the filtering criteria (predicates) as compact representations in their query states of specific tokens. This encoding can be extracted and then transported to another context to trigger the same operation. We also identify that, based on information availability, the LM can use an *eager* implementation of filtering by storing flags directly on the item latents. These dual and complimentary filtering implementations mirror the lazy vs eager evaluation from functional programming. This convergence between emergent neural mechanisms and human-designed programming primitives suggests that certain computational patterns arise naturally from task demands rather than architectural constraints. Cataloging such universal computational primitives and how they are realized may help us understand how AI systems perform complex reasoning.

ETHICS

This research investigates the internal computational mechanisms of LMs through mechanistic interpretability techniques, contributing to the scientific understanding of transformer architectures. Our identification of filter heads advances LMs transparency by revealing how models implement functional programming primitives, though we acknowledge that interpretability findings do not directly translate to safety improvements without additional work. The causal mediation techniques we develop could potentially be applied to study more sensitive model capabilities, requiring responsible application and consideration of dual-use implications in future research on mechanisms related to deception or manipulation. Our experiments require significant computational resources that may limit reproducibility to well-resourced institutions, though we commit to releasing code and datasets to facilitate broader access. While our findings about filter heads appear robust across different tasks and languages, we caution against overgeneralizing to other domains without validation, as mechanistic interpretability remains early-stage and our understanding of component interactions is incomplete.

REPRODUCIBILITY

We ran all experiments on workstations with either 80GB NVIDIA A100 GPUs or 48GB A6000 GPUs, using the HuggingFace Transformers library (Wolf et al., 2019) and PyTorch (Paszke et al., 2019). We used `NNsight` (Fiotto-Kaufman et al., 2024) for our patching experiments. The codes and the dataset produced in this work will be made publicly available.

REFERENCES

Afra Amini and Massimiliano Ciaramita. In-context probing: Toward building robust classifiers via probing large language models. *arXiv preprint arXiv:2305.14171*, 2023.

Federico Barbero, Alex Vitvitskyi, Christos Perivolaropoulos, Razvan Pascanu, and Petar Veličković. Round and round we go! what makes rotary positional encodings useful? *arXiv preprint arXiv:2410.06205*, 2024.

Xander Davies, Max Nadeau, Nikhil Prakash, Tamar Rott Shaham, and David Bau. Discovering variable binding circuitry with desiderata. *arXiv preprint arXiv:2307.03637*, 2023.

Nicola De Cao, Michael Schlichtkrull, Wilker Aziz, and Ivan Titov. How do decisions emerge across layers in neural models? interpretation with differentiable masking. *arXiv preprint arXiv:2004.14992*, 2020.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? *arXiv preprint arXiv:2310.17191*, 2023.

Sheridan Feucht, Eric Todd, Byron Wallace, and David Bau. The dual-route model of induction. *arXiv preprint arXiv:2504.03022*, 2025.

Jaden Fiotto-Kaufman, Alexander R Loftus, Eric Todd, Jannik Brinkmann, Caden Juang, Koyena Pal, Can Rager, Aaron Mueller, Samuel Marks, Arnab Sen Sharma, et al. Nnsight and ndif: Democratizing access to foundation model internals. *arXiv preprint arXiv:2407.14561*, 2024.

Daniel P Friedman, David S Wise, et al. *CONS should not evaluate its arguments*. Computer Science Department, Indiana University, 1976.

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

Christopher Grimsley, Elijah Mayfield, and Julia Bursten. Why attention is not explanation: Surgical intervention and causal reasoning about neural models. 2020.

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

Peter Henderson and James H Morris Jr. A lazy evaluator. In *Proceedings of the 3rd ACM SIGACT-SIGPLAN Symposium on Principles on Programming Languages*, pp. 95–103, 1976.

Sarthak Jain and Byron C Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

Philip Nicholas Johnson-Laird. *Mental models: Towards a cognitive science of language, inference, and consciousness*. Number 6. Harvard University Press, 1983.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition. *arXiv preprint arXiv:2502.00873*, 2025.

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. *arXiv preprint arXiv:2307.09458*, 2023.

David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 1982.

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models. *arXiv preprint arXiv:2310.08744*, 2023.

Simon Conway Morris. Evolutionary convergence. *Current Biology*, 16(19):R826–R827, 2006.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Nikhil Prakash, Natalie Shapira, Arnab Sen Sharma, Christoph Riedl, Yonatan Belinkov, Tamar Rott Shaham, David Bau, and Atticus Geiger. Language models use lookbacks to track beliefs. *arXiv preprint arXiv:2505.14685*, 2025.

Arnab Sen Sharma, David Atkinson, and David Bau. Locating and editing factual associations in mamba. *arXiv preprint arXiv:2404.03646*, 2024.

Paul Smolensky. The constituent structure of connectionist mental states: A reply to fodor and pylyshyn. In *Connectionism and the Philosophy of Mind*, pp. 281–308. Springer, 1991.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Anne M Treisman. Selective attention in man. *British medical bulletin*, 1964.

Eduard Tulchinskii, Laida Kushnareva, Kristian Kuznetsov, Anastasia Voznyuk, Andrei Andriiainen, Irina Piontkovskaya, Evgeny Burnaev, and Serguei Barannikov. Listening to the wise few: Select-and-copy attention heads for multiple-choice qa. *arXiv preprint arXiv:2410.02343*, 2024.

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

Sarah Wiegreffe, Oyvind Tafjord, Yonatan Belinkov, Hannaneh Hajishirzi, and Ashish Sabharwal. Answer, assemble, ace: Understanding how lms answer multiple choice questions. *arXiv preprint arXiv:2407.15018*, 2024.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Yukang Yang, Declan Campbell, Kaixuan Huang, Mengdi Wang, Jonathan Cohen, and Taylor Webb. Emergent symbolic mechanisms support abstract reasoning in large language models. *arXiv preprint arXiv:2502.20332*, 2025.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. *Advances in neural information processing systems*, 36:27223–27250, 2023.

A EXAMPLE PROMPTS FROM OUR DATASET

A.1 DIFFERENT TASKS

Select One – Type of Object

"Options: Bus, Peach, Scooter, Phone, Pen
Find the fruit in the options presented above.
Answer:"

Expected LM Output: " Peach"

Select One – Type of profession

"Options: Neymar, Hillary Clinton, Clint Eastwood
Who among these people mentioned above is an actor by profession?
Answer:"

Expected LM Output: " Clint"

Select One – Type of nationality

"Options: Ronaldinho, Brad Pitt, Jet Li, Ken Watanabe
Who among these people mentioned above is from China?
Answer:"

Expected LM Output: " Jet"

Select One – Location of landmark

"Options: Cabo San Lucas Arch, Plaza de Armas Cusco, Mont Saint-Michel
Which of these landmarks is in Peru?
Answer:"

Expected LM Output: " Plaza"

Select One — Rhyme

"Options: blue, debt, bright, sting, sake
Which of these words rhymes with glue?
Answer:"

Expected LM Output: " blue"

Select One (MCQ)

"a. Banana
b. Paperclip
c. Oven
d. Dress
e. Church
f. Bench
Which among these objects mentioned above is a clothing?
Answer:"

Expected LM Output: " d"

Select First

"Options: Church, Scarf, Pendant, Slow cooker, Temple
What is the first building from the list above?
Answer:"

Expected LM Output: " Church"

Select Last

"Options: Horse, Anklet, Golf ball, Cow, Necklace
What is the last animal in this list above?
Answer:"

Expected LM Output: " Cow"

Counting

"Options: Trombone, Flute, Guitar, Train, Car
How many vehicles are in this list?
Answer:"

Expected LM Output: " Two"

Check Existence

"Options: Refrigerator, Museum, Notebook, Toaster, Juicer
Do you see a kitchen appliance in the list above?
Answer:"

Expected LM Output: " Yes"

A.2 Linguistic Perturbations

A.2.1 Item Presentation

Single Line

"Options: House, Blender, Willow, Ambulance, Piano, Wrestling mat.
Which among these objects mentioned above is a kitchen appliance?
Answer:

Expected LM Output: " Blender"

Bulleted

"* Temple
* Air fryer
* Basketball
* Willow
* Van
* Harmonica
Which among these objects mentioned above is a vehicle?
Answer:"

Expected LM Output: " Van"

A.2.2 Question Placement

Question After

"Options: Elephant, Maple, Toilet, Camera, Juicer, Mall.
Which among these objects mentioned above is a bathroom item?
Answer:"

Expected LM Output: " Toilet"

Question Before

"Which object from the following list is a music instrument?
Options: Printer, Highlighter, Ukulele, Chair, Mirror, Locket.
Answer:"

Expected LM Output: " Uk"

A.2.3 From a different language

Spanish

"Opciones: Lirio, Colchoneta de lucha, Escritorio, Portátil, Refrigerador, Sandía.
¿Cuáles de estos objetos mencionados anteriormente son un(a) electrónica?
Respuesta:"

Expected LM Output: " Port"

French

"Options : Aigle, Pastèque, Accordéon, Baignoire, Ciseaux, Bibliothèque.
Lequel de ces objets mentionnés ci-dessus est un(e) fourniture de bureau ?
Réponse :"

Expected LM Output: " d"

Hindi

"विकल्प: शेर, साबुन, टेनिस बॉल, अंगूर, मिक्सर, बस.
उपरोक्त वस्तुओं में से कौन-सी एक जानवर है?
उत्तर:"

Expected LM Output: " श"

Thai

"ตัวเลือก: หัวผักกาด, สิงโต, แดฟโฟดิล, กาต้มน้ำ, ชุดเดรส, เชลโล่.
วัตถุใดในรายการข้างต้นที่เป็น เสื้อผ้า?
คำตอบ:"

Expected LM Output: " ช"

B Dual Implementation of Filtering in LMs: Question Before vs After

Our analysis reveals that transformer LMs employ distinct computational strategies for filtering depending on whether the question specifying the predicate precedes or follows the collection. We briefly discussed this in Section 5 and here we provide our detailed analysis.

Table 2(c) shows that filter heads are minimally causal when patched from question-before to question-before prompt, even when both follow the same prompt template and item presentation style. To understand this better, we perform a multi-step causal mediation analysis.

```
Which one is a fruit in this list?
Options: Cherry, Knife, Pen, Ambulance.
\nAns:
```

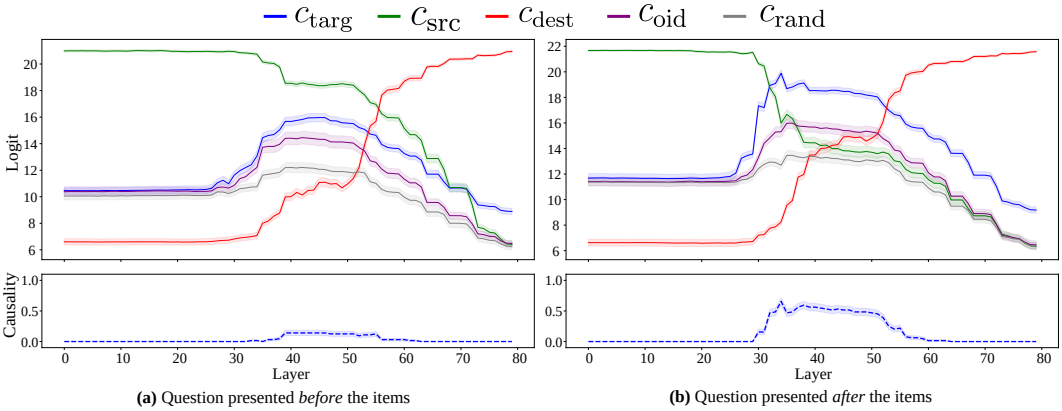
```
Which one is a vehicle in this list?
Options: Binder, Peach, Watch, Scooter, Phone.
\nAns:
```

Figure 7: Example of counterfactual prompt pair used to understand the effect of patching residual latents.

Similar to the patching setup detailed in Section 2.2, we consider two prompts — p_{src} and p_{dest}. The prompts p_{src} and p_{dest} have different predicates ($\psi_{\text{src}} \neq \psi_{\text{dest}}$) and sets of items ($\mathcal{C}_{\text{src}} \cap \mathcal{C}_{\text{dest}} = \emptyset$). But both prompts follow the question-before format (or both follow the question-after format). See Figure 7 for an example of the two prompts.

In the patched run $M(p_{\text{dest}})[\leftarrow h^{\ell}]$, we cache the residual stream latents at the last two tokens ({\Ans, :}) for a layer ℓ from the source run $M(p_{\text{src}})$ and patch them to their corresponding positions in the destination run $M(p_{\text{dest}})$. We perform this for all layers $\ell \in \{1, \ldots, L\}$ and track the scores (logits) of five tokens:

1. c_{src} : correct answer of the source prompt, $c_{\text{src}} \in \mathcal{C}_{\text{src}} \mid \psi_{\text{src}}(c_{\text{src}})$
2. c_{dest} : correct answer of the destination prompt, $c_{\text{dest}} \in \mathcal{C}_{\text{dest}} \mid \psi_{\text{dest}}(c_{\text{dest}})$
3. c_{targ} : the item in the collection in the destination prompt that satisfies the predicate of the source prompt, $c_{\text{targ}} \in \mathcal{C}_{\text{dest}} \mid \psi_{\text{src}}(c_{\text{targ}})$
4. c_{oid} : the item in the destination collection that shares its index with c_{src} in the source collection, $c_{\text{oid}} \in \mathcal{C}_{\text{dest}} \mid \text{index}(c_{\text{oid}}, \mathcal{C}_{\text{dest}}) = \text{index}(c_{\text{src}}, \mathcal{C}_{\text{src}})$. We also make sure that c_{oid} does not satisfy either predicate, $\neg\psi_{\text{src}}(c_{\text{oid}}) \wedge \neg\psi_{\text{dest}}(c_{\text{oid}})$. This token is supposed to capture if the residual states carry information about the positional information or order IDs (Feng & Steinhardt, 2023; Prakash et al., 2025) of c_{src}.
5. c_{rand} : a random item in the destination collection that does not satisfy either predicate and is different from all the other four tokens.



(a) Question presented *before* the items **(b)** Question presented *after* the items

Figure 8: Dual implementation of filtering in LMs. In both question-before and question-after formats c_{targ} (blue) shows elevated scores after patching the residual state at middle layers. But in the question-before format that score is never strong enough to dominate c_{targ} (green). The violet line shows scores for c_{oid}, the item that shares the same index in $\mathcal{C}_{\text{dest}}$ with c_{src} in $\mathcal{C}_{\text{src}}$, which also shows elevated scores in middle layers, although not as pronounced as c_{targ}.

We curate the source and destination prompts such that all five tokens are distinct and perform this experiment for both the question-before and question-after settings. We plot the results in Figure 8 and make the following observations.

O1: In both question-before and question-after settings, the score of c_{targ} (blue) increases in middle layers (30-55) where we identify the filter heads to be. However in the question-before setting, that score is never strong enough to dominate c_{dest} (green). While in the question-after setting c_{targ} becomes the highest scoring token among the four, achieving $\sim 70\%$ causality in these critical layers.

O2: We notice a bump in the score of c_{oid} (violet) in the middle layers, although it is not as pronounced as c_{targ}. This suggests that residual latents in these layers also contain the positional/order information of c_{src}. This has been observed in Feng & Steinhardt (2023) and Prakash et al. (2025). We also notice a slight bump in the score of c_{rand} (gray).

O3: If the patching is performed in late enough layers (> 60) it copies over the final decision (red) from the source run.

The distinction between the trends of c_{targ} and c_{dest} in Figure 8a indicate that the LM relies on an alternate mechanism, more than the one involving the filter heads, to perform filtering in a prompt where the question is presented before the items. We hypothesize that the question appearing before the collection allows the LM to perform *eager evaluation*: storing an *is_match* flag for each item in the collection when they are processed. If this is true then manipulating the *is_match* flag should cause predictable changes in the LM behavior in the question-first setting, while the question-after setting should not be sensitive to such manipulations.

Effect of ablating the *is_match* flag. If the LM is indeed using the *is_match* flag to perform filtering in question-before settings, we would expect that ablating this flag would significantly degrade performance.

To test this hypothesis, we ablate the *is_match* flag in an item by replacing all the residual stream latents of the tokens of that item with their corresponding latents cached for the same item in a neutral prompt (see Figure 9 for details). When we perform this ablation for each of the items in the collection, we indeed see a significant drop in LM performance for the question-before setting, while the performance remains mostly unchanged for the question-after setting (see Table 5).

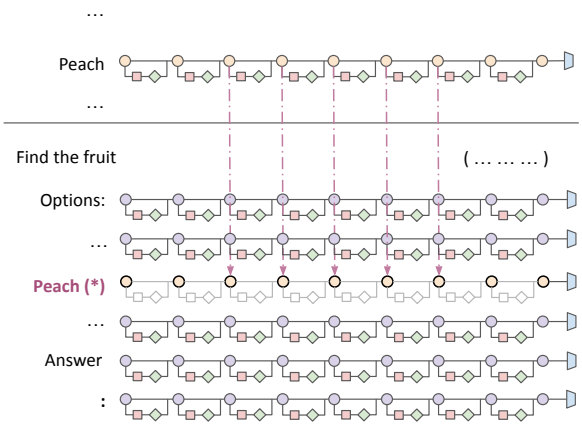


Figure 9: Ablating the *is_match* flag. For an item (e.g "Peach") in the collection, we cache the residual stream latents corresponding to the tokens of that item from a neutral prompt (e.g. any text w/o any predicate that contains the word "Peach"). Then in a separate pass, we replace the latents corresponding to that item in the original prompt with the cached latents. This effectively removes any information about whether that item satisfies the predicate or not.

This experiment supports our hypothesis that the question-before setting allows the LM to *eagerly* evaluate whether an item satisfies the predicate or not, store this intermediate result in the residual latents of the items as it processes the collection, and relies on that to make the final decision. We also notice that the question-after setting shows minimal sensitivity to this flag-ablation, which suggests that the processing of items do not rely on the context here: the LM populates the semantics (*enrichment* in Geva et al. (2023)) of each item in a context independent manner first, and then applies the predicate to perform filtering when the question is presenter after.

Effect of swapping the *is_match* flag between two items. Our most decisive evidence comes from *swapping* the *is_match* flag between items: if we swap the *is_match* flag stored in c_{pos} that satisfies the predicate with c_{neg} that does not satisfy the predicate, we should expect the LM to

Table 5: Effect of ablating *is_match*. Evaluated on 512 examples from the *SelectOne* task.

Ques Place	W/o *is_match* Acc
Before	46.09%
After	96.06%

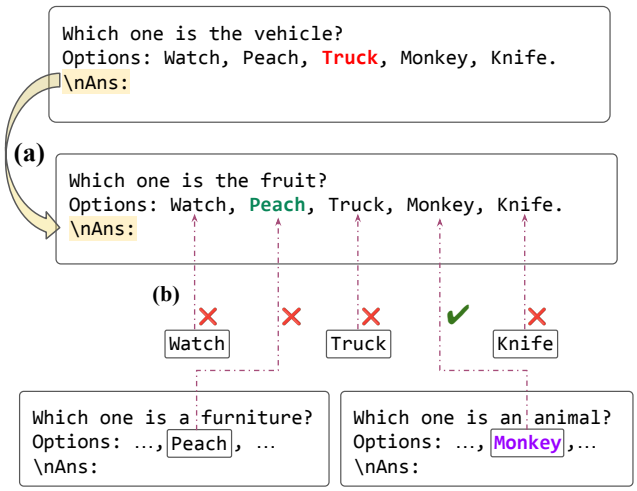


Figure 10: Counterfactual patching setup to swap the *is_match* flag. We perform a two-part intervention to determine whether the model stores filtering decisions as flags in item representations. **(a)** We patch the residual states at the final two token positions ("\nAns" and ":") from a source prompt p_{src} to a destination prompt p_{dest} at a single layer. **(b)** Additionally, for an item $c \in \mathcal{C}_{\text{dest}}$, we replace its hidden representations across *all* layers with those from a prompt p_{diff} containing a different predicate $\psi_{\text{diff}} \notin \{\psi_{\text{src}}, \psi_{\text{dest}}\}$. We use different p_{diff} (with different ψ_{diff}) per item in $\mathcal{C}_{\text{dest}}$. Crucially, we ensure exactly one item c_{flag} (distinct from both the source and destination answers) carries the *is_match* flag from its corresponding p_{diff}.

change its answer from c_{pos} to c_{neg} if it is relying on the *is_match* flag. We set up another activation patching experiment to test this hypothesis illustrated in Figure 10, which is a more elaborate version of Figure 5. We perform a 2 part intervention:

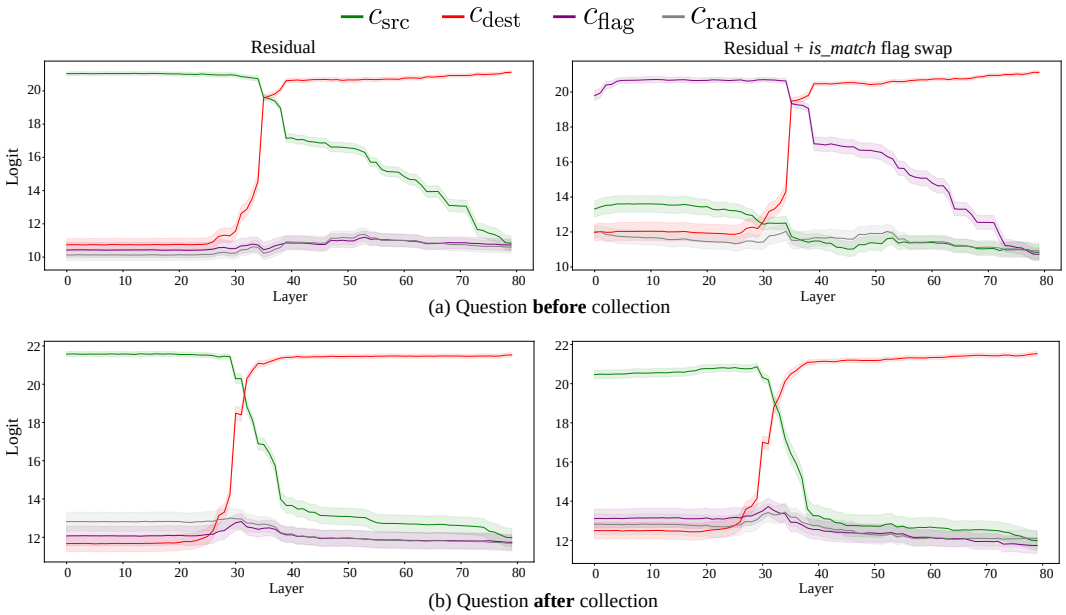
I1: Similar to Figure 7, we consider two prompts p_{src} and p_{dest} that follow the same format, either question-before or question-after, but with different predicates, $\psi_{\text{src}} \neq \psi_{\text{dest}}$. However, now they operate on the same collection of items, $\mathcal{C}_{\text{src}} = \mathcal{C}_{\text{dest}} = \mathcal{C}$. We perform the same intervention, patching the residual stream latents at the last two token positions from $M(p_{\text{src}})$ to $M(p_{\text{dest}})$ for a layer ℓ. And we track the scores of c_{src}, c_{dest}, c_{rand} as defined before. Notice that as the collections are the same, $c_{\text{targ}} = c_{\text{oid}} = c_{\text{src}}$.

I2: In addition, we choose another item $c_{\text{flag}} \in \mathcal{C}$ that is different from c_{src}, c_{dest}, c_{rand} and perform the following intervention to make sure that only c_{flag} carries the *is_match* flag while none of the other items do. In order to achieve that we cache c_{flag}'s latents from an alternate prompt p_{flag} with a predicate ψ_{flag} which is satisfied by c_{flag}, $\psi_{\text{flag}}(c_{\text{flag}})$. Then in the patched run, we replace the latents corresponding to c_{flag} in $M(p_{\text{dest}})$ with the cached latents from $M(p_{\text{flag}})$. This makes sure that c_{flag} now carries the *is_match* flag.
Similarly, to make sure that an item $c' \in \mathcal{C} \backslash \{c_{\text{flag}}\}$ does not carry the *is_match* flag, we cache its latents from another example p' with a predicate ψ' such that $\neg\psi'(c')$. Then we replace the latents corresponding to c' in $M(p_{\text{dest}})$ with the cached latents from $M(p')$. We perform this for all items in $\mathcal{C} \backslash \{c_{\text{flag}}\}$. This effectively ensures that only c_{flag} carries the *is_match* flag while all other items do not. See Figure 10 for an illustration.

In Figure 11 we plot the results of this experiment for both question-before and question-after settings. For a layer ℓ, **I1** is applied for only that layer, without or with **I2**, which is applied to *all* layers.

As expected, we see that in the question-after setting applying **I2** with **I1** is almost indistinguishable from just applying **I1**. **I2** has minimal effect because the LM cannot rely on the *is_match* flag when the question comes after.

However, in the question-before setting, we observe that the score trend of c_{flag} (violet) and c_{dest} (green) almost swap their positions in only **I1** versus when **I2** is applied in addition. With the flag-swap intervention **I2**, the LM systematically picks c_{flag} as the answer in the early layers. This further validates our hypothesis that the LM is relying on the *is_match* flag to make the final decision in the question-before setting.



(a) Question **before** collection

(b) Question **after** collection

Figure 11: Effect of swapping the *is_match* flag between items. The figures on the left shows the effect of patching only the residual states (Figure 7) and figures on the right are when we additionally swap the *is_match* with another item (Figure 10). The pair of figures on the top shows both cases in the question-before format **(a)**. We observe that c_{flag} becomes of top scoring item when the patching is performed in early layers. But this swapping of flags has no effect in the question after case, pair of figures on the bottom **(b)**.

We do not claim that the LM *only* relies on the *is_match* flag in the question-before setting. The fact that we see a bump in the score of c_{targ} (blue) in Figure 8 and patching q_{src} from question-before to question-after prompts has non-trivial causality (see Table 2c) indicates that the LM does carry the predicate information in middle layers even in the question-before setting, although not strongly as in the question-after setting.

This dual filtering implementation strategy — lazy evaluation via filter heads versus storing eager evaluation information with *is_match* flag — exemplifies a broader principle in neural computation: transformer LMs can maintain multiple pathways for core operations, dynamically selecting strategies based on what information is available. And the fact that filter heads still remain partially active even in the question-first setting shows that these mechanisms operate in parallel rather than mutual exclusion.

C Different Approaches for Locating the Filter Heads

In this section we discuss the different approaches we explored to identify the filter heads.

Filter Score. To capture the filtering behavior of the heads based on their attention pattern, we design a *filter score* that quantifies the extent to which a head focuses its attention on the elements satisfying the predicate ψ over other elements in $\mathcal{C}$.

$$\text{FilterScore}([\ell, j], \mathcal{C}, \psi) = \text{score}_{\ell j}(c \mid \psi(c)) - \max_{\neg\psi(c)} \big(\text{score}_{\ell j}(c)\big) \tag{6}$$

$$\text{where,} \quad \text{score}_{\ell j}(c) = \sum_{t \in c} \text{Attn}_{[\ell, j]}(q_{-1}, t)$$

While calculating FilterScore we make sure that there is only one item $c \in \mathcal{C}$ such that $\psi(c)$ is true. The FilterScore then select heads based on how much they focus their attention on the correct item over the most attended incorrect item. The score function sums up the attention scores over all tokens in an item c to account for multi-token items. Note that the score is calculated based on the attention pattern at the last token of the prompt (“:”).

We notice that heads in a range of middle layers exhibit stronger filtering behavior compared to those in the earlier or later layers.

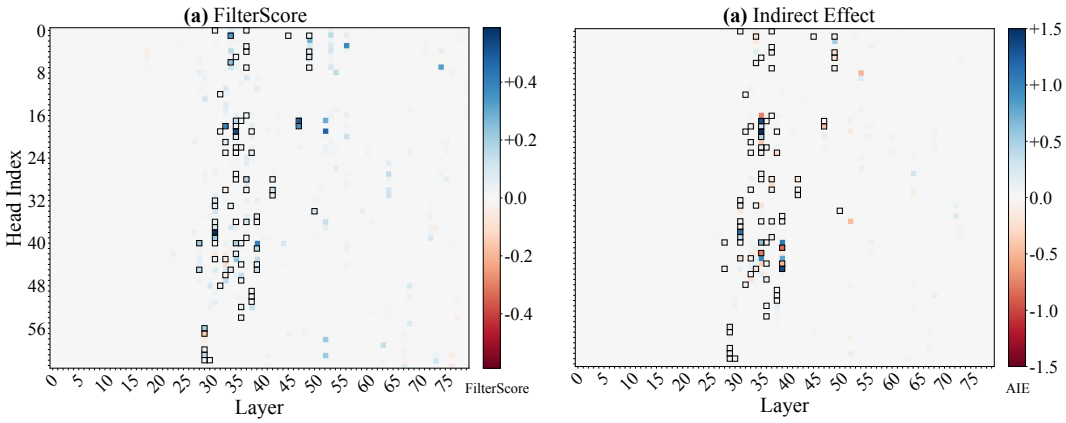


Figure 13: Location of filter heads in Llama-70B. **(a)** shows the individual FilterScore for each head: how much they attend to the correct option over others. **(b)** shows the indirect effect: how much patching q_{src} from a single head promote the predicate target. The filter heads identified with Section 2.3 are marked with black borders.

Activation Patching (w/o DCM). We can patch the attention heads *individually* and quantify their *indirect effect* at mediating the target property in the patched run. Todd et al. (2023) and Feucht et al. (2025) identified Function Vector heads and Concept heads with this approach. Specifically, for each head $[\ell, j]$ we patch q_{src} from the source run to the destination run with the method detailed in Section 2 and check its effect on boosting the score (logit) of the target item, c_{targ}. The indirect effect is measured as $\text{logit}[\leftarrow q_{\text{src}}](c_{\text{targ}}) - \text{logit}(c_{\text{targ}})$.

In Figure 12 we compare the causality of heads identified with the 3 approaches on the *SelectOne* task.

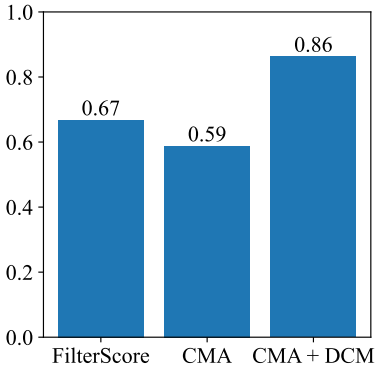


Figure 12: Collective causality of 79 filter heads identified with FilterScore, CMA, and CMA with DCM. Evaluated on the same 512 examples from the *SelectOne* task

D Vector Algebra with Predicate Representation

We explore the geometric properties of the predicate representation q_{src} by examining its behavior under vector arithmetic operations. Specifically, we investigate when we compose two predicates (*find the fruit* and *find the vehicle*) by adding their corresponding q_{src} vectors, does this resulting vector represent a meaningful combination of the two predicates?

Adding predicate representations results in disjunction of the predicates. If we add the q_{src} vectors of two sources prompts with different predicates, $p_{\text{src1}} = \mathbb{P}(\mathcal{C}_1, \psi_1 = \textit{is\_fruit})$ and $p_{\text{src2}} = \mathbb{P}(\mathcal{C}_2, \psi_2 = \textit{is\_vehicle})$, we find that the resulting vector $q_{\text{composed}} = q_{\text{src1}} + q_{\text{src2}}$ can be used on a destination prompt $p_{\text{dest}} = \mathbb{P}(\mathcal{C}_3, \psi_3 \notin \{\textit{is\_fruit}, \textit{is\_vehicle}\})$ to execute the disjunction of the two predicates (i.e., *find the fruit or vehicle*) in $\mathcal{C}_3$. The setup is illustrated in Figure 14 with an example from the *SelectOne* task.

We conduct this experiment for the *SelectOne* task. We compose q_{composed} with two prompts and curate p_{dest} such that there is only item c_{targ} in $\mathcal{C}_{\text{dest}}$ that satisfies the composed predicate $\psi_{\text{src1}} \cup \psi_{\text{src2}}$. We patch q_{composed} to the destination run and consider the intervention to be successful if the LM thinks c_{targ} is the most probable item. We achieve a causality score of 0.6523 (334 out of 512) with logit of c_{targ} increased by 6.75 ± 3.94 after the intervention.

This shows that the predicate representations are compositional, allowing for the construction of more complex predicates through simple vector operations.

E Distinguishing Active Filtering From Answer Retrieval

Our identification of filter heads raises two critical questions about their computational role. First, do these heads actively perform filtering, or do they merely attend to items that were already filtered by earlier layers? Second, do they encode the abstract predicate (e.g., "is a fruit") or simply match

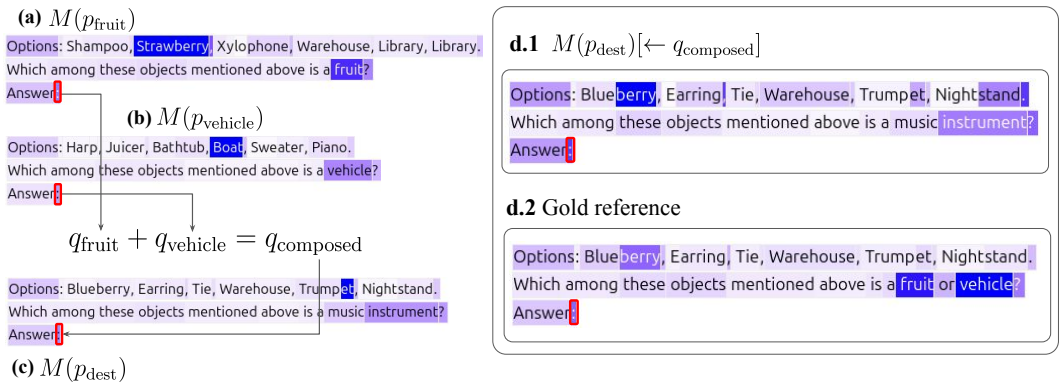


Figure 14: Aggregated attention pattern of the filter heads from the last token position, with the composition setup. The predicate encoding q_{src} is collected from 2 prompts, **(a)** $p_{\text{src1}} = \mathbb{P}(\mathcal{C}_1, \psi_1 = \texttt{is\_fruit})$ and **(b)** $p_{\text{src2}} = \mathbb{P}(\mathcal{C}_2, \psi_2 = \texttt{is\_vehicle})$. Their addition q_{composed} is patched to the destination run. The resulting attention pattern shown in **(d.1)** indicates that now filter heads select the items in p_{dest} that satisfy $\psi_{\text{src1}} \cup \psi_{\text{src2}}$. **(d.2)** shows the attention pattern for a gold prompt with the disjunction predicate.

specific answers from context? To establish that filter heads actively perform filtering rather than passively attending to pre-filtered results, we designed causal intervention experiments where query states carrying predicate information are transferred between prompts. The consistent ability of these transferred queries to trigger the transported filtering operation on entirely different collections demonstrates that the heads actively apply predicates rather than simply reading pre-computed results.

E.1 Do these heads merely match with a pre-computed answer?

A more subtle concern is whether these heads encode abstract predicates or simply store concrete answers. For instance, when the source prompt asks to "find the fruit" with answer "Plum", does the query state encode the predicate "find the fruit" or the specific item "Plum"? In the destination prompt, "Plum" would naturally show higher similarity to "Apple" (another fruit) than to "Watch" (a non-fruit), potentially explaining the selective behavior we observe in the attention pattern.

To resolve this ambiguity, we designed a critical experiment: we use source prompts that contain predicates but no valid answers. We observe that even in such cases, the filter heads retain their high causality of 0.80 (410 out of 512 examples from the *SelectOne* task, $\Delta c_{\text{targ}} = 8.08 \pm 3.1$). Combined with our ablation studies in Section 3.3, these experiments demonstrate the crucial role of filter heads in actively participating in filtering, rather than simply mediating the pre-filtered items.

F Averaging q_{src} to remove the order ID

While in Appendix E we make the case that filter heads encode predicates rather than specific answers, our error analysis reveals that sometimes filter heads do transfer the position of the answer in the source prompt.

When examining cases where our intervention fails, we find that the model sometimes selects the item at the same position as the original answer in the source prompt. Figure 8 also shows elevated scores for items matching the source answer's position c_{oid} in critical layers, although not as high as c_{targ}. This suggests that the LM also encodes the positional information or *order ID* (Feng & Steinhardt, 2023; Prakash et al., 2025) of c_{src} alongside the predicate. A probable explanation for this is: as these filter heads are distributed across a range of layers, patching q_{src} from filter heads in later layers may bring over specific contribution from the filter heads in earlier layers.

To isolate the predicate signal from this positional bias, we use a simple trick: averaging the query states across multiple source prompts. We produce n variations of the same source prompt $p_{\text{src}} = \mathbb{P}(\mathcal{C}_{\text{src}}, \psi_{\text{src}})$ by changing the index of the correct answer c_{src}. Figure 15 illustrates this idea. This trick improves our causality scores, also shown in Figure 15.

```
Options: Cherry, Knife, Pen, Ambulance.
Which one is a fruit in this list?
\nAns:
```

```
Options: Knife, Cherry, Pen, Ambulance.
Which one is a fruit in this list?
\nAns:
```

```
Options: Knife, Pen, Cherry, Ambulance.
Which one is a fruit in this list?
\nAns:
```

```
Options: Knife, Pen, Ambulance, Cherry.
Which one is a fruit in this list?
\nAns:
```

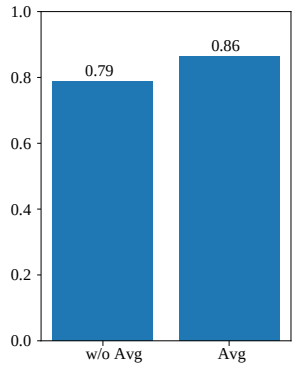


Figure 15: Averaging q_{src} to remove the order ID information. This simple trick improves the causality scores by 7 to 10 points across the board. Causality scores presented for 512 examples from *SelectOne* task.

G Adding a Priming Prefix helps with Causality.

Table 1 shows that filter head causality varies across information types for the same *SelectOne* task. Notably, tasks requiring recalling a person's nationality and the location of a landmark show lower causality scores.

Following Amini & Ciaramita (2023), we provide contextual priming and check the causality in these cases. In a question-after format, if before presenting the items we add a prefix that explicitly instructs the LM to recall relevant information, we can achieve approximately a 10-point improvement in causality scores. This experiment further validates the hypothesis that filter heads work better when the relevant semantic information required for filtering is already present in the item latents.

Table 6: Priming the context helps improve the causality score.

Filtering Task	W/O Priming	With Priming	Priming Prefix
Person Nationality	0.504 (258/512)	0.625 (320/512)	`Recall the nationality of these people:\n`
Landmark in Country	0.576 (295/512)	0.670 (343/512)	`Recall which country these landmarks are located in:\n`

H Summary of Results in Gemma

We replicated all of our experiments for Gemma-27B and we observe that the scores mostly align with Llama-70B scores. At this moment we don't have all the scores for Gemma-27B. We will include them in future revisions.

H.1 Within Task Generalizability

Table 7: Causality of filter heads on *SelectOne* tasks. Heads identified using object-type filtering generalize to semantically distinct predicates like profession identification. Compare with Llama-70B scores in Table 1.

Filtering Task	Causality	Δlogit	With Priming
Object Type	0.824	+9.95	-
Person Profession	0.770	+9.10	-
Person Nationality	0.305	+5.98	0.404
Landmark in Country	0.410	+6.48	0.455
Word rhymes with	0.018	+0.12	0.037

H.2 Across Task Generalizebility

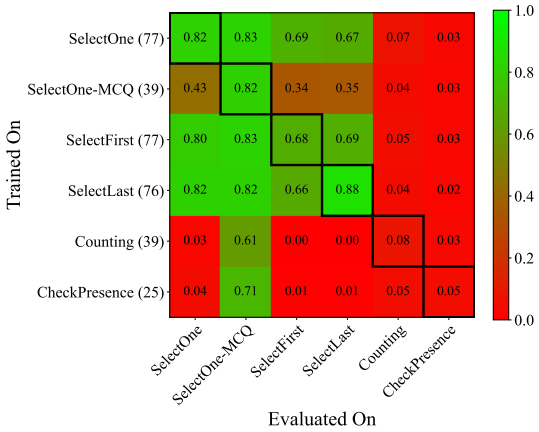
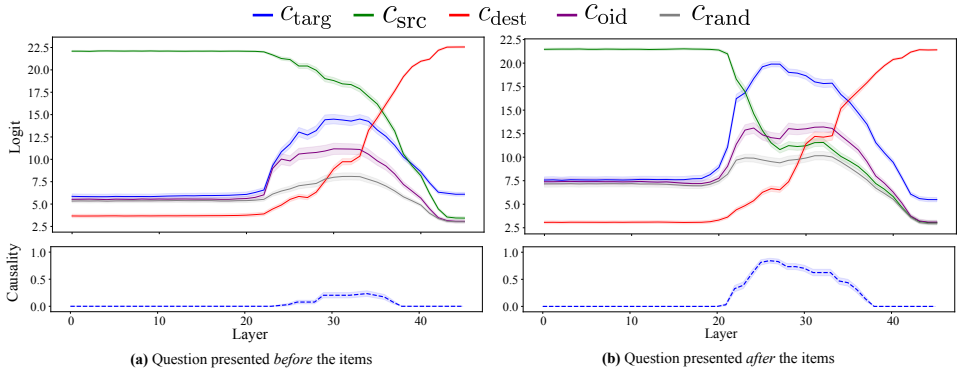


Figure 16: Task generalizability of the filter heads. Compare with Figure 3a.

H.3 Dual Implementation in Question Before vs Question After



(a) Question presented *before* the items **(b)** Question presented *after* the items

Figure 17: Effect of patching the residual latents in Gemma-27B. Compare with Figure 8.

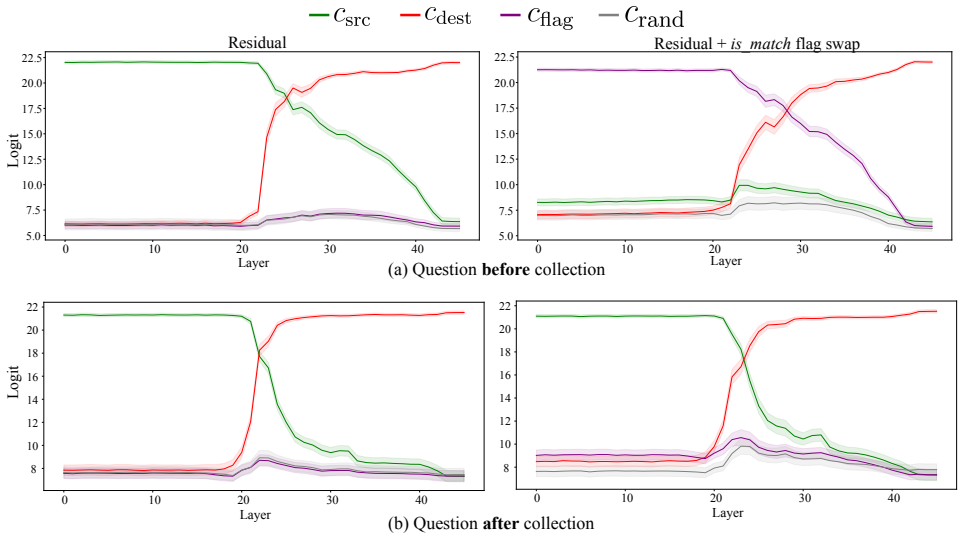


Figure 18: Effect of swapping the *is_match* flag between items in Gemma-27B. Compare with Figure 11.

I USAGE OF LLMS

Proprietery LLMs services with black-box access such as claude.ai and gemini.google.com were used as a general purpose assist tool, which is allowed as per the ICLR 2026 author guideline. We have used such LLMs to polish some of the writings in this paper. We have also used LLMs to get more items for our dataset and translating the prompts to other languages.

J LIMITATIONS

Our investigation of filtering mechanisms in LLMs, while revealing important insights, has several limitations

Task Coverage. We examined only six filter-reduce tasks, which may not capture the full diversity of filtering strategies employed by LLMs. Even within our six tasks we identified that the filter heads do not show high causality in the *CheckPresence* task, indicating that the LM uses alternate mechanism for certain filtering operations. The consistent prompt templates we used enable us to scale up our controlled experiment setup, but they may have biased us towards specific computational strategies inside the LM. LMs may adapt their filtering approach based on what information is available in ways that our limited task set and prompting strategies cannot fully reveal.

LM Coverage. We identified filter heads in Llama-70B and Gemma-27B models. The fact that we were able to identify similar mechanisms in two models of different sizes, from different families, trained on different datasets echos the idea of Evolutionary Convergence from Morris (2006): distantly related organisms (e.g. vertibrate and cephalopods) independently evolve similar adaptations (e.g. eyes) in response to similar environmental pressure.

However, such convergence is not guaranteed across all LMs. Notably, findings from Zhong et al. (2023) suggest that identical architectures trained on different datasets can potentially develop different implementations for the same operation. LMs from other families may develop a mechanism that does not make use of such filter heads. Additionally, we restrict our analysis to fairly larger LMs. It is possible that smaller LMs, where parameter constraints might enforce higher head-level superposition, entangle the predicate representation may get entangled with other unrelated stuff in the query states. Therefore, we might not see the distinct head-level causal role of filter heads we get in larger LMs.

Implementation. Our tasks are designed such that we can determine whether an answer is correct based on what the LM predicts as the next token. While curating a prompt we ensure that none of the items share a first token with each other. In all of our experiments we also ensure that the answer of the source prompt, destination prompt, and the target answer of the transported predicate are all different, in order to ensure that patching does not merely copy over the answer. However, this choice of validating with only the first predicted token has potentially restricted us from exploring other tasks that have a similar filter-reduce pattern of operation. Most of the causality scores we report in this paper were calculated on a single trial with a 512 examples sampled randomly. It is possible that these scores will change slightly on a different trial.

K QUALITATIVE ATTENTION PATTERNS OF FILTER HEADS

K.1 TYPICAL FILTER HEAD BEHAVIOR

The following are cases where the model clearly attends to the option which corresponds to the given predicate.

Select One

Options: Cucumber, Lion, Pencil, Tie, Air fryer.
Which among these objects mentioned above is a clothing?
Answer:

Select One (MCQ)

a. Yoga mat
b. Projector
c. Xylophone
d. Banana
e. House
Which among these objects mentioned above is a fruit?
Answer:

Select First

Options: Birch, Rice cooker, Ash, Tomato, Zucchini.
What is the first tree from the list above?
Answer:

Select Last

Options: Rabbit, Harmonica, Accordion, Blueberry, Cabinet, Sheep.
What is the last animal in this list above?
Answer:

Counting

Items: Blueberry, Church, Pear, Apartment, Banana
How many buildings are in this list?
Answer:

Check Existence

Items: Bike, Microphone, Surfboard, Rose, Toothpaste
Do you see a electronics in the list above?
Answer:

K.2 Filter Heads are Sensitive to Semantic Information

The following are cases where the model is prompted with tasks which require non-semantic knowledge, and the the filter head attention pattern is noisy.

Rhymes With

Options: brown, fake, that, flight.
Which of these words rhymes with hat?
Answer:

Number of Letters

Options: fake, pat, drown, row.
Which word has exactly 4 letters?
Answer:

First Alphabetically

Options: bring, play, drown, doe, set.
Which word comes first alphabetically?
Answer:

Last Alphabetically

Options: cake, flee, low, flight.
Which word comes last alphabetically?
Answer:

K.3 More Examples from the Application

We demonstrate a practical utility of filter heads in detecting the presence of certain concepts in the text: identifying false information and detecting sentiment.

We curate a paragraph/free-form text that mix factual and false statements about a topic. We break the text into sentences and then append a question asking the LM to identify the false information. We visualize the aggregated attention pattern of the filter heads, which focus on the last token of the false statements. To aid visualization we draw a red line underlining the sentences whose last token gets attention score exceeding a preset threshold.

We apply the same approach to sentiment analysis using movie reviews containing both positive and negative sentences.

Lie Detector 1

Carl Sagan was an American football player.
He is best known for his work on the television series 'Cosmos'.
Sagan made significant contributions to the field of planetary science.
He won the Pulitzer Prize for his book "The God Delusion".
Sagan's work inspires scientists around the world.

Lie Detector 2

ET is a movie about an alien who is stranded on Earth.
He befriends a young boy named Elliott and they form a close bond.
Together, they embark on a journey to help ET return home.
It is considered a classic in the romantic comedy genre.
The film explores friendship, and the importance of family.
ET's appearance and quotes made it a beloved film for all ages.
The movie was directed by Christopher Nolan.

Negative Sentiment Detector 1

Jaws is a boring movie about a shark terrorizing a town.
The music is iconic and builds suspense well.
The effects still hold up as moving today.
The characters aren't memorable and are poorly-acted.
The final scenes are thrilling and satisfying.
It's a classic that still holds power.

Negative Sentiment Detector 2

ET is a heartwarming film about a boy and an alien.
The story is simple and easy to follow.
Some scenes feel slow and outdated.
The music is beautiful and memorable.
ET's design is strange and off putting.
The acting from the kids is very strong.
The ending is memorable and emotional.

L FILTER HEADS IN LLAMA-70B

Layer	Head	Indirect Effect
35	19	3.546021
39	45	1.353394
35	17	1.306396
31	38	1.114380
35	40	0.611328
35	20	0.443115
31	39	0.340698
35	18	0.281738
29	56	0.208984
42	31	0.154541
28	40	0.151733
29	61	0.141235
36	47	0.128540
34	6	0.110474
37	30	0.106079
35	23	0.104248
31	33	0.098511
33	18	0.098145
29	57	0.076416
37	39	0.073608
34	33	0.068726
35	27	0.052734
35	28	0.052734
28	45	0.050049
33	30	0.042725
39	35	0.038818
38	19	0.028809
38	49	0.022949
36	44	0.022461
36	17	0.018066
50	34	0.017456
36	54	0.017090
37	36	0.016479
37	16	0.011108
36	52	0.010376
36	22	0.003052
32	12	0.001953
38	51	-0.001221
45	1	-0.001953
37	7	-0.003296
35	5	-0.003418
39	36	-0.003418
30	62	-0.004272
32	48	-0.004395
31	40	-0.005005
35	36	-0.011719
32	19	-0.013794
33	23	-0.014160
33	46	-0.017456
37	3	-0.021362
29	62	-0.030029
47	17	-0.036133
31	0	-0.036743
38	50	-0.042847
42	30	-0.043091
31	37	-0.055786
37	0	-0.061890
33	21	-0.063599
37	4	-0.067749
36	40	-0.068481
49	1	-0.069824
35	22	-0.079346
29	60	-0.085938
49	7	-0.087402
33	43	-0.096558
31	36	-0.100586
31	32	-0.109375
49	5	-0.162842
42	28	-0.172974
31	43	-0.183960
37	28	-0.188232
49	4	-0.216553
34	1	-0.247803
38	23	-0.250610
34	45	-0.252563
47	18	-0.437988
39	44	-0.486816
35	42	-0.770020
39	41	-0.843811

Table 8: Indirect effect scores for filter heads, sorted in descending order.