When Models Decide and When They Bind: A Two-Stage Computation for Multiple-Choice Question Answering

Hugh Mee Wong, Rick Nouwen, Albert Gatt
Utrecht University
Utrecht, The Netherlands
{h.m.wong, r.w.f.nouwen, a.gatt}@uu.nl

Abstract

Multiple-choice question answering (MCQA) is easy to evaluate but adds a meta-task: models must both solve the problem and output the symbol that *represents* the answer, conflating reasoning errors with symbol-binding failures. We study how language models implement MCQA internally using representational analyses (PCA, linear probes) as well as causal interventions. We find that option-boundary (newline) residual states often contain strong linearly decodable signals related to per-option correctness. Winner-identity probing reveals a two-stage progression: the winning *content position* becomes decodable immediately after the final option is processed, while the *output symbol* is represented closer to the answer emission position. Tests under symbol and content permutations support a two-stage mechanism in which models first select a winner in content space and then bind or route that winner to the appropriate symbol to emit.

1 Introduction

Multiple-choice question answering (MCQA) is pervasive in natural language processing (NLP) benchmarks, in part because of its ease of evaluation compared to long-form generation: assessing a model's response can often be reduced to inspecting the token (distribution) immediately following the user's prompt. However, reframing NLP tasks as MCQA prompts also introduces an additional meta-task, over and above the NLP task itself. Specifically, a language model must both determine the correct answer to the target question and recognize that it should output the symbol that *represents* that answer. Consequently, when a model selects an incorrect option, this failure may stem from a misunderstanding of the target question, the MCQA format, or both.

Prior work has shown that language models exhibit selection biases for particular option identifiers such as the label "A" (Zheng et al., 2024; Dominguez-Olmedo et al., 2024; Li and Gao, 2025; Xue et al., 2024). They may also be sensitive to the ordering of answer choices (Pezeshkpour and Hruschka, 2024), the specific symbols used to denote the options (Yang et al., 2025b; Wiegreffe et al., 2025), and the length of answer options (Zhao et al., 2025). Furthermore, the answer obtained from comparing probabilities of the first tokens in MCQA is often not consistent with the long-form generation output (Wang et al., 2024; Tsvilodub et al., 2024).

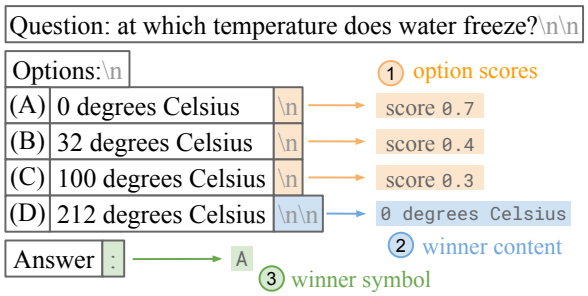


Figure 1: Overview of this work's main findings. We show that (1) an option-boundary state (the newline following an MCQA option) often contains highly linearly decodable information about an option's correctness. We further find that (2) the winning content (position) is already represented once the model has processed the final option, whereas (3) the identity of the *symbol* emerges later. The prompt in this example is drawn from the ARC-Easy dataset (Clark et al., 2018).

These issues underscore the need to better understand *how* a language model solves MCQA tasks, separating a model's task-relevant knowledge from artifacts introduced by the MCQA format itself, especially *symbol binding*, where a model has to associate an answer symbol with its corresponding option text. Mechanistic interpretability (MI) methods can be used to identify why and how answer options are represented, as well as how symbols such as option identifiers are bound to their semantic content, and how these representations are

Findings of the Association for Computational Linguistics: ACL 2026, pages 22804–22821

ultimately converted into a final decision.

Prior MI work on MCQA has identified attention heads as key contributors to predicting the answer symbol: Lieberum et al. (2023) highlight "Correct Letter Heads" that attend from the final token to the correct symbol token, while Wiegreffe et al. (2025) localize symbol-predictive signals in mid-layer heads. However, because these analyses intervene primarily at the last token position, they may miss binding dynamics if the model has already committed earlier. Leveraging results on binding mechanisms in entity tracking (Feng and Steinhardt, 2024; Dai et al., 2024) tasks, Mueller et al. (2025) find evidence that models compute an index for the correct option and then dereference it to emit the corresponding symbol.

In this work, we focus on when and where models represent the correct option and its symbol. We combine representational analyses (PCA and linear probing) with causal interventions (activation patching) to separate three stages of computation: per-option correctness evaluation, global option winner selection, and symbol binding at the answer-emission site.

Our contributions are as follows (Figure 1).[1]

- We perform representational analyses which show that models track both the position of symbols, and whether the current or previously seen option is correct. We further show that an option-boundary state (i.e., the newline after any MCQA option) often contains highly linearly decodable information related to the correctness of the option.
- Training 4-way winner-identity probes (predicting the symbol that the model outputs) and evaluating them under symbol permutations and content permutations yield a clear disassociation: the winner *content position* is already represented right after the model has processed the last option, while the identity of the *symbol* emerges later.
- We find further support for this dissociation by performing causal interventions using probe-aligned patching.

Together, these results suggest that language models use a two-stage mechanism in which they first commit to a winning option before they are prompted to respond and subsequently route/bind that winner tot he appropriate symbol closer to the point of answer emission.

[1]We release our code at https://github.com/hughmee/mcqa-binding.

2 Related Work

Symbol emission Lieberum et al. (2023) identify *Correct Letter Heads*, attention heads that attend heavily from the final token to the correct label. Low-rank analyses suggest that these heads may encode information about a letter (symbol) being the n-th item in a list. As such, these heads promote the chosen symbol based on its position in the symbol ordering. Wiegreffe et al. (2025) use vocabulary projection (*logit lens*) and activation patching at the last token position to show that attention heads in middle layers play an important role in the prediction of an answer symbol.

Binding in language models Symbol binding has been a challenging problem in both neuroscience (Burwick, 2014) and neural networks (Von Der Malsburg, 1999). A well-known example of binding can be found in *entity tracking* (Kim and Schuster, 2023), where a model must bind an entity to its attributes to later retrieve the correct attribute. Feng and Steinhardt (2024) remark that these associations are made in context, so binding must occur in the activations of the language model instead of being stored as facts in the weights. They propose the *binding ID* (BI) mechanism: the model assigns abstract IDs to entities and attributes, and bound pairs share an ID that can be referenced later. Dai et al. (2024) extend this line of work by capturing the *ordering ID* (OI), the input order in which the entities and attributes appear in the context. They show that OI is encoded in a low-rank activation subspace that influences in-context binding.

Building on this, Mueller et al. (2025) argue that MCQA may rely on ordering IDs: the model identifies the answer's position in the option list and then dereferences that index to emit the corresponding symbol, echoing a hypothesis by Lieberum et al. (2023). Lieberum et al. (2023)'s preliminary experiments to analyze the outputs to the Correct Letter Heads suggest that models may use "Content Gatherers", which attend from the final tokens to the last token of the correct option content token. In contrast, Mueller et al. (2025) report that order information shifts from the correct symbol token to the last token position in middle layers.

3 Experimental Setup

Task definition Following Wiegreffe et al. (2025) and Mueller et al. (2025), we use a synthetic 4-way MCQA task to disentangle a model's task-specific knowledge from MCQA behavior. Prior work relies on the Copying Colors from Context (*Colors*; Wiegreffe et al., 2025) dataset, adapted from the Memory Colors dataset introduced by Norlund et al. (2021). Its questions target prototypical colors of objects and are of the form "A banana is yellow. What color is a banana?" Because such facts may be stored in model weights, the way questions are framed obscures the role of symbol binding, which is done in context rather than retrieved (Feng and Steinhardt, 2024). To isolate MCQA behavior from factual recall, we modify the test set of the Colors dataset (4-way MCQA, 100 samples) by replacing the specific object names with "*this object*", as in the following example.

```
Question: This object is white. What color
is this object?

Options:
(A) white
(B) grey
(C) black
(D) brown

Answer: (
```

Given that the color copying task is rather trivial–the correct option is explicitly mentioned in the prompt–we additionally experiment with the test set of the AI2 Reasoning Challenge Easy (ARC-Easy; Clark et al., 2018), which consists of multiple multiple-choice elementary and middle-school science questions. We restrict to four-option items (2365 samples in total), convert any numeric labels to letters, and use the same `Question-Options-Answer` prompt format as in the color-copying setup.

Models We experiment with two models: Llama-3.2-Instruct-3B (28 layers; Meta AI, 2024; Grattafiori et al., 2024) and Qwen3-8B (36 layers; Yang et al., 2025a).

4 Finding Scoring Signals

Following Dai et al. (2024), we use principal component analysis (PCA) to find subspaces in the residual stream that encode the position of a symbol (such as *A, B,* ...) or content (*white, black,* ...). For each MCQA prompt, we extract the d-dimensional activations at the token positions for each of the model's n MCQA symbols and contents. This results in an activation matrix $M \in \mathbb{R}^{n \times d}$, for which the singular value decomposition can be written as $M = U\Sigma V^T$. We focus on the first two principal directions in V.

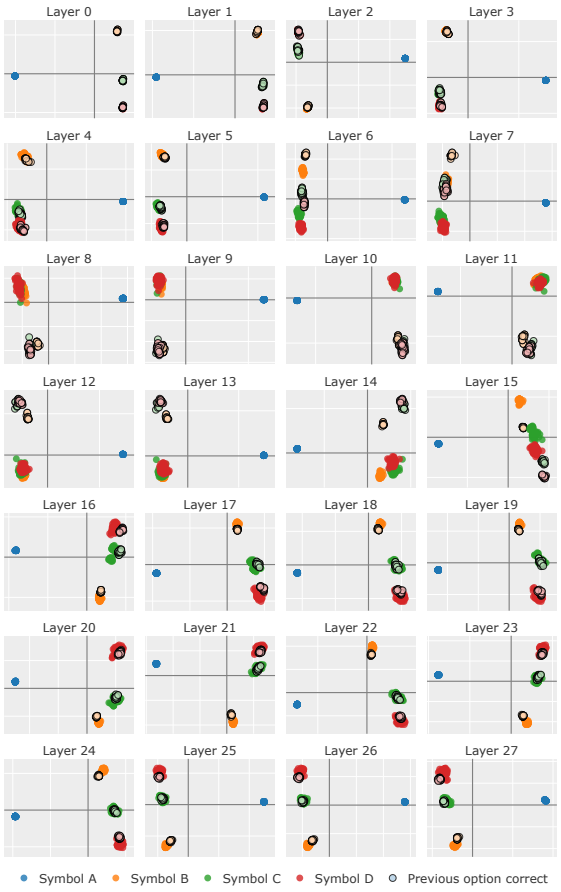


Figure 2: The first (x-axis) and second (y-axis) principal components from layerwise PCA in **Llama-3.2-3B**. Activations extracted at every **symbol** token position. Each color represents an MCQA symbol from the options. Markers with a black border (labeled "Previous option correct") indicate that, when processing symbol X, the preceding option $X-1$ had already been identified as correct. For instance, in layer 14, activations for symbol `B` (orange) separate depending on whether `A` was correct (orange markers with black borders) or incorrect (orange markers without black borders).

Distinction between symbols Figure 2 shows the PCA for Llama-3.2-3B-Instruct (see Appendix B.1 for Qwen-8B). Activations cluster by symbol position, indicating that models internally distinguish

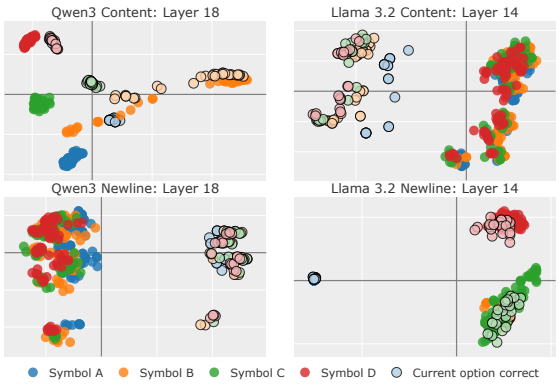


Figure 3: The first two principle components for Qwen3-8B (**left**) and Llama-3.2-3B-Instruct (**right**) when extracting the residual stream at the content tokens (**upper**) and subsequent newlines (**lower**). Note that, unlike Figure 2, this plot tracks the *current*, not the *previous* token correct. Plots for all layers in Appendix B.3.

positional information. In particular, the first principal component separates the first option from later ones. To test whether this effect is merely positional, or tied to the literal symbol "A", we permute option labels and re-run PCA (Appendix B.4, Figures 10 and 11). For Qwen3-8B, this permutation substantially alters the structure of the representation space, suggesting an entanglement between positional information with the semantic (alphabetical) ordering of the option symbols. In contrast, the representations of Llama-3.2-3B-Instruct appear to be more strongly driven by the ordering of options in the list.

In both models, the first option remains atypical. One explanation is that, when processing the first element of a list, the lists consists of a single element and no comparison with other items is possible. Consequently, the representation may not yet encode an explicit "n-th item" signal. A similar effect appears in the OI subspace visualizations of Dai et al. (2024), where the first item behaves differently despite the existence of an overall low-rank ordering direction.

Tracking of correctness The first option may look different because the model could track whether a previously seen option is correct. In an autoregressive language model, options are processed sequentially, so any "winner-so-far" signal must be stored *after* the option deemed correct; the first position cannot encode a prior-correctness trace. The idea that a symbol may store information about a *previous* answer has also been briefly suggested by Lieberum et al. (2023).
Figure 2 supports this "correctness tracking" hypothesis: within each symbol cluster, there is further separation of points depending on whether the preceding option was correct (e.g., "B" correct while processing "C"). The effect is most pronounced in middle layers, which have been associated with feature construction (Lad et al., 2024). Figure 3 suggests that this "correctness" signal may arise even earlier, at the option content token or the newline marking the option boundary.

ARC-Easy Given the trivial nature of the color copying task, the apparent separation between the correct and incorrect answers in Figure 2 and 3 may simply reflect whether a given color appeared in the question, rather than whether the option is actually correct. In other words, the signal may be tracking an option's presence in the prompt instead of its correctness. We repeat the subspace visualization using the ARC-Easy dataset. Given that ARC-Easy is less structured than the color copying dataset, it is to be expected that its patterns are less readily available in PCA space. That said, we can identify layers in Figures 17 and 16 (Appendix B.4) that suggest some degree of separability between correct and incorrect options.

5 Probing Correctness Scores

PCA offers a qualitative picture of how correctness-related variation may be structured in the residual stream: across layers and token positions, activations exhibit low-dimensional patterns consistent with a correct/incorrect separation. Based on this exploratory analysis, and noting that PCA captures directions of maximal variance that need not align with task-relevant variables, we perform more targeted tests. We use linear probes (Alain and Bengio, 2017, aka *diagnostic classifiers*; Hupkes et al. (2018)) to quantify correctness/scoring effects suggested by our PCA visualizations.

Notation Let $l \in \{0, \ldots, L\}$ be the layers of a language model and d the hidden size. For each prompt i and option $X \in \{A, B, C, D\}$, let $p_{i,X}$ be a token position associated with that option (e.g., the symbol token, or an option-content token). Let $r^{(l)}_{i,X} \in \mathbb{R}^d$ be the residual stream vector at layer l at position $p_{i,X}$. For each option X in sample i, we define $y_{i,X} = 1$ if X is the correct option in sample i and $y_{i,X} = 0$ otherwise. A linear probe at layer l is a parameter vector $w^{(l)} \in \mathbb{R}^d$ and bias $b^{(l)} \in \mathbb{R}$,

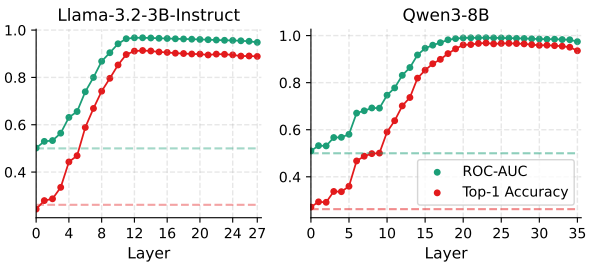


Figure 4: Layer-wise performance of **binary per-option probes** trained for the **ARC-Easy** dataset with $C = 10^{-4}$ on the residual stream. The dashed lines represent random chance baselines.

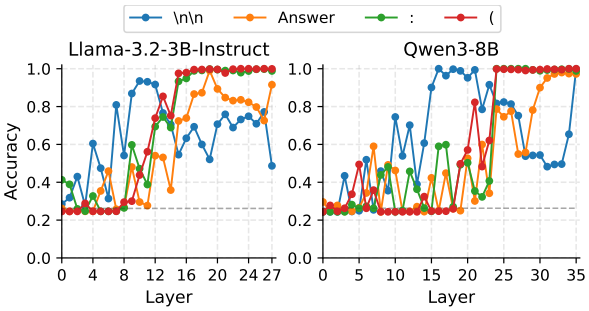


Figure 5: Layer-wise performance of **multinomial probes trained** for the **ARC-Easy** dataset with $C = 10^{-2}$. Each line corresponds to token positions of `\n\nAnswer: (`, right after the model has processed all the MCQA options. Dark gray dashed lines represent random chance baselines.

which defines a classifier $\hat{y}_{i,X} = \sigma((w^{(l)})^T r^{(l)}_{i,X} + b^{(l)})$, where $\sigma(\cdot)$ is the logistic sigmoid. Hence, a learned linear function $w^T r + b$ that best predicts whether an MCQA option is correct according to the language model.

We train linear probes (on 90% of ARC-Easy; 10% for testing) to address (1) whether the model encodes *per-option correctness* as it processes the options; and (2) whether it represents the winning content and eventual output symbol *before* the final answer-emission token.

5.1 Per-option Correctness

To test whether *per-option correctness* is linearly decodable from the model's internal representations, for each MCQA prompt we extract the residual-stream activation at the end-of-option boundary (e.g., `\n`, `.\n`, or `\n\n`) and train an L2-regularized logistic-regression probe to predict whether the option is the one the model will select as correct. Probes are trained on ARC-Easy to reduce the risk that apparent signals reflect trivial lexical overlap between the question and an option. We evaluate on held-out ARC-Easy and the synthetic color-copying set. Because options from the same question are correlated, we use question-ID grouped splits and tune the inverse regularization strength C via 10-fold GroupKFold cross-validation.

Results To account for the 1:3 class imbalance (one correct vs. three incorrect options per question), we use ROC-AUC as our primary metric. For each held-out question, we score all four options using the probe and also predict the answer as the option with the largest probe score; top-1 accuracy is then the fraction of questions for which this operation recovers the true correct option (the language model's output). This metric captures whether the probe's scores are informative *within* each question, rather than only in aggregate across the dataset.

Figure 4 shows that both AUC and top-1 accuracy improve with depth: probes are weak in early layers but rise through the middle and remain strong late in the network. Compared to PCA (Section 4), which reveals a correctness separation correctness in intermediate layers, probing indicates that correctness remains linearly decodable even in final layers. A plausible interpretation is that late layers reorganize the representation: rather than maintaining correctness as a dominant global axis of variation, the model may transform it into a more "decision-like" representation. Under such a reorganization, unsupervised PCA projections can understate separability even as the supervised decoding task becomes easier.

The ARC-Easy probe direction also transfers to the color copying task. Freezing w and scoring newline activations with $s(x) = w^T x$ yields near-perfect separation: option-level AUC and top-1 accuracy both reach 1.0 across middle-to-late layers (layers 12-27 for Llama-3.2-3B-Instruct;layers 20-35 for Qwen3-8B). Overall, this indicates a robust per-option ranking signal, a structured "option evaluation" feature in the residual stream that generalizes across MCQA datasets with the same format.

5.2 Winner Identity Probe

To test whether the model consolidates option-level evaluations into a single global decision prior to answer emission, we train a 4-way winner identity probe. For each prompt, we label the correct option index $y \in \{A, B, C, D\}$ and train a multinomial linear classifier on residual stream activations that occur after *all* options have been processed: the

newlines after the final option (the *end-of-options checkpoint*) and the `Answer: (` prefix. By sweeping layers and checkpoints, we estimate when winner identity becomes linearly decodable: decodability at the end-of-options checkpoint supports *early commitment*, whereas emergence only near answer emission supports late consolidation or binding.

Results For both Llama-3.2-3B-Instruct and Qwen3-8B, winner identity exhibits a consistent two-stage progression (Figure 5). First, it becomes reliably decodable at the end-of-options checkpoint in intermediate layers, reaching high accuracy before the model encounters the answer prefix. Second, decodability at answer prefix tokens remains weak until later layers, where it rises sharply and often saturates. This pattern is consistent with winner information being available early in the computation but only being made accessible at specific positions that directly govern the next-token distribution over answer symbols.

The contrast across checkpoint tokens refines this interpretation. The end-of-options boundary tends to show earlier decodability, while the `Answer: (` region shows the strongest late-layer decodability, which is in line with its role as immediate context for symbol emission. The end-of-option newline curves being non-monotonic suggests relocalization rather than simple accumulation: once winner identity has been actively written into the answer region by attention heads, it would no longer need to remain cleanly separable at this earlier checkpoint. Figure 8 (Appendix A) shows that these results, too, are generalizable to the color copying task.

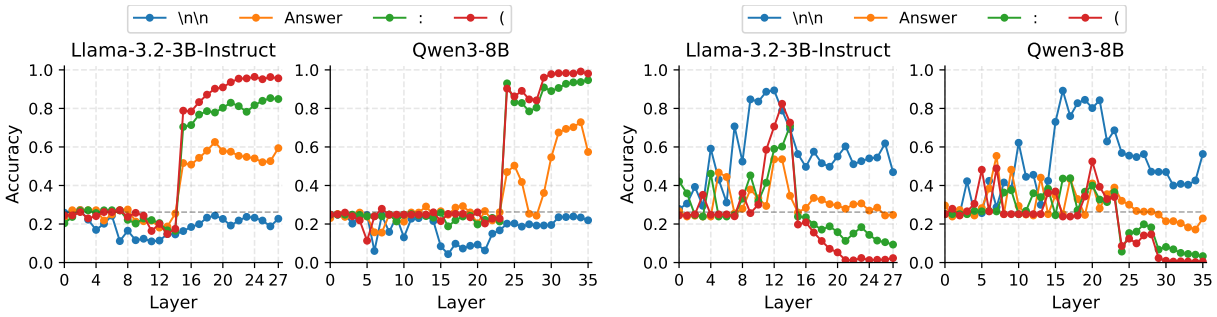
We further test whether the probes are reading out a symbol-coded decision (the symbol/letter to emit) or a content-coded decision (which option content or position is correct) using symbol-permutation generalization without retraining the probes (Figure 6). Concretely, for each prompt we randomly reassign the option symbol while leaving the order of the textual options unchanged; this breaks the alignment between "the winning option content" and "the winning symbol." We then evaluate the same permuted prompts under two target definitions. When the targets reflect the new correct (permuted) symbol, (Figure 6a), performance at the end-of-options checkpoint collapses across all layers, while accuracy in the answer region increases late and approaches ceiling. Conversely, when the targets are (we probe for) the content (position) of the winner (Figure 6b), performance remains strong at end-of-options checkpoints in middle layers but collapses near emission.

This behavior is to be expected in a two-stage MCQA mechanism: permuting labels breaks the mapping between the winning content and the symbol that represents it. Thus, a representation that primarily encodes the winning option as san index-like variable will remain mostly intact while symbol decodability drops at the end-of-options checkpoint. If binding had already been completed at this point, symbol identity would be robust to renaming. However, under this symbol permutation schema, symbol information only becomes reliably decodable later, near the colon in the prompt that triggers answer emission. Together, these results support a two-stage mechanism: the model first selects a winner in option/content space immediately after reading the candidates, then performs a late binding/routing step during the answer prefix that maps the winner identity onto the correct output symbol under the current symbol-content assignment.

6 Causal Interventions

To understand whether the signals found in previous sections are *causally relevant*, we perform a series of activation patching experiments. *Activation patching* (Vig et al., 2020; Meng et al., 2022; Geiger et al., 2020; Wang et al., 2023) is a widely used in MI to causally attribute specific model behavior to model components (e.g., specific attention heads). It requires two prompts: a *clean* prompt, on which the model exhibits some desired behavior, and a *corrupted* (or *counterfactual*) prompt that is a modified version of the original prompt. The model is first run on both prompts, and the activations (e.g., MLP output) of the clean prompt are cached. The model is then re-run on the corrupted prompt while a chosen subset of its activations are patched with those from the clean run.[2] By comparing the output of the patched corrupted run with the original corrupted run, we can quantify the causal contribution of the selected components to preserving or restoring the clean behavior. It must be noted that in the case of activation patching, the corrupted/counterfactual prompt essentially defines the task (Miller et al., 2024). In our experiments, we make use of two kinds of counterfactuals, either

[2]Note that it is also possible to patch activations from a corrupted prompt into a clean run without it necessarily being the symmetric case of patching from the clean into the corrupted prompt (Zhang and Nanda, 2024).



(a) Probing the correct **symbol**

(b) Probing the correct **content**

Figure 6: Performance of the multinomial probe from Figure 5 on ARC-Easy prompts in which option **symbols have been permuted**. We probe for **(a)** the correct output symbol under this permutation as well as for **(b)** the correct (unchanged) content position.

swapping the correct symbol with a different one, or swapping the correct content.

Metric We evaluate our patching results using the *faithfulness* metric (Wang et al., 2023) averaged over all samples:

$$\text{Faithfulness} = \frac{\ell_{\text{patched}} - \ell_{\text{corrupt}}}{\ell_{\text{clean}} - \ell_{\text{corrupt}}}, \tag{1}$$

where ℓ_x denotes the logit for the clean answer token in run x.

6.1 Binary Probe Patching

To connect the representational probes from Section 5.2 to causal tests, we perform *probe-aligned patching*, a targeted activation-editing procedure that directly intervenes on a one-dimensional subspace of the residual stream. In other words, we perform a form of subspace patching also known as *directional activation patching* (Tigges et al., 2024; Geiger et al., 2024) using the probe direction in the residual stream.

Let w be a unit-norm probe direction, and consider a clean and corrupted run. At layer l, position i, and option X, define:

$$s_{\text{clean}} = w^T r^{(l)}_{i,X,\text{clean}}, \qquad s_{\text{corr}} = w^T r^{(l)}_{i,X,\text{corrupt}}.$$

Probe-aligned patching replaces only the probe-aligned component of the corrupted residual stream with the corresponding value from the clean run, while leaving the orthogonal complement unchanged. For the binary probes, this means:

$$r^{(l)}_{i,X,patched} = r^{(l)}_{i,X,\text{corrupt}} + (s_{\text{clean}} - s_{\text{corr}})w. \tag{2}$$

This update is the unique minimal-norm modification that enforces $w^T r^{(l)}_{i,X,patched} = s_{\text{clean}}$. It matches the clean run's coordinate along w while preserving all other components of $r^{(l)}_{i,X,\text{corrupt}}$. Intuitively, this intervention "edits" the amount of the probed feature present in the corrupted activation without overwriting unrelated information.

Patch positions When we patch activations at the newline immediately following an option's content, there are therefore two natural ways to decide where a clean option's activation should be inserted into the counterfactual prompt: *content-aligned* patching, which patches a clean option into the counterfactual option that contains the same content, and *symbol-aligned* patching, which patches a clean option into the counterfactual option that carries the same symbol. We apply the chosen alignment consistently to both the true (clean) winner and the option it is swapped with, preserving their relative relationship under the intervention. Intuitively, content-aligned patching tests whether the relevant signal is attached to the option's content as it moves across labels, while symbol-aligned patching tests whether the signal is attached to the symbol itself.

Results Table 1 compares the content-aligned and symbol-aligned patching results. Across both counterfactual regimes, probe-aligned patching of the 1D "correctness" feature is markedly more faithful under symbol-aligned correspondence than under content alignment, indicating that this low-dimensional feature transfers most effectively when matched by the symbol rather than by the option content. Full-vector patching at the same newline positions reveals a complementary structure: when the content is permuted, content-aligned full patching is substantially higher than symbol-aligned, consistent with the full residual stream

	Llama-3.2-3B-Instruct		Qwen3-8B	
	Moved symbol	**Moved content**	**Moved symbol**	**Moved content**
Content-aligned	−0.01 (1.02)	−0.281 (0.702)	0.047 (0.760)	−0.003 (0.713)
Symbol-aligned	0.870 (0.937)	0.747 (0.178)	0.450 (0.717)	0.380 (0.910)

Table 1: Faithfulness scores of option-boundary patching under symbol- vs. content-aligned probe-aligned patching. Numbers between parentheses indicate scores for full residual vector patching.

at option boundaries containing rich information about the content. Notably, however, in the symbol-aligned moved-content case for Llama-3.2-3B-Instruct, full-vector patching is much less faithful than probe-directed patching. This suggests that overwriting the entire residual vector at a symbol-matched site introduces substantial mismatch with the surrounding context, whereas selectively adjusting a single probe-aligned coordinate can preserve most local structure while still reinstating the particular feature that drives behavior on the clean prompt. Intuitively, these patterns point to option-boundary representations that mix high-dimensional content-specific state with a lower-dimensional, symbol-coupled component.

This could reflect a local routing feature that later attention uses to bind the selected winner to the emitted symbol. Across the two models, the same intervention reveals different indexing structures at option boundaries: in Llama-3.2-3B-Instruct, option-boundary states appear content-context-sensitive (full vector replacement into content-mismatched positions is harmful), whereas in Qwen3-8B they appear more amenable to symbol-aligned transplantation. This suggests model-dependent variation in how option summaries are stores and later routed into the answer-emission region, without requiring that the output symbol be explicitly represented at the end-of-option token.

6.2 Multinomial Probe Patching

Let $W \in \mathbb{R}^{4\times d}$ be the weight matrix of our mulitnomial probe, and $P \in \mathbb{R}^{d\times d}$ the orthogonal projector onto the row space of W, i.e., $P = UU^T$ for an orthonormal basis U spanning $\mathrm{Row}(W)$. Then patching in the residual stream (analogous to Eq (2) for the binary probe) is defined as

$$r^{(l)}_{i,X,patched} = r^{(l)}_{i,X,\text{corr}} + P(r^{(l)}_{i,X,\text{clean}} - r^{(l)}_{i,X,\text{corr}}). \quad (3)$$

Results In Figure 7, faithfulness is negligible when patching at the end-of options checkpoint or the "Answer" prefix, but increases sharply at

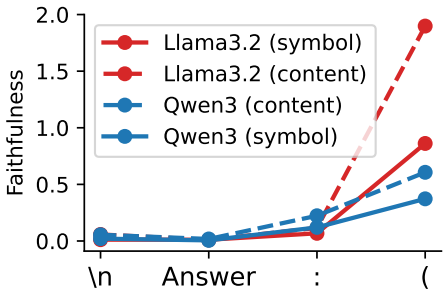


Figure 7: Faithfulness scores of multinomial winner identity probe patching under symbol and content perturbations. Dashed lines display results from using content-perturbed counterfactuals whereas solid lines are generated under symbol perturbations.

the final pre-emission token (. Thus, although correctness-related structure is detectable earlier in the residual stream, the probe-aligned representation that directly controls the answer token logits becomes causally effective only near emission. When faithfulness exceeds 1 (at the (token) reflects overshoot, patched runs assign the clean answer token a higher logit than the clean run, consistent with probe-subspace interventions isolating a strongly behaviorally-aligned component.

This pattern supports a two-stage account: First, during option processing, models compute and store local evaluation variables at boundary positions between options. Signals at the end-of-options boundary are linearly decodable, but patching evidence in Figure 7 indicates that they are not, by themselves, the control variables that determine the emitted answer token. Second, after the model encounters the answer prefix, it may convert its option-level information into a global, answer-controlling representation that is directly coupled to next-token prediction. The increase at the colon : and (is consistent with this conversion: the answer prefix cues a routing step that consolidates the decision into a representation whose geometry aligns with the multinomial probe's row space and whose value is immediately used by the unembedding to determine the next-token distribution.

7 Discussion

By combining representational measurements and visualizations with causal interventions, we find consistent evidence for a two-stage computation in language models when they perform multiple-choice question answering. First, models represent option-level evaluations and consolidate a winner in content space shortly after having processed the final option. Second, closer to answer emission, models route that winner to the appropriate output symbol.

Alignment with prior findings The two-stage mechanism is compatible with observations in the literature that MCQA accuracy is sensitive to option order and other surface-level biases, but we make a more specific claim about where those sensitivities may enter the computation. Such ordering effects are often treated as if they directly corrupt the model's underlying decision or knowledge. With our findings, option-order and surface-level biases can be reinterpreted as possibly acting through the binding stage rather than being the sole explanation of apparent knowledge or reasoning failures in MCQA.

Our results are broadly compatible with prior MI studies on MCQA, especially the finding that answer-symbol information becomes highly accessible near emission. However, our account refines the interpretation in two ways. First, by probing and intervening at earlier positions in the prompt sequence, and not just the final token position, we show that important decision-relevant computation often occurs before the model reaches the emission site. Winner identity in content space becomes decodable immediately after the final option is processed, supporting an *early commitment* regime that analyses focusing solely on the last token may miss. Second, by using symbol permutations as controlled perturbations, we disambiguate whether this winner identity is represented as a symbol-coded decision or a content-coded decision. This yields a clear dissociation: intermediate representations track the winner *content* position (in line with findings on content ordering IDs by Mueller et al., 2025), while the explicit symbol identity emerges later, near answer time. These findings tease apart processes that are conflated in previous work, namely (a) the model selecting the correct answer; and (b) the model selecting the symbol to emit. An interesting question for future work is whether these two processes can also be distinguished using mechanistic interventions on attention heads, in addition to the residual stream.

Outlook Given the high prevalence of formatted MCQA in NLP evaluations, we believe it is important to emphasize that tasks phrased in this format are not "pure reasoning" tasks, as they can conflate knowledge and binding issues. Not only could this have implications in the design of appropriate benchmarks and evaluation metrics, but it could also impact model calibration, which measures the extent to which a model's confidence matches its true probability of being correct. The two-stage computation we find suggests that we could treat MCQA confidence as a mixture of two uncertainties: selection uncertainty and binding uncertainty.

In this work, we adopt standardized prompting templates with highly structured MCQA formats and outputs. While we do not focus our attention solely on the final token of the input prompt, a necessary next step in this type of work is to relax the constraints in this setting. One way to do this is to allow the language model to generate a reasoning trace before outputting the final answer symbol. A natural extension of the current work is to track how a model's representation of the answer evolve from the final option through to the final decision over this longer span of tokens. Such an extension also provides a natural bridge to the evaluation of long-form generation.

Limitations

This work aims to characterize how instruction-tuned language models solve MCQA prompts, with an emphasis on separating option evaluation, winner selection, and symbol binding. While the results are generally consistent across the two models we study, several limitations are worth noting.

Scope of models We focus on relatively small models from two specific model families. Mechanistic findings, especially localization claims about when and where particular signals are represented, can vary with architecture, model scale and post-training procedures (e.g., supervised fine-tuning).

Prompt and task dependence Our experiments use a specific MCQA formatting template for two datasets (ARC-Easy and a synthetic color copying task) designed to control lexical confounds and cleanly separate symbol/content manipulations. Different datasets, option formatting conventions

and response constraints could shift representational geometry and causal pathways. In particular, models may adopt different strategies when prompted for explanations, chain-of-thought, or direct answer text rather than a single symbol.

Known MCQA biases We do not evaluate documented MCQA artifacts (length biases, position bias, etc.). Instead, our goal is mechanistic decomposition and consequently, additional work is required to quantify how much each artifact is explained by binding versus other factors such as lack of knowledge.

Interpretation of probes and their causal interventions Linear probes provide evidence that particular variables are linearly decodable from activations, but they do not by themselves establish that the model *uses* those variables in that form. In our work, too, the standard interpretability caution remains: high probe performance can reflect correlates or downstream consequences rather than functional intermediate representations. Additionally, probe performance depends on dataset balance, split strategy, and regularization; while we used grouped splits and cross-validation to select the inverse regularization term, subtle distributional differences in the dataset can influence absolute performance.

In the binary per-option correctness patching experiments, we intervene along a single probe direction (a 1D manipulation), which may be too restrictive if the causally relevant "option evaluation" representation is distributed across multiple dimensions. In contrast, in our winner-identity analyses we effectively consider a multi-class linear readout, which corresponds to a higher-dimensional subspace that can capture richer structure. This asymmetry means that differences in intervention success should not be over-interpreted as differences in "causal reality" of the underlying signals without accounting for the capacity of the intervention itself. More systematic comparisons (e.g., varying the dimensionality of probe-aligned interventions, or matching intervention rank across binary and multinomial settings) would better isolate whether failures arise from insufficient intervention dimensionality versus the absence of a causal signal.

Acknowledgments

We thank the three anonymous reviewers for their helpful comments. This work is funded by the Dutch Research Council (NWO) through the AiNed Fellowship Grant NGF.1607.22.002, *Dealing with Meaning Variation in NLP*.

References

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes.

Thomas Burwick. 2014. The binding problem. *WIREs Cognitive Science*, 5(3):305–315.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Qin Dai, Benjamin Heinzerling, and Kentaro Inui. 2024. Representational analysis of binding in language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17468–17493, Miami, Florida, USA. Association for Computational Linguistics.

Ricardo Dominguez-Olmedo, Moritz Hardt, and Celestine Mendler-Dünner. 2024. Questioning the survey responses of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Jiahai Feng and Jacob Steinhardt. 2024. How do language models bind entities in context? In *The Twelfth International Conference on Learning Representations*.

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

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. 2024. Finding alignments between interpretable causal variables and distributed neural representations. In *Proceedings of the Third Conference on Causal Learning and Reasoning*, volume 236 of *Proceedings of Machine Learning Research*, pages 160–187. PMLR.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Najoung Kim and Sebastian Schuster. 2023. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3835–3855, Toronto, Canada. Association for Computational Linguistics.

Vedang Lad, Wes Gurnee, and Max Tegmark. 2024. The remarkable robustness of LLMs: Stages of inference? In *ICML 2024 Workshop on Mechanistic Interpretability*.

Ruizhe Li and Yanjun Gao. 2025. Anchored answers: Unravelling positional bias in GPT-2's multiple-choice questions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2439–2465, Vienna, Austria. Association for Computational Linguistics.

Tom Lieberum, Matthew Rahtz, János Kramár, Neel Nanda, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. 2023. Does circuit analysis interpretability scale? Evidence from multiple choice capabilities in Chinchilla. *arXiv preprint arXiv:2307.09458*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Meta AI. 2024. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. Meta AI Blog.

Joseph Miller, Bilal Chughtai, and William Saunders. 2024. Transformer circuit evaluation metrics are not robust. In *First Conference on Language Modeling*.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fried Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, and 4 others. 2025. MIB: A mechanistic interpretability benchmark. In *Forty-second International Conference on Machine Learning*.

Tobias Norlund, Lovisa Hagström, and Richard Johansson. 2021. Transferring knowledge from vision to language: How to achieve it and how to measure it? In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–162, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pouya Pezeshkpour and Estevam Hruschka. 2024. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017, Mexico City, Mexico. Association for Computational Linguistics.

Curt Tigges, Oskar J. Hollinsworth, Atticus Geiger, and Neel Nanda. 2024. Language models linearly represent sentiment. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 58–87, Miami, Florida, US. Association for Computational Linguistics.

Polina Tsvilodub, Hening Wang, Sharon Grosch, and Michael Franke. 2024. Predictions from language models for multiple-choice tasks are not robust under variation of scoring methods. *arXiv preprint arXiv:2403.00998*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.

Christoph Von Der Malsburg. 1999. The What and Why of Binding. *Neuron*, 24(1):95–104.

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

Xinpeng Wang, Bolei Ma, Chengzhi Hu, Leon Weber-Genzel, Paul Röttger, Frauke Kreuter, Dirk Hovy, and Barbara Plank. 2024. "my answer is C": First-token probabilities do not match text answers in instruction-tuned language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7407–7416, Bangkok, Thailand. Association for Computational Linguistics.

Sarah Wiegreffe, Oyvind Tafjord, Yonatan Belinkov, Hannaneh Hajishirzi, and Ashish Sabharwal. 2025. Answer, assemble, ace: Understanding how LMs answer multiple choice questions. In *The Thirteenth International Conference on Learning Representations*.

Mengge Xue, Zhenyu Hu, Liqun Liu, Kuo Liao, Shuang Li, Honglin Han, Meng Zhao, and Chengguo Yin. 2024. Strengthened symbol binding makes large language models reliable multiple-choice selectors. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4331–4344, Bangkok, Thailand. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao

Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhen Yang, Ping Jian, and Chengzhi Li. 2025b. Option symbol matters: Investigating and mitigating multiple-choice option symbol bias of large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1902–1917, Albuquerque, New Mexico. Association for Computational Linguistics.

Fred Zhang and Neel Nanda. 2024. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations*.

Guangxiang Zhao, Saier Hu, Xiaoqi Jian, Wu Jinzhu, Yuhan Wu, Lin Sun, and Xiangzheng Zhang. 2025. Large language models badly generalize across option length, problem types, and irrelevant noun replacements. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26825–26834, Suzhou, China. Association for Computational Linguistics.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.

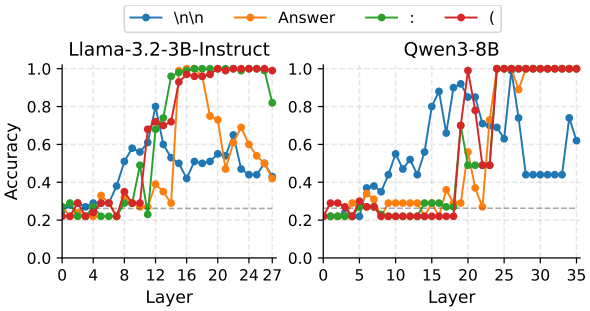


Figure 8: Performance on the color copying task of the multinomial probes trained on ARC-Easy (without additional re-training on the color copying dataset).

A Probing

Hyperparameter search for the inverse regularization strength was done over a grid $[0.0001, 0.001, 0.01, 0.1, 0.3, 1, 3]$, with a maximum of 5000 iterations. We used the `liblinear` solver for the binary probes and `lbfgs` for the multinomial probes. All probes were trained using `scikit-learn`.

Figure 8 shows performance of the probe on the color copying task after training on ARC-Easy.

B PCA Results

In this section, we present additional results from our PCA experiments.

B.1 PCA Qwen3-8B: Symbol Positions

Figure 9 shows PCA for activations at symbol positions for Qwen3-8B.

B.2 PCA with Permuted Symbols

Figures 10 and 11 show PCA for activations at symbol positions after prompt symbols are permuted.

B.3 PCA: Content and Newline Positions

The following plots are the full versions of the layer-specific plots displayed in Figure 3. Figures 12 and 13 show the PCA plots for **Llama-3.2-3B-Instruct**, with hidden activations extracted at the **content tokens** and **newline** positions, respectively. Figures 14 (**content**) and 15 (**newline**) do the same for **Qwen3-8b**.

B.4 PCA with ARC-Easy

Figures 16 and 17 show results for PCA for the two models on ARC-Easy data. Given that ARC-Easy is less structured than Colors, it is to be expected that patterns are not as easily visible in PCA space.

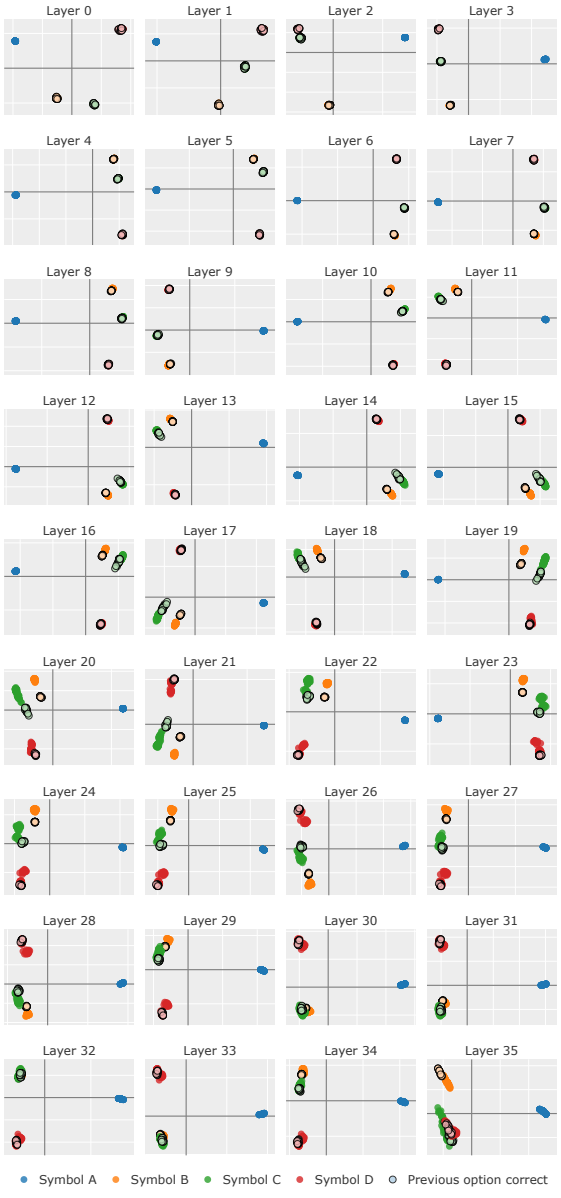


Figure 9: PCA for activations extracted at the **symbol positions** for **Qwen3-8B**, analogous to Figure 2 shown for Llama-3.2-3B in Section 4.

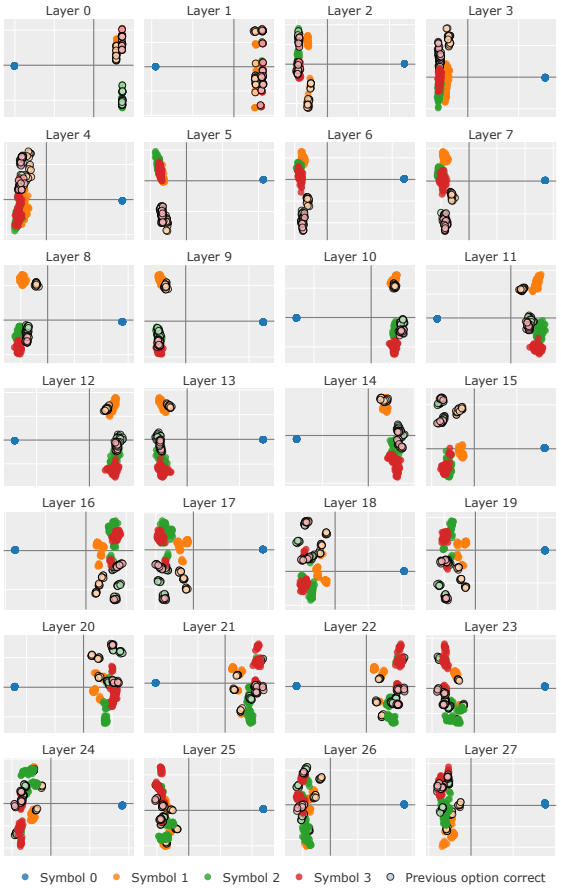


Figure 10: PCA for activations of **Llama-3.2-3B-Instruct** extracted at the **symbol positions** when the symbols in the prompt are **permuted**.

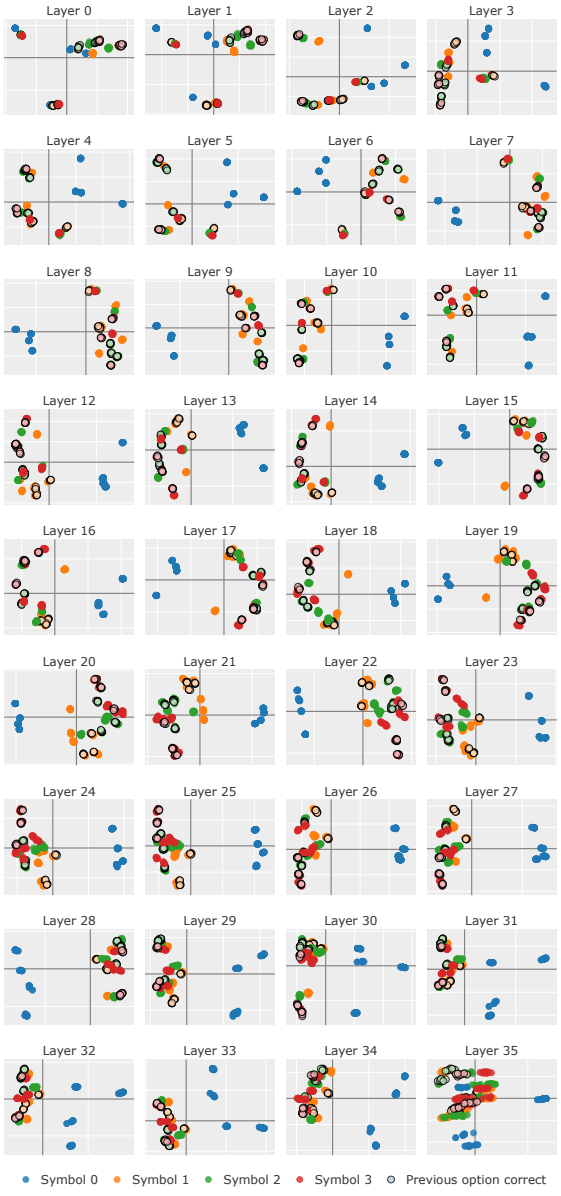


Figure 11: PCA for activations of **Qwen3-8b** extracted at the **symbol positions** when the symbols in the prompt are **permuted**

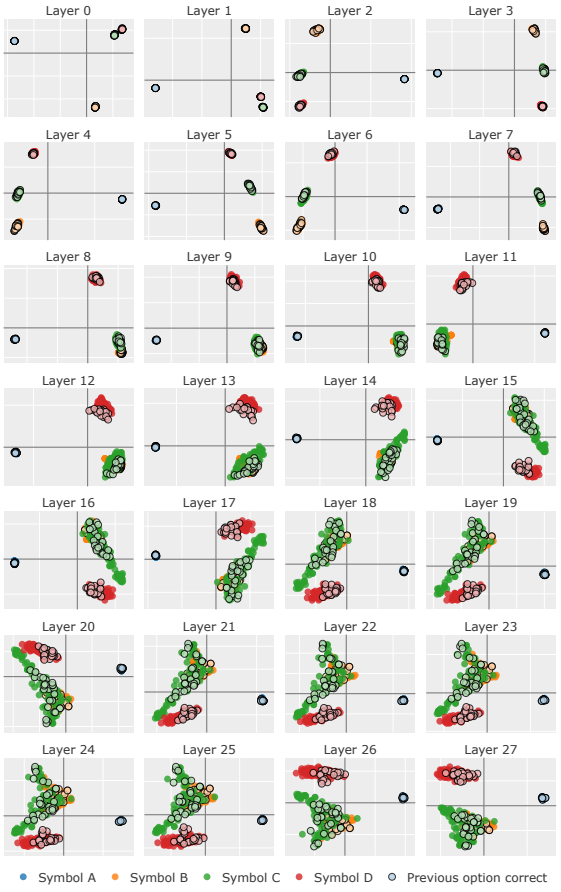


Figure 12: PCA for hidden activations extracted from **Llama-3.2-3B-Instruct** at the **content token** positions.

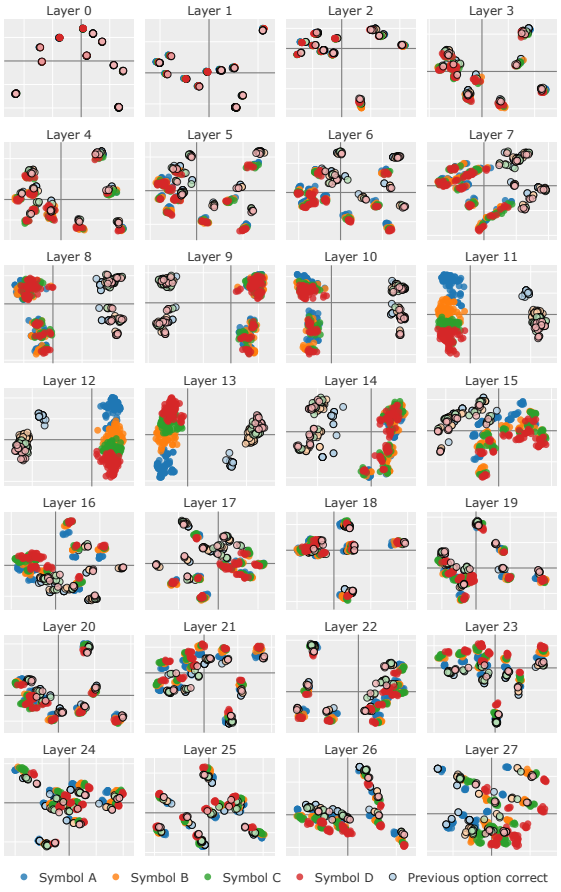


Figure 13: PCA for hidden activations extracted from **Llama-3.2-3B-Instruct** at the **newline tokens** directly after the content tokens.

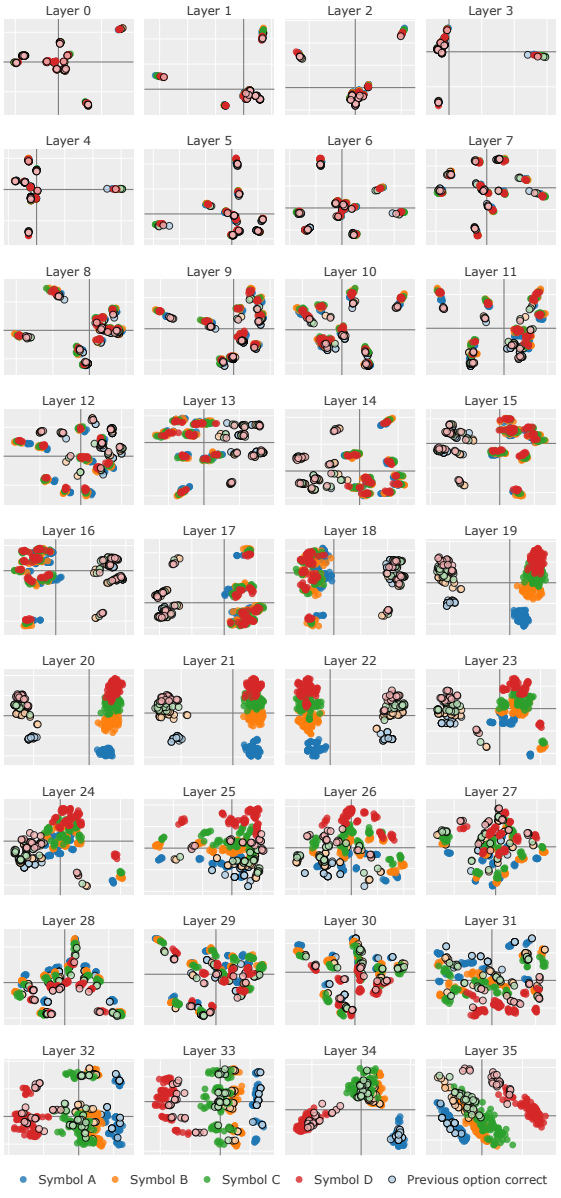


Figure 14: PCA for hidden activations extracted from **Qwen3-8b** at the **content token** positions.

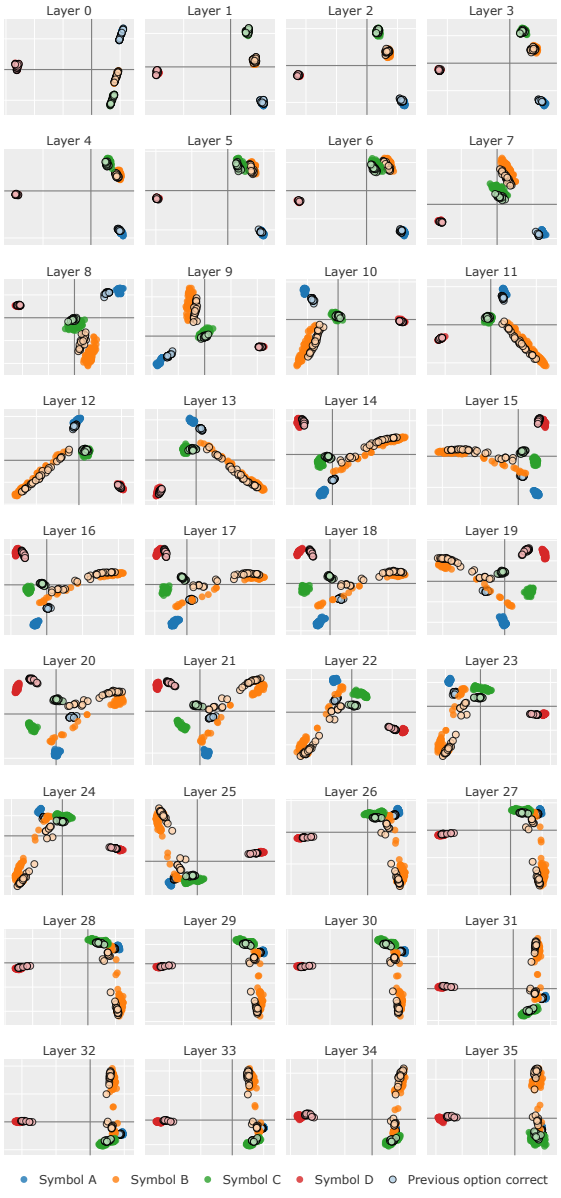


Figure 15: **Qwen3-8b** at the **newline tokens** directly after the content tokens.

However, certain layers show separability between correct and incorrect options.

Figure 16: **Qwen3-8b** at the **symbol tokens** of 200 randomly selected ARC-Easy questions. Given that ARC-Easy is less structured than Colors, it is to be expected that patterns are not as easily visible in PCA space. However, do note that layers such as Layer 20 do hint at separability between correct and incorrect options.

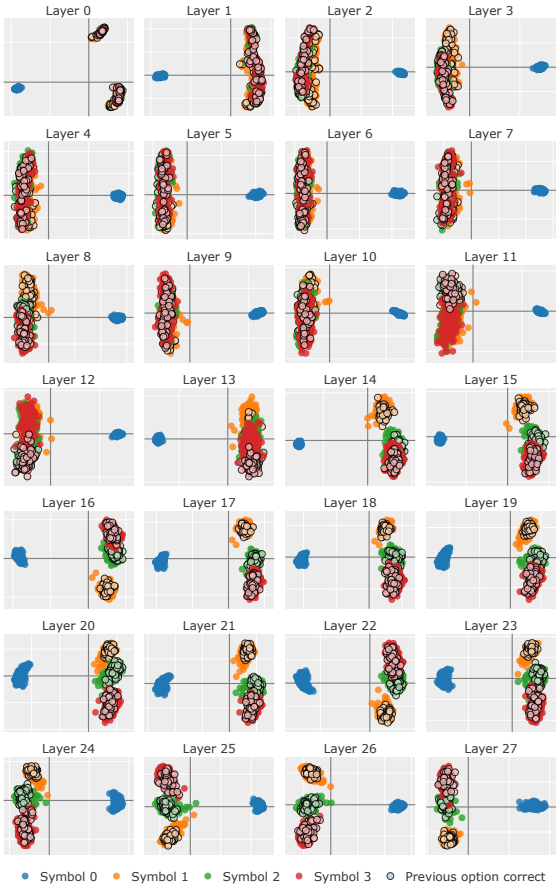


Figure 17: **Llama-3.2-3B-Instruct** at the **symbol tokens** of 200 randomly selected ARC-Easy questions. Given that ARC-Easy is less structured than Colors, it is to be expected that patterns are not as easily visible in PCA space. However, do note that layers such as Layer 12 do hint at separability between correct and incorrect options.