Internal Chain-of-Thought: Empirical Evidence for Layer-wise Subtask Scheduling in LLMs

Anonymous ACL submission

Abstract

We show that large language models (LLMs) exhibit an *internal chain-of-thought*: they sequentially decompose and execute composite tasks layer-by-layer. Two claims ground our study: (i) distinct subtasks are learned at different network depths, and (ii) these subtasks are executed sequentially across layers. On a benchmark of 15 two-step composite tasks, we employ layer-from context-masking and propose a novel cross-task patching method, confirming (i). To examine claim (ii), we apply LogitLens to decode hidden states, revealing a consistent layerwise execution pattern. We further replicate our analysis on the real-world TRACE benchmark, observing the same step-wise dynamics. Together, our results enhance LLMs transparency by showing their capacity to internally plan and execute subtasks (or instructions), opening avenues for fine-grained, instruction-level activation steering.

1 Introduction

Large Language Models (LLMs) excel at solving complex tasks such as instruction following, and multi-step problem solving (Zhang et al., 2024; Zeng et al.; Wang et al., 2024). Much recent progress relies on explicit "chain of thought" (Wei et al., 2022; Zhang et al.), which guides models to decompose multi-step problems into intermediate reasoning stages. This raises a foundational question: **Do LLMs also perform such multi-step reasoning internally, without revealing steps in their output?** In this work, we answer **yes**: LLMs exhibit an internal chain-of-thought (ICoT), meaning they internally break down composite tasks and process their components sequentially across network layers. Going beyond interpretability studies on latent factual multi-hop reasoning (Yang et al., 2024b; Biran et al., 2024; Yu et al., 2025; Anthropic., 2025), we investigate task-level reasoning rather than just chains of facts.

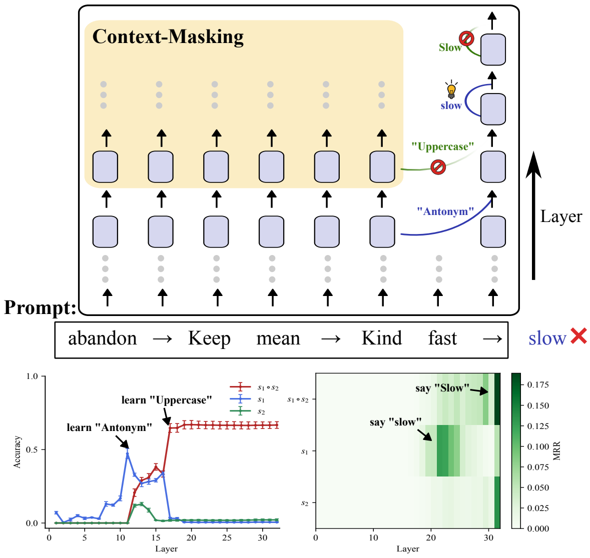

Figure 1: Evidence of an internal chain-of-thought. Selectively masking (bottom left) from specific layers can preserve the first subtask (antonym) while ablating the second (uppercase). Decoding hidden states (bottom right) on a clean run shows the intermediate answer ("`slow`") peaks at middle layers.

To illustrate this concept, consider a composite task that requires two steps: antonym then uppercase. For example, given the input "`fast`", solving this involves an intermediate step—finding the antonym "`slow`"—followed by capitalizing it to "`Slow`" (with S capitalized). If an internal chain-of-thought exists, we would expect the hidden state at some intermediate layer to represent "`slow`", and later layers to transform it into "`Slow`". In general, the presence of an ICoT implies that distinct phases of computation occur inside the model, each corresponding to a subtask in the overall problem. Figure 1 illustrates our core findings: selectively masking context from specific layers can preserve the first subtask (antonym) while ablating the second (uppercase). Meanwhile, decoding results of hidden state on clean run show that intermediate answer ("`slow`") often peaks at middle layers.

For tractability, we study tasks that decompose into two sequential subtasks (denoted $t := s_1 \circ s_2$). The task t maps an input x to an output $y^t = s_2(s_1(x))$. We frame our analysis through Task Vector framework (Hendel et al., 2023; Todd et al.; Li et al.; Saglam et al., 2025), which divides in-context learning (ICL) into two phases: (1) a *learning phase*, where the model abstracts task rules into a hidden representation, task vector, and (2) a *rule application phase*, where the query is processed using this vector. For a model T, given demonstrations S and a query x, the process is modeled as:

$$T([S, x]) \Rightarrow \underbrace{\theta = \mathcal{A}(S)}_{\text{learning phase}}, \underbrace{y = f(x; \theta)}_{\text{application phase}}, \quad (1)$$

where $\mathcal{A}$ abstracts the task vector $\theta \in \mathbb{R}^d$, and f denotes the application of task-specific rules. Extending to composite tasks yields two claims:

- **Claim 1.** Subtasks are learned at different network depths, inducing distinct subtask vectors θ^{s_1} and θ^{s_2} which generalize to their respective subtasks.
- **Claim 2.** Subtasks are executed sequentially across layers. At depth l_1, the model applies $f^{l_1}(x; \theta^{s_1})$ to compute the first subtask. At a later depth l_2, it applies $f^{l_2}(x; \theta^{s_2})$, yielding the final result.

Crucially, we distinguish between two processes: "learning" (Claim 1) and "execution" (Claim 2), corresponding to the learning phase and application phase in Task Vector framework, respectively.

We first introduce a benchmark of 15 two-step composite tasks spanning four categories (Section 2.2). We present two lines of evidence for Claim 1: (1) layer-from-context masking (Sia et al., 2024), which blocks attention to demonstrations after layer l to reveal where each subtask is learned, and (2) cross-task patching, a novel method which inserts residual activations from a composite prompt into zero-shot sub-task queries to detect reusable "subtask vectors". Across four models and 15 two-step tasks, masking reveals a sharp "X-shape" (Figure 2), indicating a sequential learning dynamics: the model first abstracts the rule for s_1 at an earlier layer, and later learns s_2 at a deeper layer. Meanwhile, patching activations in Llama-3.1-8B (see Table 2) yield transferable subtask vectors to a significant degree (66% on average).

Next, to verify Claim 2, We decode every layer with LogitLens (Nostalgebraist., 2020), projecting hidden states into token space and tracking the mean reciprocal rank of the first-step target (y^{s_1} or y^{s_2}) versus the final answer ($y^{s_1 \circ s_2}$). Decoding results show the same "handoff" (see Figure 3 and 4): intermediate answer peaks in mid-layers, then is overtaken a few layers later by the final answer. Finally, we replicate layer-from-context masking on TRACE (Zhang et al., 2024), a complex instruction-following benchmark, demonstrating that the same sequential learning dynamics emerge in real-world settings (see Figure 5). The primary contributions of this study are as follows:

- We construct a curated benchmark of 15 composite tasks spanning four categories.
- We employ context-masking and propose **cross-task patching**, demonstrating that subtasks are learned at different depths, inducing distinct subtask vectors.
- We use LogitLens to decode hidden states, revealing a consistent layerwise execution pattern.
- We replicate our method on the TRACE benchmark, confirming the same finding also emerge in practical settings.

Our findings enhance LLM transparency by revealing their capacity to internally plan and execute subtasks (or instructions). This aligns with, and extends, prior interpretability studies on multi-hop reasoning (Yang et al., 2024b; Biran et al., 2024; Yu et al., 2025; Anthropic., 2025) and look-ahead planning (Men et al., 2024). While those often focus on factual recall or predictive steps, our work investigates task-level reasoning rather than just chains of facts. Furthermore, the discovery of ICoT opens exciting avenues for fine-grained, instruction-level behavior control. For instance, by identifying the layers responsible for processing specific (potentially harmful) instructions within a user's prompt, we could directionally intervene to steer their execution for safer LLM behavior.

2 Experimental Setup

2.1 Prompt Design

We focus on composite tasks that naturally decompose into two sequential subtasks—for example, retrieving domain knowledge and then translating it, or extracting information followed by format transformation. Formally, we represent a composite task as $t := s_1 \circ s_2$, where s_1 and s_2 are sequentially applied subtasks. Given a query x,

Category	Task Description	Example (Input → Output)
Knowledge–Algorithmic	s_1: antonym s_2: uppercase	`fast` → `Slow`
Extractive–Knowledge	s_1: select adjective s_2: synonym	`artistic, captain, bring` → `creative`
Extractive–Algorithmic	s_1: select last item s_2: first letter	`spicy, cowardly, hoop` → `h`
Knowledge–Translation	s_1: retrieve country s_2: translate to French	`Cenepa River` → `Pérou`

Table 1: Representative examples from the composite task benchmark across four categories. Each task involves a sequential application of two subtasks, though no intermediate outputs are shown in-context. See Appendix A for a complete task list and definitions.

the final output is $y^t = s_2(s_1(x))$. For analysis purposes, we also compute intermediate outputs corresponding to the isolated application of each subtask: $y^{s_1} = s_1(x)$ and $y^{s_2} = s_2(x)$. As an illustrative example, consider s_1 = "antonym" and s_2 = "uppercase". For input x = "`fast`", the correct intermediate and final outputs would be y^{s_1} = "`slow`", y^{s_2} = "`Fast`", and y^t = "`Slow`".

For each composite task $t \in \mathcal{T}$ in our task suite $\mathcal{T}$, we construct a dataset $\mathcal{P}_t$ consisting of in-context prompts $p_i^t \in \mathcal{P}_t$. Each prompt includes N input-output demonstration pairs of the form (x, y^t), showing the full composite transformation, followed by a query input x_{iq} for which the model is expected to predict the corresponding target y_{iq}^t. Notably, no intermediate outputs or reasoning steps are included in the prompt. The in-context learning (ICL) prompt format is:

$$p_i^t = \left[(x_{i1}, y_{i1}^{s_1 \circ s_2}), \ldots, (x_{iN}, y_{iN}^{s_1 \circ s_2}), x_{iq}\right]. \quad (2)$$

2.2 Dataset

We construct a benchmark of 15 composite tasks spanning four categories:

- **Knowledge–Algorithmic:** Tasks that combine factual knowledge retrieval (e.g., country capitals, antonyms) with deterministic transformations (e.g., uppercase conversion).
- **Extractive–Knowledge:** Tasks that require identifying items from a list (e.g., selecting the last item) followed by a knowledge-based operation (e.g., finding a related concept).
- **Extractive–Algorithmic:** Tasks that involve list-based selection followed by symbolic transformations (e.g., case conversion, character extraction).
- **Knowledge–Translation:** Tasks that combine knowledge retrieval with language translation (e.g., translating the capital city of a given country into French or Spanish).

Each query in the dataset requires the sequential execution of two subtasks in a fixed order as defined by the task specification. However, we do not assume that LLMs necessarily follow this order during internal processing. To probe the latent execution path, we measure intermediate outputs corresponding to the isolated application of each subtask: $y^{s_1} = s_1(x)$ and $y^{s_2} = s_2(x)$. Table 1 presents illustrative examples for each category. Full details and descriptions for all 15 composite tasks can be found in Appendix A.

3 Background

We consider an autoregressive transformer language model T that takes an input prompt p and outputs a next-token distribution $T(p)$ over a vocabulary $\mathcal{V}$. Internally, T consists of L transformer layers connected via a residual stream (Elhage et al., 2021). We focus our analysis on the residual stream at the final token position. Embedding matrix $\mathbf{W}_E \in \mathbb{R}^{|\mathcal{V}| \times d}$ first maps the last token to a hidden representation as initial residual stream $\mathbf{h}^0 \in \mathbb{R}^d$. At each layer l, the model adds the outputs of the self-attention and feedforward network (FFN) modules to the residual stream from the previous layer. Formally, the residual stream at layer l is given by:

$$\mathbf{h}^l = \mathbf{h}^{l-1} + \mathbf{A}^l + \mathbf{F}^l, \quad (3)$$

where $\mathbf{A}^l \in \mathbb{R}^d$ and $\mathbf{F}^l \in \mathbb{R}^d$ denote the attention and FFN outputs at the final token position, respectively, at layer l.

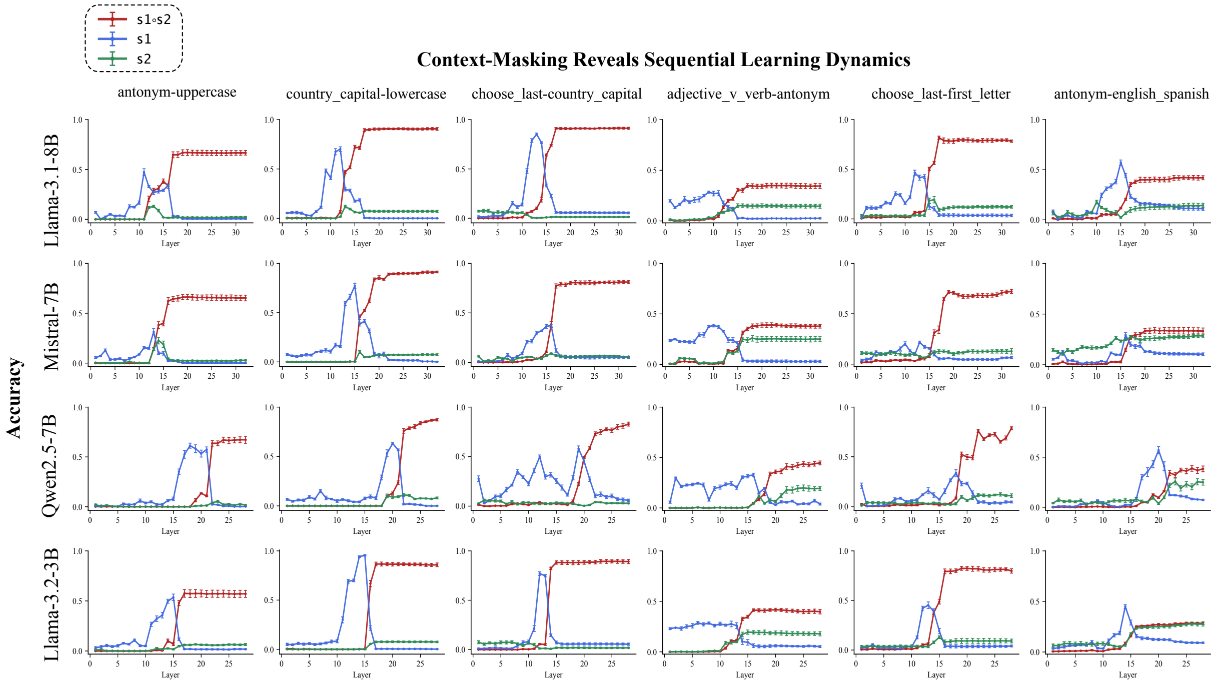

Figure 2: Layer-from context-masking results for six composite tasks across four models. The "X-shape" pattern reveals sequential learning dynamics.

4 Claim 1: Distinct Subtask Representations

We present two lines of evidence for **Claim 1**: (1) using layer-from context-masking, we show that different layers are responsible for learning each subtask; (2) with a novel cross-task patching method, we demonstrate that subtask-specific vectors emerge at the final token position, serving as abstract representations that generalize across tasks.

4.1 Layer-from Context-Masking

In-Context Learning requires model to infer a task from examples and apply it to a new input, as formalized in Equation 1. If LLMs follow a internal chain-of-thought, then each subtask in a composite task should be learned at a distinct point in the network. That is, subtask learning should unfold sequentially across layers—rather than all at once—making intermediate learning states observable. To investigate this, we employ layer-from context-masking (Sia et al., 2024). This technique disables access to the in-context examples (task demonstrations) from a specific layer onward by masking all attention to context tokens. If masking is applied from the input layer ($l = 0$), the model cannot attend to any demonstrations, and ICL should fail. However, if masking begins only after the model has learned the task, then its performance should remain intact. Crucially, by gradually shifting the start-masking layer from early to late, we can infer the sequential dynamics of the model's learning process.

Let $\mathbf{A} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D}}$ denote the raw attention scores in a decoder-only transformer, where $\mathbf{Q}$ and $\mathbf{K}$ are the query and key matrices, respectively, and D is the dimensionality of the hidden states. For token positions i and j, the element A_{ij} represents how much token i attends to token j. We apply a context masking to disable attention to in-context examples, as $A_{ij} + m(j, \mathbf{u})$. The mask $m(j, \mathbf{u})$ is defined as:

$$m(j, \mathbf{u}) = \begin{cases} 0 & \text{if } x_j \notin \mathbf{u}, \\ -\infty & \text{if } x_j \in \mathbf{u}, \end{cases} \quad (4)$$

where $\mathbf{u}$ is the set of all in-context example tokens. The mask is applied from layer l onward, such that for all $l' \geq l$, attention to context is zeroed out after Softmax. For each test prompt, we progressively increase the masking layer l from 1 to L and record the model's prediction accuracy on both intermediate and final outputs. For composite tasks, we aim to identify a two-phase masking pattern. At early layers, masking may lead the model to predict intermediate outputs—e.g., y^{s_1} or y^{s_2}—indicating that the model has learned the first subtask but not

Composite Task	Llama-3.1-8B
`antonym-uppercase`	
s_1 (antonym)	0.92 ± 0.02
s_2 (uppercase)	0.24 ± 0.03
`country_capital-lowercase`	
s_1 (country_capital)	0.88 ± 0.03
s_2 (lowercase)	0.49 ± 0.05
`choose_last-country_capital`	
s_1 (choose_last)	0.44 ± 0.03
s_2 (country_capital)	0.98 ± 0.02
`adjective_v_verb-antonym`	
s_1 (adj_v_verb)	0.38 ± 0.11
s_2 (antonym)	0.94 ± 0.03
`choose_last-first_letter`	
s_1 (choose_last)	0.29 ± 0.03
s_2 (first_letter)	1.01 ± 0.01
`antonym-english_spanish`	
s_1 (antonym)	0.91 ± 0.02
s_2 (english_spanish)	0.45 ± 0.03
Average	0.66

Table 2: Subtask vector strength for six representative composite tasks in Llama-3.1-8B. Each composite task's average residual activation is patched into each subtask, with results shown for s_1 and s_2 respectively.

yet the second. As masking is delayed to deeper layers, the model's predictions should transition sharply from intermediate answers to the final answer y^t, revealing a layered acquisition of subtasks. By contrast, if the model transitions directly from generating no meaningful output to the correct final answer as masking depth increases, without producing intermediate completions, this would suggest a monolithic in-context learning process. This distinction is central to testing whether subtask learning unfolds sequentially across layers.

Experiment. We conduct our layer-from context-masking analysis on four LLMs: Llama-3.1-8B (Grattafiori et al., 2024), Mistral-7B (Jiang et al., 2024), Qwen2.5-7B (Yang et al., 2024a), and Llama-3.2-3B (Meta., 2024), evaluating their behavior across all 15 composite tasks. For each task, we generate 500 test prompts, sampled uniformly at random from the corresponding dataset. Each prompt includes N in-context examples (following prior work (Hendel et al., 2023), we set $N = 5$). To ensure robustness, all experiments are repeated across five random seeds, and we report averaged results. In the resulting sequential learning dynamics plots (see Figure 2 and Appendix B), we observe a striking "X-shape" pattern across composite tasks. Specifically, as context masking is delayed to deeper layers, the model's output transitions from generating one of the intermediate answers (e.g., the result of s_1) to producing the correct final answer. The intersection point—where performance on the intermediate answer begins to drop while performance on the final answer rises—suggests a boundary between subtask learning phases. This structure provides compelling evidence for sequential learning dynamics: the model first abstracts the rule for s_1 at an earlier layer, and later learns s_2 at a deeper layer.

4.2 Cross-Task Patching

While context-masking reveals when subtask information is acquired, it does not directly test whether LLMs represent individual subtasks as reusable, abstract vectors. To address this, we introduce **cross-task patching**, a novel method that investigates whether sequential learning dynamics produce distinct *subtask vectors*. Prior work suggests that the residual stream at the final token position encodes a latent task representation derived from in-context examples (Hendel et al., 2023; Li et al.; Todd et al.). These representations can be replaced into the hidden states while running model on other prompts to influence model behavior. Here, we extend this idea to composite tasks. Specifically, we examine whether the activations obtained from a composite prompt can be used to improve performance on each subtask individually. We compute the average residual stream activation across composite task prompts, then patch it into zero-shot prompts from the subtask datasets. If performance on the subtask improves, we infer that the composite prompt's activation encodes the corresponding subtask vector.

Formally, we begin by running the model on a set of composite prompts $p_i^t \in \mathcal{P}_t$, each containing N examples of task t, and extract activation vector at the final token position from each layer l. Averaging over all prompts yields a layerwise task representation:

$$\bar{\mathbf{h}}_l^t = \frac{1}{|\mathcal{P}_t|} \sum_{p_i^t \in \mathcal{P}_t} \mathbf{h}^l(p_i^t). \quad (5)$$

We then patch this vector into a set of zero-shot subtask prompts $\tilde{p}_i \in \tilde{\mathcal{P}}_{s_j}$ (i.e., prompts with no in-context examples), replacing the residual stream at layer l with $\bar{\mathbf{h}}_l^t$, and evaluate the model's performance:

$$\mathrm{Acc}(\tilde{\mathcal{P}}_{s_j}, l) = \frac{1}{|\tilde{\mathcal{P}}_{s_j}|} \sum_{\tilde{p}_i} \mathbb{I}\left[T(\tilde{p}_i \mid \mathbf{h}^l := \bar{\mathbf{h}}_l^t) = y_i \right]. \quad (6)$$

To quantify how well this patched vector recovers the subtask behavior, we define a normalized *subtask vector strength*. As the patching is used on each layer, we choose the best result to calculate the subtask vector strength:

$$\mathrm{Strength}^{s_j} = \frac{\max_l \mathrm{Acc}(\tilde{\mathcal{P}}_{s_j}, l) - \mathrm{Acc}(\tilde{\mathcal{P}}_{s_j})}{\mathrm{Acc}(\mathcal{P}_{s_j}) - \mathrm{Acc}(\tilde{\mathcal{P}}_{s_j})}, \quad (7)$$

where $\mathrm{Acc}(\mathcal{P}_{s_j})$ is the subtask's performance under standard ICL (with N examples), and $\mathrm{Acc}(\tilde{\mathcal{P}}_{s_j})$ is the zero-shot baseline. A strength of 1 implies full recovery of subtask performance, indicating a fully formed subtask vector; a strength of 0 implies no transfer. We test the subtask vector strength on both two subtasks s_1 and s_2.

Experiment. To ensure independence between datasets, we first split each subtask dataset into disjoint train and test subsets (see Appendix A for details about subtask dataset). Composite datasets are constructed using only the train set, while zero-shot patching is evaluated on held-out subtask examples. We compute $\bar{h}_l^t$ using 100 composite prompts and test patching strength on 500 zero-shot subtask prompts. We repeat this process across 15 composite tasks, 4 models, and 5 random seeds.

Table 2 and Appendix C report the patching strength across tasks. We find that most composite tasks yield transferable subtask vectors to a significant degree (0.66 on average). Interestingly, some composite tasks exhibit asymmetric transfer—for instance, the composite vector may strongly support s_1 but only weakly support s_2. This asymmetry may reflect either the task type of s_2 (e.g., extractive tasks), or that s_2 is applied in a more entangled fashion atop the result of s_1, making its representation more context-dependent.

5 Claim 2: Layer-wise Rule Application

Claim 2 hypothesizes that LLMs apply rules for composite tasks in a staged process: at an earlier layer l_1, the model applies a function $f^{l_1}(x; \theta^{s_1})$ to perform the first subtask; later, at layer $l_2 > l_1$, it applies a second function $f^{l_2}(x; \theta^{s_2})$, integrating

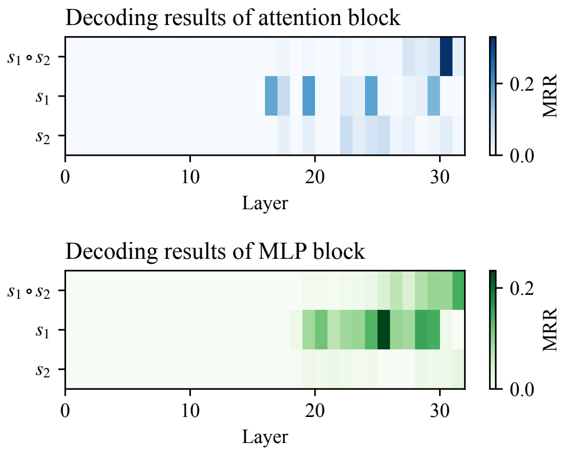

Figure 3: Heatmaps of attention and MLP block decoding results for the `country_capital-lowercase` task in Llama-3.1-8B.

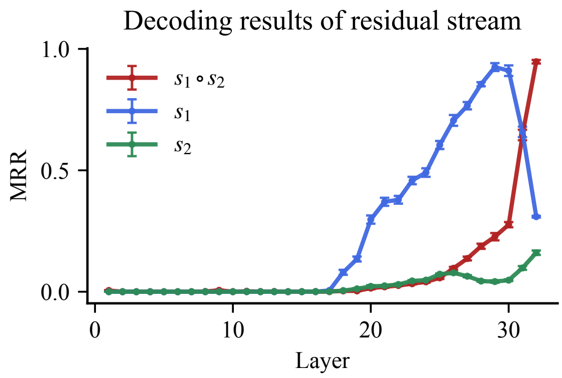

Figure 4: Decoding results of the residual stream for the `country_capital-lowercase` task in Llama-3.1-8B.

this intermediate representation with the second subtask's logic to produce the final answer. Crucially, we should be able to trace this transformation through the model's residual stream, which accumulates the outputs of each attention and MLP block.

We decode the next-token probabilities for each intermediate layer using LogitLens (Nostalgebraist., 2020). This method aims to project hidden states into the vocabulary space. Formally, let $\mathbf{h}^l$ denote the residual stream at the final token position, at layer l. To decode their outputs into probability distributions $\mathbf{p}$ over vocabulary tokens, we use the unembedding matrix $\mathbf{W_U} \in \mathbb{R}^{d \times |\mathcal{V}|}$, along with a normalization that rescales component activations relative to the final-layer logits:

$$\mathbf{p} = \mathrm{Softmax}\left(\mathbf{W_U} \cdot \frac{\mathbf{h}^l - \bar{\mathbf{h}}^l}{\alpha^*} \right), \quad (8)$$

where $\bar{\mathbf{h}}^l$ are the mean component outputs for nor-

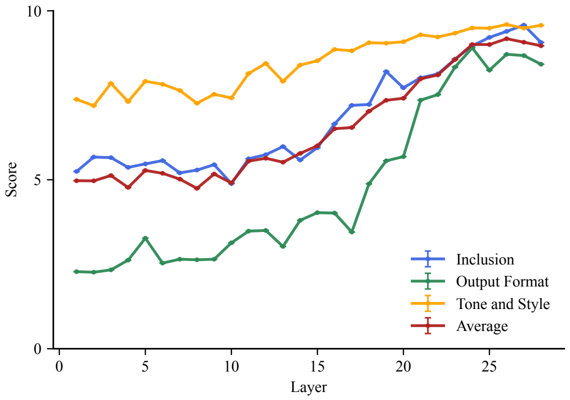

Figure 5: Layer-wise context-masking analysis on TRACE benchmark. Each curve shows model performance (scored 0–10) on a distinct constraint type, evaluated by DeepSeek-V3.

malization, and α^* is a scaling factor derived from the final layer's residual norm. Besides, we also decode each attention and MLP block's output $\mathbf{A}^l$ and $\mathbf{F}^l$. We then measure the **Mean Reciprocal Rank (MRR)** for three specific targets: y^t, y^{s_1} and y^{s_2}.

Experiment. We conduct this analysis on 500 prompts for each of the 15 composite tasks described earlier. For each prompt, we extract and decode the attention outputs, MLP outputs, and residual stream at each layer, compute mean reciprocal ranks (MRRs) for the three target outputs described above, and plot the resulting trajectories.

Figure 3 shows the heatmaps of attention and MLP block decoding results for the `country_capital-lowercase` task in Llama-3.1-8B. Figure 4 displays the decoding results of the residual stream (see Appendix D for full results). We observe a clear layerwise task execution pattern: the model produces intermediate answers in middle layers, which are progressively surpassed by the final answer in later layers. The crossover point—where MRR for y^{s_1} declines while MRR for $y^{s_1 \circ s_2}$ increases—mirrors the two-stage task execution hypothesized in Claim 2.

6 A Practical Case: TRACE Dataset

To evaluate the applicability of our analysis in real-world scenarios, we extend our experiments to TRACE (Zhang et al., 2024), a complex instruction following benchmark. TRACE is built on a manually curated taxonomy of complex instructions, incorporating 26 constraint dimensions grouped into five high-level categories. Each prompt in TRACE consists of two components: a *Task Description*, which defines the core objective (e.g., "Introduce Beijing"), and a set of *Constraints*, which specify additional requirements that the model must satisfy. For example, a representative prompt might be:

> **Task Description:** Can you introduce Beijing to me?
> **Constraints:** 1. The generated answer format needs to be JSON;
> 2. The generated answer must be objective facts and cannot contain any subjective opinions or guesses;
> 3. All characters need to be encoded in UTF-8 to ensure support for multilingual responses.

We provide the entire prompt to the model and use a LLM evaluator to assign a score (from 0 to 10) for each constraint based on how well it is satisfied. In this section, we apply layer-from context-masking, but with a twist: we selectively mask only the *Constraints* portion of the prompt from each layer onward, while retaining access to the *Task Description* throughout. Our goal is to determine whether different types of constraints are learned at different depths, thereby exhibiting a multi-step learning trajectory.

Experiment. We select a subset of TRACE (69 prompts) that include all the following constraint types: **Inclusion**, **Output Format**, and **Tone and Style**. We use Qwen2.5-7B-Instruct (Yang et al., 2024a) as the test model to complete the instructions, applying context-masking to the constraint tokens from each layer onward. To evaluate the quality of constraint satisfaction at each masking depth, we use DeepSeek-V3 (Liu et al., 2024a) as an evaluator model, producing per-constraint scores from 0 to 10.

Figure 5 shows the resulting line chart. We observe notable differences in the learning dynamics across constraint types:

- **Output Format:** The learning curve is characterized by a sharp increase in score between layers 17–20, suggesting that formatting constraints (e.g., JSON structure, character encoding) are learned relatively late in the network.
- **Inclusion** and **Tone and Style:** These constraints show more gradual and smooth improvements across layers, indicating a slower

or more distributed learning process.

These results demonstrate that different constraint types are learned at different depths in the model, further supporting our hypothesis of sequentially learning dynamics in composite instruction-following tasks.

7 Related Work

Multi-Hop Reasoning in LLMs. Recent studies have examined how large language models (LLMs) perform *latent factual multi-hop reasoning* (Press et al., 2023; Yang et al., 2024c; Li et al., 2024; Ju et al., 2024). (Yang et al., 2024b) finds that LLMs often reliably recall intermediate entities but inconsistently use them to complete complex prompts. (Biran et al., 2024) shows that LLMs resolve intermediate entities early when answering multi-hop queries, and proposes a back-patching method to improve the performance. (Yu et al., 2025) introduces logit flow to analyze latent multi-hop reasoning in LLMs and proposes back attention to improve accuracy. (Anthropic., 2025) identifies intermediate entities and reasoning path by Cross-Layer Transcoder. While those often focus on factual recall, our work investigates task-level reasoning rather than just chains of facts.

Task Representations in ICL. The ability of LLMs to perform In-Context Learning (ICL) (Brown et al., 2020) has spurred rich research into its internal mechanism. A prominent line of inquiry focuses on explicit task representations (Hendel et al., 2023; Liu et al., 2024b; Todd et al.; Li et al.; Saglam et al., 2025; Yang et al., 2025). Initial work by (Hendel et al., 2023) derived task vectors from layer activations. Other approaches include In-Context Vectors (ICVs) (Liu et al., 2024b) derived from principal components of activation differences, and Function Vectors (FVs) (Todd et al.) which emphasize the role of specific attention heads. While these foundational studies demonstrate how a singular task can be abstracted into a vector, our work extends this by investigating how composite tasks are handled.

Mechanistic Interpretability. Mechanistic interpretability (Elhage et al., 2021) aims to reverse engineer the internal mechanisms of LLMs. One type of studies focus on constructing the circuit in the model (Olsson et al., 2022; Wang et al.; Gould et al.; Marks et al., 2024). Another line of work focuses on understanding intermediate representations through tools such as the LogitLens (Nostalgebraist., 2020). This technique has been extended to trace hidden states in LLMs (Dar et al., 2023; Halawi et al.; Merullo et al., 2024; Wiegreffe et al., 2024). Another major methodology is causal mediation analysis (Todd et al.; Vig et al., 2020; Meng et al., 2022; Geva et al., 2023; Hendel et al., 2023; Wu et al., 2023; Dumas et al., 2024), which measures the effect of intervening on a hidden state to determine its causal contribution to the model's output. Recent work also investigates the superposition hypothesis (Elhage et al., 2022; Scherlis et al., 2022). To disentangle such representations, sparse autoencoders (SAEs) have been employed to extract interpretable features from high-dimensional activations (Gao et al., 2024; Marks et al., 2024; Anthropic., 2024; Ferrando et al., 2024).

8 Conclusion

We show that large language models (LLMs) exhibit an *internal chain-of-thought*. Two claims ground our study: (i) distinct subtasks are learned at different network depths, and (ii) these subtasks are executed sequentially across layers. On a benchmark of 15 two-step composite tasks, we employ layer-from context-masking and propose a novel cross-task patching method, confirming (i). To examine claim (ii), we apply LogitLens to decode hidden states, revealing a consistent layerwise execution pattern. We further replicate our analysis on the real-world TRACE benchmark, observing the same stepwise dynamics. Together, our results enhance LLMs transparency by showing their capacity to internally plan and execute subtasks (or instructions), opening avenues for fine-grained, instruction-level activation steering.

Limitations

While our findings offer mechanistic insights into how LLMs internally decompose and execute composite tasks, several limitations must be acknowledged:

Model and Scale Scope. Our experiments are conducted on four mid-sized open-source models (3B–8B parameters). While these models are representative of common deployment settings, it remains an open question whether the observed phenomena generalize to larger frontier models (e.g., GPT-4, Claude). Differences in architecture, training corpus, and alignment objectives may yield distinct patterns of subtask representation or execution.

Task Construction Bias. The distinct "X-shape" pattern observed in our context-masking experiments (Figure 2) is facilitated by the deliberate design of our benchmark tasks, which feature clearly distinguishable subtask types (e.g., knowledge retrieval followed by algorithmic transformation). This separation likely leads to more temporally distant "learning points" for each subtask across layers. However, when faced with a greater number or more nuanced types of subtasks, particularly those with high conceptual similarity, the context-masking technique might be less effective at clearly disentangling their individual learning stages. Indeed, as observed in our TRACE analysis (Section 6), the learning dynamics for closely related constraints can be more intertwined.

References

Anthropic. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. In *Transformer Circuits Thread*.

Anthropic. 2025. On the biology of a large language model. In *Transformer Circuits Thread*.

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. 2024. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14113–14130.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2023. Analyzing transformers in embedding space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16124–16170.

Clément Dumas, Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. 2024. Separating tongue from thought: Activation patching reveals language-agnostic concept representations in transformers. *arXiv preprint arXiv:2411.08745*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, and 1 others. 2022. Toy models of superposition. *arXiv preprint arXiv:2209.10652*.

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

Javier Ferrando, Oscar Obeso, Senthooran Rajamanoharan, and Neel Nanda. 2024. Do i know this entity? knowledge awareness and hallucinations in language models. *arXiv preprint arXiv:2411.14257*.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235.

Rhys Gould, Euan Ong, George Ogden, and Arthur Conmy. Successor heads: Recurring, interpretable attention heads in the wild. In *The Twelfth International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. Overthinking the truth: Understanding how language models process false demonstrations. In *The Twelfth International Conference on Learning Representations*.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. Linearity of relation decoding in transformer language models. In *The Twelfth International Conference on Learning Representations*.

AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, Ddl Casas, F Bressand, G Lengyel, G Lample, L Saulnier, and 1 others. 2024. Mistral 7b. arxiv 2023. *arXiv preprint arXiv:2310.06825*.

Tianjie Ju, Yijin Chen, Xinwei Yuan, Zhuosheng Zhang, Wei Du, Yubin Zheng, and Gongshen Liu. 2024. Investigating multi-hop factual shortcuts in knowledge editing of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8987–9001.

Zhaoyi Li, Gangwei Jiang, Hong Xie, Linqi Song, Defu Lian, and Ying Wei. 2024. Understanding and patching compositional reasoning in llms. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9668–9688.

Zhuowei Li, Zihao Xu, Ligong Han, Yunhe Gao, Song Wen, Di Liu, Hao Wang, and Dimitris N Metaxas. Implicit in-context learning. In *The Thirteenth International Conference on Learning Representations*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Sheng Liu, Haotian Ye, Lei Xing, and James Zou. 2024b. In-context vectors: making in context learning more effective and controllable through latent space steering. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32287–32307.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2024. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*.

Tianyi Men, Pengfei Cao, Zhuoran Jin, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Unlocking the future: Exploring look-ahead planning mechanistic interpretability in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7713–7724.

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

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Language models implement simple word2vec-style vector arithmetic. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5030–5047.

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

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Distinguishing antonyms and synonyms in a pattern-based neural network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 76–85.

Nostalgebraist. 2020. Interpreting gpt: the logit lens. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, and 1 others. 2022. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

Baturay Saglam, Zhuoran Yang, Dionysis Kalogerias, and Amin Karbasi. 2025. Learning task representations from in-context learning. *arXiv preprint arXiv:2502.05390*.

Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. 2022. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*.

Suzanna Sia, David Mueller, and Kevin Duh. 2024. Where does in-context learning happen in large language models? *Advances in Neural Information Processing Systems*, 37:32761–32786.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*.

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

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

Ruoyu Wang, Zhipeng Yang, Zinan Zhao, Xinyan Tong, Zhi Hong, and Kun Qian. 2024. Llm-based robot task planning with exceptional handling for general purpose service robots. In *2024 43rd Chinese Control Conference (CCC)*, pages 4439–4444. IEEE.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Sarah Wiegreffe, Oyvind Tafjord, Yonatan Belinkov, Hannaneh Hajishirzi, and Ashish Sabharwal. 2024.

Answer, assemble, ace: Understanding how transformers answer multiple choice questions. *arXiv preprint arXiv:2407.15018*.

Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. 2023. Depn: Detecting and editing privacy neurons in pretrained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2875–2886.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Liu Yang, Ziqian Lin, Kangwook Lee, Dimitris Papailiopoulos, and Robert Nowak. 2025. Task vectors in in-context learning: Emergence, formation, and benefit. *arXiv preprint arXiv:2501.09240*.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024b. Do large language models latently perform multi-hop reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10210–10229.

Sohee Yang, Nora Kassner, Elena Gribovskaya, Sebastian Riedel, and Mor Geva. 2024c. Do large language models perform latent multi-hop reasoning without exploiting shortcuts? *arXiv preprint arXiv:2411.16679*.

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

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. In *The Twelfth International Conference on Learning Representations*.

Xinghua Zhang, Haiyang Yu, Cheng Fu, Fei Huang, and Yongbin Li. 2024. Iopo: Empowering llms with complex instruction following via input-output preference optimization. *arXiv preprint arXiv:2411.06208*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

A Datasets

We evaluate our two central claims on 15 composite tasks spanning four categories. Each composite task is composed of two sequential subtasks, denoted as $s_1 \circ s_2$, and is designed to probe whether LLMs internally represent and apply these subtasks in a layered fashion. A summary of all composite tasks can be found in Table 3. We also describe the subtasks used in the cross-task patching experiment.

Antonym–Uppercase This composite dataset is constructed by capitalizing answers from an antonym dataset. The underlying antonym pairs are drawn from (Nguyen et al., 2017), which includes both antonyms and synonyms (e.g., "good → bad"). We follow the preprocessing procedure described in (Todd et al.), then capitalize the antonym response, producing pairs like "good → Bad". Intermediate answers are defined as the antonym in lowercase (e.g., "bad") and the capitalized form of the query (e.g., "Good").

Synonym–Uppercase Constructed in the same way as Antonym–Uppercase, using synonym pairs from (Nguyen et al., 2017). We capitalize the synonym to form the composite answer, and treat the lowercase synonym and capitalized query as intermediate outputs.

Country_Capital–Lowercase This dataset is built from a country–capital mapping dataset (Todd et al.). We lowercase the capital names to form the composite answers. For example, "France → paris".

Landmark_Country–Lowercase Pairs landmark names with their respective countries, based on data from (Hernandez et al.). The country name is lowercased to form the composite answer.

Product_Company–Lowercase This dataset contains commercial products paired with the companies that produce them, also curated from (Hernandez et al.). The company name is lowercased to produce the final output.

Choose_Last–Country_Capital We use the country–capital dataset (Todd et al.) to create lists of three countries sampled at random. The final answer is the capital of the last country in the list. Intermediate outputs include the last country name and the capital of the first country.

Choose_Last–Landmark_Country Follows the same format as Choose_Last–Country_Capital, using landmark–country pairs from Hernandez et al.. The model must extract the last landmark and map it to its corresponding country.

Adjective_v_Verb–Antonym This dataset tests syntactic category identification and semantic reasoning. From the antonym dataset (Nguyen et al., 2017), we select words that are unambiguously adjectives or verbs. Each list contains two verbs and one adjective. The model must identify the adjective and return its antonym.

Adjective_v_Verb–Synonym Constructed identically to Adjective_v_Verb–Antonym, but with synonym retrieval instead.

Choose_Last–First_Letter Constructed from a simple list-based selection dataset (Todd et al.). The model is prompted with a list of three items and must return the first letter of the last item.

Choose_Last–Uppercase Similar to Choose_Last–First_Letter, but instead of returning the first letter, the model is required to return the last item in uppercase form.

Antonym–English_French We translate answers from the antonym dataset to French using the Google Translate API. The composite task consists of performing the antonym transformation and then translating the result. Intermediate answers include the English antonym and the French translation of the query.

Antonym–English_Spanish Same as Antonym–English_French, but translated to Spanish.

Landmark_Country–English_French Based on (Hernandez et al.), we first retrieve the country associated with a landmark, then translate the country name to French.

Landmark_Country–English_Spanish Constructed in the same way as Landmark_Country–English_French, using Spanish as the target language.

Below, we describe the individual subtasks (s_1 and s_2) used in the cross-task patching experiments. Each subtask is a functional unit that appears as part of one or more composite tasks.

Category	**Composite Tasks**
Knowledge–Algorithmic	Antonym–Uppercase
	Synonym–Uppercase
	Country_Capital–Lowercase
	Landmark_Country–Lowercase
	Product_Company–Lowercase
Extractive–Knowledge	Choose_Last–Country_Capital
	Choose_Last–Landmark_Country
	Adjective_v_Verb–Antonym
	Adjective_v_Verb–Synonym
Extractive–Algorithmic	Choose_Last–First_Letter
	Choose_Last–Uppercase
Knowledge–Translation	Antonym–English_French
	Antonym–English_Spanish
	Landmark_Country–English_French
	Landmark_Country–English_Spanish

Table 3: Summary of the 15 composite tasks used in our experiments. Each task consists of a pair of subtasks ($s_1 \circ s_2$), spanning four categories: Knowledge–Algorithmic, Extractive–Knowledge, Extractive–Algorithmic, and Knowledge–Translation.

Antonym The antonym dataset is based on data from (Nguyen et al., 2017), which contains word pairs that are either antonyms or synonyms (e.g., "`good` → `bad`", "`spirited` → `fiery`"). We follow the same preprocessing protocol as in (Todd et al.).

Synonym This dataset is also derived from (Nguyen et al., 2017), containing word pairs with synonym relationships. Preprocessing follows the same steps as the antonym dataset.

Country_Capital This dataset consists of country–capital pairs (e.g., "`France` → `Paris`"), taken from (Todd et al.).

Landmark_Country Includes landmark–country pairs such as "`Eiffel Tower` → `France`", based on the dataset from (Hernandez et al.).

Product_Company Contains entries mapping commercial products to the companies that produce or sell them (e.g., "`iPhone` → `Apple`"). Also sourced from (Hernandez et al.).

Choose_Last Constructed by sampling three items and asking the model to return the last item. Data sourced from (Todd et al.).

Adjective_v_Verb This dataset is designed to test part-of-speech reasoning. Each example contains a list of two verbs and one adjective, and the model must identify the adjective. Source: (Todd et al.).

Uppercase A simple string transformation task where the model is required to convert the input to uppercase. Examples and format are adapted from (Todd et al.).

Lowercase Analogous to the Uppercase task, but the model is required to convert the input to lowercase. Based on the same dataset used in (Todd et al.).

First_Letter The task involves selecting the first letter of a given word. We construct this by reusing inputs from the Uppercase dataset and extracting only the first character.

Translation (English–French / English–Spanish) We use bilingual word pairs from (Conneau et al., 2017) for English–French and English–Spanish translations. Each example consists of an English word and its corresponding translation. We follow the preprocessing pipeline used in (Todd et al.).

B Results of Layer-from Context-Masking

We present the complete results of the Layer-from Context-Masking experiments across four models. Each figure visualizes the layer-wise performance on all 15 composite tasks, showing how masking context information from progressively later layers affects the model's ability to complete subtasks and composite outputs.

- Figure 6 shows results for Llama-3.1-8B.
- Figure 7 shows results for Mistral-7B.
- Figure 8 shows results for Qwen2.5-7B.
- Figure 9 shows results for Llama-3.2-3B.

C Results of Cross-Task Patching

We report the full results of the Cross-Task Patching experiment across all four models. Table 4 summarizes the subtask vector strength for each model, indicating how well activations from composite tasks can transfer to individual subtasks.

D Results of Logit Decoding

We present the complete results of the Logit Decoding analysis for all four models. Each model has two figures: (1) Mean Reciprocal Rank (MRR) scores of component outputs (attention and MLP layers), and (2) Mean Reciprocal Rank (MRR) scores of residual stream.

- Figures 10 and 11 show results for Llama-3.1-8B.
- Figures 12 and 13 show results for Mistral-7B.
- Figures 14 and 15 show results for Qwen2.5-7B.
- Figures 16 and 17 show results for Llama-3.2-3B.

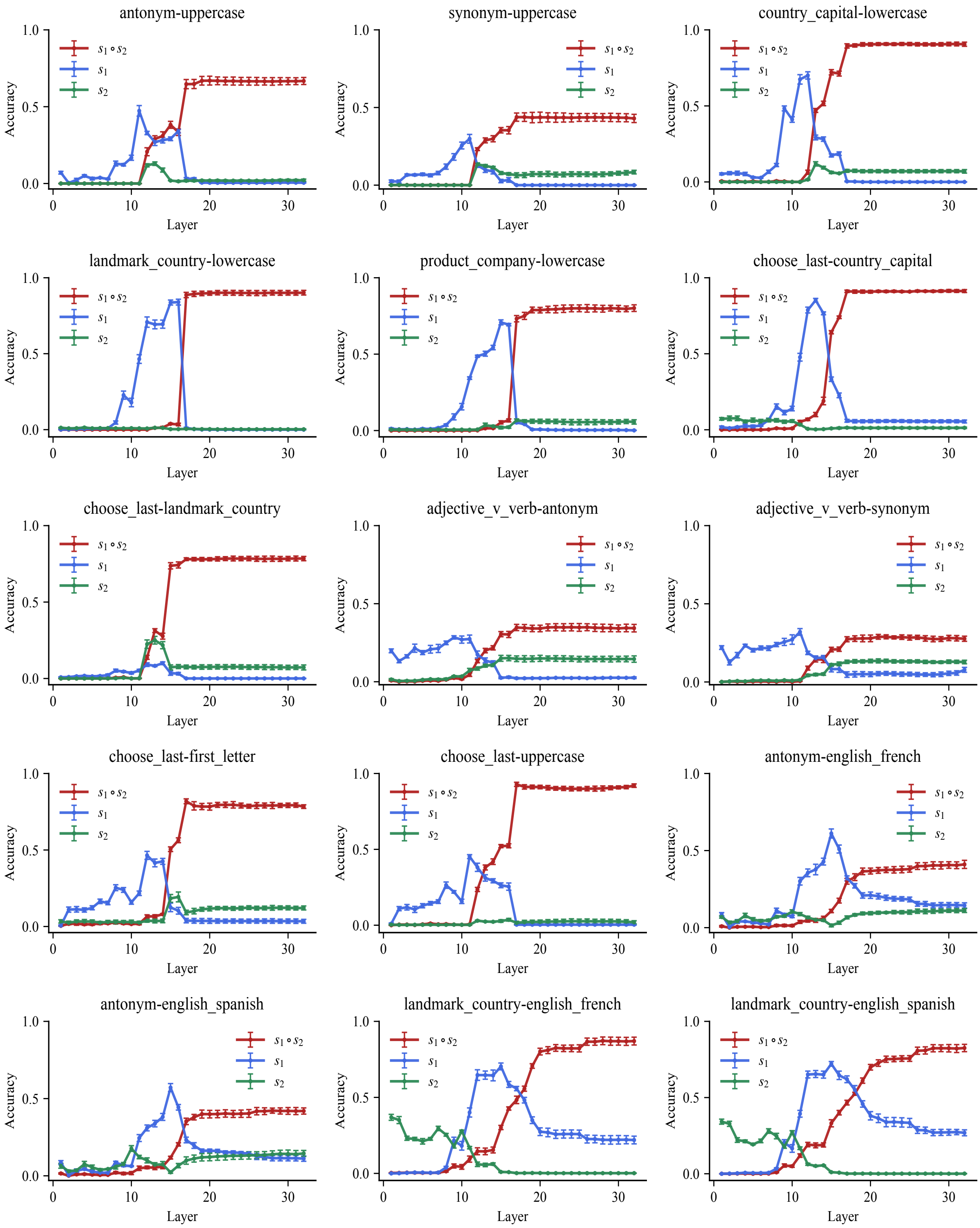

Figure 6: Layer-from context-masking results for all composite tasks in Llama-3.1-8B.

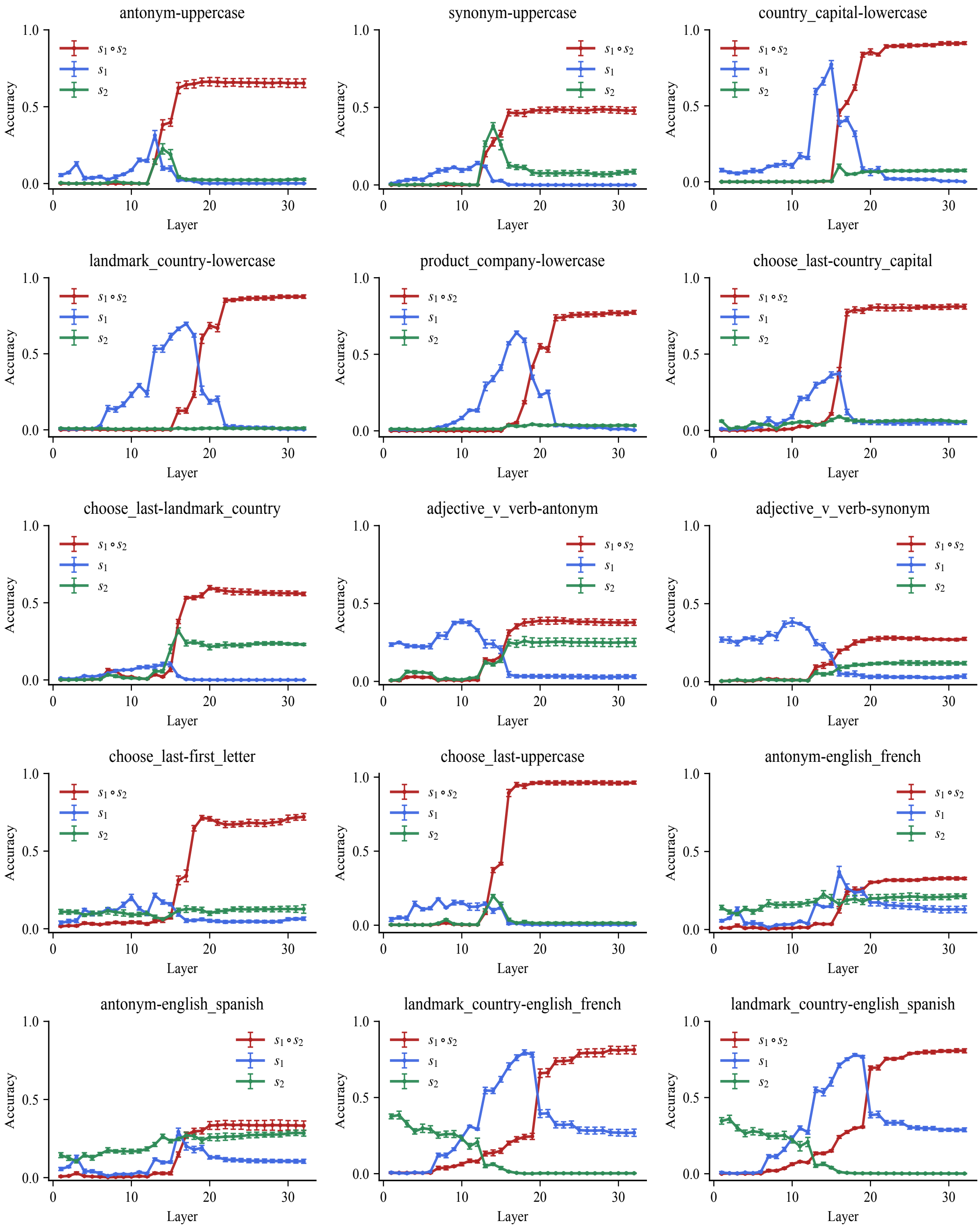

Figure 7: Layer-from context-masking results for all composite tasks in Mistral-7B.

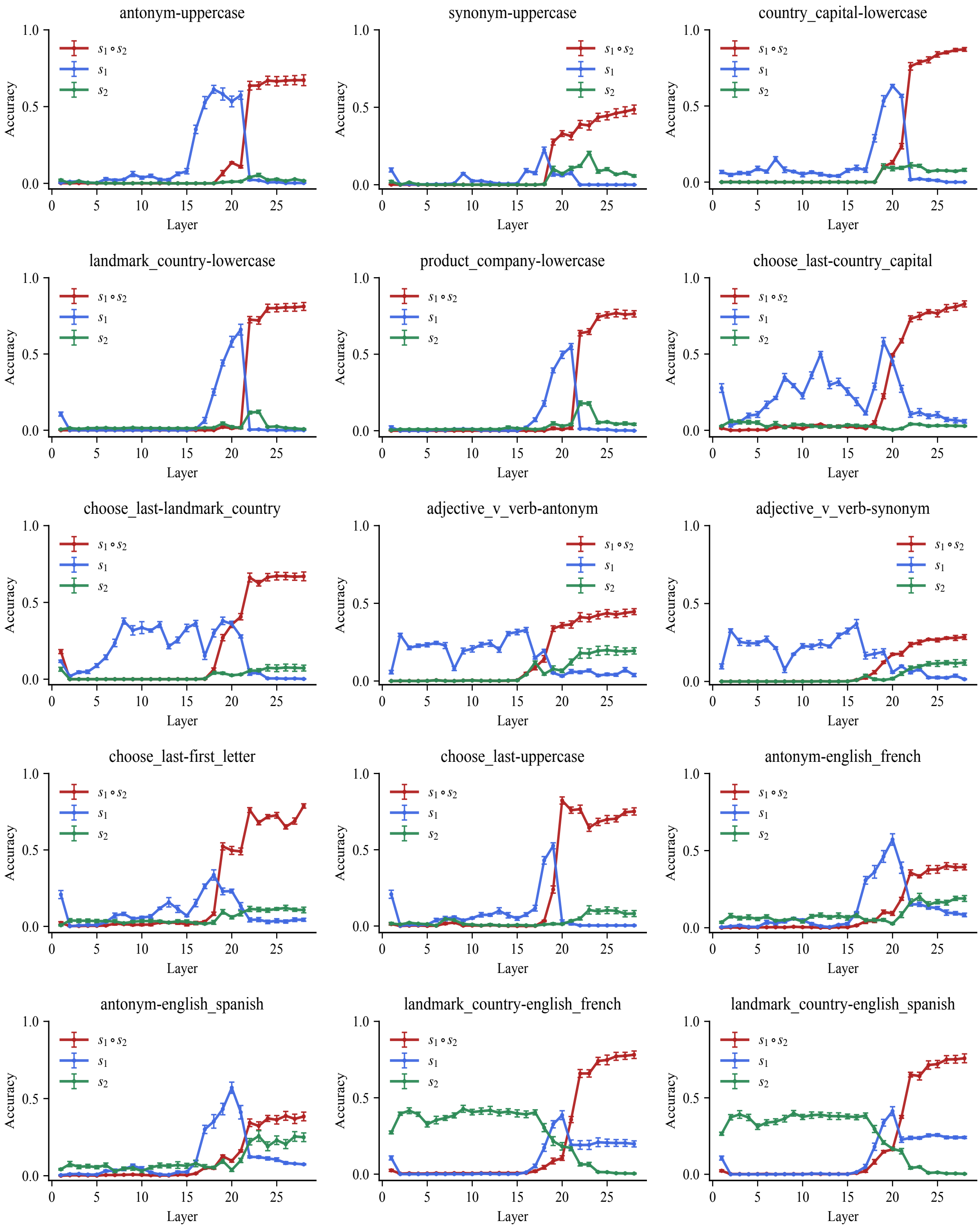

Figure 8: Layer-from context-masking results for all composite tasks in Qwen2.5-7B.

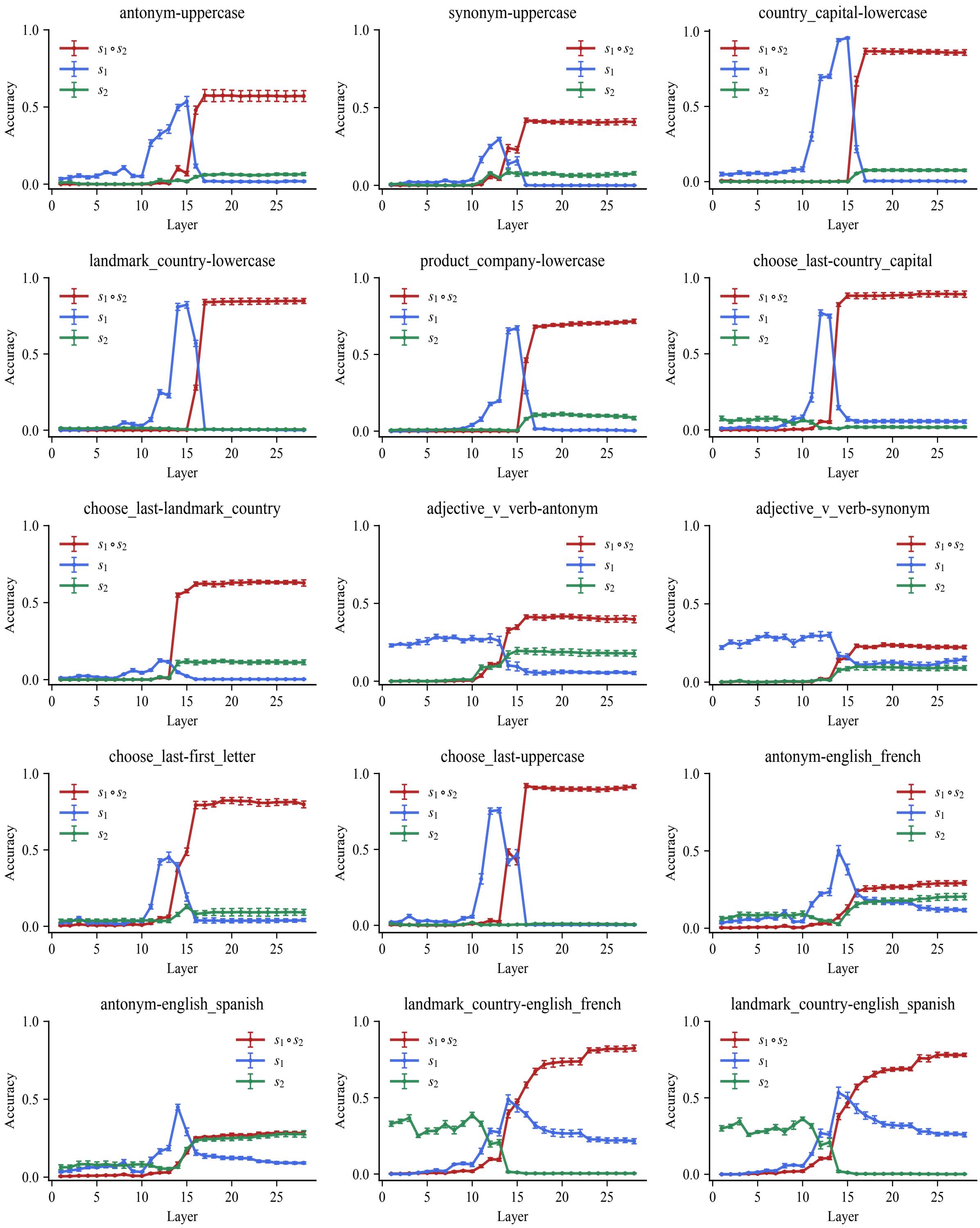

Figure 9: Layer-from context-masking results for all composite tasks in Llama-3.2-3B.

Composite Task	Llama-3.1-8B	Mistral-7B	Qwen2.5-7B	Llama-3.2-3B
antonym-uppercase				
s_1 (antonym)	0.92 ± 0.02	0.26 ± 0.07	0.83 ± 0.02	0.87 ± 0.05
s_2 (uppercase)	0.24 ± 0.03	0.35 ± 0.02	0.03 ± 0.02	0.34 ± 0.04
synonym-uppercase				
s_1 (synonym)	0.90 ± 0.05	0.06 ± 0.03	0.39 ± 0.02	0.82 ± 0.04
s_2 (uppercase)	0.19 ± 0.05	0.40 ± 0.03	0.09 ± 0.03	0.11 ± 0.02
country_capital-lowercase				
s_1 (country_capital)	0.88 ± 0.03	0.20 ± 0.03	0.49 ± 0.05	1.02 ± 0.02
s_2 (lowercase)	0.49 ± 0.05	0.45 ± 0.08	0.34 ± 0.03	0.31 ± 0.03
landmark_country-lowercase				
s_1 (landmark_country)	0.96 ± 0.02	0.92 ± 0.02	0.67 ± 0.02	0.96 ± 0.01
s_2 (lowercase)	0.07 ± 0.03	0.11 ± 0.04	0.17 ± 0.04	0.01 ± 0.00
product_company-lowercase				
s_1 (product_company)	1.01 ± 0.01	0.86 ± 0.03	0.65 ± 0.02	0.99 ± 0.03
s_2 (lowercase)	0.05 ± 0.01	0.46 ± 0.04	0.36 ± 0.05	0.06 ± 0.03
choose_last-country_capital				
s_1 (choose_last)	0.44 ± 0.03	0.32 ± 0.03	0.37 ± 0.05	0.26 ± 0.04
s_2 (country_capital)	0.98 ± 0.02	0.99 ± 0.01	0.97 ± 0.01	0.99 ± 0.01
choose_last-landmark_country				
s_1 (choose_last)	0.03 ± 0.02	0.00 ± 0.00	0.05 ± 0.04	0.03 ± 0.01
s_2 (landmark_country)	0.94 ± 0.01	0.96 ± 0.01	0.91 ± 0.01	0.96 ± 0.01
adjective_v_verb-antonym				
s_1 (adjective_v_verb)	0.38 ± 0.11	0.21 ± 0.05	0.54 ± 0.08	0.16 ± 0.04
s_2 (antonym)	0.94 ± 0.03	0.96 ± 0.04	0.95 ± 0.01	0.91 ± 0.05
adjective_v_verb-synonym				
s_1 (adjective_v_verb)	0.29 ± 0.04	0.39 ± 0.05	0.26 ± 0.07	0.17 ± 0.04
s_2 (synonym)	0.99 ± 0.05	0.76 ± 0.05	0.83 ± 0.07	0.79 ± 0.02
choose_last-first_letter				
s_1 (choose_last)	0.27 ± 0.03	0.08 ± 0.02	0.25 ± 0.04	0.41 ± 0.02
s_2 (first_letter)	1.01 ± 0.01	0.54 ± 0.03	0.95 ± 0.05	0.97 ± 0.03
choose_last-uppercase				
s_1 (choose_last)	0.44 ± 0.03	0.12 ± 0.02	0.26 ± 0.04	0.56 ± 0.03
s_2 (uppercase)	0.99 ± 0.00	0.99 ± 0.00	1.00 ± 0.00	0.98 ± 0.01
antonym-english_french				
s_1 (antonym)	0.92 ± 0.02	0.53 ± 0.05	0.88 ± 0.02	0.76 ± 0.07
s_2 (english_french)	0.39 ± 0.04	0.58 ± 0.05	0.45 ± 0.01	0.60 ± 0.06
antonym-english_spanish				
s_1 (antonym)	0.91 ± 0.02	0.35 ± 0.05	0.86 ± 0.02	0.70 ± 0.08
s_2 (english_spanish)	0.45 ± 0.03	0.69 ± 0.02	0.57 ± 0.03	0.70 ± 0.01
landmark_country-english_french				
s_1 (landmark_country)	0.93 ± 0.01	0.96 ± 0.01	0.92 ± 0.01	0.87 ± 0.01
s_2 (english_french)	0.02 ± 0.00	0.02 ± 0.01	0.02 ± 0.01	0.00 ± 0.00
landmark_country-english_spanish				
s_1 (landmark_country)	0.93 ± 0.01	0.95 ± 0.01	0.92 ± 0.01	0.90 ± 0.01
s_2 (english_spanish)	0.01 ± 0.00	0.01 ± 0.01	0.01 ± 0.01	0.00 ± 0.00

Table 4: Subtask vector strength for all composite tasks across four models.

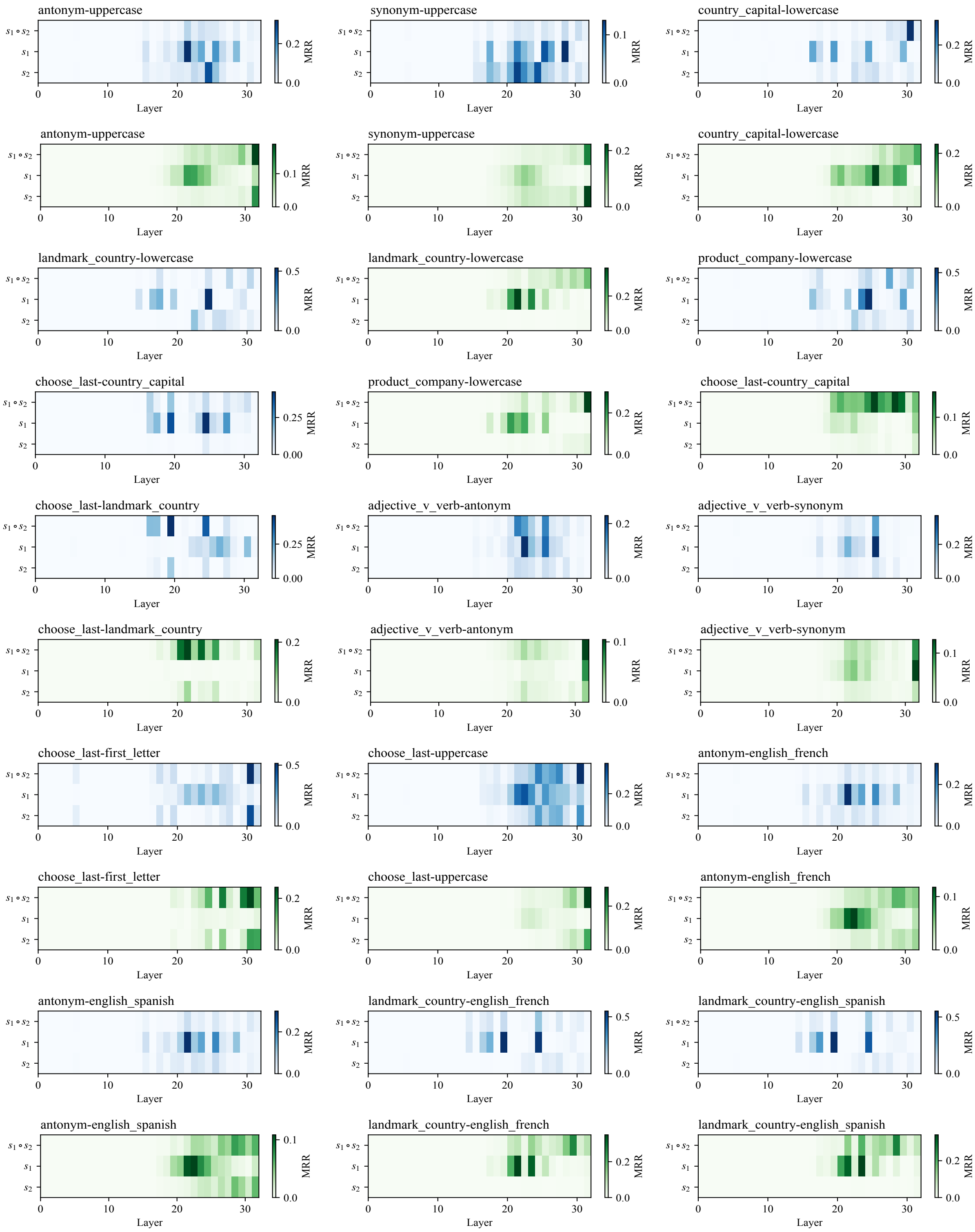

Figure 10: Heatmaps of attention and MLP block decoding results for all tasks in Llama-3.1-8B.

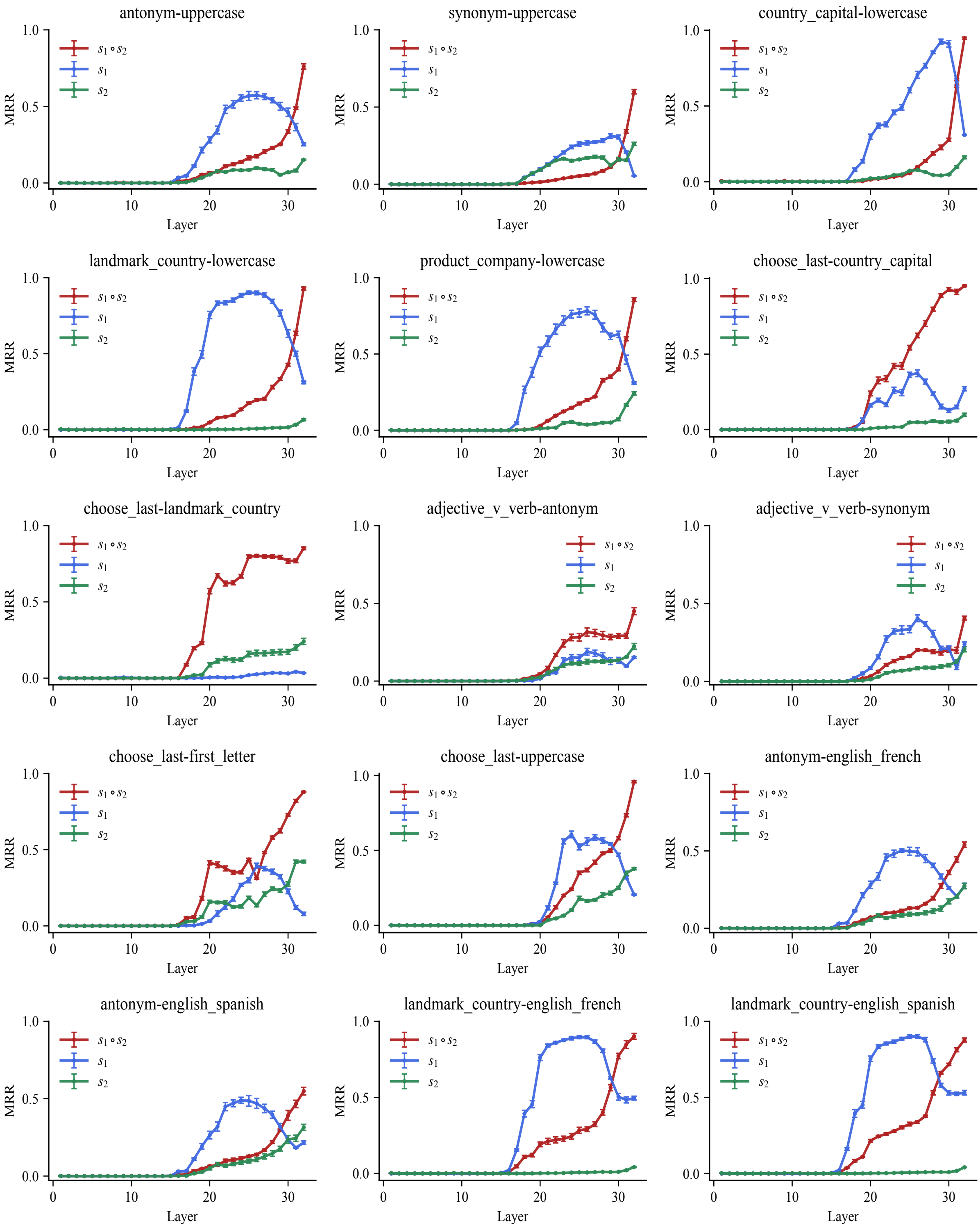

Figure 11: Decoding results of residual stream for all tasks in Llama-3.1-8B.

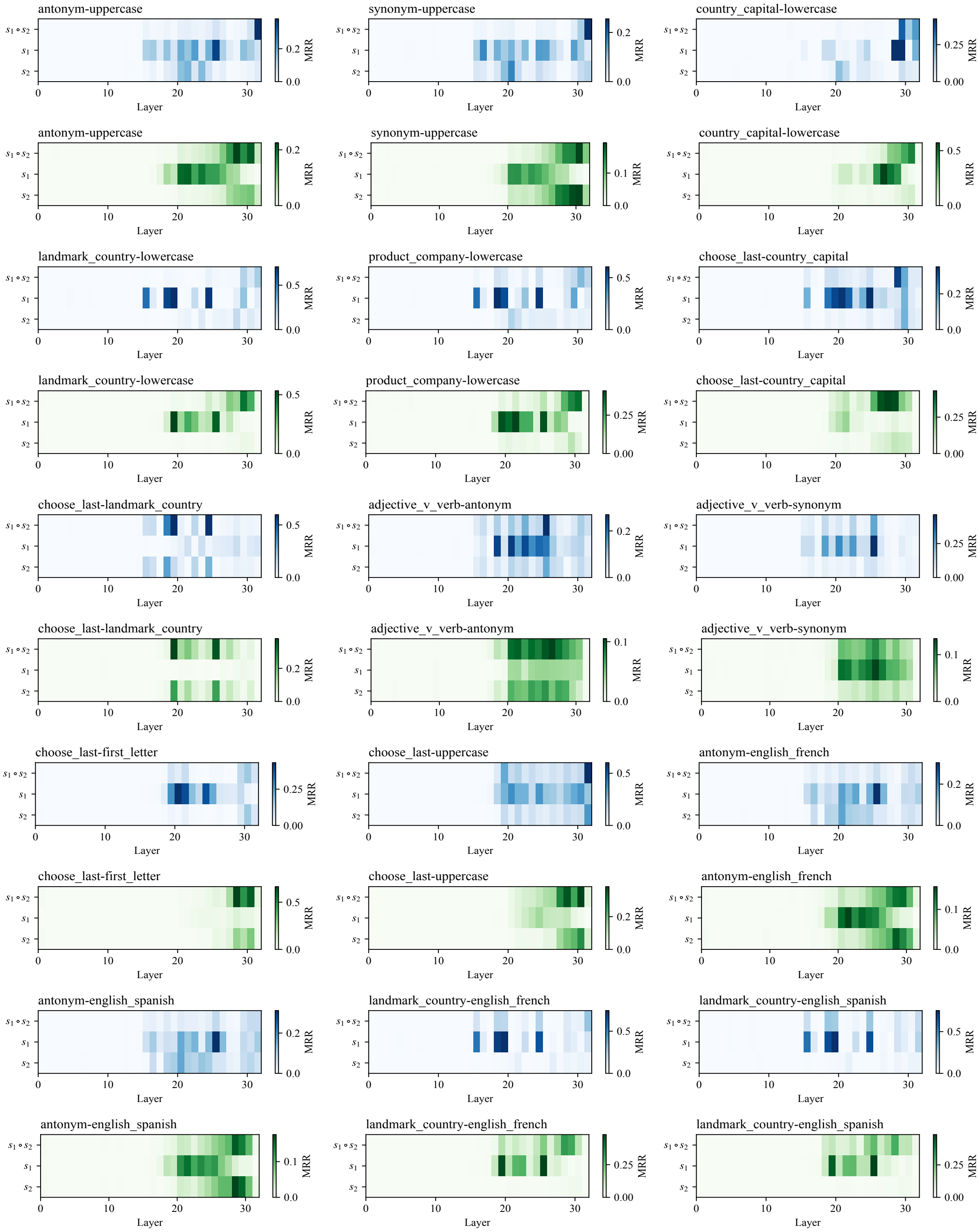

Figure 12: Heatmaps of attention and MLP block decoding results for all tasks in Mistral-7B.

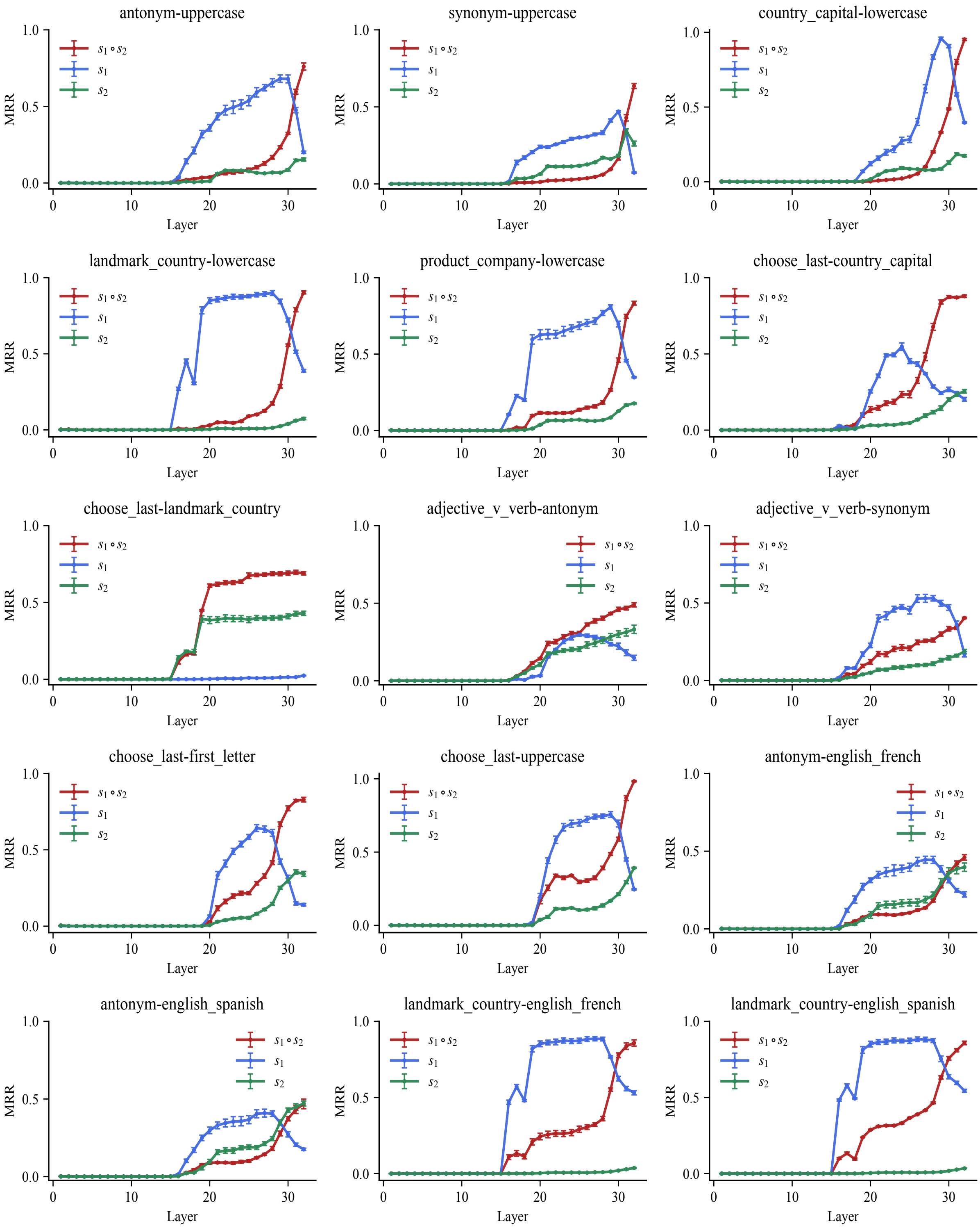

Figure 13: Decoding results of residual stream for all tasks in Mistral-7B.

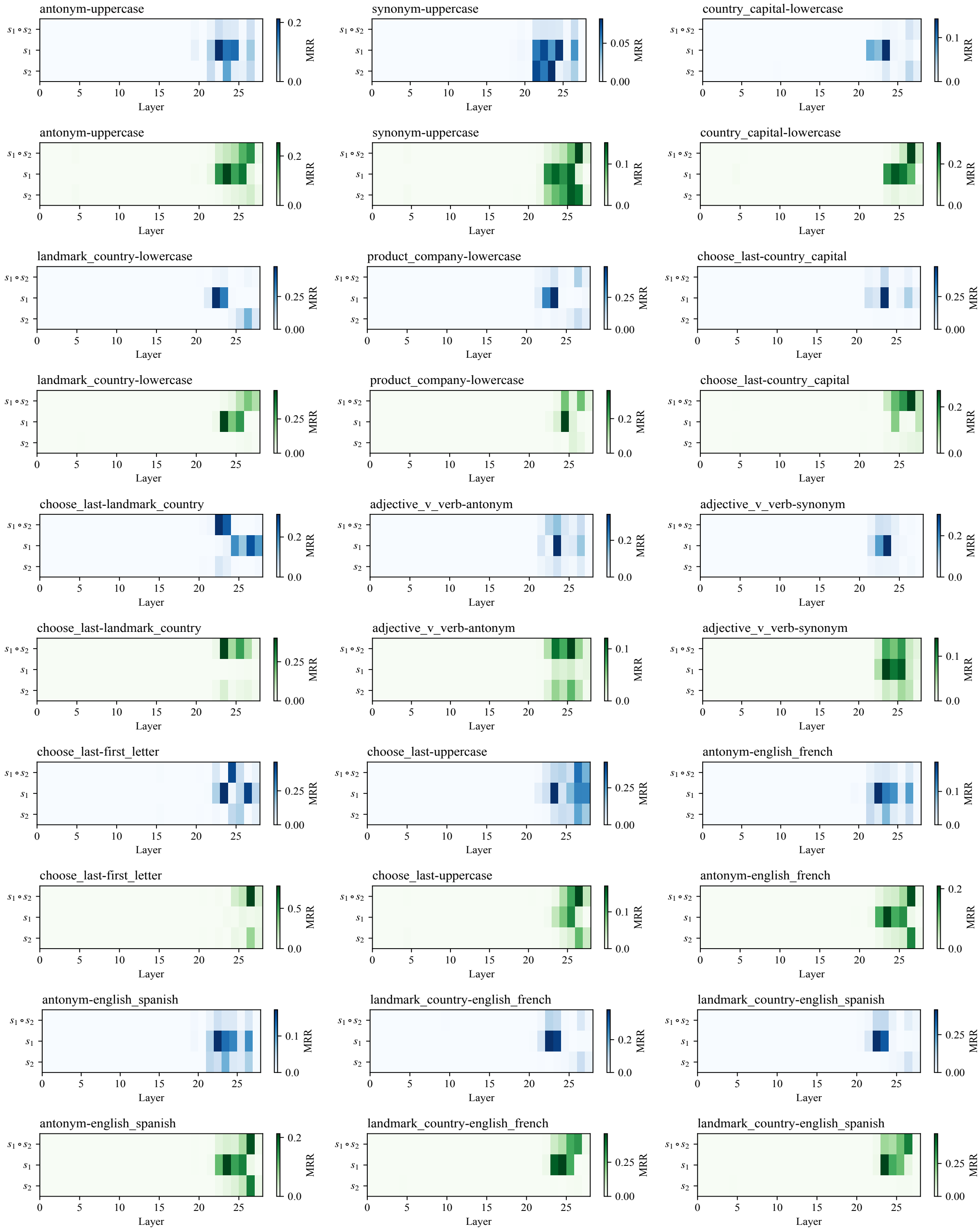

Figure 14: Heatmaps of attention and MLP block decoding results for all tasks in Qwen2.5-7B.

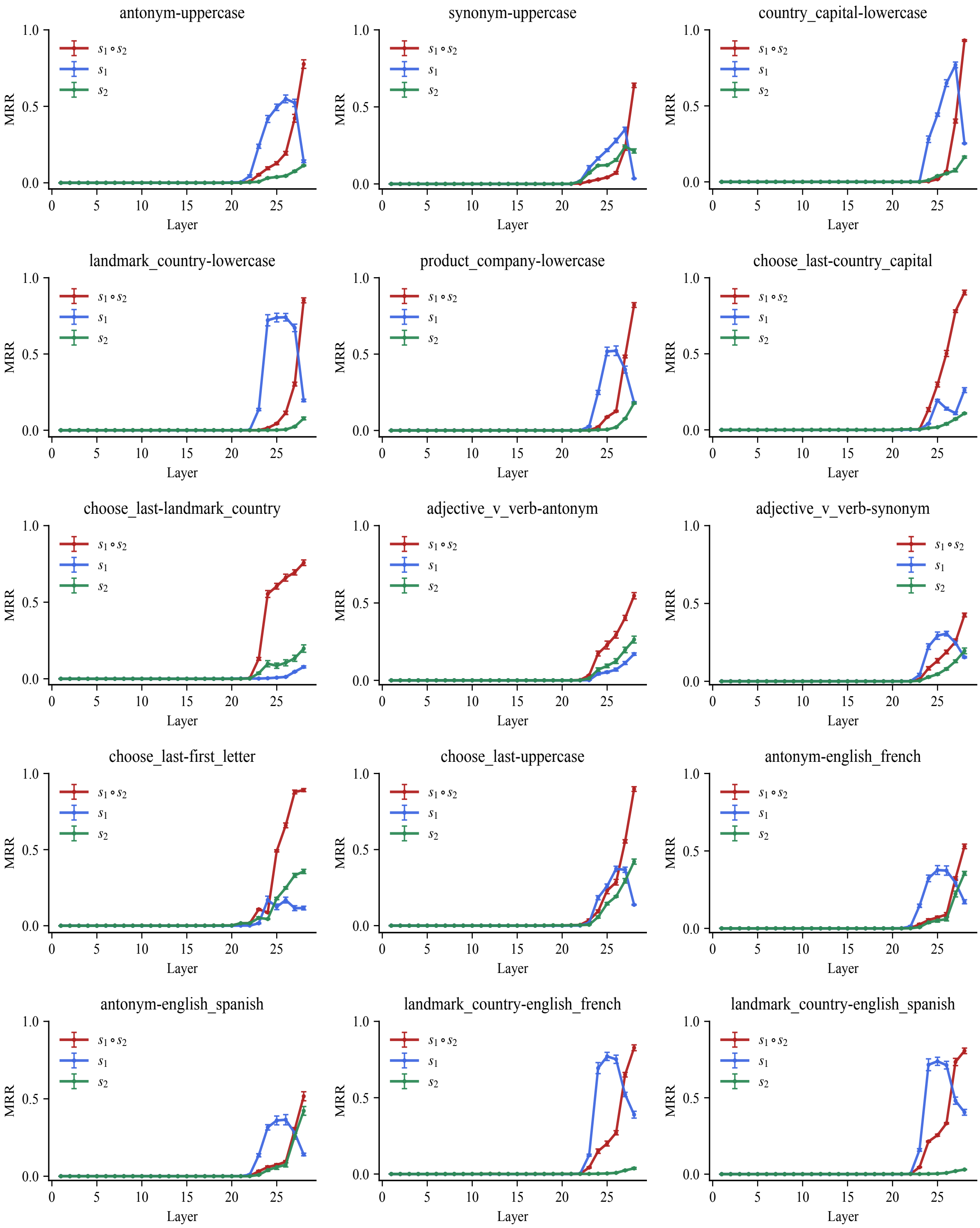

Figure 15: Decoding results of residual stream for all tasks in Qwen2.5-7B.

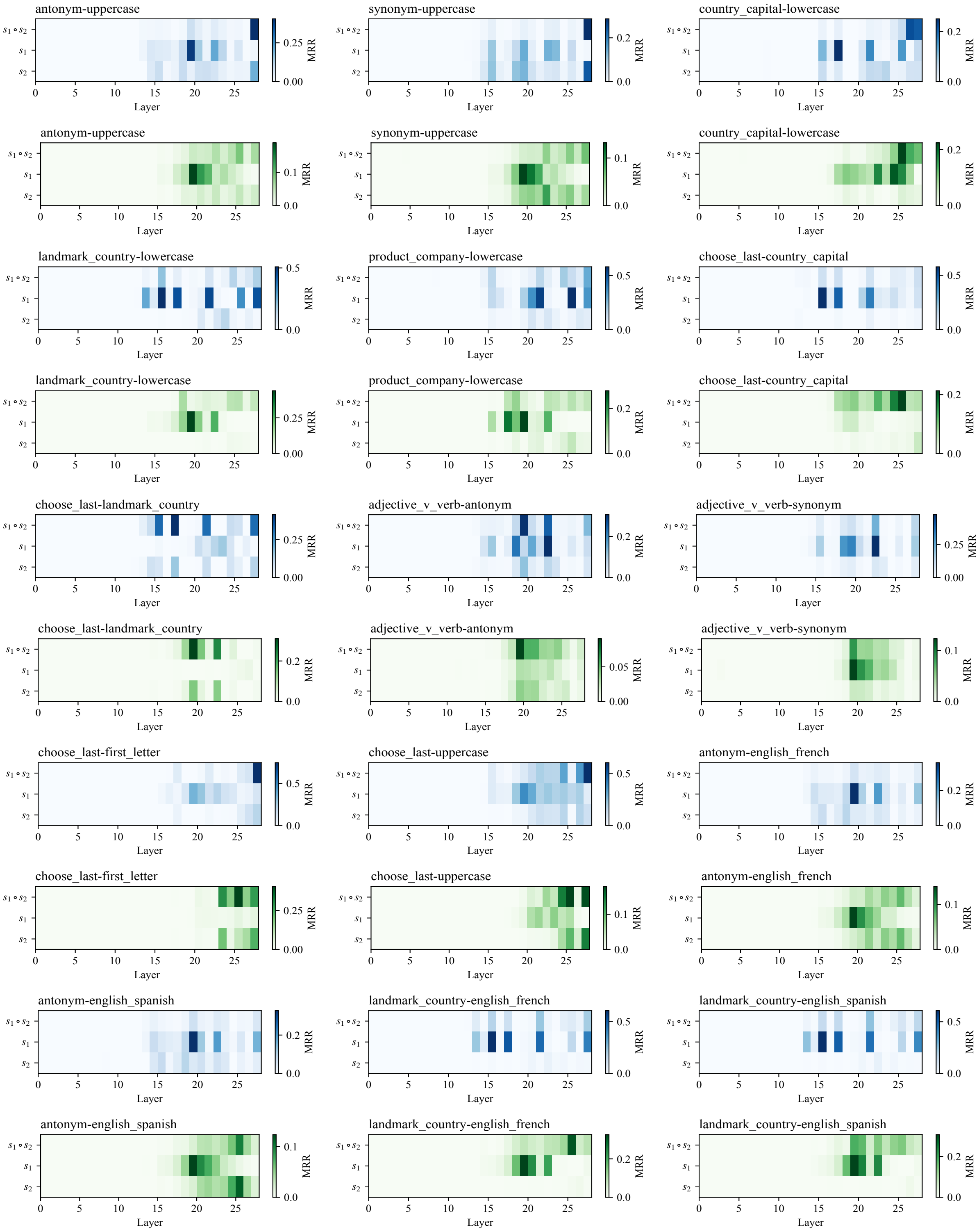

Figure 16: Heatmaps of attention and MLP block decoding results for all tasks in Llama-3.2-3B.

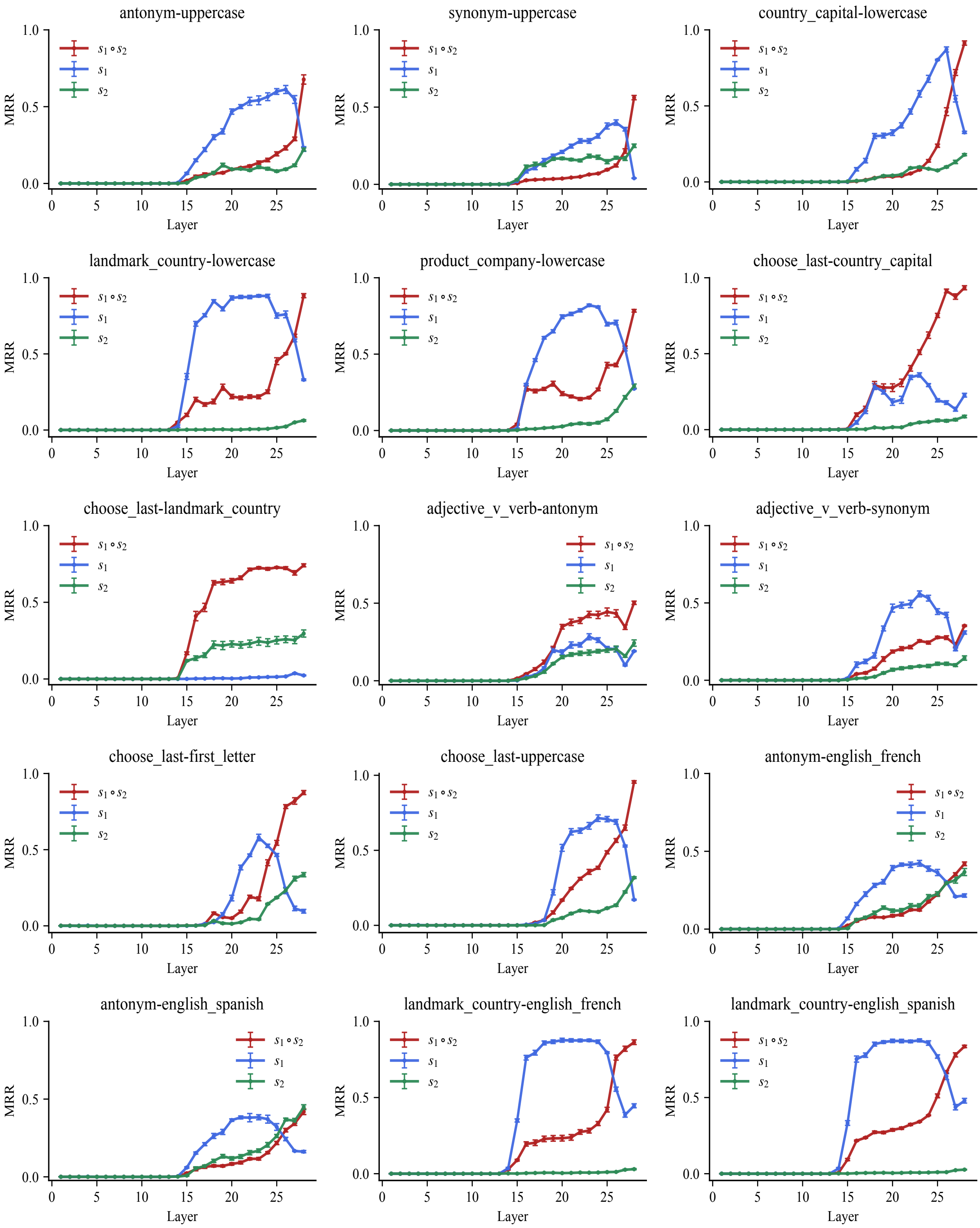

Figure 17: Decoding results of residual stream for all tasks in Llama-3.2-3B.