Tracing and Dissecting How LLMs Recall Factual Knowledge For Real World Questions

Anonymous ACL submission

Abstract

Recent advancements in large language models (LLMs) have shown promising ability to perform commonsense reasoning, bringing machine closer to human-like understanding. However, deciphering the internal reasoning processes of LLMs remains challenging due to the complex interdependencies among generated tokens, especially in practical question-answering. In this study, we introduce a two-dimensional analysis framework—comprising token back-tracing and individual token decoding—to uncover how LLMs conduct commonsense reasoning. Through explanatory analysis of three typical reasoning datasets, we identify a consistent three-phase pattern: Subject Augmentation and Broadcasting, Object Retrieval and Reranking, and Conclusion Fusion and Generation. Our findings reveal that LLMs do not lack relevant knowledge but struggle to select the most accurate information based on context during the retrieval and rerank phase. Leveraging these findings, we apply representation engineering and selective fine-tuning to target specific modules responsible for retrieval and rerank errors. Experimental results show large improvements in response accuracy for both in-domain and out-of-domain settings, validating the rationality of the interpreting result.

1 Introduction

Recent progress in large language models (LLMs) have pushed machines closer to achieving human-like capabilities (Krause and Stolzenburg, 2023; Zhou et al., 2020). These models can not only comprehend user queries, but also perform commonsense reasoning based on factual knowledge. As a result, uncovering these abilities has become a focal point of interest. It is crucial for interpreting model behavior and analyzing unexpected errors (e.g., reversal curse (Berglund et al., 2023)), ultimately overcoming the limitations of LLMs.

Research on interpreting LLMs (Geva et al., 2023; Wang et al., 2024; Dai et al., 2022; Xie et al., 2024) often simplifies reasoning by focusing on factual triplets like "*Ganesha is a Hindu god*". These studies examine how models derive the object ("*Hindu*") from the subject ("*Ganesha*") as well as the relation ("*is*"). However, in real-world scenarios, model must go beyond these triplets to understand the question, select relevant facts, and synthesize information to provide an answer. For example, when asked, "*Is Ganesha associated with Thor?*", the model must grasp the context, recognize that "*Ganesha is a Hindu god*" from all Ganesha-related facts, and conclude they are not related. In contrast, for the question, "*Does Ganesha look like a tiger?*", the model in turn focuses on appearance-related facts, such as "*Ganesha is depicted with an elephant head*". Understanding how models select appropriate factual knowledge and analyze it to reach conclusions is crucial for comprehending their overall reasoning process. This holistic approach should extend beyond simple triplet analysis and can better reflect the complexity of real-world reasoning tasks.

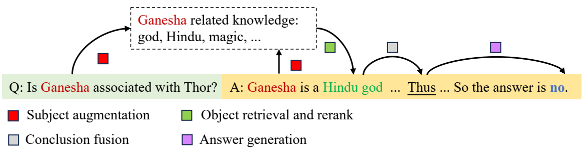

Ganesha related knowledge: god, Hindu, magic, ...
Q: Is Ganesha associated with Thor?
A: Ganesha is a Hindu god ... Thus ... So the answer is no.
🟥 Subject augmentation
🟩 Object retrieval and rerank
⬜ Conclusion fusion
🟪 Answer generation

Figure 1: Model inner reasoning process on commonsense reasoning tasks.

In this study, we aim to decipher the commonsense reasoning process within the response of LLMs. The challenge lies in the dense interconnectivity of token generation, where each generated token is influenced by multiple preceding ones, leading to a recursive analytical complexity. Existing interpretability tools cannot be directly applied to analyze this complexity. To decipher the multi-token generation process, we deign a new framework by breaking down the analysis into two dimensions: token back-tracing and individual token decoding. Token back-tracing starts from the answer and traces back to the original question. It identifies intermediate key tokens with significant direct impact through causal analysis. This reveals a chain of crucial information transfers between tokens, as shown in Fig. 1. For individual token decoding, following Wang et al. (2023), we adopt an "explain then verify" strategy. We first identify and decode the semantic information within the key modules. Then these key modules are knocked out to verify the reliability of results.

The interpreting analysis of three typical reasoning datasets revealed a consistent pattern in models' commonsense reasoning. The process unfolds in three phases: 1) **Subject augmentation and broadcast**: Firstly, the model generates extensive subject-related information through attention heads and MLP, and broadcasts it to subsequent key positions (e.g., sentence endings); 2) **Object retrieval and rerank**: the model retrieves the previously generated subject information with attention heads and reorders it using MLPs when predicting attributes.; and 3) **Conclusion fusion and generation**: the attributes are further transported to the conclusion through heads and generate corresponding conclusions, ultimately forming the answer. Based on this pattern, we further analyzed the failure cases of current models. One key finding is that *LLMs are not unaware of relevant facts, but rather struggle to select the most accurate fact during retrieval and rerank based on contextual cues*. This motivated us to develop a direct application of interpretability findings: by identifying specific modules through explanatory localization, we employed selective fine-tuning and representation engineering to optimize the attribute retrieval and rerank. Results show significant improvement in model performance, simultaneously validating the rationality of the interpretability results.

We summarize our contributions as follows: (1) We introduce an effective interpreting framework that combines token back-tracing with individual token decoding to understand how LLMs reason across multiple tokens. (2) We break down how language models perform commonsense reasoning into human-understandable steps: LLMs first augment related facts and broadcast the information into the proceeding key positions, subsequently retrieving and re-ranking these facts to predict correct object, and finally fusing and generating conclusions. (3) Using the interpreting result, we identify that on commonsense reasoning tasks, LLMs often fail to retrieve and rerank correct facts, leading to erroneous reasoning or conclusions. By selectively fine-tuning key heads and MLPs, the performance of reasoning is enhanced, especially for out-of-domain samples. It validates the reliability of the interpreting results.

2 Related Works

2.1 Mechanistic Interpretability

Mechanistic interpretability in LLMs aims to understand model behavior by reverse-engineering the internal computational processes. Logit attribution is a tool that projects internal vectors into the vocabulary space to interpret the information encoded within these representations. Several studies (Geva et al., 2021b, 2022; Dar et al., 2023; Belrose et al., 2023) have utilized this method to uncover a variety of interpretability results. Activation patching (Meng et al., 2022; Wang et al., 2023; Goldowsky-Dill et al., 2023; Conmy et al., 2023) applies causal interventions to internal model components using corrupted inputs. By examining the resulting changes in model predictions, this approach identifies critical modules and uncovers computational circuits. Numerous works (Lieberum et al., 2023; Zhang et al., 2024; Chen et al., 2024; Hanna et al., 2023) have successfully identified task-specific modules in LLMs using this method. Sparse autoencoders (Bricken et al., 2023; Templeton et al., 2024; Lieberum et al., 2024; Gao et al., 2024) have been employed to decompose internal features into interpretable feature combinations. Knockout is primarily used to verify the importance of components in a circuit (Wang et al., 2023; Olsson et al., 2022). It removes specific components and analyzes the change, where a significant change suggests importance. In this paper, we adapted some tools to interpret model's reasoning process. The comparison of interpreting tools and our selection are discussed in §A.1.

2.2 Model reasoning ability Interpretation

Numerous studies have employed interpretability tools to investigate model mechanisms in reasoning tasks. Geva et al. (2023) explored factual knowledge recall, finding that subject information is enriched in the subject token in early layers, while relation information is passed to the final token, which then uses attention heads to extract the corresponding attribute from the subject representa-

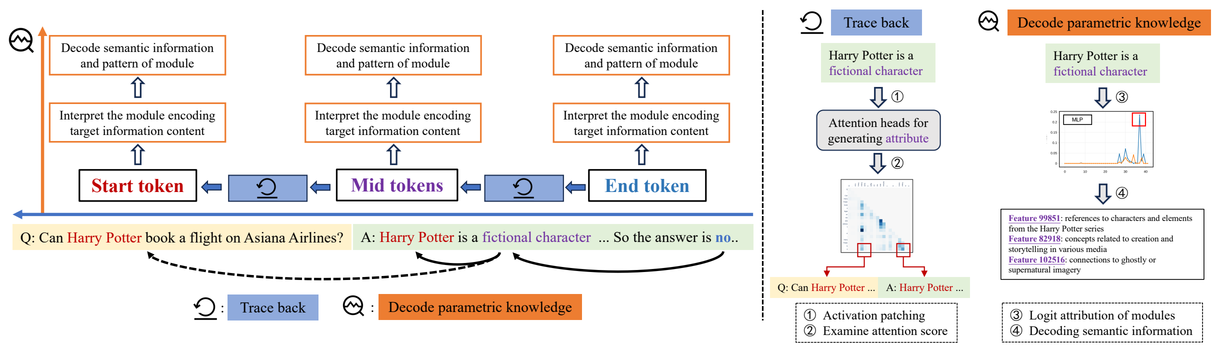

Decode semantic information and pattern of module
Interpret the module encoding target information content
Start token
Mid tokens
End token
Q: Can Harry Potter book a flight on Asiana Airlines?
A: Harry Potter is a fictional character ... So the answer is no..
Trace back
Decode parametric knowledge
Harry Potter is a fictional character
Attention heads for generating attribute
Q: Can Harry Potter ...
A: Harry Potter ...
MLP
Feature 99851: references to characters and elements from the Harry Potter series
Feature 82918: concepts related to creation and storytelling in various media
Feature 102516: connections to ghostly or supernatural imagery
① Activation patching
② Examine attention score
③ Logit attribution of modules
④ Decoding semantic information

Figure 2: Overview of interpreting pipeline: 1) **Tracing back (horizontal)**: we use activation patching to identify the head with causal effect and trace the origin of the information iteratively. 2) **Decoding concept knowledge (vertical)**: we use logit attribution to identify the key module for generating concepts during reasoning at the key position and decode the semantic information within in it.

tion. Building on this, Wang et al. (2024); Dai et al. (2022); Yu and Ananiadou (2024); Geva et al. (2022) identified MLP neurons involved in factual knowledge recall and demonstrated how modulating their activations can control model behavior. Additionally, works such as Yu et al. (2024); Ortu et al. (2024); Yu et al. (2023); Xie et al. (2024) analyzed the balance between retrieved knowledge and parametric memory. Dutta et al. (2024); Hou et al. (2023) examined model mechanisms in reasoning generation tasks. These studies largely focus on elementary retrieval tasks, such as recalling a single fact o from a triplet (s, r, o). In this study, we focus on interpreting the model reasoning process in more complicated commonsense reasoning tasks.

3 Methods

3.1 Preliminary

In our experiments, we uncovered several key token positions in the reasoning process through back tracing: *Subject*, *Object*, *Answer*. These tokens are observed special in experiments and therefore highlighted for better comprehension. (1) **Subject** ($\mathcal{S}$): The subject of inquiry in the question; this is a concept node in a knowledge graph (Speer et al., 2017), representing any entity, idea, or object relevant to commonsense (e.g., "*Harry Potter*" in Figure 2). (2) **Object** ($\mathcal{O}$): The object, paired with $\mathcal{S}$ contains some factual knowledge, is also a concept node. These objects, according to their relevance as accurate fact for the question, can be categorized into predicted objects $\mathcal{O}_p$ (e.g., "*Harry Potter is a 'fictional character'*") and candidate objects $\mathcal{O}_c$ (e.g., "*Harry Potter is a 'wizard'*"). (3) **Answer** ($\mathcal{A}$): The answer to the question, which varies based on the type of question. It may be a binary judgment (e.g., "*yes/no*") or a selection (e.g., "*(2) Kayla*"). We denote the correct and false answers as $\mathcal{A}_t$ and $\mathcal{A}_f$.

Furthermore, through back-tracing, we identified several positions that entail reasoning-related information: (4) **Reasoning conjunctive adverb** ($\mathcal{R}$): we find conjunctive adverbs that connect reasoning steps (e.g., "*Thus*") encodes rich information related to the answer. (5) **Conclusion** ($\mathcal{C}$): terms that convey the affirmative or negating essence of the conclusion sentence, clarifying the stance to the question. (e.g., "*cannot*" in "*Thus, Harry Potter cannot book a flight on Asiana Airlines.*") (6) **Question end** ($\mathcal{Q}_e$): we find abundant subject-related information is encoded at the end of question.

3.2 Methodology

As illustrated in Figure 2, the interpretation process is divided into two orthogonal pipelines. **1) Token back-tracing**: The horizontal pipeline traces the path of tokens from the end to the start. Through causal back-tracing, tokens that are strongly correlated with a target token can be effectively identified, allowing us to focus on the most relevant information flows rather than exhaustively analyzing the dense connections across all tokens. This approach helps identify the key relationships between tokens, thereby pinpointing the crucial positions of key tokens in commonsense reasoning (as defined in §3.1). **2) Decode parametric concept or attribute**: The second pipeline, shown vertically, analyzes the patterns within LLMs when generating a specific token, including inner behaviors and activation characteristics. It explains the behavior of modules (e.g., residual blocks, Attention heads, and MLPs) by evaluating the information related to the target content (e.g., $\mathcal{A}_t$, $\mathcal{A}_f$, $\mathcal{O}_p$ and $\mathcal{O}_c$) within modules' output. Subsequently, it decodes the semantic information and patterns encoded in these modules into human-understandable formats.

Instantiation of tracing token-to-token path. We employ activation patching (Wang et al., 2023) as an effective tool for causal back-tracing. This method, which originates from causal mediation analysis (Vig et al., 2020), enables us to identify significant attention heads through direct effect analysis (as shown in the right side of Fig. 2). We made several improvements to the original method: (1) Developing a refined metric to reduce noise in key module identification. (2) Extending the analysis from final logits to the middle layer outputs. (3) Automating the generation of counterfactual data to obtain large-scale results. See §A.2 for details. In our implementation, we identify heads with the Top-5 direct effect as key contributors to token generation. By analyzing the attention patterns in these important heads, we select the top 2 previous tokens with the highest attention scores as being correlated with the current token, serving as the basis for further tracing and analysis. This process is then iteratively applied to discover the complete transition path across tokens.

Instantiation of decoding parametric concept or attribute. We use logit attribution (nostalgebraist, 2021) to interpret the module behavior across layers. The method projects hidden states into the vocabulary space using the model's pretrained unembedding matrix and obtains its distribution on vocabulary space. Therefore, the method reveals the information contained in current hidden states and explains the contribution of specific heads or MLPs or residual blocks to the predicted token. To address the false identification issue, we calculate the softmax probability of the multiple tokens ($\mathcal{O}_p$, $\mathcal{O}_c$, $\mathcal{A}_t$ or $\mathcal{A}_f$) after projection. This improvement ensures the identified key modules contribute specifically to target prediction rather than all related tokens. The probabilities across layers will form the curves (see examples in Figure 3a), indicating the module's inner reasoning process.

To validate the interpreting results obtained by logit attribution, for MLP, we adopt Sparse Autoencoder (SAE) (Templeton et al., 2024) to decode the semantic information embedded in the parameters and activations. (e.g., Information related to "*magic*" is decoded in MLP of layer 8 when feeding "*Harry Potter*" to the model.) More details are introduced in §A.2.2. Regarding attention heads, we use probing to decode the semantic information. We project the outputs of the heads into the vocabulary space and examine the top-20 tokens in the head's output distribution to decode the semantic information. To validate the functional roles of these key components, we knockout these modules to observe the influence on output.

Method selection and improvement details. We discuss the selection of existing tools in §A.1. The details in instantiations and improvements to all the interpretability tools are available in §A.2.

4 Experiments

4.1 Experiments Overview

Consider a question: "*Q: Can Harry Potter book a flight on Asiana Airlines?*" and Gemma2-9B's output is "*Harry Potter is a fictional character. Fictional characters cannot book flights. Thus, Harry Potter cannot book a flight on Asiana Airlines. So the answer is no.*". Through extensive experimental results, we find the reasoning process: (1) **Subject Augmentation and Broadcast**, at subject token position ($\mathcal{S}$, "*Harry Potter*"), the model extends from the subject to augment relevant object (e.g., "*Wizard*" and "*fictional*"). (2) **Object Retrieval and Rerank**, when predicting object token ($\mathcal{O}_p$, "*fictional*") attention is responsible for retrieving related objects while MLP layers rank the most appropriate one as output. (3) **Conclusion Fusion and Generation**, when predicting conclusion ($\mathcal{C}$, "*cannot*") and answer ($\mathcal{A}$, "*no*") tokens, the model integrates the previous information and generates the final answer through attention heads and MLPs.

Given our analytical pipeline's back-tracing nature, we present the results in reverse chronological order to align with the original investigation procedure. We begin tracing back from $\mathcal{A} \rightarrow \mathcal{C} \rightarrow \mathcal{O}$, and decoding to find the **conclusion fusion and generation** (§4.3). Diving deeper into $\mathcal{O}$, we further observe **object retrieval and rerank** (§4.4). Further tracing the origin of $\mathcal{O}$ leads us to $\mathcal{S}$, uncovering **subject augmentation and broadcast** (§4.5). Additionally, We extend our investigation across different models and datasets in §4.6.

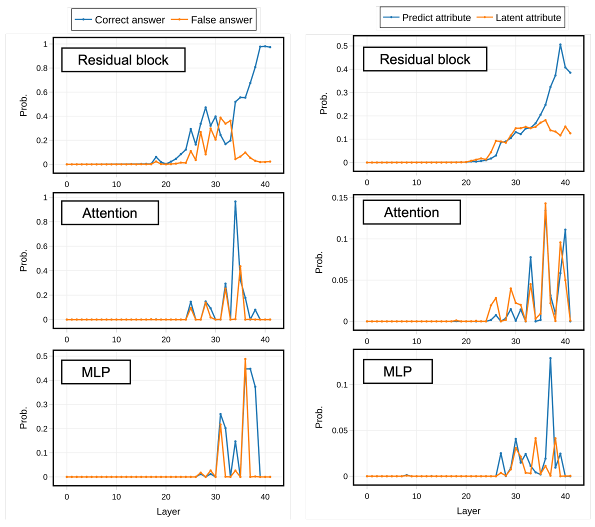
4.2 Experiments Settings

Models. We conducted experiments on two popular open-sourced models, Gemma2-9B (Team et al., 2024) and Llama2-7B (Touvron et al., 2023). The results in the Section 4 primarily focus on Gemma2-9B, as Sparse Autoencoders (SAEs) have been trained for all its layers (including residual and MLP layers) (Lieberum et al., 2024), enabling comprehensive validation of our analyses. See Appendix A.8 for results on Llama2-7B.

Datasets. We selected three widely used commonsense reasoning benchmark datasets: **StrategyQA** (Geva et al., 2021a), **CommonsenseQA** (Talmor et al., 2018), and **SocialIQA** (Sap et al., 2019). These three datasets evaluate distinct aspects of reasoning capabilities (see Tab 16 for details). Given that StrategyQA presents more sophisticated reasoning challenges, we primarily present the StrategyQA results in the main text. Detailed results for the other two datasets are presented in § A.6.

Settings. Following the experimental protocols in Geva et al. (2023); Lieberum et al. (2023), we randomly sampled 1,000 instances from each dataset for our experiments. All figures and tables presented in this paper are averaged results across this sample size, ensuring statistical reliability while maintaining computational feasibility. Detailed analysis of result stability across various sizes of samples is presented in Appendix A.4.

4.3 Conclusion Fusion and Generation



(a) (b)

Figure 3: (a) Logit attribution of $\mathcal{A}_t$ and $\mathcal{A}_f$ at predicting $\mathcal{A}$ on StrategyQA. (b) Logit attribution of $\mathcal{O}_p$ and $\mathcal{O}_c$ at predicting $\mathcal{O}$ on StrategyQA.

We start from decoding the information of $\mathcal{A}_t$ and $\mathcal{A}_f$ (i.e. "*yes*" and "*no*") in residual blocks, attention, and MLP layers at the position of predicting $\mathcal{A}$ as shown in Figure 3a. The curves of residual blocks depicts how the model predicts $\mathcal{A}$ across layers while curves of attention and MLP layers depict the module contribution to the $\mathcal{A}_t$ and $\mathcal{A}_f$. The prediction of $\mathcal{A}$ can be divided into three stages: (i) **Stage** 1 (l $0-24$): Little to no answer-related information is present in residual blocks, attention and MLP layers, indicating the model is still processing the input. (ii) **Stage** 2 (l $25-33$): Information related to the answer increases, yet the probabilities for $\mathcal{A}_t$ and $\mathcal{A}_f$ are close across residual blocks. Within the modules, attention heads begin to convey answer-related information from layer 25 and the MLP follows to encode this information from layer 26. Notably, the heads' outputs show similar information for both $\mathcal{A}_t$ and $\mathcal{A}_f$, but the MLPs' outputs assign a higher probability to $\mathcal{A}_t$. In this stage, the model start to generate an answer but has not yet identified the correct one. (iii) **Stage** 3: By layer 34, the model distinguishes the correct answer $\mathcal{A}_t$, with its probability sharply rising and the $\mathcal{A}_f$'s probability decreasing. At the same layer, the attention output sharply spikes for $\mathcal{A}_t$ (probability near 1.0), while the MLP output is much lower (≈ 0.1). Afterward, the outputs of MLPs further increase $\mathcal{A}_t$'s probability (l $37-38$), leading to the final prediction. In conclusion, attention head is responsible for fusing related information, while the MLP enhances the probability of the correct answer, contributing to generating the final answer.

Head	Top tokens in projection
28.06	`yes, yeah, no, nil, Yes`
32.07	`Noah, node, Noah, no, Nora`
34.09	`denying, denied, denial, deny`
35.14	`ye, Ye, Yea, YE, yes, YES, Yeh`

Table 1: Top-scoring tokens in the key attention heads output when predicting $\mathcal{A}$.

Layer	ID	Feature Explanation
27	76551	questions and answers related to decision-making and assessments.
30	21336	affirmative and negative responses to questions.
38	101266	answers presented in a structured format, particularly in multiple-choice or quiz contexts.

Table 2: Top-scoring features decoded by SAE in the output of MLP when predicting $\mathcal{A}$.

We further investigated the semantic information encoded in the outputs of MLP and attention heads for verification. In attention heads, we found that in stage 2 and 3, the key heads encoded information related to both $\mathcal{A}_t$ and $\mathcal{A}_f$ (see the outputs of heads in Tab. 1). Meanwhile, numerous features related to decision-making (see Tab. 1) are identified in MLPs. These findings provide additional evidence supporting the critical role of the MLP and Attention layer in the answer generation process.

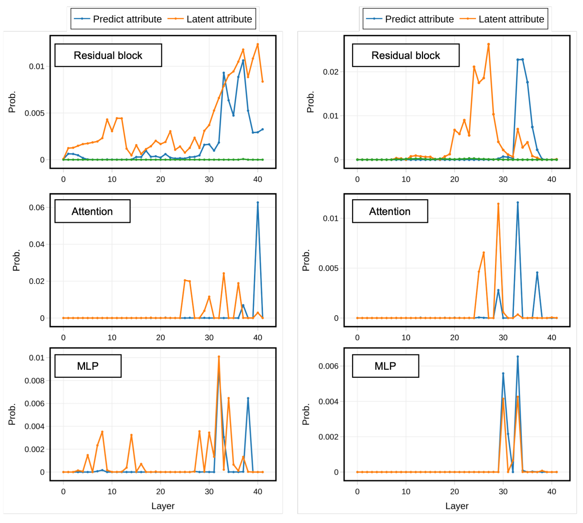
Finally, we applied activation patching to identify key heads and trace the information for generating $\mathcal{A}$. Tracing the information flow, the path began at the conclusion $\mathcal{S}$, progressed to the reasoning conjunctive adverb $\mathcal{R}$, and finally arrived at object $\mathcal{O}$. In the process, we discovered that $\mathcal{R}$ act as **anchors for the fusion and transport of conclusion-related information** in the reasoning process. For a detailed examination of the trace from $\mathcal{A}$ to $\mathcal{O}$, and an in-depth analysis of answer-related information at $\mathcal{R}$, refer to §A.5.

4.4 Object Retrieval and Rerank

The object information $\mathcal{O}$ decoded in the outputs of the residual block, attention layers and MLP layers are shown in Fig. 3b. We compare as many related objects, including the predicted object $\mathcal{O}_p$ and candidate object $\mathcal{O}_c$, as possible. For residual block, the object information emerges at around layer 26. However, $\mathcal{O}_p$ is not dominant in the first place, as the probabilities of $\mathcal{O}_p$ and $\mathcal{O}_c$ increase alternately. For attention heads, $\mathcal{O}_p$ and $\mathcal{O}_c$ interleave, with neither showing explicit dominance throughout the whole layers. On the contrary, MLP shows obvious preference on $\mathcal{O}_p$, where correct object information is prominent across almost all layers. Notably, at layer 37, $\mathcal{O}_p$ is clearly dominant, while $\mathcal{O}_c$ remains minimal. This sharp spike aligns with a key transition point in the curve of residual block. From these observations, it seems that 1) both $\mathcal{O}_p$ and $\mathcal{O}_c$ are integrated during the process of object token generation. 2) The attention heads initially retrieve the information for both $\mathcal{O}_p$ and $\mathcal{O}_c$, while MLPs subsequently rerank $\mathcal{O}_p$ to the top position.

To validate our finding, we look into the output of heads and MLP. As shown in Tab. 13, attention heads encode a rich set of attribute information relevant to the subject (e.g., "*British*", "*wizard*", "*book*", and etc). Meanwhile, in Tab. 11, the decoded features by MLP are strongly related to "*identity and character*". It is high correlated to $\mathcal{O}_p$, but none of them is related to $\mathcal{O}_c$. These results validates the function of retrieving and reranking for attention head and MLP, respectively.

Finally, we utilize activation patching to identify the heads with causal effect (see Fig. 15a) and find these heads focus on two critical token positions, $\mathcal{S}$ and end of question. Therefore, we trace back to $\mathcal{S}$ to investigate the origin of $\mathcal{O}$.



(a) (b)

Figure 4: (a) Logit attribution of $\mathcal{O}_p$ and $\mathcal{O}_c$ at $\mathcal{S}$ in StrategyQA. (b) Logit attribution of $\mathcal{O}_p$ and $\mathcal{O}_c$ at the end of question in StrategyQA.

4.5 Subject Augmentation and Broadcast

Generally, in commonsense reasoning datasets, the $\mathcal{S}$ always appears in both the question and the rationale. Through analysis, we observe that the $\mathcal{S}$ in the rationale can also be back-traced to the $\mathcal{S}$ in the question. Therefore, we treat the position of $\mathcal{S}$ in the question as a focal point for deeper analysis.

Figure 4a illustrates the information of $\mathcal{O}_p$ and $\mathcal{O}_c$ decoded in the outputs. Notably, we observe that: 1) Residual block contains obvious information regarding both $\mathcal{O}_p$ and $\mathcal{O}_c$ across various layers, with $\mathcal{O}_c$ being more prominent at the end. 2) another two curves show that both attention heads and MLPs have a large influence on $\mathcal{O}_p$ and $\mathcal{O}_c$. To further decode information, we identifies that MLPs in layers 7 and 32 encode abundant features related to $\mathcal{O}$ (see Tab. 12). Meanwhile, Probing also reveals that heads in layers 29 and 39 rank the $\mathcal{O}_c$ at top. In addition to diminishing the impact of the information from any previous token, we also examine the three corresponding curves at the position before $\mathcal{S}$ (for instance, "Question: **Can** Harry Potter"). The results (green line in Fig. 4a) reveal that the information regarding $\mathcal{O}$ is virtually zero. It indicates that the emergence of $\mathcal{O}_p$ and $\mathcal{O}_c$ is indeed contingent upon the appearance of $\mathcal{C}$ and is independent of any previous tokens. In conclusion, both the MLP and heads play essential roles in assisting the model to associate and extend from $\mathcal{S}$ to related $\mathcal{O}_p$ and $\mathcal{O}_c$. We refer to this stage, along with the contributions of the MLP and heads, as **subject augmentation**.

Regarding the question's end token position, Fig. 4b also presents the three corresponding curves. (1) In the residual, both $\mathcal{O}_p$ and $\mathcal{O}_c$ appear across multiple layers. On the contrary to concept token position, $\mathcal{O}_p$ has a greater presence than $\mathcal{O}_c$. (2) The curves for the MLP and heads also encapsulate information about both $\mathcal{O}_p$ and $\mathcal{O}_c$, and further enhance the importance of $\mathcal{O}_p$. It indicates that even at unrelated token positions, the $\mathcal{O}$ corresponding to the $\mathcal{S}$ (or the knowledge they encompass) can be broadcast. The original order of $\mathcal{O}$ may be broadcast based on the current context, ultimately influencing the generation of $\mathcal{O}_p$. We term this stage as **subject broadcasting**.

4.6 Verification and generalization of findings

To validate our interpretations, we conducted knockout analysis on critical model components (Fig. 22). Ablating the 10 key attention heads for answer generation resulted in a 60% decrease in correct answer prediction probability. Similarly, removing five critical heads and two MLP layers involved in object recall led to a 50% and 68% decreases in object prediction probability, respectively. These results provide strong causal evidence supporting our identified key mechanisms.

Further analysis of Gemma2-9B's reasoning process on CommonsenseQA and SocialIQA (§A.6) using 1,000 samples revealed similar patterns of object retrieval, reranking, and conclusion generation. However, subject augmentation was less prominent in these datasets, presumably due to the explicit provision of factual knowledge within the question context. These findings were also replicated using the Llama2-7B model across all three datasets (detailed results in §A.8).

5 Application of Interpreting Results

In this section, we analyze the failure of LLMs in commonsense reasoning (§5.1) and then introduce two applications of interpreting results to enhance the model's reasoning capability (§5.2 and §5.3).

5.1 Failure Case Analysis

We manually analyze all the failure cases of Gemma2-7B on StrategyQA test set. The results reveal four error types (Fig. 5): 1) Reference Errors: retrieving irrelevant or incorrect objects; 2) Logic Errors: insufficient knowledge to support conclusions; 3) Conclusion Errors: wrong answers despite correct reasoning; and 4) Concept Errors: misidentification of target concepts to analyze. Reference Errors dominate at 74% of all cases. Further probing reveals that these errors primarily stem from object reranking issues rather than knowledge gaps (see §A.7 for details), as correct objects typically appear within the model's top-5 predicted tokens. Based on this finding, we propose enhancing commonsense reasoning by using selective supervised fine-tuning and representation engineering.

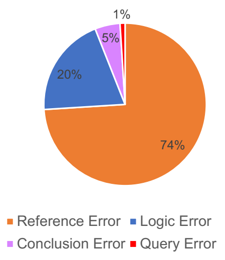
Models	ID Task	OOD Task		
	Strategy	CSQA	SIQA	Wino
Gemma2-9B	70.7	75.7	73.0	61.2
+ SFT (9B)	79.0	74.3	70.9	60.3
+ SSFT (0.3B)	80.3	76.2	74.0	65.2
Llama2-7B	62.5	68.3	67.9	55.5
+ SFT (7B)	77.3	54.8	59.0	52.7
+ SSFT (0.2B)	78.5	64.1	63.2	61.1

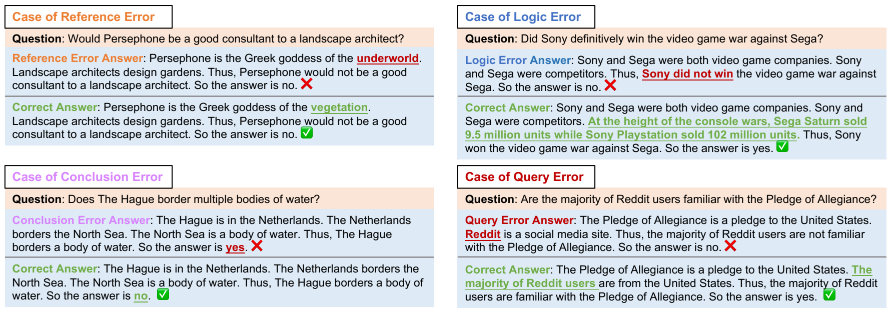
Table 3: Results on four commonsense reasoning tasks (i.e., StrategyQA, CSQA, Winogrande, and SocialIQA) before and after tuning on the StrategyQA dataset.

5.2 Selective Supervised Fine-tuning

Zhang et al. (2024); Chen et al. (2024) proposed a method to enhance model's capability through updating a small set of parameters. Specifically, given a sequence of attention heads and MLPs ordered by their significance, denoted as $(MLP.l_1), (Head.l_2.h_2), (Head.l_3.h_3), \ldots$, where l_i represents the layer index and h_i represents the head index of the i^{th} ranked head, only parameters of top K heads and top M MLPs are exclusively updated during fine-tuning. Following the same setting, we selectively fine-tune the top 32 Attention heads (for knowledge retrieval, i.e., red squares in Fig. 13a) and top 1 MLP layers (for knowledge reranking, i.e., peak in Fig. 14a). Considering the generalization, we introduce another commonsense reasoning test dataset, WinoGrande (Sakaguchi et al., 2021). See §A.9 for more detailed experiment setting.

Experiment Results. The comparative results between SSFT and SFT are presented in Table 3. For the experiments of Gemma2-9B on StrategyQA, both SSFT and SFT improved performance, achieving gains of $+8.3\%$ and $+9.6\%$, respectively. While SFT shows a comparable enhancement for the StrategyQA task, it adversely affected performance on OOD tasks, with an average decrease




Case of Reference Error

Question: Would Persephone be a good consultant to a landscape architect?

Reference Error Answer: Persephone is the Greek goddess of the **underworld**. Landscape architects design gardens. Thus, Persephone would not be a good consultant to a landscape architect. So the answer is no. ❌

Correct Answer: Persephone is the Greek goddess of the **vegetation**. Landscape architects design gardens. Thus, Persephone would not be a good consultant to a landscape architect. So the answer is no. ✅

Case of Logic Error

Question: Did Sony definitively win the video game war against Sega?

Logic Error Answer: Sony and Sega were both video game companies. Sony and Sega were competitors. Thus, **Sony did not win** the video game war against Sega. So the answer is no. ❌

Correct Answer: Sony and Sega were both video game companies. Sony and Sega were competitors. **At the height of the console wars, Sega Saturn sold 9.5 million units while Sony Playstation sold 102 million units.** Thus, Sony won the video game war against Sega. So the answer is yes. ✅

Case of Conclusion Error

Question: Does The Hague border multiple bodies of water?

Conclusion Error Answer: The Hague is in the Netherlands. The Netherlands borders the North Sea. The North Sea is a body of water. Thus, The Hague borders a body of water. So the answer is **yes**. ❌

Correct Answer: The Hague is in the Netherlands. The Netherlands borders the North Sea. The North Sea is a body of water. Thus, The Hague borders a body of water. So the answer is **no**. ✅

Case of Query Error

Question: Are the majority of Reddit users familiar with the Pledge of Allegiance?

Query Error Answer: The Pledge of Allegiance is a pledge to the United States. **Reddit** is a social media site. Thus, the majority of Reddit users are not familiar with the Pledge of Allegiance. So the answer is no. ❌

Correct Answer: The Pledge of Allegiance is a pledge to the United States. **The majority of Reddit users** are from the United States. Thus, the majority of Reddit users are familiar with the Pledge of Allegiance. So the answer is yes. ✅

Figure 5: The distribution of the four types of errors encountered by Gemma2-7B on StrategyQA. 1) Reference Error: The model retrieves irrelevant or wrong attributes. 2) Logic Error: incomplete reasoning steps. 3) Conclusion Error: reaches an incorrect answer but based on correct rationale. 4) Concept Error: incorrectly identifies the target concept for analysis. We first sample 50 cases to manually summarize the error types and then use GPT-4 to automatically classify the remainings. See Fig. 24 for the prompt.

of -1.5%. In contrast, SSFT continued to bolster the model's reasoning ability across all OOD commonsense reasoning tasks, improving the performance by an average of $+2.6\%$. These findings suggest that selectively fine-tuning a small fraction of key components for commonsense reasoning can boost performance on ID tasks while maintaining generalizability, highlighting the effectiveness of our previous exploration. A similar trend was observed in the Llama2-7B results. Through mechanism analysis of the model before and after SSFT, we further validate that SSFT enhances the model's knowledge retrieval and reranking capabilities. (See Fig. 23). Additionally, we further validate the effectiveness of SSFT through training on two other datasets (Tab. 14 and 15).

5.3 Representation Engineering

Representation engineering, which adjusts the model's internal hidden states to influence its behavior, has proven to be an effective method for modulating model performance (Zou et al., 2023). Following the approach outlined in Xiao et al. (2024); Templeton et al. (2024), we correct the model's erroneous behavior using:

$$\tilde{\mathbf{h}}_l = \mathbf{h}_l + k\mathbf{x}_t, \tag{1}$$

where $\mathbf{h}_l$ represents the original output of residual block at layer l, $\mathbf{x}_t$ is the feature direction corresponding to the correct knowledge identified using SAE, k is the steering magnitude which we set 5. See §A.3 for more details.

Experiment Results. We utilize representation engineering to correct the model's (Gemma2-9B) failure in recalling correct object. For example, in question "*Would Persephone be a suitable consultant to a landscape architect?*". The model initially defaults to identifying "*Persephone as the Greek goddess of the underworld*", leading to an incorrect assessment. The correct reference is "*Persephone is the Greek goddess of spring*". By introducing feature directions related to deities or nature into the residual block at layer 37 (object retrieval), we strengthened the model's tendency to associate "*Persephone*" with "*spring*". This tendency can largely contribute to the correct answer, and rectify the model's response. As a result, 93% failure cases can be rectified, illustrating the rationality of the identified interpreting results.

6 Conclusion

In conclusion, our research sheds light on the intricate dynamics of commonsense reasoning within LLMs, revealing a structured process that parallels human cognitive reasoning. By meticulously analyzing the hidden states across various transformer layers and token positions, we identified a multi-faceted mechanism that integrates knowledge augmentation, retrieval, and answer generation—essentially resembling a retrieval-augmented generation framework. Our findings underscore the pivotal roles played by both attention heads and MLPs in the manifestation of factual knowledge, highlighting a dual approach to knowledge processing. Furthermore, our experiments demonstrated that while LLMs often possess relevant factual knowledge, they frequently struggle to retrieve the correct information during inference. Through selective fine-tuning of key components, we achieved notable enhancements in reasoning performance across diverse contexts, indicating that targeted adjustments can effectively optimize the reasoning capabilities of LLMs.

7 Limitations

While the methods and findings presented in this study provide valuable insights into the internal mechanisms of large language models (LLMs), there are several limitations:

Scope of Evaluation: The experiments primarily focus on commonsense reasoning tasks, and the results may not fully generalize to other types of reasoning or NLP tasks. Future work could extend the methodology to explore how these internal mechanisms behave across a wider range of tasks.

Model Dependency: Our analysis is based on the specific architectures and pretrained models used in this study. While the interpretability tools such as logit attribution, activation patching, and sparse autoencoders provide useful insights, the observed behaviors may vary with different models or architectures. The findings may be influenced by the particular training data and the design choices of the models.

Complexity of Causal Back-Tracing: The causal back-tracing method, while effective in identifying key tokens and correlations, remains computationally expensive and may require further optimization for large-scale models. Additionally, accurately interpreting causal relationships in highly complex networks like transformers is a non-trivial task and may be subject to noise or inaccuracies, especially in deep layers.

Interpretability Limitations: While we provide insights into model behavior by examining attention heads, MLPs, and other components, the level of interpretability remains limited. Fully understanding the underlying reasons for model decisions, especially in tasks involving nuanced or implicit commonsense knowledge, may still be out of reach with current methods.

Human Evaluation: While the interpretability tools offer a mechanistic view of the model, the final conclusions and explanations are still subject to human interpretation. There is a risk of oversimplification or misinterpretation when mapping complex internal mechanisms to human-understandable explanations, particularly in highly abstract or nonlinear decision-making processes.

References

Nora Belrose, Zach Furman, Logan Smith, and et al. 2023. Eliciting latent predictions from transformers with the tuned lens. *CoRR*, abs/2303.08112.

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2023. The reversal curse: Llms trained on “a is b” fail to learn “b is a”. In *The Twelfth International Conference on Learning Representations*.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Wei Chen, Zhen Huang, Liang Xie, Binbin Lin, Houqiang Li, Le Lu, Xinmei Tian, Deng Cai, Yonggang Zhang, Wenxiao Wan, et al. 2024. From yes-men to truth-tellers: Addressing sycophancy in large language models with pinpoint tuning. *arXiv preprint arXiv:2409.01658*.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, pages 16318–16352. Curran Associates, Inc.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Guy Dar, Mor Geva, Ankit Gupta, and et al. 2023. Analyzing transformers in embedding space. In *ACL*, pages 16124–16170. Association for Computational Linguistics.

Subhabrata Dutta, Joykirat Singh, Soumen Chakrabarti, and Tanmoy Chakraborty. 2024. How to think step-by-step: A mechanistic understanding of chain-of-thought reasoning. *ArXiv*, abs/2402.18312.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

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, Singapore. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and et al. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *EMNLP*, pages 30–45. Association for Computational Linguistics.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021a. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Mor Geva, Roei Schuster, Jonathan Berant, and et al. 2021b. Transformer feed-forward layers are key-value memories. In *EMNLP*, pages 5484–5495. Association for Computational Linguistics.

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

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *arXiv preprint arXiv:2305.00586*.

Yifan Hou, Jiaoda Li, Yu Fei, Alessandro Stolfo, Wangchunshu Zhou, Guangtao Zeng, Antoine Bosselut, and Mrinmaya Sachan. 2023. Towards a mechanistic interpretation of multi-step reasoning capabilities of language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4919, Singapore. Association for Computational Linguistics.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Stefanie Krause and Frieder Stolzenburg. 2023. Commonsense reasoning and explainable artificial intelligence using large language models. In *European Conference on Artificial Intelligence*, pages 302–319. Springer.

Tom Lieberum, Matthew Rahtz, J'anos Kram'ar, G. Irving, Rohin Shah, and Vladimir Mikulik. 2023. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *ArXiv*, abs/2307.09458.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. 2024. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv preprint arXiv:2408.05147*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Neural Information Processing Systems*.

nostalgebraist. 2021. Interpreting gpt: The logit lens.

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

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

Francesco Ortu, Zhijing Jin, Diego Doimo, Mrinmaya Sachan, Alberto Cazzaniga, and Bernhard Schölkopf. 2024. Competition of mechanisms: Tracing how language models handle facts and counterfactuals. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8420–8436, Bangkok, Thailand. Association for Computational Linguistics.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. 2024. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *NeurIPS*.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, and et al. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *ICLR*.

Yifei Wang, Yuheng Chen, Wanting Wen, Yu Sheng, Linjing Li, and Daniel Dajun Zeng. 2024. Unveiling factual recall behaviors of large language models through knowledge neurons. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7388–7402, Miami, Florida, USA. Association for Computational Linguistics.

Yuxin Xiao, Chaoqun Wan, Yonggang Zhang, Wenxiao Wang, Binbin Lin, Xiaofei He, Xu Shen, and Jieping Ye. 2024. Enhancing multiple dimensions of trustworthiness in llms via sparse activation control. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*.

Lei Yu, Meng Cao, Jackie CK Cheung, and Yue Dong. 2024. Mechanistic understanding and mitigation of language model non-factual hallucinations. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7943–7956, Miami, Florida, USA. Association for Computational Linguistics.

Qinan Yu, Jack Merullo, and Ellie Pavlick. 2023. Characterizing mechanisms for factual recall in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9924–9959.

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

Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu-ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. 2024. Interpreting and improving large language models in arithmetic calculation. *arXiv preprint arXiv:2409.01659*.

Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. 2020. Evaluating commonsense in pre-trained language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9733–9740.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

A Appendix

A.1 Discussion about interpreting techniques

We briefly introduce existing interpreting tools here: (1) **Logit lens** projects intermediate activations to vocabulary space, enabling researchers to understand how predictions evolve across layers. (2) **Probing** techniques employ shallow classifiers to detect specific information encoded in model activations, though they only reveal correlational rather than causal relationships. (3) **Sparse Autoencoders (SAEs)** help discover independent features from superposed representations by mapping activations to a higher-dimensional sparse space. (4) **Visualization tools** aid in hypothesis generation and qualitative analysis, particularly for understanding attention patterns and neuron activations. (5) **Automated feature explanation** leverage LLMs themselves to automatically generate and validate feature labels, reducing the need for manual annotation while providing quantitative measures of explanation quality. (6) **Knockout**: To identify crucial model components, knockout/ablation methods systematically remove or modify specific parts while observing behavioral changes. (7) **Activation patching**, helps locate important components and connections within model circuits by comparing clean and corrupted runs. We recommend reading paper (Rai et al., 2024) for a comprehensive overview of mechanistic interpretability in LLMs. (8) **Information flow** analysis, based on Taylor expansion, provides insights into how information propagates between tokens through attention mechanisms, though its correlational nature limits causal interpretability.

This paper focuses on understanding the internal mechanisms of how the model performs commonsense reasoning. Specifically, we studied how the model generates rationales before the final answers. We break down this problem into three sub-problems: 1) Locating the key token positions in the rationale; 2) For each key token, identifying the key components during its generation; 3) Analyzing the behavior of the key components. To address each of these problems, we select distinct tools from the candidate tools above. Tab 4 provides justifications for our choices.

Simplifying assumptions of the tools While our interpretability analysis yields valuable insights, it is important to acknowledge the inherent limitations of the tools employed. These tools necessarily operate under certain simplifying assumptions to make the complex reasoning processes of LLMs more tractable for analysis. In Tab. 5, we systematically summarize the assumptions and reliability of the tools used in this paper. These simplifying assumptions, while potentially not capturing all nuances of LLM reasoning, provide a structured framework for investigating specific aspects of rationale generation within LLMs. Based on this, the analysis results are reliable, especially when combined with the verification method outlined in Wang et al. (2023); Lieberum et al. (2023). The only problem may lie in the lack of comprehensiveness in interpreting the whole reasoning procedure.

In summary, based on the assumptions of circuits, similar embedding spaces, and sparse feature representation, this paper utilized these interpretability tools to uncover the interpreting results in the paper. These results are reliable. Indeed, some potential complex mechanisms should remain uncovered, necessitating the design of more suitable tools and methods. Addressing this will be a focus of our future work.

A.2 Improvements to the interpreting tools

Due to the complex challenge (multi-token rationale and dense token connection) of interpreting the reasoning mechanisms, directly applying these tools to interpret the commonsense reasoning process is infeasible. Therefore we made many improvements to existing interpreting tools. In Tab 6, we outline the key problems we encountered with these tools and the modest improvements we made to address them. Specifically, our detailed improvement to activation patching and SAE are depicted in §A.2.1 and §A.2.2 respectively.

A.2.1 Activation patching details

Counterfactual data generation We use GPT-4 to assist in automatically generating the counterfactual data required for activation patching, with the prompt shown in Figure 6 and an example in Table 8. Additionally, we implement a post-processing step: if the predicted token for the counterfactual data matches the prediction for the data under investigation (which would fail to perturb the model's behavior), GPT-4 is prompted to regenerate the counterfactual data.

We conduct experiments to compare the performance of "GPT-4" and "human". we engaged ten master's students specializing in Natural Language Processing as volunteers. Five students were manu-

Table 4: Analysis of interpretability tools selection for different investigation problems

Problems	Candidate Tools	Selected Tools	Justification for not Using Other Tools	Justification for Using the Selected Tools
Locate the key token positions in the rationale of commonsense reasoning	Activation patching, information flow	Activation patching	**Not information flow:** i) **Lack of causality**: Information Flow applies Taylor expansion to assess the importance of connections between tokens, which lacks causal interpretation. ii) **Poor interpretability**: our preliminary experiments indicate that the key heads identified using the Information Flow do not offer good interpretability. Furthermore, knocking out these heads has no significant impact on the model's predictions.	**Use activation patching:** i) **Causality**: activation patching pinpoints the key attention heads using causal analysis by directly modifying activations and observing output changes. ii) **Robust interpretability**: Many works have successfully interpreted the model's specific behavior using activation patching, such as mathematical calculation (Zhang et al., 2024), multi-choice question answering (Lieberum et al., 2023), and sycophancy (Chen et al., 2024)
Identify the key components for commonsense reasoning	Probing, logit attribution, activation patching	Logit attribution	**Not probing:** i) **Hard to Probe Diverse Knowledge**: probing involves an external classifier, and the outputs of knowledge recall tasks are difficult to categorize due to the diversity of knowledge. ii) **Correlation not causality**: Probing primarily analyzes correlation rather than causal relationships. **Not activation patching:** high cost (5 minutes to analyze a sample on a 7B model using a single A100 GPU.)	**Use logit attribution: high efficiency**: logit attribution has proven to be an efficient tool to identify and analyze the key components within LLMs in many works (Wang et al., 2023; Lieberum et al., 2023; Zhang et al., 2024; Yu et al., 2023).
Analyze the behavior of the key components in commonsense reasoning	Logit attribution, automated feature explanation, SAE, knockout	Logit attribution, SAE, knockout	**Not neuron activation pattern explanation:** i) **polysemanticity**: neurons may not be easily explainable because they often encapsulate multiple features and can be polysemantic. ii) **Interpretability illusion**: neuron analysis often focuses on top-activating dataset examples, which may create an illusion of interpretability by neglecting the neuron's varied behaviors across different activation levels	**Use the combination of three tools:** Logit attribution is an efficient analyzing tool but may cause false interpretation results. Therefore we introduce SAE and knockout, which are used to verify the correctness of the key components. We use SAE to decode semantic information in the module's output to verify the semantic consistency. We use knockout to verify that the identified module can actually affect the generated result. If the results are not consistent, the identified module from the Logit attribution will be discarded. Combining these two tools can further improve the reliability of the identified modules and the analysis of their behavior by removing noisy components.

ally executing all procedures, including generating X_c, analyzing key component behaviors, and developing data templates. The remaining students then compared their annotations with those generated by GPT-4 to judge which more accurately represented the component behavior. Overall, the results (Table 7) demonstrate that GPT-4 is highly accepted by human evaluators, with the combination of "GPT wins" and "Ties" exceeding 80%, underscoring its robust reliability. These indicate that *GPT-4's outputs are almost consistent with those generated by humans.*

Activation patching metric We design a special metric to evaluate the causal effect: predicted token's probability divided by the sum of probabilities of the top k tokens. For example, when LLM is generating the next token for input "*Harry Potter is a*", the top k predicting tokens include "*fictional, wizard, British, ...*". Then the metric is:

$$\frac{\text{prob}(fictional)}{\text{prob}(fictional) + \text{prob}(wizard) + \text{prob}(...)}. \qquad (2)$$

While directly using the logit change of "*fictional*" leads to the identification of irrelevant modules that increase all the logits of "*fictional, wizard, British, ...*".

Trace the information source within the middle layers While standard activation patching reveals causal effects by modifying network activations and observing changes in the model's output, we identified a significant limitation during our experiments. We discovered that answer-relevant information (e.g., "*no*") can emerge strongly in intermediate layers at reasoning adverb positions $\mathcal{R}$ (e.g., "*Thus*"), yet become almost imperceptible in the final layer output (as shown in Fig. 9b). This phenomenon renders traditional activation patching ineffective for tracing information sources, as it relies solely on observing the model's final output. To address this limitation, we enhanced the activation patching methodology: when investigating information sources at layer k, we iteratively corrupted the output of each attention head from layers 0 through k using activations from counterfactual data. This approach allows us to identify key attention heads by measuring their negative influence on the probability of tokens of interest in the layer

Tools	Simplified Assumption	Reliability of the Assumption
Activation Patching	**Analogous circuits assumption**: Model can be viewed as a computational graph M where nodes are terms in its forward pass (neurons, attention heads, embeddings, etc.) and edges are the interactions between those terms (residual connections, attention, projections, etc.), a circuit C is a subgraph of M responsible for some behavior. *Analogous circuits appear across different models and tasks, which suggests that neural networks tend to converge on similar mechanisms for solving similar problems.*	**1. Analogous features across different vision models**: Certain low-level features, such as Gabor filters and curve detectors, reliably appear in early layers across multiple vision models (e.g., AlexNet, InceptionV1, VGG19, ResNet) trained on different datasets like ImageNet and Places365 (Olah et al., 2020). This suggests that neural networks converge on similar basic structures for solving visual tasks. **2. Task-specific circuits identified in LLMs**: Many works have identified the circuit for different tasks (e.g. Indirect object identification (Wang et al., 2023), multiple-choice question answering (Lieberum et al., 2023) and greater-than computation (Hanna et al., 2023)), which indicates analogous circuits performing certain tasks appear across different models and tasks. **3. Function-specific components identified in LLMs**: Research has identified elements like induction heads, which are specialized mechanisms within transformer models that perform specific functions such as copying patterns from prior sequences (Olsson et al., 2022) and neurons correlate with specific grammatical features are discovered in (Geva et al., 2022).
Logit Attribution	**Projection assumption**: the outputs of each module in the model can be projected into the vocabulary space via the unembedding matrix to encode the semantic information within hidden states.	A bunch of work has proved the feasibility of understanding the hidden state and weights within transformer-based LLMs through projection to the vocabulary space (Wang et al., 2023; Lieberum et al., 2023; Geva et al., 2022).
Sparse Autoencoder (SAE)	**Sparse representation assumption**: The hidden state within models can be efficiently represented by a small number of interpretable salient monosemantic features	The SAE is based on sparse dictionary learning. Many works (Lieberum et al., 2024; Gao et al., 2024) have trained SAE to decompose and explain the hidden state within LLMs and prove SAE to be an efficient interpreting tool.

Table 5: Comparison of Different Tools and Their Assumptions

Interpreting stage	**Sub steps**	**Interpreting Tools**	**Specific Problems**	**Our Modest Improvement**	**Detailed illustrations**
Locate the key token positions	Generate x_r, x_c pairs	Activation patching	High labor cost to scale the results	Automate this process using GPT-4 to get scalable results	When explaining how model generates "fictional (character)" given "Harry Potter is a", we prompt GPT-4 to generate counterfactual data
Locate the key token positions	Compute logit change of key tokens	Activation patching	Noisy metric	Design a special metric to evaluate the causal effect	Using the same case above, the top k predicting tokens include "fictional, wizard, British, ..."
Locate the key token positions	Trace the information source within the middle layers	Activation patching	Unable to trace the source of middle layer information	Design a metric to examine perturbations effect	We identify rich answer-related information responsible for generating conclusion token
Identify the key components	Choose the target tokens for observation	Logit attribution	False identification of the key modules	Introduce probabilities of candidate tokens as comparison	Using the same case above, we identify the modules where logit attribution of predicted token is high
Analyze the behavior of the key components	Verify the reliability of key components	Logit attribution	Projection assumption failure	Use SAE to decode semantic information	Use the same case above, we use logit attribution to evaluate each modules' contribution
Analyze the behavior of the key components	SAE Training	SAE	High computation cost	Logit attribution is first used to identify key layers	For the LLaMA model, training SAE for MLPs of every layer requires substantial resources
Analyze the behavior of the key components	Evaluating the relevance between SAE features and reasoning task	SAE	High labor cost to select the feature	Use GPT-4 to automatically analyze correlation	When using SAE decomposing the output of key MLP, we use GPT-4 to select object-related features

Table 6: Interpreting tools and improvements

k residual block output, thereby enabling effective tracing of information flow through intermediate layers.

A.2.2 SAE details

Mechanism of SAE: Based on dictionary learning, SAE translates the hidden states of LLMs into several interpretable pieces, or termed *features*. These features are activated on sparse token sequences with specific patterns, and most can be interpreted by GPT-4 (Lieberum et al., 2024) into concrete semantic descriptions.

Strategic Training of SAE: While Google has publicly released SAE checkpoints for all layers of Gemma-7B, such resources remain unavailable for the LLaMA-2 model family. Consequently, we undertook the task of training our own SAE models. Given the substantial computational requirements, we strategically limited our training to specific MLP layers (layers 16 and 20) that are crucial for object reranking.

The training code for our Sparse Autoencoder (SAE) builds upon the open-source implementation provided by OpenAI (`https://github.com/openai/sparse_autoencoder`), which employs a Top-K activation function to maintain sparse latent representations. Our training configuration utilized 2 billion tokens from the Pile dataset, structured in 64-token sequences. The SAE architecture incorporates 512,000 latent variables with a Top-K activation parameter of 32. We implemented a distributed training setup with tensor parallelism of 2 and data parallelism of 8, processing batches of 131,072 tokens. The learning rate was set to 1.24e-4, determined through scaling laws derived from the GPT-2 architecture. The entire training process consisted of a single epoch. The computational requirements were still substantial: generating MLP outputs for the LLaMA2-7B model across 2 billion tokens consumed approximately 5 hours on 64 A100 GPUs, while the subsequent SAE training phase required an additional six hours utilizing 16 A100 GPUs.

SAE feature relevance evaluation: We primarily use SAE to investigate the information contained in the MLP and residual block outputs at the concept token position. Specifically, we selected the top 64 activated features (Top-64) based on SAE activations. Since these features include a substantial number of general-purpose activations (e.g., those representing syntax, specific words, etc.), we employed GPT to automatically analyze whether these activated features are related to the concept. The prompt used for this analysis is provided in Fig. 7.

Table 7: Comparison of differences between GPT-4 and human annotations for counterfactual data generation.

GPT-4 Wins	Human Wins	Ties
8%	12%	80%

A.3 Details of representation engineering

Representation engineering serves as a downstream application of our interpretability results, primarily to verify their reliability. This technique enables behavioral adjustments of the model through targeted modifications of internal representations. For instance, Templeton et al. (2024) demonstrated how introducing a security-related feature into the model's middle layer residual stream can guide it toward generating safer content. Our experimental protocol consists of four key steps: (1) **Object Identification**: Leveraging the ground truth rationales provided in the StrategyQA dataset, we employ GPT-4 to detect cases of incorrect object retrieval by the model. In instances where errors are identified, we determine the correct objects that should have been retrieved. (2) **Layer Selection**: We target the MLP layer that exhibits significant contribution to the retrieval of predicted objects ($\mathcal{O}_p$). Specifically, we focus on layer 36, which corresponds to the peak responsibility for object reranking, as shown in Fig. 3b. (3) **SAE Feature Selection**: We decompose the hidden state at the factual knowledge prediction position using SAE. To identify steering-relevant features, we employ GPT-4 for automated assessment of feature relevance to the correct factual knowledge (detailed methodology in A.2.2), selecting the most pertinent feature for modification. (4) **Magnitude Calibration**: Through grid search across a range of 1-10, we empirically determine the optimal perturbation magnitude, settling on $k = 5$ for our interventions.

A.4 Stability analysis of interpretability results

To validate the robustness of our findings, we conducted comprehensive scaling experiments. Take logit attribution of the answer generation stage in StrategyQA as an example, we examined different sample sizes (50, 100, and 1000 instances) and included an additional random resampling of 1000

Prompt Template for Counterfactual Data Generation

~~~
<Inputs><topic> The particular topic being studied</topic>
<input_sentence> The original sentence provided for analysis</input_sentence>
<predicted_content> The specific words reflecting model behavior</predicted_content>
<first_word_predicted> The first word initially predicted by the model</first_word_predicted></Inputs>

<Instructions Structure>
1. Instruct the assistant to begin by analyzing the original input sentence and why it leads to the specific predicted word.
2. Guide the assistant to think about changes that could alter the model's prediction.
3. Instruct the assistant to provide the reason for the model's original prediction.
4. Request the assistant to modify the original sentence so that the model’s prediction changes.
5. Instruct the assistant to explain the modification's rationale, focusing on why the modified sentence now influences a different predicted outcome.
6. Ensure the output is formatted in the specified JSON structure.
</Instructions Structure>

<Instructions>
Your task is to analyze and modify a sentence to influence the predictive behavior of a language model. You will be given a topic, an input sentence,
the specific words predicted by the model, and the model’s first predicted word.

Here is the topic and input sentence to modify: <topic>{$TOPIC}</topic> <input_sentence>{$INPUT_SENTENCE}</input_sentence>

Here are the words generated by model given the input sentence: <predicted_content>{$PREDICTED_CONTENT}</predicted_content>

Here is the first predicted word:
<first_word_predicted>{$FIRST_WORD_PREDICTED}</first_word_predicted>

Follow these steps carefully to complete the task:

1. **Analyze the Original Prediction**: Start by understanding the **input sentence** and why it leads the model to predict the
   **first_word_predicted** as the output under the specific **topic**. Consider the context, tone, or structure of the sentence that prompts this
    specific word choice by the model.
2. **Plan the Modification**: Think about how you could change the **input_sentence** minimally (by changing only 3-4 words) to alter the
   model's behavior so that it no longer predicts the original word or instead predicts a word with an opposite meaning. It's acceptable to change some
    of the sentence's meaning if it helps influence the output.
3. **Provide Analysis and Modification**:
   - Write the **reason for the original prediction** based on your analysis in Step 1.
   - Rewrite the **input_sentence** in a modified form that will change or flip the model's predicted word.
   - Explain your **reason for the modification**, focusing on how the changes you made will influence the model to predict a different word.

4. **Output the Final Result**: Format your response in JSON, as shown below:

```json
{
    "Reason for original prediction": "Explain why the original input caused the model to predict the initial word.",
    "Modified input": "Write the modified sentence here.",
    "Reason for modification": "Explain why the modified input will lead to a different prediction from the model.”
}
```
Make sure each section is clear and precise. End your response with this JSON structure.
</Instructions>
~~~

Figure 6: Prompt for using GPT-4 to generate counterfactual data in activation patching.

instances. As illustrated in Fig. 8, the progression patterns of correct and false answers across different model components (Residual block, Attention, and MLP) remain remarkably consistent regardless of sample size. Specifically, the characteristic peaks in the attention mechanism around layer 34 and the distinctive MLP activation patterns in the later layers (30-40) are preserved across all sample sizes. The resampled 1000-instance experiment further corroborates these findings, exhibiting nearly identical behavioral patterns to the original 1000-instance sample. This consistency across different sample sizes and random resampling strongly suggests that our choice of 1000 instances provides a reliable representation of the model’s behavior patterns. Moreover, the clear separation between correct and false answer trajectories remains stable across all experimental conditions, indicating that our interpretability findings are not artifacts of sample size but rather reflect genuine computational patterns within the model. While we demonstrate this stability using the answer generation phase, similar consistency is observed in other reasoning stages.

A.5 Details of tracing from answer $\mathcal{A}$ to object $\mathcal{O}$

We found that the attention heads responsible for generating $\mathcal{A}$ primarily focus on the conclusion token $\mathcal{C}$, as demonstrated by the pattern of head 25.08 in Tab. 9. Therefore, we traced back to the $\mathcal{C}$, Fig. 9a shows the probabilities of $\mathcal{A}_t$ and $\mathcal{A}_f$ in the residual block, attention, and MLP outputs at the conclusion token position. It is evident that the model distinguishes the correct answer $\mathcal{A}_t$ in the deep layers, with both the attention and MLP outputs containing substantial information related to $\mathcal{A}_t$.

Next, we identified the heads for generating $\mathcal{C}$ using activation patching and discovered that the key

Table 8: Example of probing data X_r and counterfactual data X_c generated by GPT-4. Counterfactual data change the model (Gemma2-9B) prediction behavior by applying minimal change to the probing data.

Data	Model Input	Model Predict
X_r	Question: Kendall opened their mouth to speak and what came out shocked everyone. How would you describe Kendall? (1) a very quiet person (2) a very passive person (3) a very aggressive and talkative person Answer: Kendall opened their mouth to speak and what came out shocked everyone. Thus, Kendall is a very __	aggressive
X_c	Question: Kendall opened their mouth to speak and what came out **was softer than expected**. How would you describe Kendall? (1) a very quiet person (2) a very passive person (3) a very aggressive and talkative person Answer: Kendall opened their mouth to speak and what came out **was softer than expected**. Thus, Kendall is a very __	quiet

attention heads primarily focus on the reasoning conjunctive adverb $\mathcal{R}$ (i.e., "*Thus*" in head 31.03 pattern in Tab. 9). We also observed that the attention head outputs contain information related to the correct answer $\mathcal{A}_t$, such as "*yes*," "*indeed,*" and "*true.*" Based on these findings, we conducted further probing at $\mathcal{R}$ to trace the origin of $\mathcal{A}_t$.

Through decoding information of $\mathcal{A}_t$ and $\mathcal{A}_f$ at $\mathcal{R}$ (Fig. 9b), we find that deep layers ($30-34$) already encode rich information related to the correct answer $\mathcal{A}_t$. To trace the origin of the answer-related information, we employed a modified activation patching to identify the key Attention heads. Specifically, we iteratively corrupted the output of each attention head from layer $0-30$ using the activation in counterfactual data, then identified the key attention heads that have a significant negative influence on the probability of $\mathcal{A}_t$ in residual block (layer 30) output. Three key Attention heads (25.7, 25.8 and 25.9) are identified that primarily focus on the position of attribute $\mathcal{A}$ (e.g., "company"). From the observation above, we conclude a key finding: *reasoning conjunctive adverbs serve as an anchor for gathering and transferring conclusion-related information in reasoning process*. Therefore, our investigation continuously traces back to the position of object $\mathcal{O}$ prediction.

A.6 Results on CommonsenseQA and SocialIQA

We further apply our interpreting method to CommonsenseQA (Talmor et al., 2018) and SocialIQA (Sap et al., 2019) and find the model's reasoning process within these two datasets consists of **attribute retrieval**, **attribute rerank**, and **answer generation** as shown in Fig. 10. Similarly, we start by decoding the probability of $\mathcal{A}_t$ and $\mathcal{A}_f$ at the position of predicting answer $\mathcal{A}_t$. The decoding curve of CommonsenseQA is in Fig. 12b and SocialIQA result is in Fig. 14b. It is observed that the information trend in residual block, Attention, and MLP is similar across the two datasets. Specifically, the probability of $\mathcal{A}_t$ increases significantly at layer 30, while Attention output encodes $\mathcal{A}_t$ related information before layer 30 and $\mathcal{A}_t$ relate information emerges in MLP at layer around 32. Therefore, we conclude the answer generation process as follows: attention is responsible for copying and generating $\mathcal{A}_t$ related information and MLP is responsible for augmenting this information. Through backtracing, we identified the key heads for generating the correct answer (see key head distribution in Fig. 11b and 13b). As shown in Tab. 10, we find the head output encodes rich information related to the correct answer and mainly attends to the object in rationale and choices in question. Therefore, we first trace back to the position of $\mathcal{C}$.

Since both datasets are in the form of multiple-choice questions, the answer (object) is already provided as one of the options. Therefore, we treat the correct answer as the predicted object $\mathcal{O}_p$ and the other options as candidate objects $\mathcal{O}_c$. The logit attribution curves for $\mathcal{A}_t$ and $\mathcal{A}_f$ are shown in Fig. 12a and 14a for CommonsenseQA and SocialIQA respectively. As shown in the figure, the attention output contains both $\mathcal{O}_p$ and $\mathcal{O}_c$, while the MLP output only contains the $\mathcal{O}_p$. This finding aligns with our previous discovery on StrategyQA regarding the **object retrieval and reranking** mechanism: attention heads first aggregate all relevant objects, and then the MLP ranks these objects based on their relevance, selecting the $\mathcal{O}_p$ for the final output. These results further validate the generalizability of our approach and findings.

Finally, we used activation patching to identify the key attention heads responsible for generating

Prompt Template for SAE feature filtering

```
Istruction: evaluate the relevance of a feature that activates
on certain texts to the concept of `{concept}'.

You will be provided with a possible explanation of the feature,
a set of texts where the feature has been activated, along with
the most activated word(s) in each text.

$<$Example$>$
Concept: `Environmental Protection'

Possible explanation of the feature: feature identifies
texts related to protecting the natural environment.

Activated texts and most activated word:
- must take action to reduce carbon emissions and combat
  climate change. | most activated word: emissions
- Deforestation is a major threat to biodiversity and
  contributes to global warming | most activated word: deforestation
$<$/Example$>$

$<$Expected Output$>$
The feature is highly relevant to the concept of environmental
protection as it identifies texts discussing environmental
issues and solutions.
Relevance Score: 10
$<$/Expected Output$>$

Based on the given explanation of the feature and the activated
texts, please rate the relevance of the feature to the concept
of `{concept}' on a scale of 0 to 10.
- 0: Not at all relevant, the feature is not related to the concept.
- 5: Neutral, the feature is not directly related to the concept
    but share some common traits with the concept, e.g. apple
    and banana are both fruits.
- 10: Very relevant, the feature is directly related to the concept.

Please conclude your response in the following format:
`Relevance Score: [SCORE]', where [SCORE] is an integer between
0 and 10.

Here is the concept: {concept} and the explanation of the feature:

Concept: {concept}

Possible explanation of the feature: {explanation}

Activated texts and most activated word: {texts}
</Instructions>
```

Figure 7: Prompt for using GPT-4 to evaluate the relevance between feature and task.

$\mathcal{O}_p$. The distribution of important heads is shown in Fig. 11a and 13a. We found that the key heads primarily focus on the options in the question (see head pattern of 34.14 in Tab. 10), which serve as the source for all objects. With this, the complete reasoning process is concluded.

A.7 Results of logit attribution on failure cases

Our interpretability framework was applied to analyze cases of Reference Errors, where the model retrieves irrelevant or incorrect objects. As shown in Fig. 17, several noteworthy patterns emerge: First, although the model ultimately outputs incorrect objects, information related to the correct answer remains present in the final layers. This observation is also supported by our examination of the model's top-5 token predictions, which consistently include the correct object among the candidates, albeit not as the primary prediction. Second, we observe a distinct pattern in the information flow: Layer 35's attention mechanisms exhibit strong signals related to the correct object (a pattern absent in earlier layers), while Layer 32's MLP shows pronounced activation patterns associated with incorrect object. This pattern diverges from our previously observed object recall process in successful cases, where attention heads first gather relevant object information (object retrieval), and MLPs subsequently rerank the most pertinent object for output (object rerank).

Based on these observations, we hypothesize that Reference Errors stem from two primary mechanisms: (1) Insufficient information gathering by attention mechanisms in the middle-to-late layers (25-35), leading to incomplete object collection. (2) Incorrect reranking by MLP layers, which erroneously prioritize irrelevant objects over pertinent ones. This mechanistic understanding informed our fine-tuning strategy, where we simultaneously target both attention heads and MLP layers for optimization.

A.8 Experiment results on Llama2-7B

On Llama2-7B, we apply the same method to interpret the reasoning process in StrategyQA (see Fig. 16, CommonsenseQA (Fig. 20) and SocialIQA (Fig. 14. Three phases of reasoning, i.e. **subject augmentation and broadcast**, **object retrieval and rerank**, **conclusion fusion and generation** are observed on StrategyQA. Similarly, **object retrieval and rerank** and **conclusion generation** are observed on CommonsenseQA and SocialIQA.

A.9 Details of selective supervised finetuning

In selective supervised fine-tuning, a sequence of attention heads and MLPs ordered by their significance will be given, denoted as

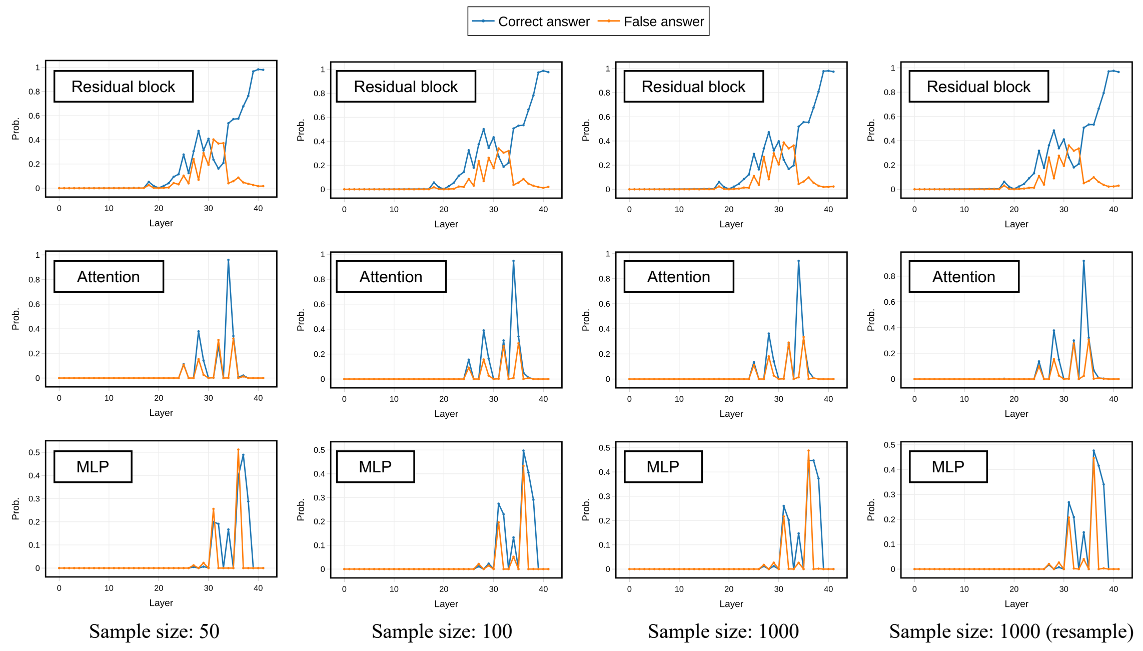


Figure 8: Scaling analysis of model behavior patterns across different sample sizes during answer generation on StrategyQA.

Pos.	Head	Attention score	Projection
$\mathcal{O}$	25.02	Q: Is Canon Inc. a Kabushiki gaisha?<newline> A: Canon Inc. is a	Japan, Japanese, Jepang, Japón, japan, Tokyo
$\mathcal{R}$	25.08	Q: Is Canon Inc. a Kabushiki gaisha?<newline> A: Canon Inc. is a Japanese company. Japanese companies are Kabushiki gaisha. Thus	confirmation, confirmación, Personendaten, verification
$\mathcal{S}$	31.03	Q: Is Canon Inc. a Kabushiki gaisha?<newline> A: Canon Inc. is a Japanese company. Japanese companies are Kabushiki gaisha. Thus, Canon Inc. is	yes, Yes, indeed YES, true, Indeed
$\mathcal{A}$	25.08	Q: Is Canon Inc. a Kabushiki gaisha?<newline> A: Canon Inc. is a Japanese company. Japanese companies are Kabushiki gaisha. Thus, Canon Inc. is a Kabushiki gaisha. So the answer is	confirmation, confirmación, confirmer, verification

Table 9: Attention score of the key attention heads (on StrategyQA in Gemma2-9B) on different tokens and top-k tokens after projecting the output of heads into the vocabulary space. The attention heads are obtained according to the activation patching result in Figure 15. The term Head 25.02 denotes the 2nd head in the attention layer of the 25th layer of the model.

$(MLP.l_1), (Head.l_2.h_2), (Head.l_3.h_3), \ldots,$ where l_i represents the layer index and h_i represents the head index of the i^{th} ranked head, only parameters of top K heads and top M MLPs are exclusively updated during fine-tuning. We optimize both the corresponding input mapping matrix $\{W_{l_1}^{h_1}, W_{l_2}^{h_2}, ..., W_{l_K}^{h_K}\}$ and the output mapping matrix $\{O_{l_1}^{h_1}, O_{l_2}^{h_1}, ..., O_{l_K}^{h_K}\}$ in top K heads simultaneously. For the selected MLP layer, we update all parameters in this layer.

For the format of training data, following Fu et al. (2023) and Huang et al. (2022), each sample in our training data is organized with the format of "`{Few-shot CoT prompt} Q: {Question}. A: {Rationale}`". We train the model using a learning rate of 1×10^{-4} and a batch size of 32 for 2 epochs. For supervised fine-tuning, a learning rate of 1×10^{-5} is utilized, while all other configurations remain consistent with SSFT training. Experiments are conducted on 8 NVIDIA A100 (80GB) GPUs.

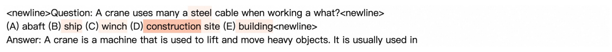
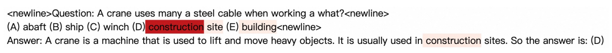
Pos.	Head	Attention score	Projection
$\mathcal{O}$	34.14	 <newline>Question: A crane uses many a steel cable when working a what?<newline> (A) abaft (B) ship (C) winch (D) construction site (E) building<newline> Answer: A crane is a machine that is used to lift and move heavy objects. It is usually used in	construction, Konstruktion, autorytatywna, Construction
$\mathcal{A}$	31.15	 <newline>Question: A crane uses many a steel cable when working a what?<newline> (A) abaft (B) ship (C) winch (D) construction site (E) building<newline> Answer: A crane is a machine that is used to lift and move heavy objects. It is usually used in construction sites. So the answer is: (D)	construction, constructions, struction, traction, construcción

Table 10: Attention score of the key attention heads (on CommonsenseQA in Gemma2-9B) on different tokens and top-k tokens after projecting the output of heads into the vocabulary space. The attention heads are obtained according to the activation patching result in Figure 11. The term Head 34.14 denotes the 14nd head in the attention layer of the 34th layer of the model.

ID	Feature Explanation
115620	Phrases related to confrontation and dynamics involving identity.
99851	References to characters and elements from the Harry Potter series.
82918	Concepts related to creation and storytelling in various media.
114490	Elements related to character dynamics and development in storytelling.

Table 11: Top-scoring features decoded by SAE in the output of MLP at layer 37 when predicting $\mathcal{O}$.

Layer	ID	Feature Explanation
7	106518	References to specific characters and items from a fictional universe.
7	113897	References to characters and locations from the Harry Potter series.
32	5548	References to specific characters and events from the Harry Potter series.
32	94534	References to the concept of "world" or "global" themes

Table 12: Top-scoring features decoded by SAE in the output of MLP at layer 7 and 37 at $\mathcal{S}$.

Head	Top tokens in projection
25.01	`Hogwarts, wizard, wizards, children,`
25.02	`Brito, British, London, Westminster`
29.06	`book, chapters, books, Book, bookId`
29.14	`wizards, wizard, Hogwarts, Harry`

Table 13: Top-scoring tokens in the key attention heads output when predicting $\mathcal{O}$. (i.e., "*fictional character*" for "*Harry Potter*".)

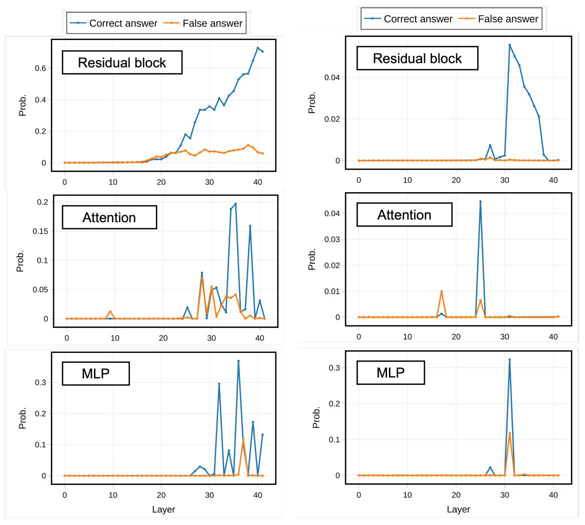


Figure 9: Logit attribution results on StrategyQA of Gemma2-9B. (a) Probability of $\mathcal{A}_t$ and $\mathcal{A}_f$ when predicting $\mathcal{C}$. (b) Probability of $\mathcal{A}_t$ and $\mathcal{A}_f$ when at $\mathcal{R}$.

Table 14: SSFT results using CommonsenseQA as the training dataset.

Models	Tuned Params.	ID Task CSQA Acc.	ID Task CSQA Δ	OOD Task Winogrande Acc.	Winogrande Δ	StrategyQA Acc.	StrategyQA Δ	SocialIQA Acc.	SocialIQA Δ	Average Acc.	Average Δ
Gemma2-9B	-	75.7	-	61.2	-	70.7	-	73.0	-	68.3	-
+ SFT	9B	81.3	+5.6	59.8	-1.4	71.0	+0.3	77.4	-5.6	66.1	-2.2
+ SSFT	0.2B	82.1	+6.4	65.1	+3.9	70.7	-	74.3	+1.3	70.0	+1.7
Llama2-7B	-	61.1	-	62.5	-	53.4	-	60.2	-	58.7	-
+ SFT	6.7B	72.3	+11.2	57.8	-4.7	53.5	+0.1	55.7	-3.0	56.2	-2.5
+ SSFT	0.2B	73.5	+12.4	63.1	+0.6	56.2	+2.8	63.2	+3.0	61.8	+3.1

Table 15: SSFT results using SocialIQA as the training dataset.

Models	Tuned Params.	ID Task SocialIQA Acc.	ID Task SocialIQA Δ	OOD Task Winogrande Acc.	Winogrande Δ	StrategyQA Acc.	StrategyQA Δ	CSQA Acc.	CSQA Δ	Average Acc.	Average Δ
Gemma2-9B	-	73.0	-	61.2	-	70.7	-	75.7	-	69.2	-
+ SFT	9B	80.2	+7.2	59.0	-2.2	72.0	+1.3	72.1	-3.6	67.7	-1.5
+ SSFT	0.2B	81.1	+8.1	64.2	+3.0	70.9	+0.2	77.0	+1.3	70.7	+1.5
Llama2-7B	-	61.1	-	62.5	-	53.4	-	60.2	-	58.7	-
+ SFT	6.7B	72.3	+11.2	57.8	-4.7	53.5	+0.1	55.7	-3.0	56.2	-2.5
+ SSFT	0.2B	73.5	+12.4	63.1	+0.6	56.2	+2.8	63.2	+3.0	61.8	+3.1

Table 16: Examples of Reasoning Cases from StrategyQA, CommonsenseQA, and SocialIQA Datasets. Three datasets evaluate distinct aspects of reasoning capabilities: CommonsenseQA focuses on basic conceptual understanding, SocialIQA assesses social and emotional intelligence, and StrategyQA tests multi-hop reasoning abilities that require the integration of multiple pieces of evidence through complex inference chains. The answer is generated by Gemma2-9B. In CommonsenseQA and SocialIQA, the entities are often abstract names or professions with no specific meaning. Therefore, we treat the options in the context as attributes, the final predicted option as the predicted attribute, and the remaining options as candidate objects.

Dataset	**StrategyQA**	**CommonsenseQA**	**SocialIQA**
Question	Is **Ganesha** associated with a Norse god?	The artist was sitting quietly pondering, then suddenly he began to paint when what struck him? (A) sadness (B) anxiety (C) inspiration (D) discomfort (E) insights	remy had a good talk with aubrey so aubrey understood remy better now. How would Remy feel as a result? (1) unsatisfied (2) calm (3) anxious
Answer	Ganesha is a **Hindu** god. Norse gods are associated with Norse mythology. Thus, Ganesha is not associated with a Norse god. So the answer is no.	The **artist** was sitting quietly pondering, then suddenly he began to paint when **inspiration** struck him. So the answer is: (C) inspiration.	**Remy** had a good talk with Aubrey. Thus, Aubrey understands Remy better. Remy will feel **calm** as a result. So the answer is: (2) calm.
Answer Type	Yes / No	Multiple Choice	Multiple Choice
Answer Token	no	(C) inspiration	(2) clam
Object	Ganesha	artist	Remy
Predicted Object	Hindu	inspiration	calm
Candidate Object	elephant, deity, god	sadness, anxiety, discomfort	unsatisfied, anxious

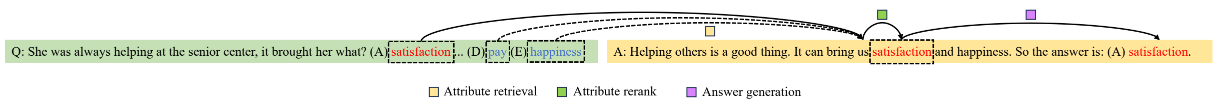
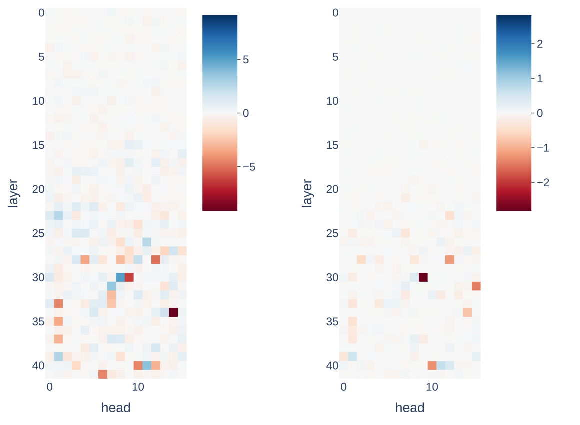
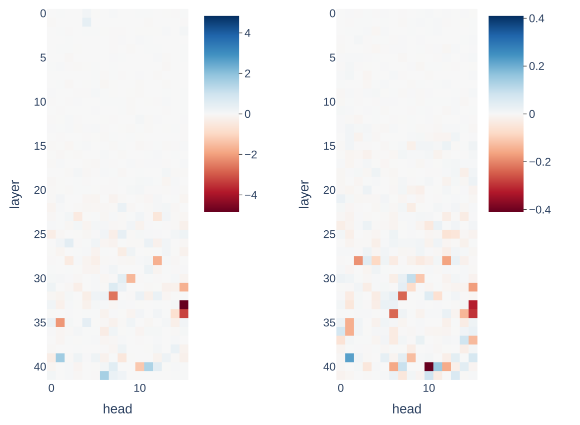

Q: She was always helping at the senior center, it brought her what? (A) satisfaction ... (D) pay (E) happiness
A: Helping others is a good thing. It can bring us satisfaction and happiness. So the answer is: (A) satisfaction.
Attribute retrieval
Attribute rerank
Answer generation

Figure 10: Model inner reasoning process on CommonsenseQA.



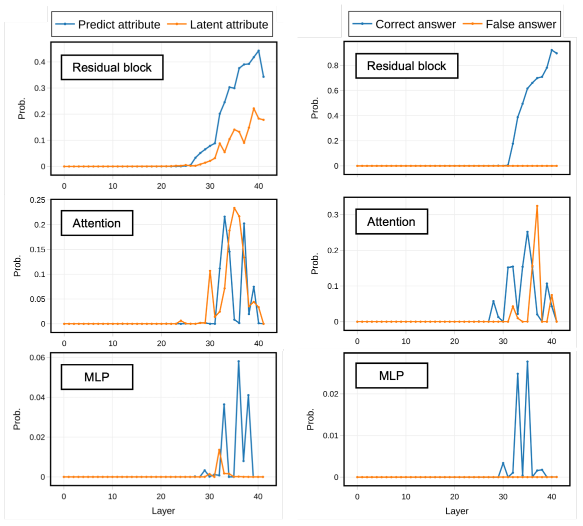
(a) Trace back at $\mathcal{A}$ (to $\mathcal{O}$) (b) Trace back at $\mathcal{O}$ (to choices)

Figure 11: Distribution of key heads (Gemma2-9B) during tracing back at different token positions on CommonsenseQA (averaged on 100 samples). The red squares indicate heads that have a significant positive impact on predicting the output token.



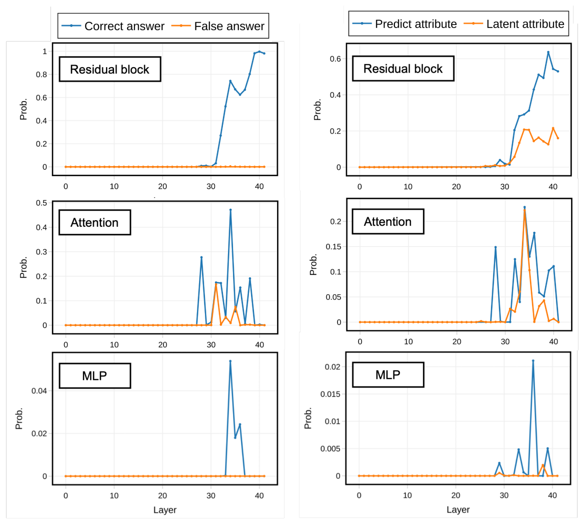
(a) Trace back at $\mathcal{A}$ (to $\mathcal{O}$) (b) Trace back at $\mathcal{O}$ (to choices)

Figure 13: Distribution of key heads (Gemma2-9B) during tracing back at different token positions on SocialIQA (averaged on 100 samples). The red squares indicate heads that have a significant positive impact on predicting the output token.



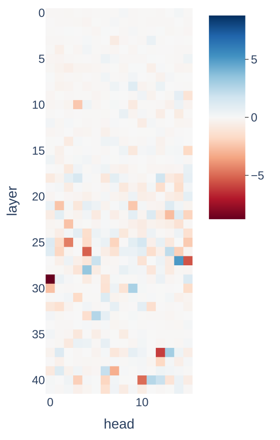
(a) (b)

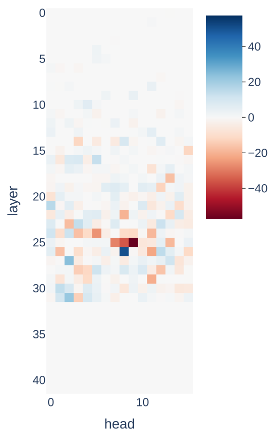
Figure 12: Logit attribution results on CommonsenseQA of Gemma2-9B. (a) Probability of $\mathcal{O}_p$ and $\mathcal{O}_c$ at the position of predicting $\mathcal{O}$. (b) Probability of $\mathcal{A}_l$ and $\mathcal{A}_f$ at the position of predicting $\mathcal{A}$.

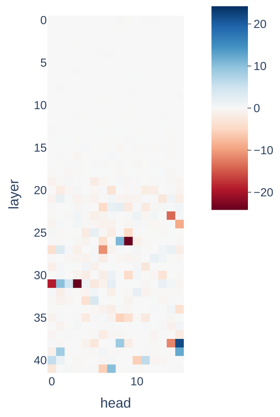


(a) (b)

Figure 14: Logit attribution results on SocialIQA of Gemma2-9B. (a) Probability of $\mathcal{O}_p$ and $\mathcal{O}_c$ at the position of predicting $\mathcal{O}$. (b) Probability of $\mathcal{A}_l$ and $\mathcal{A}_f$ at the position of predicting $\mathcal{A}$.


(a) Trace back at $\mathcal{O}$ (to $\mathcal{S}$)


(b) Trace back at $\mathcal{R}$ (to $\mathcal{O}$)


(c) Trace back at $\mathcal{C}$ (to $\mathcal{R}$)

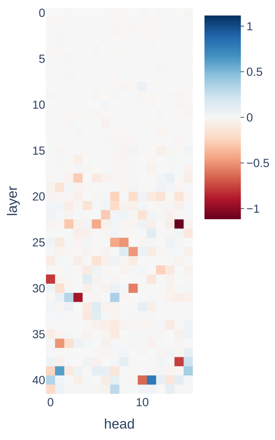
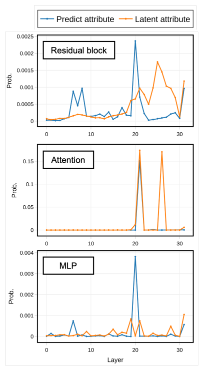
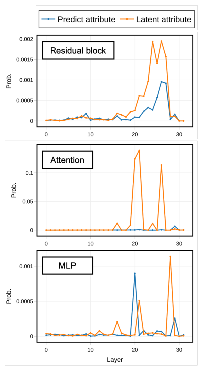
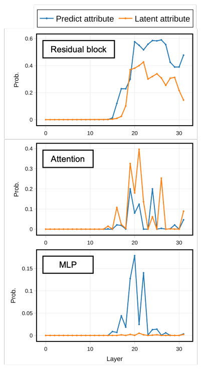
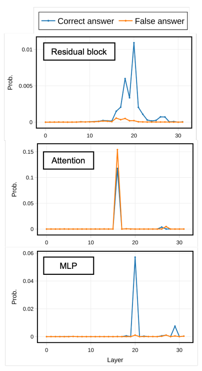

(d) Trace back at $\mathcal{A}$ (to $\mathcal{C}$)

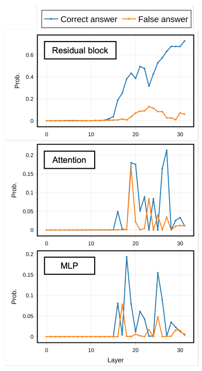
Figure 15: Distribution of key heads during tracing back at different token positions on StrategyQA (averaged on 100 samples). The red squares indicate heads that have a significant positive impact on predicting the output token.


(a)


(b)


(c)


(d)


(e)

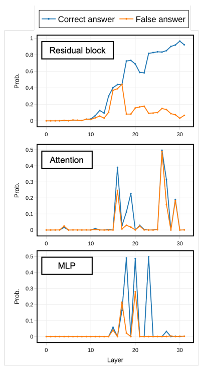

(f)

Figure 16: Logit attribution results on StrategyQA of Llama2-7B. (a) Probability of $\mathcal{O}_p$ and $\mathcal{O}_c$ at $\mathcal{C}$. (b) Probability of $\mathcal{O}_p$ and $\mathcal{O}_c$ at the end of question. (c) Probability of $\mathcal{O}_p$ and $\mathcal{O}_c$ at $\mathcal{O}$ prediction. (d) Probability of $\mathcal{A}_l$ and $\mathcal{A}_f$ at $\mathcal{R}$. (e) Probability of $\mathcal{A}_l$ and $\mathcal{A}_f$ at $\mathcal{S}$ prediction. (d) Probability of $\mathcal{A}_l$ and $\mathcal{A}_f$ at $\mathcal{A}$ prediction.

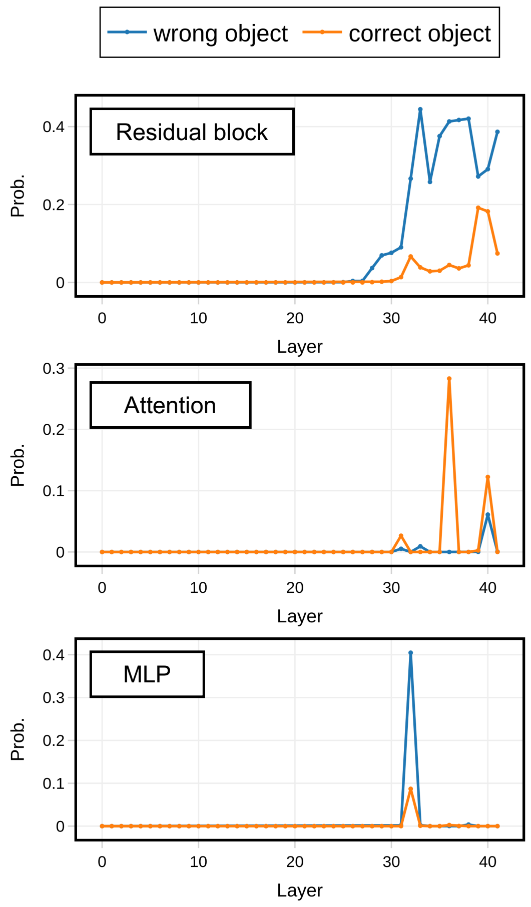
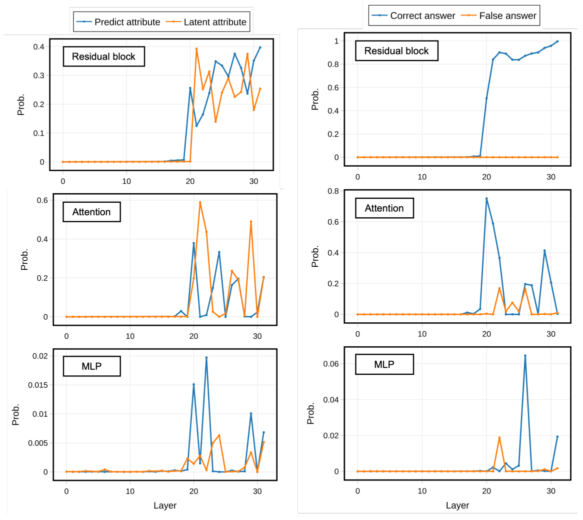
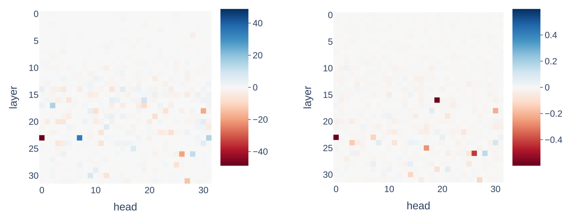


Figure 17: Comparative analysis of model behavior during reference errors. The plots show logit attributions for correct (orange) and wrong (blue) objects across different layers in Residual blocks, Attention mechanisms, and MLPs.



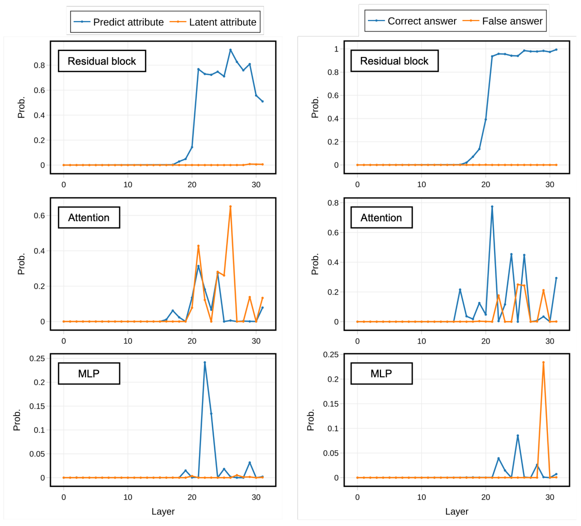
(a) (b)

Figure 18: Logit attribution results on SocialIQA of Llama2-7B. (a) Probability of $\mathcal{O}_p$ and $\mathcal{O}_c$ at the position of predicting $\mathcal{O}$. (b) Probability of $\mathcal{A}_l$ and $\mathcal{A}_f$ at the position of predicting $\mathcal{A}$.



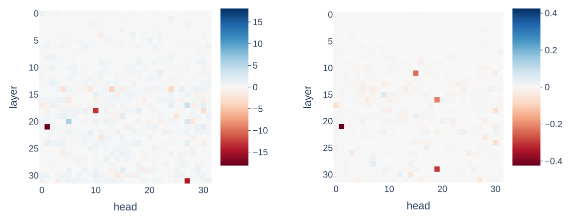
(a) Trace back at $\mathcal{O}$ (b) Trace back at $\mathcal{A}$ (to $\mathcal{O}$)

Figure 19: Distribution of key heads (Llama2-7B) during tracing back at different token positions on SocialIQA (averaged on 100 samples). The red squares indicate heads that have a significant positive impact on predicting the output token.



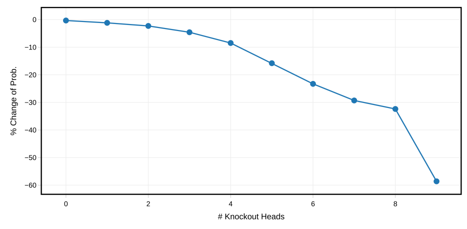
(a) (b)

Figure 20: Logit attribution results on CommonsenseQA of Llama2-7B. (a) Probability of $\mathcal{O}_p$ and $\mathcal{O}_c$ at the position of predicting $\mathcal{O}$. (b) Probability of $\mathcal{A}_l$ and $\mathcal{A}_f$ at the position of predicting $\mathcal{A}$.

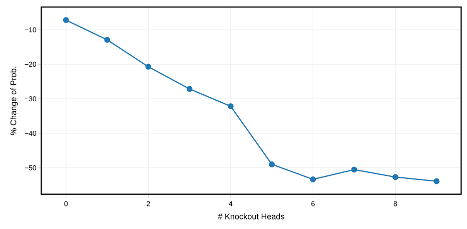


(a) Trace back at $\mathcal{O}$ (b) Trace back at $\mathcal{A}$ (to $\mathcal{O}$)

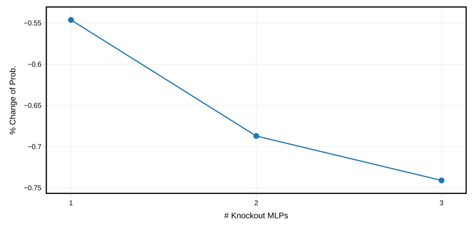
Figure 21: Distribution of key heads (Llama2-7B) during tracing back at different token positions on CommonsenseQA (averaged on 100 samples). The red squares indicate heads that have a significant positive impact on predicting the output token.



(a) Knockout top 10 heads when predicting $\mathcal{A}$



(b) Knockout top 10 heads when predicting $\mathcal{O}$

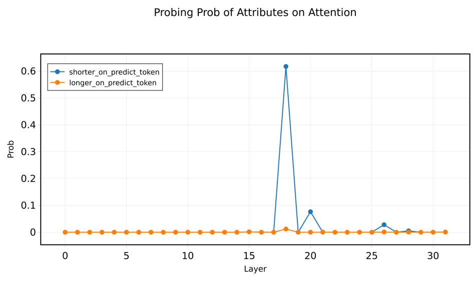


(c) Knockout top 3 MLPs when predicting $\mathcal{O}$

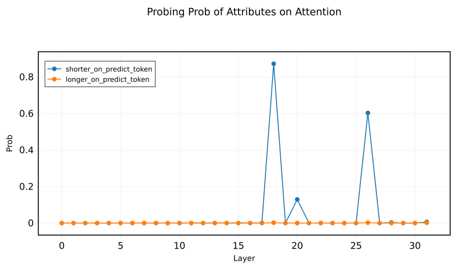
Figure 22: Knockout results on Gemma2-9B: (a) Decrease in probability of $\mathcal{A}_t$ when cumulatively intervene the top 10 heads for generating $\mathcal{A}$. (b) Decrease in probability of $\mathcal{O}_p$ when cumulatively intervene the top 10 heads for generating $\mathcal{O}$. (b) Decrease in probability of $\mathcal{O}_p$ when cumulatively intervene the top 3 MLPs for generating $\mathcal{O}$.

Input	Question: John cannot run the entire length of the track, he had been used to the field. The _ is short. (1) track (2) field Answer: A person who cannot run the entire length of a track likely feels uncomfortable or out of practice on a surface that is different from what they are used to. If John had been used to the field, it suggests that he is more accustomed to that environment. Therefore, the track must be
SSFT model output	**longer** than the field, making it difficult for him to run its entire length. ✅
Base model output	**shorter** than the field, as he struggles to run its entire length. ❌

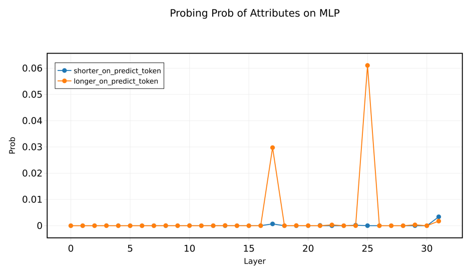
(a) Case study: output of SSFT and Base model



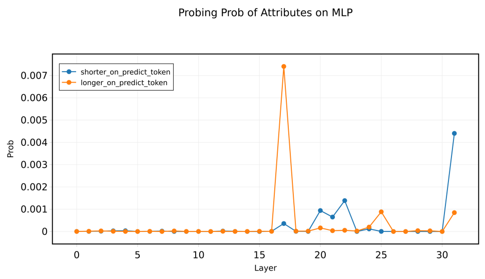
(b) Probing attention layer output for "shorter" and "longer" on SSFT model



(c) Probing attention layer output for "shorter" and "longer" on Base model



(d) Probing MLP layer output for "shorter" and "longer" on SSFT model



(e) Probing MLP layer output for "shorter" and "longer" on Base model

Figure 23: Comparison between the SSFT and Base models: (a) Case study highlights that the SSFT model correctly predicts the answer, while the Base model fails. (b, c) Probing results for attention layers show enhanced knowledge retrieval in the SSFT model compared to the Base model. (d, e) Probing results for MLP layers demonstrate improved reranking capability in the SSFT model. These findings confirm that the identified modules—attention heads for knowledge retrieval and MLP layers for reranking—are critical for accurate reasoning and were effectively strengthened through SSFT.

Prompt Template for Failure Case Classification

~~~
I am testing the accuracy of a large language model's responses on the multi-hop reasoning dataset, StrategyQA. Your task is to
classify the errors in the model's answers based on specific error types. For each question, I will provide the input question,
the model's answer, the correct answer and the reasoning steps needed for the correct answer. Your goal is to accurately classify
the errors using the following four error types:

1. **Entity Selection Error**: This occurs when the model picks the wrong entity from the input, leading to incorrect reasoning
in subsequent steps.
# Example 1:
Input:
```json
{
    "question": "Are the majority of Reddit users familiar with the Pledge of Allegiance?",
    "model_answer": "The Pledge of Allegiance is a pledge to the United States. Reddit is a social media site. Thus,
    the majority of Reddit users are not familiar with the Pledge of Allegiance. So the answer is no.",
    "correct_answer": "yes",
    "decomposition": [
        "What country do most Reddit users come from?",
        "What country is the Pledge of Allegiance associated with?",
        "Is #1 the same as #2?"
    ]
}
~~~
Classification: {"type": "Entity Selection Error", "explanation": "The model incorrectly selected Reddit as the entity
it spoke about, while the correct entity for reasoning should be 'Reddit users.' Therefore, this question should
be classified as an 'Entity Selection Error'".}

2. **Knowledge Retrieval Error**: This occurs when the model retrieves irrelevant, incomplete, or incorrect knowledge,
leading to flawed conclusions in the reasoning process.
# Example 1:
...

# Example 2:
...

3. **Conclusion Misalignment Error**: This occurs when the model's reasoning steps are correct, but the final conclusion is wrong.
# Example 1:
...

4. **Reasoning Logic Error**: This occurs when the logical connection between the reasoning steps and the final
conclusion breaks down. In this error, even if individual reasoning steps are correct, they fail to coherently lead
to the intended conclusion, causing the reasoning process to result in an illogical or incorrect outcome.
# Example 1:
...

Instructions: If the error does not fit into any of these four categories, please suggest a new category with a clear explanation.

For each input, I will provide the question, the model's answer, the correct answer, and the decomposition of reasoning steps.
You should return your classification and a brief explanation as
follows:
```json
{"type": "Entity Selection Error" or "Knowledge Retrieval Error" or "Conclusion Misalignment Error" or
"Incomplete Reasoning Error", "explanation": "Explain why this question belongs to the chosen category."}
```
Classficiation:
```

Figure 24: Prompt for using GPT-4 to automatically classify the category of failure case.
```