RADAR: Mechanistic Pathways for Detecting Data Contamination in LLM Evaluation

Ashish Kattamuri *, Harshwardhan Fartale, Arpita Vats, Rahul Raja

Posted Date: 29 September 2025

doi: 10.20944/preprints202509.2396.v1

Keywords: recall vs. reasoning; activation analysis; mechanistic interpretability; representation disentanglement; semantic recall detection; reasoning pathway identification; layer-wise activation analysis; latent representation mapping; attention attribution; causal tracing; neuron activation profiling; circuit discovery; alignment analysis; large language models; recommender systems; retrieval-augmented generation; hallucination detection; reasoning robustness; safety and alignment; embedding drift; contamination detection; semantic genome mapping; clustering quality metrics; interpretability benchmark

 doi:10.20944/preprints202509.2396.v1

Article

RADAR: Mechanistic Pathways for Detecting Data Contamination in LLM Evaluation

Ashish Kattamuri [1,*], Harshwardhan Fartale [2], Arpita Vats [1] and Rahul Raja [1]

[1] Independent Researcher, USA
[2] Indian Institute of Science
* Correspondence: ashishkattamuri@gmail.com

Abstract

Data contamination poses a significant challenge to reliable LLM evaluation, where models may achieve high performance by memorizing training data rather than demonstrating genuine reasoning capabilities. We introduce RADAR (Recall vs. Reasoning Detection through Activation Representation), a novel framework that leverages mechanistic interpretability to detect contamination by distinguishing recall-based from reasoning-based model responses. RADAR extracts 37 features spanning surface-level confidence trajectories and deep mechanistic properties including attention specialization, circuit dynamics, and activation flow patterns. Using an ensemble of classifiers trained on these features, RADAR achieves 93% accuracy on a diverse evaluation set, with perfect performance on clear cases and 76.7% accuracy on challenging ambiguous examples. This work demonstrates the potential of mechanistic interpretability for advancing LLM evaluation beyond traditional surface-level metrics. The code used in this work is publicly available.

Keywords: recall vs. reasoning; activation analysis; mechanistic interpretability; representation disentanglement; semantic recall detection; reasoning pathway identification; layer-wise activation analysis; latent representation mapping; attention attribution; causal tracing; neuron activation profiling; circuit discovery; alignment analysis; large language models; recommender systems; retrieval-augmented generation; hallucination detection; reasoning robustness; safety and alignment; embedding drift; contamination detection; semantic genome mapping; clustering quality metrics; interpretability benchmark

1. Introduction

Large Language Models (LLMs) show strong performance across tasks, but data contamination remains a major challenge in evaluation. Overlap between training and evaluation sets inflates metrics and obscures the distinction between genuine reasoning and memorization [1–3].

Existing detection methods typically compare evaluation data to training corpora, check n-gram overlaps, or flag verbatim outputs [4]. These approaches are limited: they require access to training data, fail with paraphrased contamination, and cannot reveal whether a model solved a task by recall or reasoning.

We propose **RADAR**, which instead analyzes internal computation dynamics. Leveraging mechanistic interpretability, RADAR extracts features from attention, hidden states, and activation flows [5,6]. Recall exhibits focused attention and rapid confidence convergence, while reasoning shows distributed activation and gradual stabilization.

Our contributions are: (1) We demonstrate that mechanistic features can reliably distinguish recall from reasoning with 93% accuracy, (2) We provide interpretable insights into the internal signatures of these cognitive processes, and (3) We offer a practical tool for contamination detection that works without access to training data.

 doi:10.20944/preprints202509.2396.v1

2. Methodology

2.1. Framework Architecture

RADAR operates through three integrated components: (1) **Mechanistic Analyzer** that extracts internal model states, (2) **Feature Extraction** that computes surface and mechanistic features, and (3) **Classifier** that predicts recall vs. reasoning, , as illustrated in Figure 1..

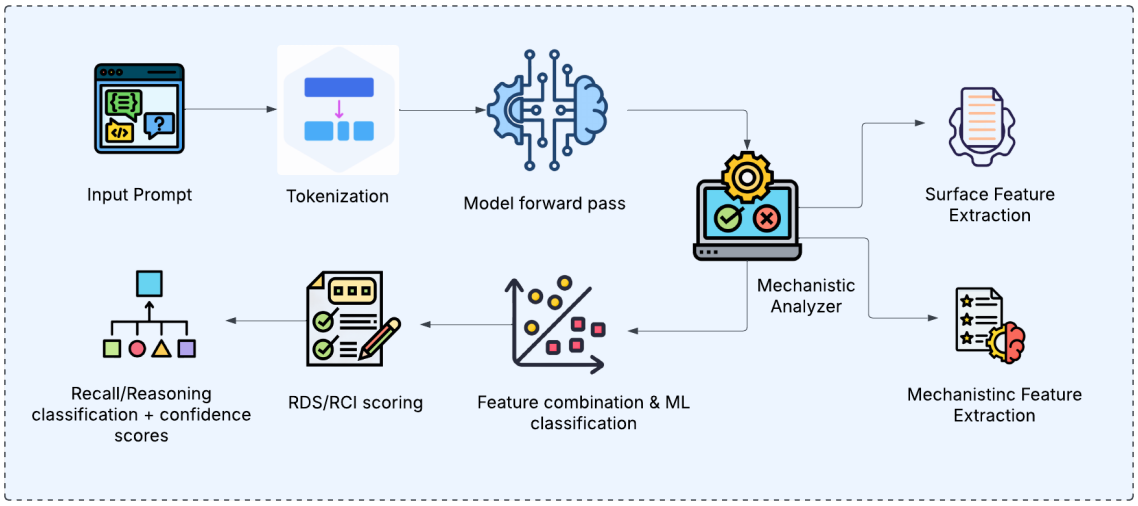

Figure 1. RADAR Framework Architecture: Input prompts are processed by the Mechanistic Analyzer to extract internal states, which are converted to Surface and Mechanistic Features, then classified by an ensemble to predict recall vs. reasoning with confidence scores.

The Mechanistic Analyzer interfaces with target LLMs (e.g., DialoGPT-medium) configured to output attention weights and hidden states. For each prompt, it analyzes attention patterns across all heads and layers, computing entropy and specialization metrics, and examines hidden state dynamics, including variance, norms, and effective rank.

2.2. Feature Engineering

We extract 37 features organized into two complementary categories:

Surface Features (17): Derived from the model's output trajectory across layers, these features capture prediction dynamics through confidence statistics (mean, std, max, min, range), convergence properties (layer, speed, slope), entropy measures (mean, change, information gain), and stability metrics.

Mechanistic Features (20): Derived from attention weights and hidden states across all layers and heads, these features capture internal computational mechanisms, including attention specialization (specialized heads, specialization scores, entropy), circuit dynamics (depth, complexity, activation flow), intervention sensitivity (ablation robustness, critical components), working memory (hidden state variance, norm trajectories), and causal effects (logit attribution, mediation scores).

2.3. Classification System

The classification module employs an ensemble of four supervised learning models: Random Forest[7], Gradient Boosting[8], Support Vector Machine (SVM)[9], and Logistic Regression[10]. Each model is trained on the extracted feature vectors after normalization with `StandardScaler`. The scaling ensures zero mean and unit variance across features:

$$x_i' = \frac{x_i - \mu_i}{\sigma_i},$$

where μ_i and σ_i denote the mean and standard deviation of feature i.

 doi:10.20944/preprints202509.2396.v1

For prediction, each base classifier $j \in \{1, \ldots, M\}$ (with $M = 4$) outputs a hard label $\hat{y}_j$ and a probability estimate $p_j = P(y = 1 \mid x')$, where $y = 1$ corresponds to *recall* and $y = 0$ to *reasoning*. The ensemble aggregates these outputs as:

$$\hat{y} = 1\left[\frac{1}{M}\sum_{j=1}^{M}\hat{y}_j > \frac{1}{2}\right], \qquad \bar{p} = \frac{1}{M}\sum_{j=1}^{M} p_j$$

The final confidence score is defined consistently with the predicted label:

$$\text{conf} = \bar{p},\ \hat{y} = 1\ (\text{recall}), \qquad 1 - \bar{p},\ \hat{y} = 0\ (\text{reasoning})$$

3. Experiments and Results

3.1. Experimental Setup and Results

We curated two datasets: a balanced training set (30 examples: 15 recall, 15 reasoning) and a diverse test set (100 examples: 20 clear recall, 20 clear reasoning, 30 challenging cases, 30 complex reasoning). The classifier achieved 96.7% cross-validation accuracy during training.

RADAR achieved an overall accuracy of 93.0% on the test set, with task-specific performance of 97.7% on recall tasks and 89.3% on reasoning tasks. A detailed breakdown of performance across different categories is shown in Table 1.

Table 1. RADAR Performance Results.

Overall Performance		*Category-wise Performance*	
Overall Accuracy	93.0%	Clear Recall	100% (20/20)
Recall Tasks	97.7%	Clear Reasoning	100% (20/20)
Reasoning Tasks	89.3%	Challenging Cases	76.7% (23/30)
		Complex Reasoning	100% (30/30)

3.2. Feature Analysis

Key discriminative features include specialized attention heads (higher for recall), circuit complexity (higher for reasoning), and confidence convergence patterns (faster for recall). Recall tasks showed a mean Recall Detection Score (RDS) of 0.933 compared to 0.375 for reasoning, demonstrating clear separability, as shown in Figure 2.

 doi:10.20944/preprints202509.2396.v1

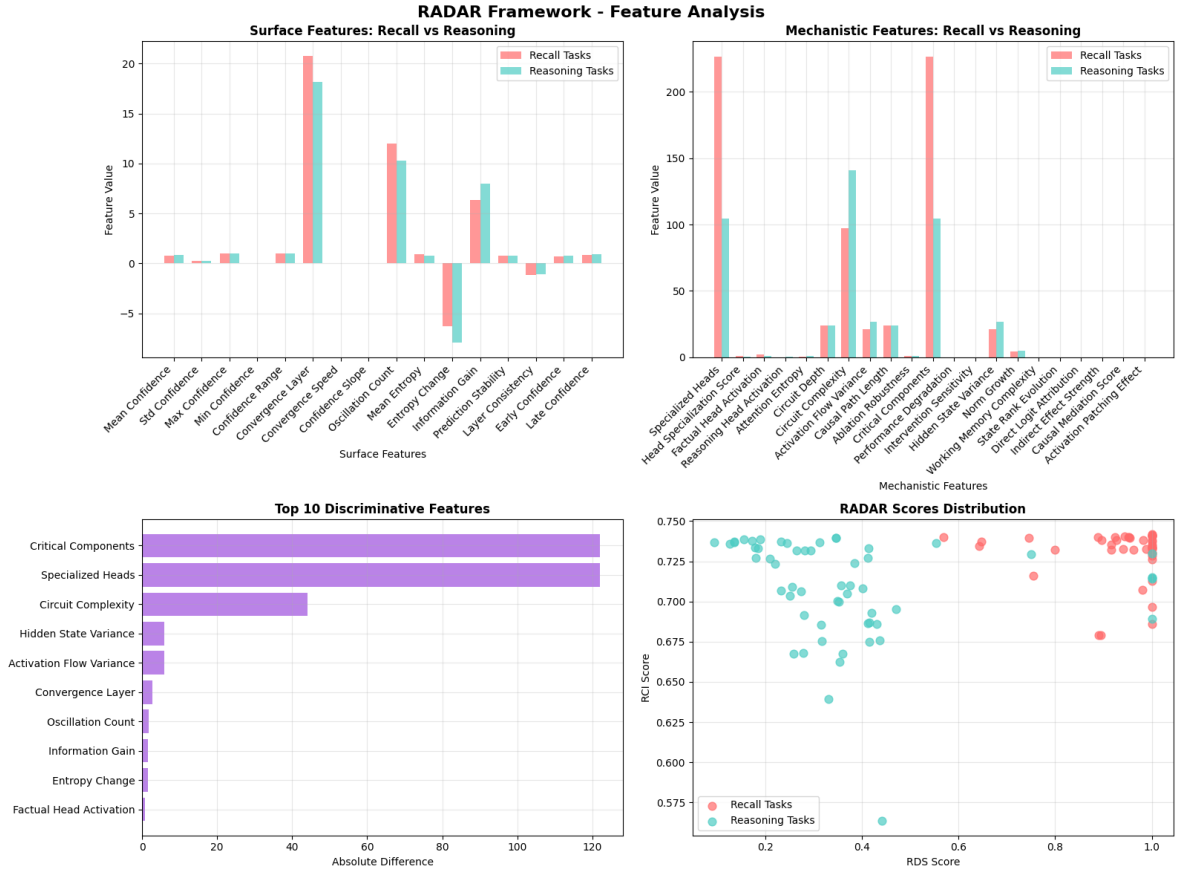

Figure 2. RADAR Feature Analysis: Comparison of surface and mechanistic features for recall and reasoning tasks, highlighting top discriminative features and RDS–RCI score distribution. The results show recall tasks characterized by early confidence and specialized heads, while reasoning tasks rely on broader circuit complexity and higher activation flow variance. The scatter plot demonstrates strong clustering, with recall tasks in the high-RDS region and reasoning tasks distributed in lower-RDS regions.

Surface features revealed that recall tasks exhibit higher early confidence and faster convergence, whereas reasoning tasks show gradual confidence build-up and later stabilization. Mechanistic features highlighted that recall relies on focused attention patterns and specialized heads, while reasoning engages broader network resources with higher activation flow variance. The scatter plot visualization confirms clear separation between recall and reasoning tasks in the RDS–RCI score space.

4. Discussion and Implications

4.1. Contamination Detection Applications

RADAR's ability to distinguish recall from reasoning has direct implications for contamination detection. When reasoning-type prompts elicit recall-like internal signatures (high confidence, fast convergence, specialized attention), this indicates potential contamination where the model "knows" rather than "computes" the answer.

Our approach offers several benefits: (1) Works without access to training data, (2) Analyzes computational processes rather than just outputs, (3) Provides interpretable features explaining classifications, (4) Complements existing external detection methods, and (5) Scales to different model architectures.

4.2. Interpretability Insights

The feature analysis confirms that recall and reasoning leave distinct mechanistic signatures. Recall processes exhibit focused attention patterns with rapid confidence convergence, suggesting direct retrieval pathways. Reasoning processes show distributed attention, gradual confidence build-

 doi:10.20944/preprints202509.2396.v1

up, and higher circuit complexity, indicating multi-step computational processes engaging broader network resources. Top discriminative features (Specialized Heads, Circuit Complexity, Hidden State Variance) capture fundamental differences in how the model processes information, providing interpretable insights into the underlying cognitive mechanisms.

5. Conclusion

RADAR demonstrates that mechanistic interpretability can effectively detect data contamination by analyzing internal LLM processing signatures. Our framework achieves 93% accuracy in distinguishing recall from reasoning, providing interpretable insights into the cognitive processes underlying model responses. This work opens new directions for LLM evaluation that move beyond surface-level metrics to examine computational mechanisms. The ability to detect contamination without training data access, combined with interpretable mechanistic features, makes RADAR a valuable tool for improving LLM evaluation reliability.

Future work will explore scaling to larger models, developing unsupervised detection methods, and extending to other contamination types. The integration of mechanistic interpretability with traditional evaluation methods promises more robust and trustworthy LLM assessment frameworks.

Appendix A. Implementation Details

Appendix A.1. Model Configuration

The target model (microsoft/DialoGPT-medium) was configured with `output_attentions=True` and `output_hidden_states=True`. Analysis focused on input prompt tokens to capture reasoning during comprehension.

Appendix A.2. Feature Computation

Surface features tracked confidence and entropy trajectories across layers. Mechanistic features analyzed attention weight distributions using entropy measures and computed hidden state statistics including effective rank via SVD decomposition.

Appendix A.3. Training Procedure

Features were preprocessed and standardized using StandardScaler. Each ensemble model was trained with k-fold cross-validation for robust performance estimation.

Appendix B. Training and Test Datasets

This appendix provides details on the datasets used for training and evaluating the Enhanced RADAR Framework's Recall–Reasoning Classifier.

Appendix B.1. Training Dataset

The training dataset was used exclusively for training the classifier. It consists of 30 examples, each containing a prompt and a corresponding label indicating whether the expected response is based on "recall" or "reasoning." The composition is as follows:

Table A1. Composition of the training dataset.

Category	Count
Total Examples	30
Recall Examples	15
Reasoning Examples	15

This dataset provides the classifier with a basic representation of the internal features and patterns that distinguish factual retrieval from logical inference. Representative examples are shown below:

 doi:10.20944/preprints202509.2396.v1

Table A2. Sample training dataset prompts and labels.

Prompt	Label
"The capital of France is"	recall
"If X is the capital of France, then X is"	reasoning
"2 + 2 equals"	recall
"If a triangle has angles 60, 60, and X degrees, then X equals"	reasoning

Appendix B.2. Test Dataset

The test dataset was used only for evaluating the trained classifier on unseen data. It comprises 100 examples, with broader coverage to assess generalization across different levels of difficulty and ambiguity:

Table A3. Composition of the test dataset.

Category	Count
Total Examples	100
Clear Recall Examples	20
Clear Reasoning Examples	20
Challenging/Ambiguous Cases	30
Complex Reasoning Cases	30

The inclusion of challenging and complex reasoning cases is important for evaluating robustness, especially in detecting possible data contamination where a reasoning task could be solved by recall.

Examples from each category are shown below:

Table A4. Sample test dataset prompts by category.

Category	Example Prompt	Label
Clear Recall	"The capital of Germany is"	recall
Clear Reasoning	"If a rectangle has length 5 and width 3, its area is"	reasoning
Challenging/Ambiguous	"What is the sum of 10 and 15?"	reasoning
Complex Reasoning	"If a store has 100 items and sells 30% of them, how many items remain?"	reasoning

Appendix B.3. Why Challenging or Ambiguous Prompts Are Difficult

Challenging or ambiguous prompts are difficult because they blur the line between recall and reasoning.

- Some prompts may appear to require reasoning (e.g., arithmetic) but can be solved by memorized recall if the model has seen similar examples during training.
- Conversely, some factual prompts may trigger reasoning-like processing if the information is incomplete or framed indirectly.
- Ambiguity arises when the surface form of the task does not clearly signal whether the solution requires stored knowledge or active inference.

These cases are crucial for evaluation because they reveal whether the classifier is robust to subtle shifts in task framing and whether it can correctly separate recall-driven answers from reasoning-based ones.

Appendix C. Scoring

In addition to the binary classification, the RADAR Framework computes several continuous scores that provide a more nuanced perspective:

 doi:10.20944/preprints202509.2396.v1

- **Recall Detection Score (RDS):** Indicates how strongly the analysis suggests a recall-based process, combining specific surface and mechanistic features.
- **Reasoning Complexity Index (RCI):** Reflects the complexity and depth of processing, suggesting a reasoning-based process. Derived from a combination of surface and mechanistic features.
- **Mechanistic Score:** Focuses on features related to causal effects and intervention sensitivity.
- **Circuit Complexity Score:** Based on features describing the depth and complexity of the activated computational graph.

These scores are calculated using predefined formulas that weigh different features according to their relevance to recall and reasoning processes. They provide complementary information to the classifier's binary output.

Appendix D. Feature Documentation

The RADAR (Recall And Deliberative Analysis of Reasoning) Framework extracts 37 features from language model behavior to distinguish between recall-based and reasoning-based tasks. These features are organized into two categories: **Surface Features** (16 features) that capture observable trajectory patterns, and **Mechanistic Features** (21 features) that analyze internal model dynamics through attention patterns and activation analysis.

Appendix D.1. Surface Features (16 Features)

Surface features analyze the confidence and entropy trajectories across all model layers to capture behavioral patterns without requiring deep mechanistic analysis.

Appendix D.1.1. Confidence-Based Features (8 Features)

Feature	Type	Definition & Computation
`mean_confidence`	float	Mean confidence across all layers: $\bar{c} = \frac{1}{L}\sum_{l=1}^{L} c_l$ where c_l is the maximum softmax probability at layer l, and L is the total number of layers.
`std_confidence`	float	Standard deviation of confidence trajectory: $\sigma_c = \sqrt{\frac{1}{L-1}\sum_{l=1}^{L}(c_l - \bar{c})^2}$. Higher values indicate more variable confidence across layers.
`max_confidence`	float	Maximum confidence achieved: $c_{max} = \max_{l \in [1,L]} c_l$. Indicates peak certainty reached by the model.
`min_confidence`	float	Minimum confidence observed: $c_{min} = \min_{l \in [1,L]} c_l$. Represents lowest certainty point in processing.
`confidence_range`	float	Range of confidence values: $\Delta c = c_{max} - c_{min}$. Measures the span of confidence variation across layers.
`convergence_layer`	int	Layer index where maximum confidence is achieved: $l^* = \arg\max_l c_l$. Earlier convergence may indicate simpler recall tasks.
`convergence_speed`	float	Inverse of convergence layer: $v_{conv} = \frac{1}{l^*+1}$. Higher values indicate faster convergence to high confidence.
`confidence_slope`	float	Linear regression slope of confidence trajectory: $\beta = \frac{\sum_{l=1}^{L}(l-\bar{l})(c_l-\bar{c})}{\sum_{l=1}^{L}(l-\bar{l})^2}$ where $\bar{l} = \frac{L+1}{2}$. Positive slopes indicate increasing confidence.

Appendix D.1.2. Trajectory Dynamics Features (4 Features)

 doi:10.20944/preprints202509.2396.v1

Feature	Type	Definition & Computation
`oscillation_count`	int	Number of sign changes in the discrete confidence derivative. Let $\Delta c_l = c_{l+1} - c_l$ for $l = 1, \ldots, L-1$. Then $$\text{oscillation\_count} = \#\{\, l \in \{1, \ldots, L-2\} : (\Delta c_l)(\Delta c_{l+1}) < 0 \,\},$$ i.e., consecutive derivatives with opposite sign. Zeros in Δc_l are ignored for sign changes.
`early_confidence`	float	Mean confidence in the first half of layers: $$c_{\text{early}} = \frac{1}{\lfloor L/2 \rfloor} \sum_{l=1}^{\lfloor L/2 \rfloor} c_l.$$ Captures initial model certainty.
`late_confidence`	float	Mean confidence in the second half of layers: $$c_{\text{late}} = \frac{1}{\lceil L/2 \rceil} \sum_{l=\lfloor L/2 \rfloor + 1}^{L} c_l.$$ Captures final model certainty.
`prediction_stability`	float	Inverse of confidence standard deviation: $$s_{\text{pred}} = 1 - \sigma_c, \quad \sigma_c = \sqrt{\frac{1}{L-1} \sum_{l=1}^{L} (c_l - \bar{c})^2}, \quad \bar{c} = \frac{1}{L} \sum_{l=1}^{L} c_l.$$ Higher values indicate more stable predictions across layers.

Appendix D.1.3. Information-Theoretic Features (4 Features)

Feature	Type	Definition & Computation
`mean_entropy`	float	Average entropy across layers: $$\bar{H} = \frac{1}{L} \sum_{l=1}^{L} H_l, \quad H_l = -\sum_i p_{l,i} \log p_{l,i},$$ where $p_{l,i}$ is the probability of token i at layer l and log is the natural logarithm.
`entropy_change`	float	Change from first to last layer: $$\Delta H = H_L - H_1.$$ Negative values indicate uncertainty reduction.
`information_gain`	float	Negative entropy change: $$IG = -\Delta H = H_1 - H_L.$$ Positive values indicate successful uncertainty reduction.

Continued on next page

 doi:10.20944/preprints202509.2396.v1

Feature	Type	Definition & Computation
`layer_consistency`	float	Inverse of entropy standard deviation: $$\text{consistency} = 1 - \sqrt{\frac{1}{L-1}\sum_{l=1}^{L}(H_l - \bar{H})^2}.$$ Higher values indicate more consistent information processing across layers.

Appendix D.2. Mechanistic Features (21 Features)

Mechanistic features analyze internal model dynamics through attention patterns, activation flows, and causal intervention proxies to understand the computational mechanisms underlying different task types.

Appendix D.2.1. Attention Specialization Features (5 Features)

Feature	Type	Definition & Computation
`num_specialized_heads`	int	Total count of attention heads with entropy below a specialization threshold (typically $\tau = 1.5$): $$N_{\text{spec}} = \sum_{l=1}^{L}\sum_{h=1}^{H}\mathbf{1}[H_{l,h} < \tau],$$ where $H_{l,h}$ is the entropy of head h in layer l.
`head_specialization_score`	float	Normalized specialization measure: $$S_{\text{head}} = 1 - \frac{\bar{H}_{\text{attn}}}{3.0},$$ where $\bar{H}_{\text{attn}}$ is the mean attention entropy across all heads. Higher values indicate more specialized attention patterns.
`factual_head_activation`	float	Inverse relationship with attention entropy: $$A_{\text{fact}} = \frac{1}{\bar{H}_{\text{attn}} + \epsilon}, \quad \epsilon = 10^{-8}.$$ Higher values suggest factual recall patterns (low entropy, focused attention).
`reasoning_head_activation`	float	Proportional to attention entropy: $$A_{\text{reason}} = \frac{\bar{H}_{\text{attn}}}{3.0}.$$ Higher values suggest reasoning patterns (high entropy, distributed attention).

 doi:10.20944/preprints202509.2396.v1

`attention_entropy`	float	Mean entropy across all attention heads: $\bar{H}_{\text{attn}} = \frac{1}{LH}\sum_{l=1}^{L}\sum_{h=1}^{H} H_{l,h}, \quad H_{l,h} = -\sum_{i,j} A_{l,h}^{(i,j)} \log A_{l,h}^{(i,j)},$ where $A_{l,h}^{(i,j)}$ is the attention weight from position i to j.

Appendix D.2.2. Circuit Dynamics Features (4 Features)

Feature	Type	Definition & Computation
`effective_circuit_depth`	float	Number of layers with significant causal effects. Equal to the number of attention layers analyzed. Represents the depth of the computational circuit.
`circuit_complexity`	float	Product of variance and norm growth: $C_{\text{circuit}} = \sigma_{\text{var}}^2 \cdot \gamma_{\text{norm}},$ where σ_{var}^2 is activation variance growth and γ_{norm} is the norm growth trajectory.
`activation_flow_variance`	float	Variance in activation magnitudes across layers. Measures how much activation patterns change between layers, indicating computational complexity.
`causal_path_length`	float	Length of the causal computation path. Currently equal to circuit depth, representing the number of processing steps in the causal chain.

Appendix D.2.3. Intervention Sensitivity Features (4 Features)

Feature	Type	Definition & Computation
`ablation_robustness`	float	Robustness to component removal: $R_{\text{ablation}} = 1 - \frac{\bar{H}_{\text{attn}}}{5.0}.$ Higher entropy (distributed attention) leads to lower robustness.
`critical_component_count`	int	Number of critical components: $N_{\text{critical}} = \max(1, N_{\text{spec}}).$ Uses specialized head count as a proxy for critical components.
`performance_degradation_slope`	float	Rate of performance degradation under intervention: $\beta_{\text{degrad}} = \lvert\sigma_{\text{causal}}\rvert,$ where σ_{causal} is the standard deviation of causal effect estimates across layers.

 doi:10.20944/preprints202509.2396.v1

`intervention_sensitivity`	float	Sensitivity to interventions: $$S_{\text{interv}} = 1 - R_{\text{ablation}}.$$ Inverse of ablation robustness; higher values indicate greater sensitivity.

Appendix D.2.4. Working Memory Features (4 Features)

Feature	Type	Definition & Computation
`hidden_state_variance`	float	Variance in hidden state activations. Measures variability in internal representations across layers, indicating working memory usage.
`norm_growth_trajectory`	float	Growth pattern of activation norms. γ_{norm} tracks how activation magnitudes change across layers, indicating information accumulation.
`working_memory_complexity`	float	Complexity of working memory usage. Currently uses rank evolution as a proxy for working memory complexity.
`state_rank_evolution`	float	Evolution of representation rank. $R_{\text{evolution}}$ measures how the effective dimensionality of representations changes across layers.

Appendix D.2.5. Causal Effect Features (4 Features)

Feature	Type	Definition & Computation
`direct_logit_attribution`	float	Direct causal effect on output: $$E_{\text{direct}} = \frac{1}{L}\sum_{l=1}^{L}\frac{\bar{H}_{\text{attn},l}}{10},$$ where $\bar{H}_{\text{attn},l}$ is mean attention entropy at layer l. Proxy for direct causal contribution.
`indirect_effect_strength`	float	Strength of indirect causal effects: $$E_{\text{indirect}} = \sigma_{\text{causal}},$$ where σ_{causal} is the standard deviation of layer-wise causal effect estimates.
`causal_mediation_score`	float	Mediation effect strength: $$M_{\text{causal}} = E_{\text{direct}} \times E_{\text{indirect}}.$$ Product of direct and indirect effects, measuring causal mediation.

 doi:10.20944/preprints202509.2396.v1

`activation_patching_effect`	float	Proxy measure for activation patching: $P_{\text{patch}} = E_{\text{direct}}$. Note: This is computed from attention entropy, not from actual activation patching experiments.

Appendix E. Feature Computation Pipeline

Appendix E.1. Surface Feature Extraction

1. Extract confidence trajectory:
$$\{c_l\}_{l=1}^{L}, \quad c_l = \max_i p_{l,i}$$
2. Extract entropy trajectory:
$$\{H_l\}_{l=1}^{L}, \quad H_l = -\sum_i p_{l,i} \log p_{l,i}$$
3. Compute statistical measures: mean, standard deviation, minimum, maximum, and range.
4. Analyze trajectory dynamics: slope, oscillations, and convergence properties.
5. Calculate information-theoretic measures.

Appendix E.2. Mechanistic Feature Extraction

1. Analyze attention patterns across all layers and heads.
2. Compute attention entropy for each head:
$$H_{l,h} = -\sum_{i,j} A_{l,h}^{(i,j)} \log A_{l,h}^{(i,j)}$$
3. Identify specialized heads:
$$N_{\text{spec}} = \sum_{l,h} \mathbf{1}[H_{l,h} < 1.5]$$
4. Analyze activation patterns (variance, norms, and rank evolution).
5. Compute proxy causal effects from attention entropy.
6. Calculate intervention sensitivity measures.

Appendix F. Important Notes and Limitations

Appendix F.1. Proxy Measures

Several features rely on proxy measures rather than direct computation:

- **Causal effects**: Derived from attention entropy instead of actual interventions.
- **Activation patching**: Approximated via attention entropy proxy, not true patching experiments.
- **Critical components**: Approximated using specialized head counts.
- **Working memory**: Approximated using rank evolution as a complexity proxy.

Appendix F.2. Computational Considerations

- All features can be computed in a single forward pass.
- No gradient computation is required for feature extraction.
- Attention patterns are analyzed across all layers and heads.
- Surface features require only the output probability distributions.

 doi:10.20944/preprints202509.2396.v1

Appendix G. Usage in Classification

The 37 features are concatenated into a single feature vector:

$$\mathbf{f} = [\mathbf{f}_{\text{surface}}, \mathbf{f}_{\text{mechanistic}}] \in \mathbb{R}^{37}$$

This vector is then used to train classifiers (Random Forest, Gradient Boosting, SVM, Logistic Regression) to distinguish between recall and reasoning tasks.

References

1. Golchin, S.; Surdeanu, M. Time travel in llms: Tracing data contamination in large language models. *arXiv preprint arXiv:2308.08493* **2023**.
2. Deng, C.; Zhao, Y.; Tang, X.; Gerstein, M. Investigating data contamination in modern benchmarks for large language models. *arXiv preprint arXiv:2311.09783* **2023**.
3. Feldman, V. Does learning require memorization? a short tale about a long tail. In Proceedings of the Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, 2020, pp. 954–959.
4. Carlini, N.; Tramer, F.; Wallace, E.; Jagielski, M.; Herbert-Voss, A.; Lee, K.; et al. Extracting training data from large language models. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 2633–2650.
5. Elhage, N.; Nanda, N.; Olsson, C.; Henighan, T.; Joseph, N.; Mann, B.; et al. A mathematical framework for transformer circuits. *Anthropic* **2021**.
6. Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; Carter, S. Zoom in: An introduction to circuits. *Distill* **2020**, *5*, e00024–001.
7. Breiman, L. Random forests. *Machine learning* **2001**, *45*, 5–32.
8. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Annals of statistics* **2001**, pp. 1189–1232.
9. Cortes, C.; Vapnik, V. Support-vector networks. *Machine learning* **1995**, *20*, 273–297.
10. Hosmer, D.W.; Lemeshow, S.; Sturdivant, R.X. *Applied Logistic Regression*; Wiley, 2013.