Token-Weight Trajectory Interpretability: A Practical Framework for Understanding LLM Behavior

T. Harutyunyan

October 22, 2025

Abstract

We present a practical, hybrid framework for interpreting large language models (LLMs) that reconstructs token-level decision trajectories by combining token probability logs with weight-activation traces. A lightweight, deterministic analysis program ingests raw model instrumentation (top-N token distributions per step, per-token activations, and selected weight/head influence metrics), produces structured token-weight trajectories, and renders readable traces for human review. Optionally, the processed traces can be fed to an LLM for concise summarization, anomaly detection, or clustering; this second step is purely assistive and not required for the method to function. The approach provides a unified way to (1) explain token-level choices, (2) detect and localize hallucinations, and (3) expose patterns of systematic bias—without modifying model weights or relying on gradient-based attribution. We include formal definitions, an implementation-agnostic pipeline, toy examples, an evaluation plan, and a discussion of limitations and ethical considerations.

1 Introduction

Modern transformer-based LLMs produce fluent and useful text but remain largely opaque: their internal reasoning is not directly accessible, and when they err (hallucinate) or reproduce bias, engineers and stakeholders struggle to explain why. Interpretability research offers many tools (attention visualization, gradient-based attributions, neuron inspection), but none deliver a consistent, token-level, human-readable reconstruction of the *decision path* that led a model to pick a particular token over close alternatives.

We propose a practical pipeline—Token-Weight Trajectory Interpretability (TWTI)—that:

- Records the token-level decision landscape (top-N candidate tokens + scores) for each generation step.
- Logs a compact, thresholded snapshot of the influential activations (weights, heads, and layer outputs) that contributed to the selection.
- Converts these raw logs into structured token-weight trajectories using a deterministic analysis program.

- Produces readable traces for human consumption and optional LLM-assisted summarization.

This hybrid approach is intentionally pragmatic: the core interpretability is provided by a simple, auditable program, while model-assisted summarization remains optional. The result is a scalable, transparent, and actionable interpretability tool usable in debugging, compliance, and safety workflows.

2 Contributions

1. **Framework:** A clear, implementation-agnostic pipeline for reconstructing token-by-token decision trajectories.
2. **Programmatic interpreter:** A specification and pseudocode for a simple program that converts raw logs into structured, human-readable traces.
3. **Demonstrative examples:** Toy examples illustrating a correct factual choice, a hallucination, and a bias case.
4. **Evaluation plan:** Objective criteria and benchmarks for validating the usefulness and fidelity of explanations.
5. **Ethical and security guidance:** Recommendations to avoid misuse and protect sensitive model internals.

3 Related Work

This section positions TWTI relative to established interpretability methods:

- **Attention-based interpretation:** Explores how attention scores reveal token-to-token influence, but attention alone does not fully explain generation decisions.
- **Gradient and saliency methods:** Provide sensitivity information but are often noisy and hard to interpret at token granularity.
- **Mechanistic interpretability / circuit analysis:** Seeks to map causal substructures but typically requires manual, time-consuming work; circuits are difficult to scale.
- **Posthoc explanation via model querying:** Asking models to explain outputs has merit but conflates the black box with the explainer; TWTI's program-first approach avoids that conflation.

TWTI complements these lines of work by providing deterministic, token-aligned traces that can feed into or be compared against existing methods.

4 Formal Definitions and Notation

Let a model generate a sequence of tokens $(t_1, t_2, \ldots, t_T)$.

For each generation step i:

- Let $C_i = [(c_{i,1}, p_{i,1}), (c_{i,2}, p_{i,2}), \ldots, (c_{i,N}, p_{i,N})]$ be the top-N candidate tokens and their probabilities (sorted by p). The selected token is $t_i = c_{i,1}$.
- Let W denote the full set of model parameters. For tractability, define an influence metric $I(w; i)$ that scores how much parameter $w \in W$ contributed to the logit differences at step i.
- Define $S_i = \{w \mid I(w; i) \geq \tau\}$, the set of parameters whose influence exceeds threshold τ. In practice, we store compressed identifiers (layer, head, neuron index) and a numeric influence score.

A **token-weight trajectory** for step i is the tuple (t_i, C_i, S_i, M_i) where M_i are metafields (context window references, relative position, attention sparsity, etc.). A full trajectory for a run is

$$TWT = [(t_1, C_1, S_1, M_1), \ldots, (t_T, C_T, S_T, M_T)].$$

5 Instrumentation and Influence Metrics

The TWTI pipeline relies on two instrumentation streams that are common or easily added to transformer implementations:

1. **Token Logits / Top-N Extraction:** Capture the model's pre-softmax logits (or post-softmax probabilities) for the top-N candidates at each generation step.
2. **Activation / Influence Snapshots:** Capture compact sub-summaries of which parameters/activations influenced the logits. We do not store full parameter matrices. Instead, compute *influence scores* at runtime using lightweight metrics:
    - **Activation-product score:** For linear layers where output logit components are computed by dot-products, compute the product of input activation and weight magnitude per neuron and sum across the components that map to the candidate logits. Normalize per-layer.
    - **Attention-contribution score:** For attention heads, compute head-specific contribution to the output distribution using the magnitude of query-key-value interactions projected to logits.
    - **Layer ablation delta (cheap proxy):** Temporarily mask or scale a layer/head by a small amount and estimate the delta in candidate logits.
    - **Gradient-proxy (static):** Compute the dot product of the logit change with the parameter's last-updated gradient vector as a proxy for influence.

All influence computations should be configurable by budget (e.g., sample every k tokens, select top-k layers per step).

6 Analysis Program Specification

```
# Input: TWT raw logs (per-step C_i, raw influence candidates with scores)
# Config: N (top tokens), tau (influence threshold), K (cluster size)

for each step i in run:
    selected_token = C_i[0]
    candidates = C_i[0:N]

    S_i = select_top_by_score(raw_influences_i, threshold=tau, max_items=K)
    labeled_S_i = map_ids_to_labels(S_i)
    explanation_i = []

    if semantic_tag_present(labeled_S_i):
        explanation_i.append("Semantic pattern: " + dominant_semantic_tag(labeled_S_i))

    if attention_heads_present(labeled_S_i):
        explanation_i.append("Attention heads: " + list_attention_heads(labeled_S_i))

    alt_influence_table = compute_candidate_influence_table(candidates, labeled_S_i)

    trace_entry = {
        'step': i,
        'selected_token': selected_token,
        'candidates': candidates,
        'top_influences': labeled_S_i,
        'alt_influence_table': alt_influence_table,
        'human_template': render_human_template(selected_token, candidates, labeled_S_i,
    }

    append trace_entry to structured_trace

clusters = cluster_trace_entries(structured_trace)
produce final report (structured_trace, clusters)
```

7 Concrete Examples (Toy)

7.1 Correct Fact Example

Context: "The Eiffel Tower is located in __."

Top-5 candidates: Paris (0.72), France (0.18), Europe (0.05), the (0.03), Germany (0.02)

Top influences: Layer 12 Head 3 (landmark–city association), Layer 14 Head 7 (geolocation phrase attention), Neuron 4312 (proper-noun selection bias)

Program output: Step 15: Selected "Paris" (72%) over ["France" (18%), "Europe" (5%), ...]. Dominant signals: Layer 12 Head 3 produced strong landmark→city activation; Layer 14 Head 7 reinforced geolocation context. The model favored a specific city token over broader geography due to higher specific semantic match.

7.2 Hallucination Example

Context: "The asteroid Hermes landed in 1985 near the city of __."

Top-5 candidates: Springfield (0.45), New York (0.20), Geneva (0.15), N/A (0.12), Paris (0.08)

Top influences: Layer 9 Head 2 (fictional-entity association), Layer 10 Head 5 (temporal co-occurrence bias), Neuron 2187 (popular-city bias)

Program output: Step 22: Selected "Springfield" (45%). Influences: Layer 9 indicates a pattern linking fictional named asteroids with common fiction-locality names; Layer 10 shows temporal co-occurrence boosting locally frequent city tokens. This pattern suggests a hallucination driven by distributional co-occurrence rather than factual grounding.

7.3 Bias Example

Context: "The new hire previously worked at __."

Top-5 candidates: Google (0.65), Microsoft (0.10), Startup (0.08), Consultancy (0.07), University (0.05)

Top influences: Layer 11 Head 2 (prestige-company association; 0.50), Neuron 500 (big-company prior; 0.20)

Program output: Step 9: Selected "Google" (65%) over alternatives. Recorded influences show a strong prestige-company activation pattern: Layer 11 Head 2 amplifies signals that associate candidate quality with large well-known employers; Neuron 500 contributes a prior favoring big-company affiliations.

8 Limitations and Ethical Notes

TWTI is a diagnostic and interpretive framework rather than a causal model analysis. While it provides structured, human-readable explanations of model behavior, these explanations should not be mistaken for direct evidence of internal reasoning.

- **Approximation and non-uniqueness:** Influence metrics are approximations; multiple internal trajectories can yield similar outputs.
- **Scale and cost:** Instrumentation and storage overhead grow with model size; efficient sampling and compression may be required.
- **Interpretation risk:** Readable traces increase transparency but may be overinterpreted as literal cognitive steps.
- **Ethical handling:** Instrumented traces may expose sensitive model information; they should be redacted or access-controlled in applied settings.

Overall, TWTI is intended as a practical interpretability tool to aid debugging and bias analysis, not as definitive causal attribution of internal model processes.

9 Conclusion

TWTI offers a practical, auditable, and flexible approach to reconstruct token-level decision trajectories in LLMs. By combining lightweight instrumentation, a deterministic analysis program, and optional model-assisted summarization, the method gives engineers and researchers actionable explanations for token choices, hallucinations, and biased outputs. It is designed for real-world debugging and compliance workflows and bridges the gap between theoretical interpretability and deployable tools.

A Appendix A: Example Deterministic Templates

- Factual choice template: "Step {i}: Selected '{t}' ({p}%) over {alts}. Dominant signals: {signals}."
- Hallucination flag template: "Step {i}: Possible hallucination — selected '{t}' ({p}%). Top influences point to distributional co-occurrence rather than verified factual evidence."
- Bias flag template: "Step {i}: Bias signal — selected '{t}' ({p}%). Influences show consistent association between group {G} and role {R}."

B Appendix B: Pseudocode for Candidate Influence Table

```
# Input: candidates [(token, prob)...], influences [(param_id, score)...], param_to_logi
# Output: table mapping candidate -> estimated influence fraction
for each candidate in candidates:
    est_influence[candidate] = 0
    for each (param, score) in influences:
        contribution = map_param_to_candidate_logit_contribution(param, candidate)
        est_influence[candidate] += score * contribution
normalize est_influence to percentages
return est_influence
```