Tokengeist: Multi-Turn Attribution Tracing in Agentic Conversations

Anonymous ACL submission

Abstract

When a language model produces a response in a multi-turn conversation, which tokens from prior turns shaped that answer—and how did those dependencies propagate across prior turns? Existing context attribution methods process the full context in a single pass, recovering surface-level dependencies but missing the layered, non-linear structure of real-world dialogues and multi-step reasoning tasks. We introduce *multi-turn context attribution* (MTCA): given a target span in a model response, the task of tracing attribution backward across turns to identify not only which prior turns were directly relevant, but also how those turns themselves depended on earlier context. We propose TOKENGEIST, an attribution-method-agnostic and scalable framework that recovers full dependency paths by casting attribution as a recursive traversal of a directed acyclic graph (DAG) over conversation turns. We will release MT-CABENCH, a benchmark of 3,845 target spans across 665 multi-turn conversations, annotated with gold provenance graphs reaching depths of up to 14, across four dependency types. Across four open-weight models, flat attribution methods fail to recover multi-hop dependencies, achieving under 20% source recall, while TOKENGEIST reaches 90%. Our results reveal systematic failure modes of single-pass attribution—which we term *provenance collapse*—and motivate attribution methods that reason recursively across turns.

1 Introduction

Large language models (LLMs) are increasingly deployed in settings where a single query is not an isolated event but rather a step in a longer sequence of interactions. Multi-turn dialogues, tool-calling agents, and retrieval-augmented pipelines all produce contexts in which a model's final answer may depend on information introduced several turns prior—possibly after that information has been summarized, reformatted, or transformed by intermediate assistant turns. This raises a fundamental question for interpretability and trust: *given a specific claim in a model's response, what chain of prior context explains its provenance?* This question is especially important in agentic conversations, where tools are automatically invoked for the user, sometimes in high-value scenarios such as finance, shopping, and enterprise workflows. Identifying the provenance of a response can help debug such workflows, allowing either an engineer or agentic loop to make corrections.

Prior work has studied context attribution in single-turn settings. Methods include attention-based heuristics (Cohen-Wang et al., 2025), gradient-based approaches (Yin and Neubig, 2022), learned surrogate models (Wang et al., 2025a; Cohen-Wang et al., 2024), and prompting the model to explain its own reasoning (Tahaei et al., 2024; Laban et al., 2024). However, these methods typically treat the available context as a *flat* sequence: a prompt or retrieved context followed by a response. When applied to multi-turn conversations, they can assign importance scores to earlier tokens or spans, but they do not explicitly recover the *layered* structure of dialogue, where an assistant turn may itself be a derived artifact that condenses, filters, or transforms earlier context.

We argue this gap is consequential. Liu et al. (2024) and Laban et al. (2026) show that LLMs do not uniformly attend to information across long contexts. We hypothesize that attribution in multi-turn settings has a *non-linear, multi-hop structure*: the answer to turn N may depend on a claim in turn $N-2$, which in turn depended on a tool output in turn 1. A flat attribution method looking at the full context window cannot recover this structure; it can only identify which turns *appear* relevant to the final answer, not how relevance was *propagated* forward through the conversation.

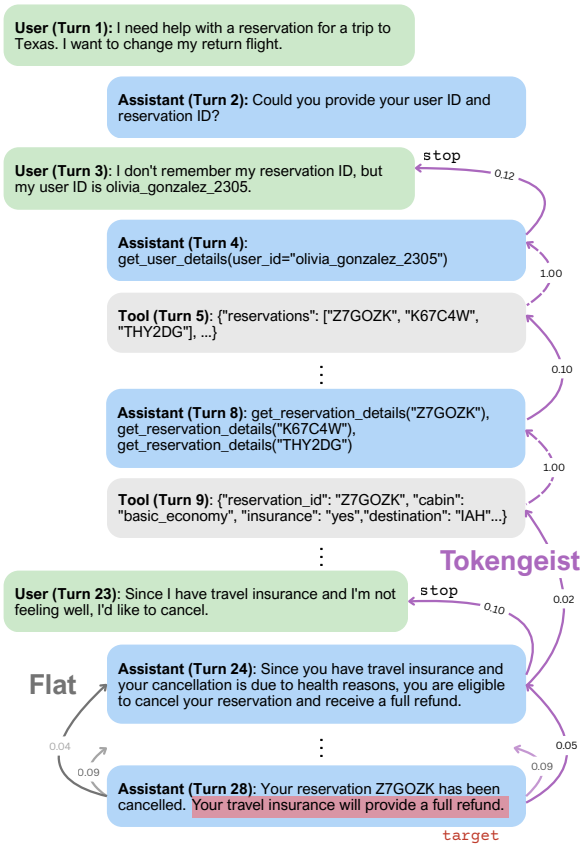

Figure 1: **Recursive provenance tracing example.** Right-hand arrows (purple) show TOKENGEIST's recursive trace, stopping at user-authored turns (T23, T3). Left-hand arrows (gray) show the flat attribution trace.

Motivating example. Figure 1 illustrates this challenge on a TauBench airline conversation. The agent's claim *"your travel insurance will provide a full refund"* (T28) traces back through an intermediate eligibility determination (T24) that combined two earlier sources: the user's stated health reason (T23) and an insurance field in a tool result (T9) fetched via a lookup chain seeded by the user ID eight turns prior (T3). Flat attribution applied to T28 assigns its highest score to T27—the cancellation tool result, which confirms the booking was cancelled but says nothing about insurance eligibility—and misses the sources T23 and T3.

Multi-Turn Context Attribution (MTCA). We formalize this problem as MTCA. Given a target span in the final model response, the goal is to recover its full provenance: which prior turns or context does it depend on, and through what chain of intermediate steps did that information reach the response? Unlike single-turn attribution, MTCA must account for the layered structure of dialogues.

Tokengeist. We propose TOKENGEIST, a framework for MTCA that represents provenance as a DAG over conversation turns and is agnostic to the choice of base attribution method. Starting from the target span, TOKENGEIST applies a single-turn attribution method to identify the most relevant prior turns, then recursively repeats this step for each candidate intermediate turn until it reaches original user inputs or system prompts.

MTCABench. To evaluate MTCA, we introduce MTCABENCH, a benchmark of 3,845 annotated target spans across 665 multi-turn conversations (688 from ConFETTI (Alkhouli et al., 2025), 3,157 from TauBench (Yao et al., 2025)) with gold provenance graphs. Conversations include tool-calling traces and cases that stress-test known LLM failure modes including hallucination and distraction from irrelevant turns. Annotation follows a structured protocol described in Appendix C.

Contributions.

1. We introduce **multi-turn context attribution** (MTCA), a new task for recovering the full cross-turn provenance of a model response, rather than only single-pass token attributions.
2. We propose **TOKENGEIST**, a framework that formalizes MTCA as DAG construction via recursive attribution and provides flexible path extraction over the resulting graph.
3. We will release **MTCABENCH**, a benchmark of 3,845 annotated target spans across 665 multi-turn conversations with gold provenance graphs for evaluating attribution methods on realistic conversations.
4. We present empirical findings showing systematic failures of flat attribution methods and evaluate base attribution methods under TOKENGEIST.

2 Related Work

Single-turn context attribution. A growing body of work identifies which parts of an input influence a model's generated output. Early token- or feature-level methods include contrastive explanations, which identify input tokens that explain why a model preferred one prediction over another (Yin and Neubig, 2022), and perturbation-based approaches such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017),

which estimate feature importance from input perturbations or feature coalitions.

More recent work targets content-grounded generation, connecting generated statements to evidence in the provided context. ContextCite (Cohen-Wang et al., 2024) learns a surrogate model that predicts how a generation changes when context sources are included or excluded, while AT2 (Cohen-Wang et al., 2025) avoids expensive ablations by combining attention-head signals under ablation-derived supervision. Long-context traceback methods such as TracLLM (Wang et al., 2025b) and AttnTrace (Wang et al., 2025a) identify sentences, passages, or paragraphs in long inputs via informed search, contribution-score denoising, or attention-based traceback.

Several benchmarks evaluate evidence grounding: Li et al. (2024b) show that automatic attribution evaluation remains difficult; Li et al. (2024a) target knowledge-aware generation over structured sources; and Hirsch et al. (2025) localize user-selected generated spans to source spans. A related line evaluates attributed generation and citation quality (Gao et al., 2023; Yu et al., 2026). These works typically treat the context as a flat input and return relevant tokens, spans, passages, documents, or citations for a single generation. In contrast, MTCA asks for the recursive provenance structure of a multi-turn conversation: not only which event influenced the target span, but which *earlier* events influenced that event itself.

Multi-turn and long-context understanding. Liu et al. (2024) and Laban et al. (2026) document that LLMs do not uniformly use information distributed across long contexts. Bai et al. (2024) and Kwan et al. (2024) evaluate multi-turn conversational abilities, and recent tool-use benchmarks (Alkhouli et al., 2025; Yao et al., 2025; Patil et al., 2025) move evaluation closer to realistic interaction histories where actions depend on prior turns, tool calls, and tool results. However, they primarily score task success, response quality, or tool-call correctness, and do not ask whether a method can explain a generated span by recovering the multi-hop provenance graph.

Graph-structured provenance. Our formulation represents provenance as a DAG whose edges point from a later event to the earlier event it depends on, connecting MTCA to workflow provenance and program-dependence analysis, which capture how entities are derived from prior entities and activities (Acar et al., 2010) or make data and control dependencies explicit in program executions (Ferrante et al., 1987). Recent NLP work has begun to study attribution over structured artifacts; Suri et al. (2025), for example, ground model responses in flowchart components via graph-based reasoning. MTCA differs in that the graph is not given to the model as an input artifact but must be reconstructed from a conversation trace whose intermediate nodes may themselves be model-generated. This makes recursive tracing central, since a method may identify a nearby assistant summary while missing the original source from which it was derived—a failure mode we formalize as *provenance collapse*.

3 Problem Formulation

Notation. A *conversation* $\mathcal{C}$ is a sequence of turns $T_1, T_2, \ldots, T_N$, where each turn consists of a *role* $r_i \in \{\textsc{System}, \textsc{User}, \textsc{Assistant}, \textsc{Tool}\}$ and *content* c_i, a sequence of tokens. We refer to turns with $r_i \in \{\textsc{System}, \textsc{User}\}$ as *exogenous turns*, since their content is externally supplied. Turns with $r_i = \textsc{Assistant}$ are *endogenous turns*, generated by the model. Turns with $r_i = \textsc{Tool}$ are *tool-result turns*: their content is produced by an external tool executor rather than the model, so they carry no model-generated attention weights and are handled separately during attribution (Section 4).

T_T denotes the *target turn*: any assistant turn in $\mathcal{C}$ whose content we aim to explain. A *target span* is a contiguous sub-sequence $\bar{y} = (y_a, \ldots, y_b)$ of T_T's content, with $1 \leq a \leq b \leq |c_T|$.

3.1 Single-Turn Attribution

Attribution operates across three granularities that serve distinct roles.

Token level is where attribution is computed. Let $\mathcal{A}$ denote a *base attribution method*: given a target span $\bar{y}$ (a contiguous slice of output tokens) and a context X (the preceding token sequence), $\mathcal{A}$ produces for every source token $x_j \in X$ a score quantifying how much x_j influenced $\bar{y}$. Concrete instantiations are described in Section 4.4.

Sentence level is the scoring unit. Rather than reporting one score per token, we segment X into sentences $\mathrm{seg}(X) = (s_1, \ldots, s_M)$; each sentence is scored as an atomic unit over its token index

range, yielding a sentence-level score vector

$$\mathbf{a} = \mathcal{A}(\bar{y},\, s_1, \ldots, s_M) \in \mathbb{R}^M_{\geq 0}. \tag{1}$$

Scoring sentence ranges rather than individual tokens reduces noise from attention spikes on punctuation or common tokens.

Turn level is the provenance unit. When X concatenates multiple turns, each s_i belongs to exactly one turn (tracked via character offsets). Turn aggregation reduces sentence scores to one score per turn:

$$\hat{a}_k = \max_{i\,:\,s_i \in T_k} a_i, \tag{2}$$

yielding $\hat{a}_k \geq 0$ per preceding T_k, reflecting the strongest sentence-level signal within that turn.

In **flat attribution**, $\mathcal{A}$ is applied once with X equal to all turns preceding the target, then turn aggregation (Eq. 2) yields one score per turn. This is the flat attribution baseline we compare against.

3.2 Multi-Turn Context Attribution (MTCA)

Given $\mathcal{C}$, $\bar{y}$, and $\mathcal{A}$, the goal of MTCA is to recover a *provenance graph* $G = (V, E, w)$: a weighted DAG over conversation turns in which T_T is the unique root, each edge $(T_i, T_j) \in E$ denotes that T_i depends on T_j with $\mathrm{idx}(j) < \mathrm{idx}(i)$, and leaf nodes are exogenous turns. The DAG accommodates multi-source dependencies, where a single turn draws on multiple independent earlier turns.

4 Tokengeist

4.1 Recursive attribution

TOKENGEIST (Figure 2; Appendix E) builds G by backward breadth-first search (BFS) from T_T, using $\bar{y}$ as the initial attribution target. At each frontier node v representing turn T_t, the tracer: (i) concatenates turns $T_0, \ldots, T_{t-1}$ into a context string X_t; (ii) calls $\mathcal{A}(c_t,\, \mathrm{seg}(X_t))$ to obtain sentence-level scores over X_t, where c_t is the token span being explained: $\bar{y}$ at depth 0, or the full token sequence of turn T_t at deeper steps; (iii) applies turn aggregation (Eq. 2) to convert sentence scores to one score per preceding turn; (iv) discards turns with aggregated score $< \theta$ (default $\theta = 0$) and selects the top k turns (default $k = 3$).

The role of each selected turn determines how recursion proceeds:

- **Exogenous turn** is added as a leaf node; recursion stops.
- **Tool-result turn** carries no model-generated attention weights. TOKENGEIST instead follows a deterministic *structural edge* (score 1.0) to the nearest preceding assistant turn that issued the invocation, then resumes recursive attribution from that invoking turn. This encodes the causal chain: **target** $\rightarrow$ **tool result** $\xrightarrow{\textit{struct.}}$ **tool invocation** $\rightarrow$ **argument sources**.
- **Assistant turn** is added as a node and recurse, attributing its full content against the earlier context.

Recursion also terminates once the maximum depth $d_{\max}$ is reached (default $d_{\max} = 8$).

4.2 Design choices

Turn-level DAG nodes. Sentence-level scores from $\mathcal{A}$ identify *which turns to follow*, while DAG nodes are the turns themselves: the attribution computation operates at sentence granularity to find the signal, but finding the responsible turn is the operationally useful answer. Turn-level nodes also make recursion well-defined: we recurse on an endogenous turn's *full* content. Since the model generates each turn in a single forward pass, it is natural to treat the turn as the atomic unit of provenance. Span F1 (Section 6.2) measures the precision of within-turn span localization.

DAG deduplication. The tracer maintains a visited set of turn indices. When a turn already in G is reached via a second attribution path—the structural pattern of multi-source dependencies—a convergent edge is added to the existing node rather than duplicating it, ensuring G is a DAG.

Path extraction. After DAG construction, we apply *relative-threshold extraction* to prune G before evaluation. At each node v, let $s^* = \max_u s(v, u)$ be the maximum parent edge score. Given $\alpha \in [0, 1]$, edge (v, u) is retained iff

$$s(v, u) \;\geq\; \alpha \cdot s^*. \tag{3}$$

Unlike an absolute score threshold, this adapts to the local score distribution at each node, which is important because attribution scores are not globally calibrated: they vary across different parts of the graph and across base attributors. Setting $\alpha = 1.0$ reduces to a greedy single chain; lower values admit additional branches when a lower-ranked parent receives comparable attribution to the top candidate. We set $\alpha = 0.85$; performance is insensitive to this choice within a flat plateau ($\alpha \in [0.65, 0.85]$, $\Delta\mathrm{F1} < 0.01$, Appendix B.2).

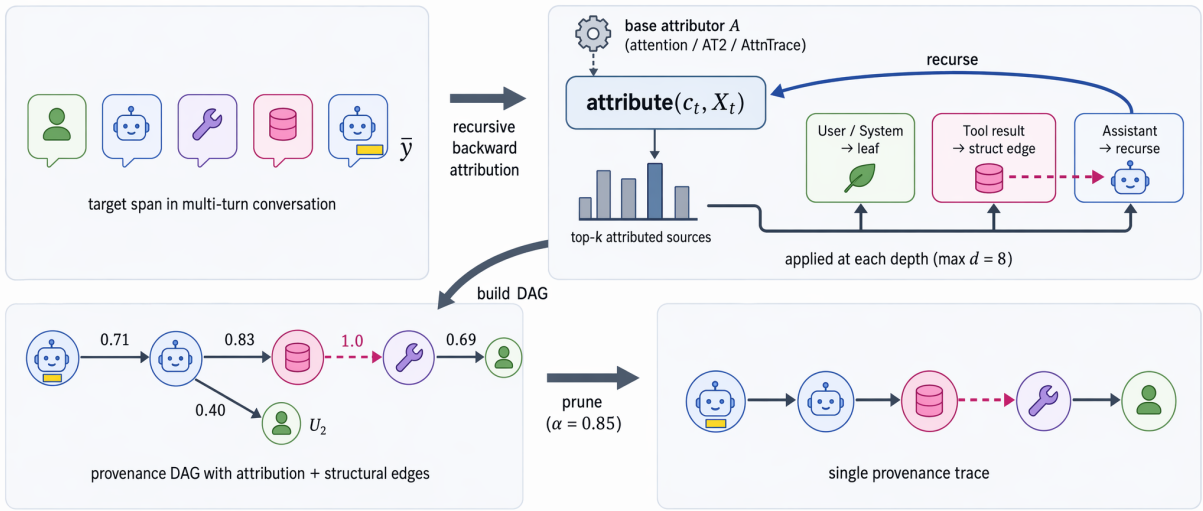

Figure 2: **Overview of TOKENGEIST.** Starting from a target span $\bar{y}$ in a multi-turn conversation, a base attributor $\mathcal{A}$ is applied recursively to select the top-k relevant prior turns at each step. Role-aware dispatch determines what happens next: user / system turns become exogenous leaves, tool-result turns follow a deterministic structural edge (dashed) to their invoking tool-call turn, and assistant turns are recursed into. The resulting provenance DAG is pruned by relative-threshold extraction ($\alpha = 0.85$) to produce the final provenance graph.

Complexity. TOKENGEIST makes $O(k^{d_{\max}})$ attribution calls in the worst case; in practice the frontier shrinks rapidly as recursion terminates at exogenous turns. Timings are in Appendix A.

4.3 Relation to Flat Attribution

Flat attribution is the special case of TOKENGEIST with $d_{\max} = 1$: it applies $\mathcal{A}$ once to $\bar{y}$ over the full context, without recursion. This fails to recover intermediate dependencies when the turns most directly attributed by the final step are themselves endogenous turns that summarize or restate earlier information. We formalize this failure mode as **provenance collapse**: the true exogenous source receives low attribution because the model never attended to it directly in the final turn, having already incorporated its content into an intermediate endogenous turn. TOKENGEIST addresses provenance collapse by recursing into each endogenous turn rather than treating it as terminal.

4.4 Attribution Backends

TOKENGEIST is agnostic to the choice of $\mathcal{A}$; we evaluate three instantiations. **Avg. Attention** averages attention weights uniformly across all layers and heads (no training). **AT2** (Cohen-Wang et al., 2025) replaces uniform averaging with a pretrained linear probe over attention features. **AttnTrace** (Wang et al., 2025a) estimates attribution scores via bootstrap subsampling of forward passes. Full implementation details and hyperparameters are in Appendix A.1.

	ConFETTI	TauBench
Conversations / simulations	109	556
Annotation targets	688	3,157
Avg. targets per conversation	6.3	5.7
Avg. turns per conversation	16.8	27.3
Avg. provenance depth	3.4	6.0
Max. provenance depth	12	14
Targets with tool-call path (%)	96.1	96.2
Domain families	10	2

Table 1: **MTCABENCH dataset overview.** Provenance depth is the longest root-to-leaf path in the gold DAG.

5 MTCABench

We introduce MTCABENCH, a benchmark for evaluating multi-turn context attribution. Each instance contains a multi-turn conversation, a target span in an assistant response, and a gold provenance graph identifying the earlier turns the target span draws from. The benchmark tests whether attribution methods can recover both direct evidence and layered dependencies through intermediate assistant turns, tool calls, and tool results.

5.1 Dataset Details

MTCABENCH draws conversations from two agentic function-calling benchmarks, ConFETTI (Alkhouli et al., 2025) and TauBench (Yao et al., 2025) (Table 1).

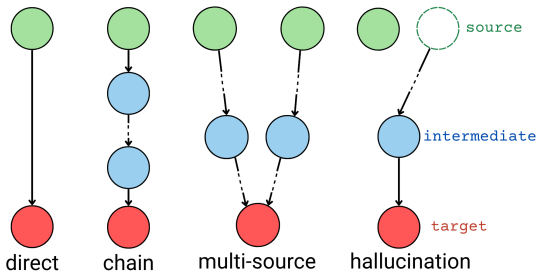

Figure 3: **Dependency types in MTCABENCH.** Dashed edges omit intermediate hops for clarity.

Domain coverage. ConFETTI conversations span 10 task families identified by semantic clustering of API call signatures: Accommodation & Car Rental, Vacation Planning, Business Travel & HR, Medical Services, Insurance & Benefits, Lifestyle & Local, Telecom & Social, HR Operations, Banking, and Productivity. We cluster `all-MiniLM-L6-v2` embeddings with k-means and select k by silhouette score. TauBench covers two domains, airline and retail; airline overlaps with ConFETTI's travel clusters. Together, MTCABENCH covers 11 distinct real-world task families across both splits.

Dependency types. Illustrated in Figure 3, each target carries a base hop-depth tag—*direct* (single hop from source to target) or *chained* (multi-hop through intermediate turns)—and may additionally be tagged *multi-source* (two or more independent source turns) or *hallucination* (model output factually incorrect; gold provenance traces where the wrong value came from; Appendix D). Per-tag counts and distributions are in Appendix C.

5.2 Annotation Protocol

We describe below the two-phase annotation pipeline applied to every conversation in MTCABENCH, followed by human quality review; full prompts are in Appendix C.2.

Target identification. We select 5–8 target spans per conversation from assistant turns, excluding user and tool-result messages since attribution targets must be model-generated. We use `GPT-5.4` to do this selection, prioritizing spans that are specific and factual—an ID, date, amount, or paraphrase of a tool result—and traceable to a prior turn through well-defined provenance. `GPT-5.4` is prompted to select a mix of dependency types and provenance depths, including hallucination cases. The final per-tag distribution is in Appendix C.

Provenance annotation. For each target span, the annotation model is prompted to produce a provenance graph tracing all dependencies back to exogenous source nodes, without a depth limit. Tool-call chains follow a strict three-node structure: *tool_result* $\rightarrow$ *invocation* $\rightarrow$ *argument source*. We use `GPT-5.4` to annotate and assign dependency-type tags, with up to three retries per target to ensure quality. If annotation fails after three attempts, we escalate to `GPT-5.4-Pro`.

Human review. A preliminary spot-check by one author of 100 stratified targets (50 per split; 2.6% of the benchmark) found a 98% acceptance rate before correction and 100% after re-labeling two wrong cases; we plan independent multi-annotator review prior to benchmark release. Full methodology and results are in Appendix C.5.

6 Evaluation

Our experiments evaluate the central hypothesis of the paper: flat attribution can identify locally relevant context, but fails to recover layered provenance paths in multi-turn conversations. We therefore compare flat attribution methods against recursive variants produced by TOKENGEIST, using the same underlying attribution scorer.

6.1 Experimental setup

We evaluate on four open-weight instruction-tuned models: `Llama-3.1-8B-Instruct` and `Qwen2.5-7B-Instruct` (full results in Table 2); `Phi-4-Mini-Instruct` and `DeepSeek-R1-Distill-Qwen-7B` (Appendix A.2). GPT-family models are used only for dataset construction (Section 5.2), not attribution. Since AT2 requires a pretrained linear probe per model architecture, model selection is constrained by publicly available AT2 probes (Cohen-Wang et al., 2025). For each of the three base scorers introduced in Section 4.4 (Avg. Attention, AT2, AttnTrace), we compare its flat application against its TOKENGEIST counterpart, isolating the contribution of recursion.

6.2 Metrics

We evaluate each predicted provenance graph $\hat{G}$ against a gold provenance DAG G^*. Because MTCA represents dependencies as directed edges between conversation events, our primary structural metric is edge-level F1.

Edge F1. We compute precision, recall, and F1 over directed dependency edges. An edge (u, v) is correct if both $\hat{G}$ and G^* include v as a provenance source of u. Edge F1 measures whether a method recovers the local dependency structure of the provenance graph.

Node F1. We compute F1 over non-target provenance nodes. Node F1 asks whether the method found the correct supporting turns, regardless of connection. This helps distinguish evidence-retrieval failures from graph-structure failures.

Source recall. We compute recall over gold leaf-source nodes. A leaf source is an exogenous node—an originating source with no further dependencies. This metric measures whether a method reaches the original source of the information rather than stopping at an intermediate turn.

Span F1. We compute word-level F1 between the attributed text spans stored in predicted provenance nodes and the gold span annotations. For each gold turn with a span annotation, we compute word-overlap F1; turns absent from the predicted graph contribute F1 $= 0$. We then macro-average across all gold-annotated turns. Span F1 is strictly harder than Node F1, since both the correct supporting turn (*coverage*) and sub-turn excerpt (*granularity*) must be correctly identified.

6.3 Main comparison: flat versus recursive attribution

We compare flat attribution with recursive attribution across all target spans in MTCABENCH. Across all four models, recursive attribution substantially outperforms its flat counterpart in every metric across both datasets (Table 2). Across the Avg. Attention and AT2 backends, TOKENGEIST achieves a 20–40 percentage-point gain in Edge F1 over flat methods; gains for ATTNTRACE are smaller, reflecting its weaker flat baseline (see Limitations). On TauBench, source recall jumps from below 20% to over 90%—an indication of the *provenance collapse* formalized in Section 4.

Three implications follow. *(i) Recursion, not the base attributor, is the dominant factor.* The gains hold across all four models and all three base scorers—including the weakest, ATTNTRACE—confirming that TOKENGEIST is a drop-in upgrade for any base attribution scorer, and we expect future improvements in base scorers to compose with it directly. *(ii) Source recall crosses a usability threshold.* Verification, audit, and hallucination grounding require reaching the originating user or tool event, not the nearest assistant turn—a bar flat methods miss but TOKENGEIST clears. *(iii) Learned probes scale better with agentic context.* The relative advantage of AT2 over raw attention increases with context depth: results are mixed on ConFETTI, while AT2 leads on TauBench for three of four models (Table 2 and Table 3), suggesting learned probes generalize better with longer tool-call chains.

6.4 Performance by dependency type

Next, we stratify performance by dependency type, directly testing whether gains from recursion are concentrated in the cases that motivate MTCA: chained and multi-source dependencies. Figure 4 shows on direct (single-hop) dependencies, flat methods perform well because no intermediate turn needs to be traversed. On chained dependencies, flat methods drop to under 35% Edge F1 while TOKENGEIST reaches 70%. Multi-source targets are the hardest category: flat attention averages 23% Edge F1 while TOKENGEIST reaches 52%; flat methods frequently return only one of the two contributing branches.

On hallucination targets, TOKENGEIST can recover both the followed intent and the missed correction, while flat attribution recovers neither (Appendix D). By following tool result $\xrightarrow{\textit{struct.}}$ invocation, TOKENGEIST runs attribution from the invocation's context—the moment the wrong decision was made, when the overridden correction was still a live competing signal. Flat attribution on the final turn operates after the fact, attending primarily to the tool result confirming the completed action. Flat attribution is therefore structurally inadequate for multi-hop, multi-source, and hallucination provenance.

Depth-stratified signal-coverage analysis in Appendix B.4 shows flat methods collapse to near-zero coverage after depth 2, while TOKENGEIST sustains coverage above 75% to depth 6 and more, accounting for most of the Edge F1 gap, since 87.9% of ConFETTI and 96.5% of TauBench targets are chained (depth ≥ 2).

6.5 Ablations

Appendix B reports ablations over recursion depth $d_{\max}$, branching factor k, pruning threshold α, and source granularity, supporting the defaults

Method	*ConFETTI* (n=495)				*TauBench* (n=3,157)			
	Edge F1	Node F1	Src. Rec.	Span F1	Edge F1	Node F1	Src. Rec.	Span F1
Llama-3.1-8B-Instruct								
Flat Attention	40.2	45.0	36.2	14.0	19.2	23.1	14.7	3.7
Flat AT2	39.3	44.8	36.8	14.4	16.1	20.8	16.9	3.9
Flat AttnTrace	25.7	49.7	54.8	19.3	10.0	17.8	28.3	4.8
TG + Attention	**72.5**	**81.0**	**92.7**	**49.7**	**51.4**	**66.9**	**90.7**	**31.9**
TG + AT2	69.4	78.9	92.6	48.3	39.0	58.9	89.7	29.3
TG + AttnTrace	46.9	68.3	92.7	38.9	15.2	27.1	74.4	10.6
Qwen2.5-7B-Instruct								
Flat Attention	40.4	44.9	35.0	14.2	20.0	23.0	12.0	3.3
Flat AT2	40.1	44.8	45.8	14.6	18.3	22.4	18.2	4.0
Flat AttnTrace	25.0	48.4	55.0	19.0	10.0	17.8	28.3	4.8
TG + Attention	**73.9**	**82.4**	**94.7**	**50.6**	53.8	69.8	89.8	33.9
TG + AT2	72.5	81.0	92.5	49.2	**61.0**	**70.3**	**90.9**	**38.1**
TG + AttnTrace	46.3	67.0	92.7	38.1	14.7	25.8	68.8	9.3

Table 2: **Flat versus recursive attribution on MTCABENCH.** All scores are percentages. TG = TOKENGEIST. Bold marks the best recursive method per metric per model. All results are mean across 3 independent runs; AT2 and Avg. Attention are deterministic; AttnTrace standard deviation is $\lesssim$1.1 points and never changes method rankings (Appendix A). Results for Phi-4-Mini-Instruct and DeepSeek-R1-Distill-Qwen-7B follow the same pattern and appear in Appendix A.2 (Table 3).

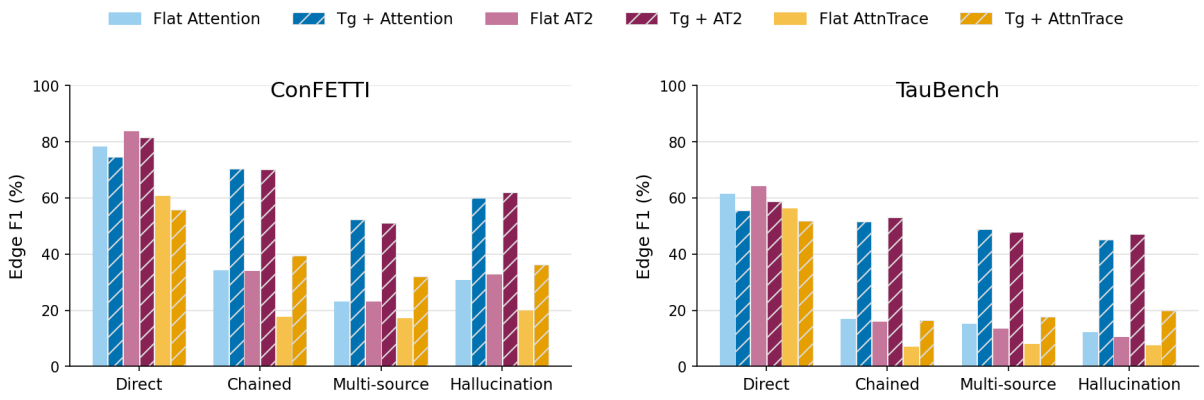

Figure 4: **Edge F1 by dependency type** on ConFETTI (left) and TauBench (right). Light bars: flat methods; dark bars: TG (= TOKENGEIST).

($d_{\max}$=8, k=3, α=0.85, sentence-level segmentation) used throughout the main results.

7 Conclusion

We introduced *multi-turn context attribution* (MTCA), which recasts provenance recovery as a recursive traversal over a DAG of conversation events, and instantiated it as TOKENGEIST, an attribution-method-agnostic framework we will release alongside MTCABENCH, a benchmark of 3,845 annotated target spans across 665 multi-turn conversations. Across four open-weight models, TOKENGEIST raises source recall from under 20% to over 90%. It roughly doubles edge F1 over flat baselines, exposing *provenance collapse* as a systematic failure of single-pass attribution.

Looking ahead, we see MTCA as a foundation rather than an endpoint. The recursive structure invites tighter coupling with the agent loop itself: provenance graphs could be consumed online by planners and self-correction mechanisms, allowing agents to detect when a claim rests on a shaky intermediate turn and re-query or repair before acting. Scaling MTCA to longer horizons—persistent memory, cross-session histories, and multi-agent transcripts—will likely require attribution backends that amortize cost across overlapping subgraphs and prune more aggressively than the depth-bounded BFS we study here.

Limitations

We outline the main limitations of TOKENGEIST and MTCABENCH.

Quality is bounded by the base attribution method. TOKENGEIST composes multiple calls to a single-turn attribution scorer $\mathcal{A}$ and inherits the strengths and weaknesses of that scorer. This is most visible for AttnTrace, which performs poorly even in its flat form (e.g., 5.4–16.7 edge F1 on MTCABENCH); TG+AttnTrace correspondingly underperforms TG+Attention and TG+AT2 by a wide margin, especially on TauBench where its flat attribution scores are weakest. That said, our recursive solution still improves over the flat baseline in every model$\times$backend cell we report (Table 2), so the benefit of recursive tracing is robust to the choice of backend even when the absolute numbers are not. Better single-turn attribution methods, when they appear, should compose with TOKENGEIST without modification and lift the ceiling further.

Attention-based attribution is correlational, not causal. All three backends we evaluate score sentences by how strongly the generating model attends to or weights them, not by counterfactual ablation. A high attribution score therefore indicates that a turn was *used* in the model's computation, not that removing it would have changed the answer. Counterfactual or perturbation-based scorers (Cohen-Wang et al., 2024; Wang et al., 2025b) could in principle be plugged in as $\mathcal{A}$, but we leave that integration to future work.

Turn-level granularity. DAG nodes in TOKENGEIST are conversation turns rather than spans. This choice is motivated in Section 4, but it caps within-turn localization: a method that selects the correct turn but returns its full content is penalized by Span F1, which is consistently the lowest of our four metrics. Span- or sentence-level recursion is a natural extension but requires resolving what it means to recurse on a sub-turn fragment that was generated as part of a single autoregressive pass.

Computational cost. Recursive attribution costs $O(k^{d_{\max}})$ calls to $\mathcal{A}$ in the worst case. For attention-aggregation backends each call is a single forward pass, but for AttnTrace, which adds bootstrap subsampling, the cost compounds. Practitioners running TOKENGEIST on very long agentic transcripts or in latency-sensitive settings will need to tune k, $d_{\max}$, and the pruning threshold α accordingly.

Coverage of MTCABENCH. MTCABENCH is built from two English-language sources—ConFETTI (Alkhouli et al., 2025) and TauBench (Yao et al., 2025) (airline and retail domains)—and is text-only. It does not cover multilingual dialogue, multimodal inputs, very long persistent-memory sessions, or multi-agent transcripts, and its dependency-type distribution reflects the editorial choices of those source datasets. Generalization of our results to other domains, languages, and modalities is plausible but unverified. Hallucination targets are a small fraction of the benchmark (59 ConFETTI, 35 TauBench; 8.6% and 1.1% of each split); hallucination-specific findings should be interpreted with that sample-size caveat in mind.

Model-assisted annotation and reproducibility of attribution backends. Gold provenance DAGs are produced by a two-phase pipeline that uses GPT-5.4 as the sole annotator (Appendix C.2); our evaluation therefore measures agreement with this model-assisted annotation standard rather than with a human-only gold standard. For attribution backends, AT2 and AVG. ATTENTION are deterministic up to float16 GPU non-determinism ($\lesssim 0.5$ point variance, never affecting method rankings), while ATTNTRACE is stochastic by design and we follow the upstream implementation in not fixing the bootstrap seed; fixing the seed would not eliminate the GPU non-determinism shared with the other backends.

Model scope. Since our backends require access to model internals (attention weights or hidden states), we evaluate on four open-weight 7–8B instruction-tuned models. The specific models are further constrained by AT2 probe availability: AT2 requires a pretrained linear probe per model architecture, and we use the four architectures for which probes are publicly released (Cohen-Wang et al., 2025). We do not report results on much larger open-weight models, closed-weight API models, or base (non-instruction-tuned) checkpoints, and we cannot rule out that provenance collapse manifests differently at other scales or after different post-training procedures.

Attribution model need not match the generating model. TOKENGEIST applies attribu-

tion post-hoc to recorded conversations, so the model performing attribution may differ from whichever model originally produced the assistant turns. Post-hoc attribution asks what the *attribution* model attends to when re-reading a fixed conversation, not what the generating model attended to during generation—these are distinct questions, and TOKENGEIST answers the former. The gold DAGs annotate semantic information dependencies in the conversation text, which are properties of the content rather than of any particular model's internals; our empirical results confirm that attention signals track these dependencies reliably in the post-hoc setting. Attribution quality may differ when the attribution model and the generating model are identical, and we leave that comparison to future work.

Ground-truth provenance is itself a constructed annotation. Gold provenance DAGs are produced via the structured annotation protocol described in Appendix C with model-assisted labeling and human review. While we plan to report inter-annotator agreement, there is often no oracle "true" provenance for a conversation: our metrics therefore measure agreement with a carefully constructed annotation standard rather than with the model's internal causal graph.

Hyperparameter sensitivity. The defaults reported in the main text ($k = 3$, $d_{\max} = 8$, $\alpha = 0.85$, $\theta = 0$) were selected on a held-out slice of CONFETTI and reused across all models and datasets. We report ablations over each in Section 6, but performance under distribution shift to new conversation styles may benefit from re-tuning.

Impact Statement

This paper is about understanding where a model's answer comes from in a multi-turn conversation. In many real deployments, a response is not produced from a single prompt alone: it may depend on user instructions, earlier assistant summaries, or tool outputs spread across many turns. When that information is copied, condensed, or transformed along the way, it becomes difficult to tell what the final answer is actually grounded in. Our work aims to make this flow of information easier to inspect.

The main benefit of this work is transparency. A method for multi-turn attribution can help developers and researchers see whether a particular claim is grounded in an original source, an intermediate assistant message, a tool result, or a spurious piece of context. This can be useful for debugging long-context systems, analyzing agent traces, building better benchmarks, and studying when attribution methods stop too early at a nearby summary instead of tracing back to the true source. More broadly, we hope this encourages systems to preserve better provenance information across conversations and tool-use workflows.

At the same time, attribution is not a substitute for correctness or safety. A recovered provenance path can help explain what context appears to have influenced a response, but it does not prove that the response is true, complete, or safe to act on. There is also a risk that users may over-trust attribution outputs, especially when the underlying attribution scores are noisy or incomplete. Because multi-turn traces may include private user messages, retrieved documents, or tool outputs, any practical use of this kind of method should include appropriate privacy, access-control, and data-retention safeguards.

Finally, MTCABench reflects the domains and annotation choices used to build it. It should be treated as a useful starting point for studying multi-turn provenance, not as a complete map of all possible conversational settings. Overall, we see this work as a step toward more inspectable model interactions: it helps show how information moves through a conversation, but it should be used alongside careful evaluation, privacy review, and domain-specific safety testing.

References

Umut Acar, Peter Buneman, James Cheney, Jan Van den Bussche, Natalia Kwasnikowska, and Stijn Vansummeren. 2010. A graph model of data and workflow provenance. In *2nd Workshop on the Theory and Practice of Provenance (TaPP '10)*.

Sharad Agarwal, Pooria Namyar, Alec Wolman, Rahul Ambavat, Ankur Gupta, and Qizheng Zhang. 2026. Switchcraft: Ai model router for agentic tool calling. *arXiv preprint arXiv:2605.07112*.

Tamer Alkhouli, Katerina Margatina, James Gung, Raphael Shu, Claudia Zaghi, Monica Sunkara, and Yi Zhang. 2025. CONFETTI: Conversational function-calling evaluation through turn-level interactions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7993–8006, Vienna, Austria. Association for Computational Linguistics.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. 2024. MT-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7421–7454, Bangkok, Thailand. Association for Computational Linguistics.

Benjamin Cohen-Wang, Yung-Sung Chuang, and Aleksander Madry. 2025. Learning to attribute with attention. *arXiv preprint arXiv:2504.13752*.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. 2024. ContextCite: Attributing model generation to context. In *Advances in Neural Information Processing Systems*, volume 37, pages 95764–95807.

Jeanne Ferrante, Karl J Ottenstein, and Joe D Warren. 1987. The program dependence graph and its use in optimization. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 9(3):319–349.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488, Singapore. Association for Computational Linguistics.

Eran Hirsch, Aviv Slobodkin, David Wan, Elias Stengel-Eskin, Mohit Bansal, and Ido Dagan. 2025. LAQuer: Localized attribution queries in content-grounded generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15355–15370, Vienna, Austria. Association for Computational Linguistics.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2024. MT-eval: A multi-turn capabilities evaluation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20153–20177, Miami, Florida, USA. Association for Computational Linguistics.

Philippe Laban, Alexander Fabbri, Caiming Xiong, and Chien-Sheng Wu. 2024. Summary of a haystack: A challenge to long-context LLMs and RAG systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9885–9903, Miami, Florida, USA. Association for Computational Linguistics.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2026. LLMs get lost in multi-turn conversation. In *The Fourteenth International Conference on Learning Representations*.

Xinze Li, Yixin Cao, Liangming Pan, Yubo Ma, and Aixin Sun. 2024a. Towards verifiable generation: A benchmark for knowledge-aware language model attribution. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 493–516, Bangkok, Thailand. Association for Computational Linguistics.

Yifei Li, Xiang Yue, Zeyi Liao, and Huan Sun. 2024b. AttributionBench: How hard is automatic attribution evaluation? In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14919–14935, Bangkok, Thailand. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A. Rossi, Vivek Gupta, and Dinesh Manocha. 2025. Follow the flow: Fine-grained flowchart attribution with neurosymbolic agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 22485–22508, Suzhou, China. Association for Computational Linguistics.

Marzieh Tahaei, Aref Jafari, Ahmad Rashid, David Alfonso-Hermelo, Khalil Bibi, Yimeng Wu, Ali Ghodsi, Boxing Chen, and Mehdi Rezagholizadeh. 2024. Efficient citer: Tuning large language models for enhanced answer quality and verification. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4443–4450, Mexico City, Mexico. Association for Computational Linguistics.

Yanting Wang, Runpeng Geng, Ying Chen, and Jinyuan Jia. 2025a. AttnTrace: Contextual attribution of prompt injection and knowledge corruption. *arXiv preprint arXiv:2508.03793*.

Yanting Wang, Wei Zou, Runpeng Geng, and Jinyuan Jia. 2025b. TracLLM: A generic framework for attributing long context LLMs. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3845–3864. USENIX Association.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2025. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*.

Kayo Yin and Graham Neubig. 2022. Interpreting language models with contrastive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 184–198.

Yue Yu, Ting Bai, Hengzhi Lan, Li Qian, Li Peng, Jie Wu, Wei Liu, Jian Luan, and Chuan Shi. 2026. C^2-Cite: Contextual-aware citation generation for attributed large language models. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, pages 871–880.

A Experiment Details

A.1 Implementation Details

Software stack. TOKENGEIST is implemented in Python 3.10 on top of HuggingFace `transformers` 4.51.3, `accelerate` 1.1.1, and PyTorch 2.10 (CUDA 13.0 runtime, cuDNN 9.2). Sentence segmentation uses NLTK `punkt_tab`. All experiments use the same code path for flat and recursive attribution: a flat run is exactly the special case $d_{\max}=1$ of the recursive tracer, so the only difference between paired (flat, recursive) numbers in Table 2 is the recursion depth.

Models. We evaluate on four open-weight instruction-tuned causal language models, loaded in `float16` on a single GPU per process: `Qwen/Qwen2.5-7B-Instruct`, `meta-llama/Llama-3.1-8B-Instruct`, `microsoft/phi-4-mini-instruct`, and `deepseek-ai/DeepSeek-R1-Distill-Qwen-7B`. All models are used as released, with no fine-tuning. For all backends we set `output_hidden_states=True` on the attribution forward pass so that the same hidden states can be reused across the attention extraction step and downstream attribution scoring.

For ConFETTI, conversation history is presented to the model in the flat `Role: content` format used by the underlying tracer; for TauBench, where the upstream agent was driven through the model's native chat template, we set `use_chat_template=True` so that the attribution forward pass sees inputs in the same format as the original generation, including tool-call delimiters. Tool-call names and JSON arguments are serialized into the textual content of the issuing assistant turn so that attention from later turns can resolve to function names and argument values.

Attribution backends. We use three single-turn attribution backends, all of which expose the same interface and can be slotted into TOKENGEIST without changes to the recursive tracer:

- **Avg. Attention.** Native attention scores averaged uniformly across all transformer layers and attention heads, then summed over each candidate source span and averaged over the target tokens (Appendix A.3).
- **AT2** (Cohen-Wang et al., 2025). A pretrained linear probe over attention features, loaded from the public `madrylab/at2-*` HuggingFace probes corresponding to each evaluated model. No additional training is performed.
- **AttnTrace** (Wang et al., 2025a). Bootstrap-style attention attribution with $B=30$ subsampled forward passes, subsampling fraction $q=0.4$, and top-$k_{\text{attn}}=5$ token aggregation per segment. These are the original AttnTrace defaults.

The same backend is used for the corresponding flat baseline and TG+backend rows in Tables 2 and 3, isolating the effect of recursion.

TOKENGEIST hyperparameters. Unless stated otherwise, we use the following configuration for every method and benchmark: maximum recursion depth $d_{\max}=8$ (flat: $d_{\max}=1$); per-step branching factor $k=3$; score threshold $\theta=0$ (no pre-filtering of low-score candidates before top-k); source segmentation granularity `sentence`; DAG node granularity `turn` (per-turn aggregation by max over span scores); tool-result $\rightarrow$ tool-invocation structural edges with fixed score 1.0; and relative-threshold path extraction with $\alpha=0.85$ applied uniformly to all predicted graphs before scoring (Section B.2). **Span F1** (Section 6.2) is computed on the same DAGs without re-tracing: per-turn node text is tokenized by

whitespace and matched against the gold span annotation.

Annotation pipeline. Ground-truth provenance DAGs are produced by the two-phase pipeline described in Section 5.2 using `GPT-5.4` for both target identification (Phase 1) and provenance annotation (Phase 2). Generation uses the deployment's default sampling parameters with JSON-mode output, retried up to three times on transient errors. If annotation still fails after three retries, the target is escalated to `GPT-5.4-Pro`. The labeling pipeline is fully separate from the attribution pipeline: GPT-family models are never used as attribution models or scorers.

Reproducibility. All results in the paper are averaged over 3 independent runs. The two AT2 backends (flat and recursive) are deterministic given the same model checkpoint and inputs. AVG. ATTENTION is deterministic up to float16 GPU non-determinism (we observe $\lesssim 0.2$ point variance on individual metrics across re-runs on the same hardware, which never changes method rankings). AttnTrace is stochastic by design (bootstrap sampling); we use NumPy's default RNG without a fixed seed, which mirrors the original implementation. Across all 4 models and both flat and recursive AttnTrace variants, measured run-to-run standard deviation on TauBench is $\lesssim 0.6$ points on all four reported metrics. On ConFETTI, Edge F1, Node F1, and Span F1 are within 0.7 points; Src. Rec. standard deviation reaches up to 1.1 points (largest cases: Qwen2.5-7B flat and Phi-4-Mini recursive), driven by the bootstrap resampling interacting with the sparser source-recall signal on this benchmark. In no case does run-to-run variance change method rankings. Code, model checkpoints, label files, and per-target predicted graphs will be released with the camera-ready version.

A.2 Additional Model Results

Table 3 reports the same flat-versus-recursive comparison as Table 2 on the remaining two evaluated models, `Phi-4-Mini-Instruct` and `DeepSeek-R1-Distill-Qwen-7B`. The same headline patterns observed in the main table also hold here: recursive TOKENGEIST substantially outperforms every flat baseline, source recall on TauBench crosses the 20%-to-80–90% threshold, and AT2 leads AVG. ATTENTION on TauBench on both models. TG+AttnTrace remains the weakest recursive variant on both models, consistent with the backend-dependence discussion in the Limitations.

A.3 Attention Extraction Details

The AVG. ATTENTION backend reads attention weights directly from the attribution forward pass rather than calling `model(..., output_attentions=True)`, which would force the slower `eager` attention path and require materialising the full $O(L^2)$ matrix. We instead capture per-layer hidden states via `output_hidden_states=True`, then re-project queries and keys at attribution time using the model's own Q/K projection layers, rotary embeddings (`apply_rotary_pos_emb`), and GQA key-expansion (`repeat_kv`), restricting the attention slice to the target token range $[t_{\text{start}}, t_{\text{end}})$. We then average uniformly over all L transformer layers and all H attention heads:

$$\bar{A}_{ij} = \frac{1}{L \cdot H} \sum_{\ell=1}^{L} \sum_{h=1}^{H} A_{ij}^{(\ell,h)}, \tag{4}$$

where $A^{(\ell,h)} \in \mathbb{R}^{|\bar{y}| \times |X|}$ is the post-softmax attention from target tokens to context tokens at layer ℓ and head h. The per-source score is then $s_k = \frac{1}{|\bar{y}|} \sum_{i \in \bar{y}} \sum_{j \in \text{src}_k} \bar{A}_{ij}$, with span ranges src_k taken from the sentence-level segmentation of the context. Scores are non-negative because individual $A_{ij}^{(\ell,h)}$ are post-softmax.

We support three dispatch paths (`llama`, `qwen2`, `phi3`) through a small dispatch layer that looks up the right `apply_rotary_pos_emb` / `repeat_kv` helpers per architecture; `DeepSeek-R1-Distill-Qwen-7B` routes to the `qwen2` path automatically via name matching, since it shares the Qwen2 attention architecture. We use uniform layer-and-head averaging rather than the popular early-layer-only or last-layer-only heuristics; the AT2 probe is included precisely to compare against a learned reweighting of the same per-head features.

Tool-result handling. Tool-result turns contain no model-generated tokens, so their attention weights are not informative. When recursion reaches a tool-result turn, TOKENGEIST short-circuits the attribution call: it adds a deterministic structural edge with score 1.0 to the nearest preceding assistant turn (the issuer of the call), then resumes normal attribution from that assistant turn. This encodes the causal chain *target* $\rightarrow$ *tool result*

Method	*ConFETTI* (n=495)				*TauBench* (n=3,157)			
	Edge F1	Node F1	Src. Rec.	Span F1	Edge F1	Node F1	Src. Rec.	Span F1
Phi-4-Mini-Instruct								
Flat Attention	39.8	44.8	40.5	14.0	18.1	20.9	12.2	3.2
Flat AT2	40.0	45.4	38.5	14.4	17.6	22.1	15.7	3.8
Flat AttnTrace	16.8	31.0	43.2	10.2	5.4	10.6	22.7	2.5
TG + Attention	70.8	78.9	**91.8**	48.9	51.4	67.4	**89.1**	34.8
TG + AT2	**72.1**	**80.7**	91.1	**48.9**	**60.7**	**70.8**	82.1	**38.9**
TG + AttnTrace	29.2	43.6	77.2	24.5	5.8	12.1	34.1	3.2
DeepSeek-R1-Distill-Qwen-7B								
Flat Attention	36.9	41.4	28.4	12.3	15.6	18.5	8.7	2.3
Flat AT2	39.7	45.3	36.3	14.9	17.7	21.4	12.9	3.4
Flat AttnTrace	23.2	38.0	41.7	12.3	8.5	12.8	9.4	1.6
TG + Attention	66.1	75.0	89.3	47.8	49.7	67.7	85.2	**34.8**
TG + AT2	**71.8**	**80.7**	**92.4**	**49.3**	**51.9**	**67.8**	**89.2**	33.7
TG + AttnTrace	43.9	62.1	90.3	36.1	33.7	52.4	74.7	26.6

Table 3: **Flat versus recursive attribution on MTCABENCH (additional models).** Same setup, metrics, and bolding convention as Table 2. All scores are percentages; mean across 3 independent runs.

$\xrightarrow{\text{struct.}}$ *tool invocation* $\rightarrow$ *argument sources* without giving structural edges spurious soft weight.

A.4 Computation Details

Hardware. All experiments run on two Ubuntu 22.04 servers, each equipped with 4× NVIDIA A100 80GB PCIe GPUs (8 GPUs in total; driver 580.159.03, CUDA toolkit 13.2). Each attribution process is pinned to a single GPU via `CUDA_VISIBLE_DEVICES` (a 7–8B-parameter model loaded in `float16` comfortably fits on one card), and conversations are sharded round-robin across the available GPUs by the runner scripts; each process loads its own model copy. We do not use `flash-attn`: the attribution forward pass relies on cached hidden states via `output_hidden_states=True`, which is incompatible with the fused flash-attention kernel, so all four models run with the `eager` attention implementation.

Per-target runtime. The asymptotic cost of TOKENGEIST is $O(k^{d_{\max}})$ attribution calls per target in the worst case, but the effective frontier shrinks rapidly because (a) recursion terminates at exogenous user/system turns, (b) tool-result $\rightarrow$ tool-invocation hops are zero-cost (structural edges, no model call), and (c) the visited set ensures each (turn, span) pair is attributed at most once per DAG. With k=3 and $d_{\max}$=8, observed DAG sizes on ConFETTI are 4–12 nodes (mean ≈7) and on TauBench 8–30 nodes (mean ≈18), reflecting the longer tool-call chains in agentic simulations.

Mean wall-clock attribution time per target (single A100): on ConFETTI, flat AVG. ATTENTION and AT2 each take ≈0.2 s (probe inference adds negligible overhead), and flat AttnTrace takes ≈3 s (B=30 forward passes dominate; range 2.7–3.5 s across models). Phi-4-mini is the fastest model evaluated (0.14 s flat AVG. ATTENTION) due to its smaller hidden dimension; the three 7–8B models are within 0.06 s of each other. Recursive variants take ≈0.6 s (AVG. ATTENTION, AT2) and ≈12 s (AttnTrace) on ConFETTI, and ≈1.2 s, ≈1.4 s, and ≈17 s respectively on TauBench (longer tool-call chains increase the effective DAG size). This corresponds to a 3–4× overhead over the flat baseline on both datasets; the frontier stays small because most recursion paths terminate quickly at exogenous turns and structural tool-result $\rightarrow$ tool-invocation hops incur no model call. Recursive AttnTrace is the most expensive method because it pays the B=30 bootstrap cost at every frontier node; recursive AVG. ATTENTION and AT2 are within 10% of each other in absolute wall time.

Total compute. The full set of results in Tables 2 and 3 requires four models × six methods × (495+3,157) targets = 87,648 attribution runs. On the hardware above this corresponds to approximately 117 GPU-hours of compute (≈9 h on ConFETTI, ≈108 h on TauBench; summed from per-target `timings` fields in output `meta.json` files), plus a one-time labeling cost in GPT-5.4 inference for the two-phase annotation pipeline (109 ConFETTI

conversations—79 used for evaluation, 30 held out for α selection—and 556 TauBench simulations). The 117 GPU-hour total reflects the main evaluation runs only. Additional attribution compute was incurred for the α sweep on the 30 held-out ConFETTI conversations (Section B.2) and the source granularity ablation on 10-conversation subsets per benchmark (Section B.5); the branching-factor ablation (Section B.3) added no further cost, as it re-analyzed stored graphs without additional attribution runs. No hyperparameter was tuned separately per method: α=0.85, sentence granularity, and k=3 are applied uniformly across all backends and benchmarks.

B Ablation Studies

All ablations use the hyperparameters described in Appendix A.1 unless stated otherwise.

B.1 Recursion Depth Ablation

We vary the maximum provenance depth $d_{\max}$ swept over $\{1, 2, 4, 8, 12\}$ for ConFETTI and $\{1, 2, 4, 8, 12, 16\}$ for TauBench, using Qwen2.5-7B-Instruct. $d_{\max}$ is measured in DAG hops from the target node, counting every edge including structural tool-result $\rightarrow$ tool-invocation hops, so it is directly comparable to the provenance depth of gold annotations (Table 1). $d_{\max}$=1 retains only the target's direct parents, equivalent to flat attribution.

Results are shown in Figure 5. On ConFETTI (average gold depth 3.4), AVG. ATTENTION and AT2 rise sharply from $d_{\max}$=1 to $d_{\max}$=4 and plateau thereafter. The plateau at $d_{\max}$=4 aligns directly with the dataset's average provenance depth: once the tracer can reach nodes four hops from the target, it has covered the majority of gold paths.

On TauBench (average gold depth 6.0), the curve saturates later, around $d_{\max}$=8, again tracking the deeper average gold structure. The two datasets thus show different plateau points that match their respective depth distributions, validating that the ablation reflects genuine coverage of the gold provenance rather than a computational ceiling.

ATTNTRACE follows the same shape but at lower absolute values on both datasets, consistent with its weaker base-scorer performance in the main results. The default $d_{\max}$=8 used in the main experiments corresponds to coverage well above the plateau on ConFETTI and at the plateau on TauBench, ensuring no gold paths are truncated.

B.2 Pruning Threshold Ablation

The pruning threshold (relative-threshold extraction) parameter $\alpha \in [0, 1]$ (Equation 3) controls how aggressively edges are pruned from the predicted provenance graph before scoring. We fix α=0.85 and verify on a held-out 30-conversation ConFETTI split (disjoint from the 79-conversation evaluation set). As shown in Figure 6, both AVG. ATTENTION and AT2 with α=0.85 lie within ΔF1 < 0.01 of the peak across the plateau $\alpha \in [0.65, 0.85]$. This confirms the choice is not sensitive to small perturbations, and we apply $\alpha = 0.85$ uniformly across all methods and both benchmarks.

B.3 Branching Factor Ablation

The per-step branching factor k controls how many candidate parent spans the recursive tracer follows at each hop. k=1 produces a greedy single chain (only the highest-scoring parent is recursed into at each step); k>1 allows multi-parent (DAG) recovery where multiple source spans jointly support a single intermediate. We sweep $k \in \{1, 2, 3\}$ post-hoc on the same stored graphs used in the main evaluation, applying relative-threshold extraction (α=0.85) before top-k selection so that results are directly comparable to Table 2.

Results for Qwen2.5-7B-Instruct on ConFETTI are shown in Figure 7. AVG. ATTENTION and AT2 each gain approximately +0.01 edge F1 from k=1 to k=2 (precision stays high; recall rises from 0.71 to 0.75 as multi-parent gold edges become recoverable), and plateau at k=3 (ΔF1<0.001). ATTNTRACE shows a slightly larger gain (+0.06 from k=1 to k=2) because its bootstrap-sampled scores are noisier and the single top-scored parent is less reliably the best one, but it also saturates by k=3. The plateau reflects that most gold DAG nodes have at most two high-confidence parents after α=0.85 pruning; a third branch virtually never passes the threshold.

For comparison, we also evaluate raw top-k extraction without alpha pre-filtering: F1 degrades monotonically with k (from 0.727 at k=1 to 0.500 at k=3 for AVG. ATTENTION) as predicted graphs become denser than the gold and precision collapses. This confirms that α is the precision-restoring mechanism; k governs recall coverage of multi-parent structure conditional on that pruning.

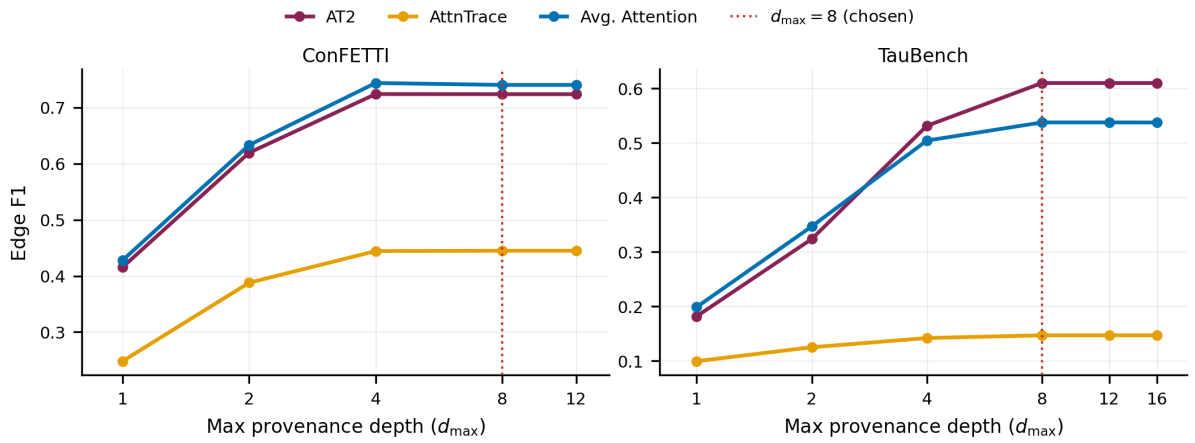

Figure 5: **Recursion depth ablation** (Qwen2.5-7B-Instruct; ConFETTI, left; TauBench, right). Edge F1 as a function of maximum provenance depth $d_{\max}$, measured in DAG hops (including structural tool-result $\rightarrow$ invocation edges). ConFETTI saturates at $d_{\max}$=4 (average gold depth 3.4); TauBench saturates at $d_{\max}$=8 (average gold depth 6.0). Plateau depth tracks the gold depth distribution on each dataset.

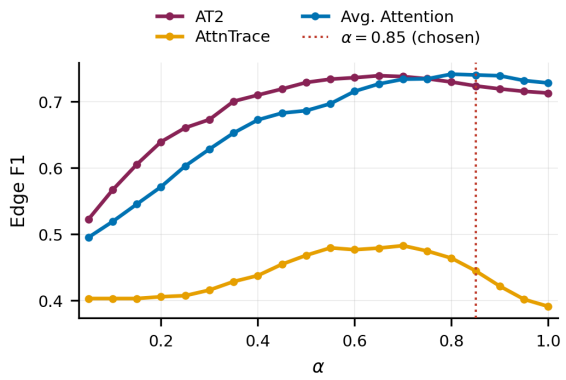

Figure 6: **Pruning threshold ablation** (Qwen2.5-7B-Instruct, held-out 30-conversation ConFETTI split). Edge F1 as a function of the relative-threshold extraction parameter α (Equation 3), which controls how aggressively low-scoring edges are pruned from the predicted provenance graph before evaluation. Dashed line marks the chosen value $\alpha = 0.85$ with ΔF1 < 0.01 across $\alpha \in [0.65, 0.85]$.

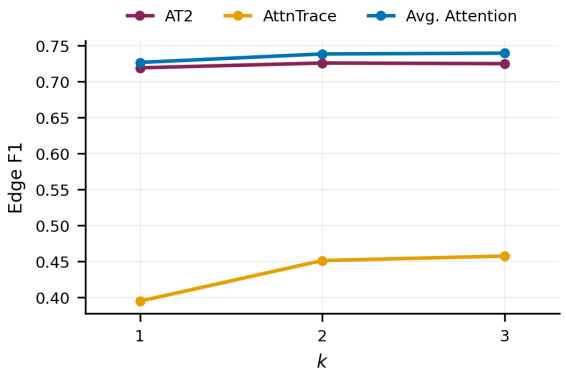

Figure 7: **Branching factor ablation** (Qwen2.5-7B-Instruct, ConFETTI, α=0.85 relative-threshold prefiltering). Edge F1 as a function of per-step branching factor $k \in \{1, 2, 3\}$. Results saturate at k=2; k=3 adds negligible further gain (ΔF1<0.001).

We adopt k=3 as the default: it recovers the full gain over the greedy chain baseline and leaves a conservative margin for rare 3-parent convergences, at negligible additional computation (the visited-set deduplication ensures each (turn, span) pair is attributed at most once per DAG regardless of k).

B.4 Attribution Signal vs. Provenance Depth

We measure the *signal coverage* of gold nodes (the fraction receiving any non-zero attribution score) as a function of (a) their provenance depth in the gold DAG and (b) their absolute turn distance from the target. Depth is computed by BFS from the target along GROUND_TRUTH_DEPS edges; invocation turns (assistant tool-call turns linked to their results via structural edges) appear at even depths 2, 4, ... These experiments use TauBench with Qwen2.5-7B-Instruct and all six methods.

Figure 8 (left) shows signal coverage by depth. All flat methods (dashed) cover roughly 70% of depth-1 gold nodes but drop below 20% at depth 2 and approach zero beyond depth 3. Recursive TOKENGEIST variants (solid) maintain signal coverage above 80% at odd depths 3, 5, 7 (exogenous turns: tool results, user messages) and above 50% at even depths (invocation turns). The even-depth

dip reflects that invocation turns are connected via zero-cost structural edges that carry no attribution signal; they are still reached by the tracer but their soft score is zero unless the recursive call above them also assigns a direct non-structural edge. TG+Attention and TG+AT2 reach ≈90% signal coverage at depth ≥6.

Figure 8 (right) shows signal coverage by turn distance. The advantage of recursion emerges from distance 3 onward: flat signal coverage halves by distance 2 and falls below 10% beyond distance 5, while recursive methods sustain coverage above 75% out to distance 15 and beyond.

B.5 Source Granularity Ablation

We compare two source segmentation granularities, *sentence-level* (default) and *word-level*, across all three attribution backends on 10 ConFETTI conversations (62 targets) and 10 TauBench simulations (60 targets) using Llama-3.1-8B-Instruct. Results after relative-threshold extraction ($\alpha = 0.85$) are shown in Figure 9.

The effect of granularity is method-dependent. For AVG. ATTENTION, sentence granularity outperforms word on the recursive variants across both datasets: on ConFETTI, edge F1 is +0.16 higher and source recall +0.13 higher; on TauBench, edge F1 is +0.13 higher. AT2 follows a similar pattern on ConFETTI (+0.11 edge F1, +0.15 source recall) but the advantage is smaller and mixed on TauBench. This is consistent with the well-documented tendency of attention weights to concentrate on punctuation and special tokens (Xiao et al., 2024): scoring sentence-length spans as atomic units averages out these spikes, whereas word-level segments are small enough that a single high-attention function word can dominate the score for that segment.

ATTNTRACE shows the opposite pattern: word granularity is consistently better, particularly on TauBench where recursive ATTNTRACE gains +0.07 edge F1 and +0.28 source recall with word segmentation. ATTNTRACE scores segments by bootstrap subsampling of context groups, so smaller word-level segments give the sampler finer-grained control over which content is included or excluded per iteration. However, word-level segmentation substantially increases runtime for ATTNTRACE: the number of bootstrap iterations scales with the number of segments, so word-level inputs require far more forward passes than sentence-level inputs, making it impractical at scale.

Since sentence granularity is optimal for both AVG. ATTENTION and AT2 on both benchmarks, and word-level segmentation is impractical for ATTNTRACE at scale, we adopt sentence-level granularity as the fixed default for all experiments.

C MTCABench Details

C.1 Dataset Statistics

Dependency type counts. Table 4 gives per-tag counts for both splits. Base tags (*direct*/*chained*) are mutually exclusive; additional tags can co-occur with either base tag.

	ConFETTI		TauBench	
Tag	Count	%	Count	%
Base tag				
direct	83	12.1	112	3.5
chained	605	87.9	3,045	96.5
Additional tag				
multi_source	162	23.5	1,653	52.4
hallucination	59	8.6	35	1.1

Table 4: **Dependency-type tag distribution in MTCABENCH.** Percentages are out of the total number of targets in each split: ConFETTI (n=688), TauBench (n=3,157). A single target may carry multiple additional tags but only one base tag. *multi-source* includes targets tagged *diamond* (4 / 14 targets in ConFETTI / TauBench), a convergent-path subtype where two independent chains merge at the same intermediate node.

Dependency type distribution. Figure 10 shows the percentage of targets carrying each dependency tag, for both splits.

Conversation length statistics. Table 5 reports conversation length for both splits. TauBench token counts are exact values from the `usage.prompt_tokens` field recorded during simulation; ConFETTI token counts are character-based estimates (≈chars/4). The longer contexts in TauBench reflect multi-step agentic tool-call chains, which also drive the higher average provenance depth (Table 1).

The 556 TauBench simulations cover two domains: airline (125 simulations, 22.5%) and retail (431 simulations, 77.5%).

Domain coverage. Table 6 lists the 10 ConFETTI task families identified by semantic clustering (Section C.3) and the 2 TauBench domains.

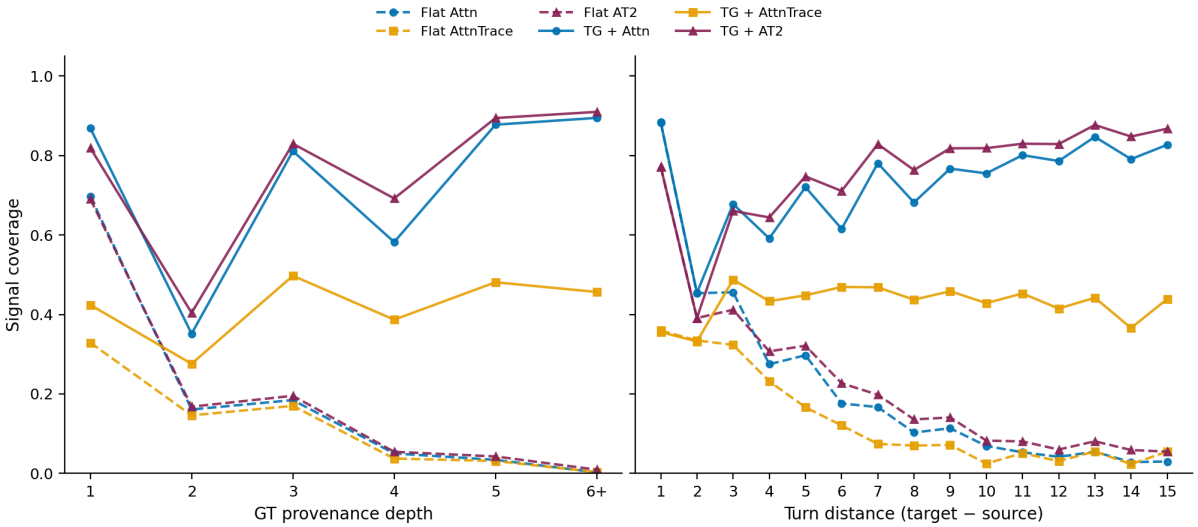

Figure 8: **Signal coverage of gold nodes vs. provenance depth (left) and turn distance (right)** (TauBench, Qwen-2.5-7B, n=26,478 gold-node records). Signal coverage = fraction of gold nodes whose turn receives any non-zero attribution score. Flat methods (dashed lines) collapse beyond depth 1 / distance 2; recursive TOKENGEIST variants (solid lines) sustain signal coverage above 75% at all depths and distances tested. Even-depth dips in recursive curves reflect invocation turns propagated via structural (zero-signal) edges.

	ConFETTI	**TauBench**
Turns per conv: mean	16.8	27.3
Turns per conv: range	9–35	14–55
Input tokens: mean	≈2,143	7,082
Input tokens: range	≈1,000–7,300	4,161–12,495

Table 5: **Conversation length statistics.** Input token counts for ConFETTI are character-based estimates; TauBench values are exact prompt-token counts from simulation logs.

Provenance depth distribution. Figure 11 shows the full depth distribution for both splits.

C.2 Annotation Prompts

Full prompts for the two-phase pipeline described in Section 5 are shown in Listings 1 and 2.

C.3 ConFETTI Dataset Corrections

The ConFETTI dataset (Alkhouli et al., 2025) provides 109 multi-turn conversations (506 function-calling turns) used in the ConFETTI split of MT-CABENCH. Prior to annotation, we applied input disambiguation corrections following a previously described LLM-assisted human-in-the-loop methodology (Agarwal et al., 2026).

86 turns (17%) referenced dates without specifying a year, while the gold labels contained hard-coded year values. Without the year present in the conversation, the correct date is unattributable to any conversational turn—the model must either rely on world knowledge or guess—making the attribution target ill-defined. We appended the missing year (or month and year) to the relevant user utterances so that the conversation unambiguously determines the expected function call. These changes do not alter gold labels.

C.4 Annotation Quality Recovery

43 TauBench simulations failed schema validation or did not complete in the initial two-phase run. The most common phase-2 failures were: omitting the target turn as the root node of the provenance DAG (the graph must include the turn being explained, not only its sources), and applying mutually exclusive base tags (*direct*+*chained*) to the same target. These simulations were re-annotated in a third pass using `GPT-5.4-Pro` with identical prompts; all 43 were recovered and included in the final benchmark.

C.5 Preliminary Human Spot-Check

We conducted a preliminary human spot-check to assess annotation quality. A stratified sample of 100 targets was drawn from the full benchmark (50 ConFETTI, 50 TauBench), balanced across four dependency types (20–29 examples per bucket in total). One author reviewed each example using a three-level judgment:

- **Correct**: the provenance DAG and dependency-type tags are accurate.

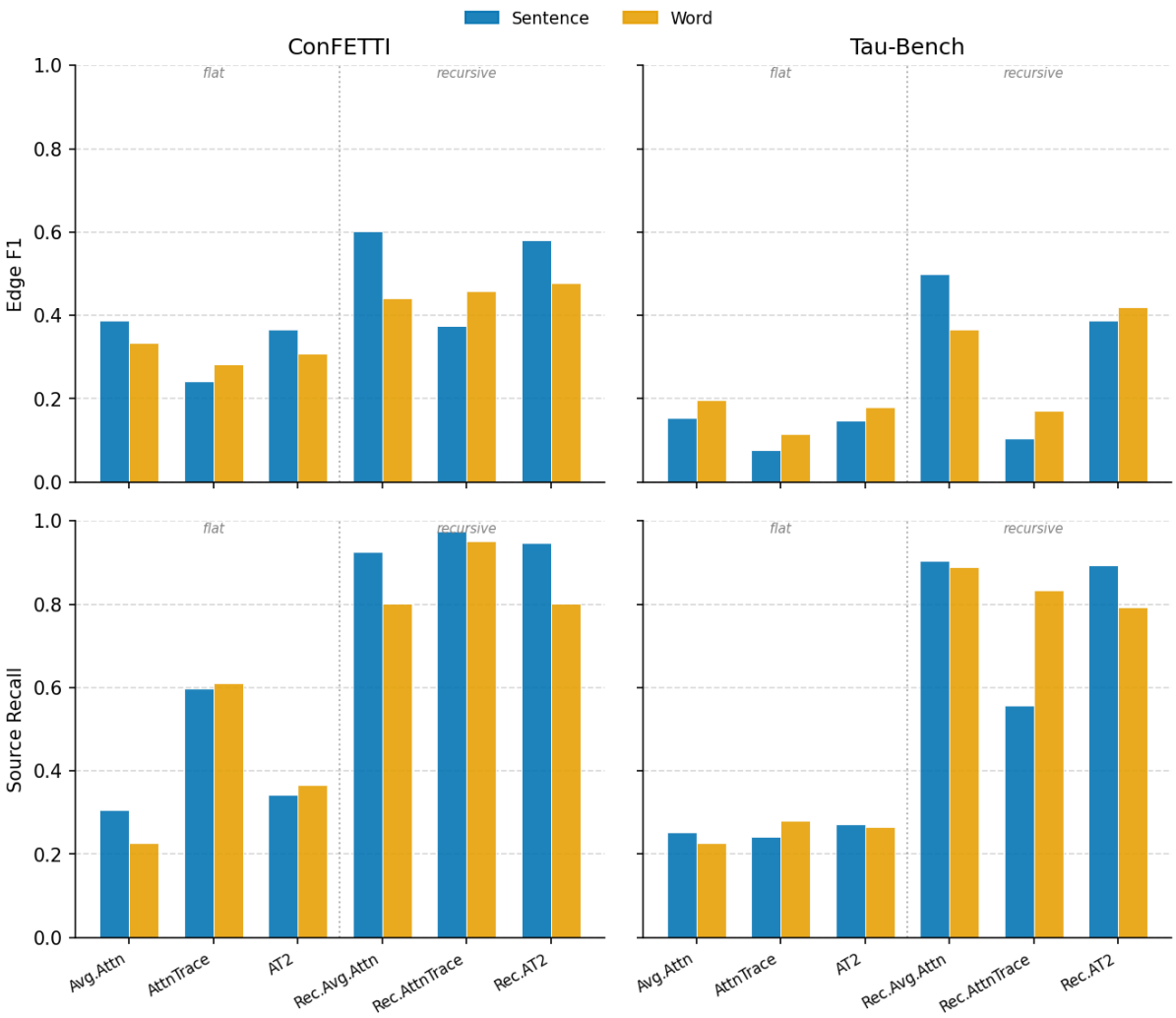

Figure 9: **Source granularity ablation** (Llama-3.1-8B-Instruct, 10-conversation subset). Edge F1 (top) and source recall (bottom) for sentence- vs. word-level segmentation across all backends on ConFETTI (left) and TauBench (right). Sentence is better for AVG. ATTENTION and AT2; word is better for ATTNTRACE due to its bootstrap segment-sampling design, but at much higher runtime cost.

- **Minor issues**: one missing or extra node, wrong span text, or borderline tag, but overall structure is right.
- **Wrong**: incorrect turns included, clearly wrong dependency type, or fundamentally broken chain.

Results. Review produced 87% correct, 11% with minor issues, and 2% wrong (2 examples). Both wrong cases involve chains that stop at an intermediate endogenous assistant turn rather than recursing to the true exogenous source—a violation of the hard stopping-condition constraint in the annotation protocol. These cases are being relabeled using the standard two-phase pipeline.

Table 7 summarizes acceptance rates by bucket. The *chained* and *chained+multi_source* buckets were error-free; *direct* and *hallucination* each contributed one failure. The *hallucination* bucket also had the highest rate of minor issues, reflecting the inherent ambiguity of annotating spans that the model *should* have attended to.

Limitations and ongoing work. These results are preliminary: the sample covers 100 targets (≈2.6% of the full benchmark), was reviewed by a single author, and used a subjective three-level judgment scale. They should be interpreted as an early indication of annotation quality rather than a definitive quality estimate.

We are actively extending this quality assessment with a second author reviewing the same sample independently, enabling inter-annotator agreement measurement and adjudication of dis-

Split	Domain	Conv.	Representative APIs
ConFETTI	Accommodation & Car Rental	18	BookHotel, CarRental
	Lifestyle & Local	16	RestaurantSearch, Shopping, NewsSearch
	Medical Services	15	DoctorsAppointment, Telehealth
	Insurance & Benefits	14	Insurance, DoctorsAppointment
	Business Travel & HR	12	BookFlight, HRTimeOff, HRCompanyDirectory
	Telecom & Social	11	Telecom, Discord
	Banking	7	Banking
	HR Operations	6	HRCompanyDirectory, HRPayrollBenefits
	Productivity	6	MapMyFitness, GMail, Slack
	Vacation Planning	4	BookFlight, CarRental, Weather
TauBench	Retail	431	order management, product search, returns
	Airline	125	flight booking, reservations, baggage

Table 6: **Task domain coverage in MTCABENCH.** ConFETTI domains are derived from k-means clustering (k=10) of `all-MiniLM-L6-v2` embeddings of API call signatures; TauBench domains are provided by the dataset. TauBench airline overlaps topically with ConFETTI travel clusters, giving 11 distinct task families in total.

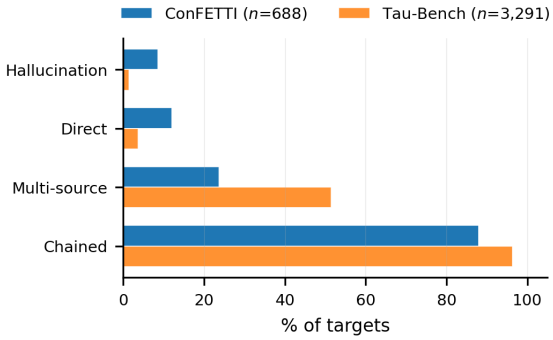

Figure 10: **Dependency-type tag distribution in MTCABENCH.** Bars show the percentage of targets in each split carrying each tag.

Bucket	n	Pre	Post
chained	27	100%	100%
chained+multi_source	29	100%	100%
direct	24	96%	100%
hallucination	20	95%	100%
Overall	100	98%	100%

Table 7: **Human spot-check acceptance rates.** Acceptance = correct + minor issues (i.e., not fundamentally wrong). Pre/post refer to before and after re-labeling two wrong cases.

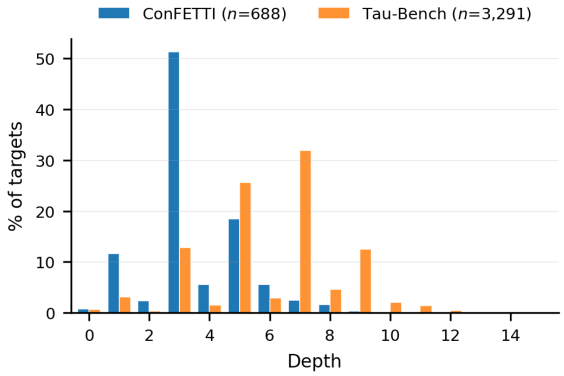

Figure 11: Ground-truth provenance depth distribution across the 688 ConFETTI targets and 3,157 TauBench targets in MTCABENCH. Depth is the maximum number of edges on any root-to-leaf path in the provenance graph. ConFETTI peaks at depth 3 (n=336), while TauBench peaks at depth 7 (n=1,008), reflecting the longer tool-call chains in agentic simulations. Both datasets span depths 1–14, covering simple single-hop and deep multi-hop attribution scenarios.

agreements.

D Hallucination Provenance Example

Hallucination targets carry a qualitatively distinct provenance structure: the DAG encodes *where the wrong value came from*. In multi-source hallucination cases, a graph that carries both the path the model took and the signal it overrode as leaf sources is most informative for understanding the model's failure mode.

Figure 12 illustrates this with a ConFETTI example. The user requests flight and hotel cancellations (Turn 0), then immediately corrects to hotel only—"not the flight"—in Turn 2. The agent cancels the flight anyway and reports success in Turn 9. The gold trace for "I have successfully cancelled your flight ticket" carries *two* leaf sources: Turn 0 (the intent the agent followed) and Turn 2 (the correction it missed). TOKENGEIST recovers both branches; flat attribution recovers neither.

The structural edge is key: TOKENGEIST follows tool result (Turn 8) $\xrightarrow{struct.}$ invocation (Turn 6), then runs attribution from the invocation's context. The invocation was generated at

Listing 1: **Phase 1: Target Identification Prompt.** Used to select 5–8 attribution target spans per conversation. {conversation} is replaced by the full serialised conversation.

```
You are selecting annotation targets for a multi-turn LLM conversation benchmark
on token attribution.

A good target span is:
- Specific text in an ASSISTANT turn that carries factual content (a value, date,
ID, location, amount, or a natural-language paraphrase of tool output). NEVER
target tool result turns or user turns, only assistant turns.
- Traceable back to at least one earlier turn through a clear dependency path.
- Varied across dep types: aim for a mix of direct (1-hop from source), chained
(multi-hop through tool calls or intermediary turns), and multi-source (target
requires information from two distinct originating turns)
- Look for hallucination targets - spans where the model used the wrong value,
traceable to a spurious prior turn rather than the correct source. For example: an
argument value copied from a previous tool call instead of read from the current
tool result. These are particularly valuable benchmark cases.

Aim for 5-8 targets. Spread them across different turns and different parts of the
conversation.
Prefer concrete specificity - an exact ID, date, or quoted value is better than a
vague phrase.

CONVERSATION: {conversation}
Return a JSON object with a single key "targets" containing the array, no other
text.
{
    "targets": [
        {
            "name": "<short snake_case identifier describing the target>",
            "target_turn_idx": <int, 0-based index of the turn containing the span>,
            "target_text": "<exact verbatim span copied from that turn>"
        }
    ]
}
```

the moment when Turn 2 was still a live competing signal, so its attention weights may encode Turn 2 even though the model ultimately overrode it. Flat attribution on Turn 9 operates after the fact and attends primarily to the tool result confirming the completed action, so the overridden correction does not surface.

E Tokengeist Procedure Walkthrough

Figure 13 visualizes the TOKENGEIST procedure formalized in Section 4 on a concrete five-turn tool-using example. It is intended as a reference for readers following the algorithm step-by-step; the body of the paper relies on the compact teaser in Figure 2.

Why this example. The five-turn refund conversation is constructed to exercise all three role-aware dispatch branches of the recursion in a single backward trace: a recurse-into-assistant hop ($T_5 \rightarrow T_4$), a tool-result structural hop ($T_3 \rightarrow T_2$, dashed magenta), and termination at an exogenous user leaf (T_1). Real MTCABENCH traces typically combine several such branches; the depth and branching statistics across the full benchmark are reported in Figure 11.

What the example does not exhibit. Two aspects of the algorithm are implicit in the figure but not visually exercised by this single-chain trace. First, **DAG deduplication**: when a turn is reached via a second attribution path, a convergent edge is added to the existing node rather than creating a duplicate (Section 4). Second, **multi-source branching**: the bar chart in each Stage 2 box collapses to a single dominant turn here, whereas multi-source targets retain the top $k = 3$ candidates and recurse into each.

Reading the figure alongside the algorithm. The Stage 2 legend, the role-aware dispatch arrows, and the stopping criteria all correspond one-to-one with the per-step procedure in Section 4 and its reference implementation in `RecursiveProvenanceTracer._trace_recursive`. The pruning rule shown in Stage 3 is the relative-

threshold extraction defined in Equation 3; the choice $\alpha = 0.85$ is justified by the sweep in Appendix A.

Figure 14 provides a complementary conceptual view, contrasting single-pass and recursive attribution at the level of context-window coverage.

Phase 2: Provenance Annotation Prompt (part 1 of 2).

```
You are annotating a multi-turn LLM conversation for a research benchmark on token
attribution. The goal is to build a ground-truth provenance graph that captures
which prior turns causally produced a specific target span - the kind of
provenance an attribution method should recover.
A conversation consists of alternating user and assistant turns, indexed from 0.
Tool call turns (where the assistant invokes a function) and tool result turns
(the returned JSON) are included and indexed.

DEFINITIONS
- Target: a specific text span within a specific turn whose provenance you are
tracing.
- Dependency: turn T depends on prior turn S if the specific content of S was
causally necessary to produce T. A turn is NOT a dependency if it is merely
contextually relevant - only include turns without which T would be wrong,
incomplete, or different.
- Source node (leaf): a turn with no further dependencies - typically an
originating user turn. Always included with depends_on: [].
- Intermediate node: a turn that both depends on earlier turns and is depended on
by later turns.
- Direct dependencies: Identify every prior turn that the target span directly
depends on.
- Recursive dependencies: For each direct dependency, identify what it directly
depends on. Continue recursively until you reach source nodes. Do not apply a
depth limit - trace all the way back to the originating user turn(s).

RULES
Tool calls:
- If the target depends on information returned by a tool, the dependency is the
tool result turn, not the invocation turn.
- A tool result turn's ONLY direct dependency is the invocation turn that
triggered it.
- An invocation turn's dependencies are the prior turns whose values were passed
as arguments.
- The correct structure is a strict chain: tool_result --> invocation --> [arg
sources]. Never connect a tool result directly to arg sources - that flattens the
chain incorrectly.
    CORRECT: "4": {{"depends_on": [3]}}, // tool result --> only its invocation
    "3": {{"depends_on": [2]}} // invocation --> arg source
    WRONG:
    "4": {{"depends_on": [3, 2]}}, // tool result must NOT list arg sources
    "3": {{"depends_on": [2]}}

Spans:
- Populate spans for every node in the graph - copy the exact text from that turn
that is causally necessary for the turn that depends on it.
- Span text must be copied exactly and contiguously from the source turn.
- spans encodes the edge meaning: for node X with depends_on [A, B], spans
contains one entry per parent showing what text from that parent X needed.

Graph structure:
- Each turn appears at most once in ground_truth_deps, keyed by its turn index.
- If the same turn is reachable via multiple paths, include it once. All turns
that depend on it list it in their own depends_on arrays.
- Source nodes are included with depends_on: [] and spans: {{}}.
```

Listing 2: **Phase 2: Provenance Annotation Prompt** (part 2 of 2). Used to trace the full provenance graph for each target back to exogenous source turns. `{conversation}` and `{targets_json}` are replaced at call time; `{conv_id}` is the conversation identifier.

```
DEPENDENCY TYPE TAGS
Every variant gets exactly one base tag (mutually exclusive):
- Direct: The target depends on one or more prior turns all reachable in a single
hop. No intermediate turns exist between any source and the target.
- Chained: At least one dependency of the target is itself dependent on a prior
turn, forming a sequence of two or more hops between a source node and the target.
Add any additional tags that apply:
- Diamond: One source node reaches the target via two independent intermediate
paths. The two paths diverge from the same source and converge back at the target
through different intermediate nodes.
- Multi_source: Two or more source nodes are ancestors of the target. The target's
content requires information from multiple distinct originating turns.
- Hallucination: The target span is factually wrong. The model used a value or
took an action traceable to a spurious prior turn rather than the correct source.
The DAG still traces where the wrong value came from.
For hallucination targets involving a wrong action or wrong parameter choice (e.g.
the model cancelled the wrong item, reused a value from a prior context it should
not have), trace BOTH branches: (a) where the wrong parameter value came from, AND
(b) what caused the model to make this choice - include the turns whose content
triggered the wrong decision. If the model ignored a turn that contained the
correct instruction, include that turn as a source node so the graph captures what
the model should have attended to but did not.

CONVERSATION: {conversation}
TARGETS: {targets_json}

Return a single JSON object. No other text. Schema:
{
  "conversation_id": "{conv_id}",
  "targets": [
    {
        "name": "<name>",
        "target_turn_idx": <int>,
        "target_text": "<exact text>",
        "comment": "<one sentence tracing the main dependency chain>",
        "dep_type_tags": ["<tag>", ...],
        "ground_truth_deps": {
            "<turn_idx>": {
                "depends_on": [<turn_idx>, ...],
                "spans": {"<turn_idx>": "<exact span copied from that turn>"}
            }
        }
    }
```

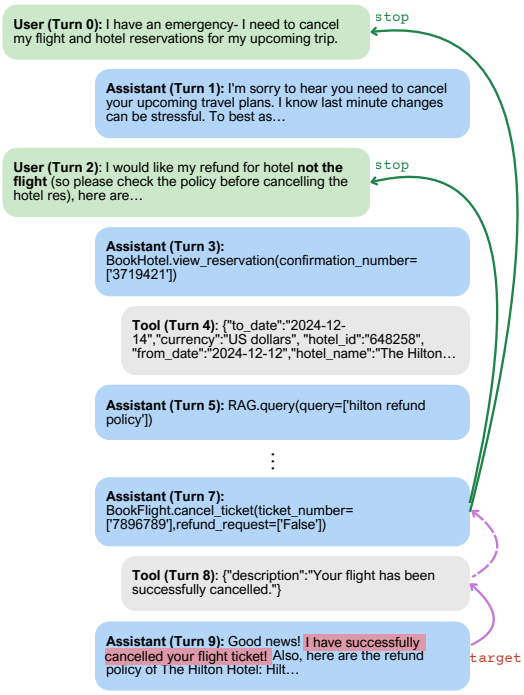

Figure 12: **Hallucination example from ConFETTI.** The gold provenance trace for the target span "I have successfully cancelled your flight ticket" (Turn 9) is shown with arrows indicating directed dependencies.

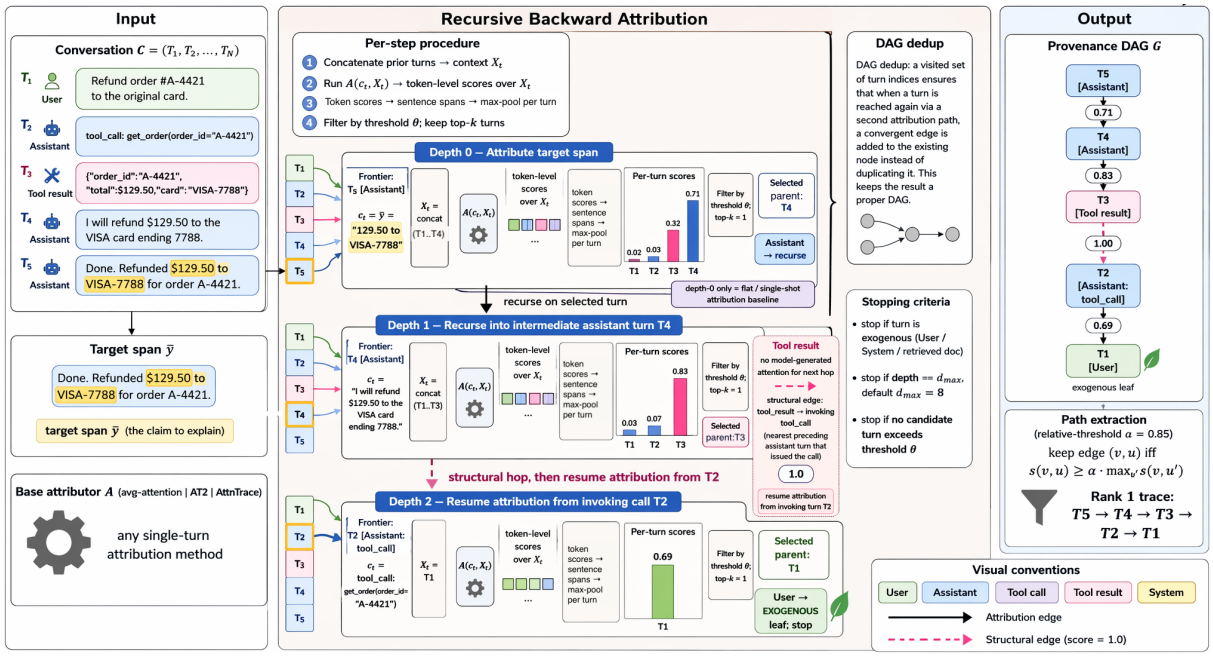

Figure 13: **Tokengeist procedure on a five-turn refund-tool example.** Stage 1 lists the inputs (the conversation, the target span $\bar{y}$, and the base attributor $\mathcal{A}$). Stage 2 unrolls three recursion steps and shows the per-step substeps (concatenate, score, map to turns, top-k) once, in a legend box that applies to every step. Stage 3 shows the constructed provenance DAG and the final trace after relative-threshold extraction with $\alpha = 0.85$ (Section 4). Solid arrows denote attribution edges; the dashed magenta arrow denotes the deterministic structural edge that bridges a tool-result turn to its invoking tool-call turn.

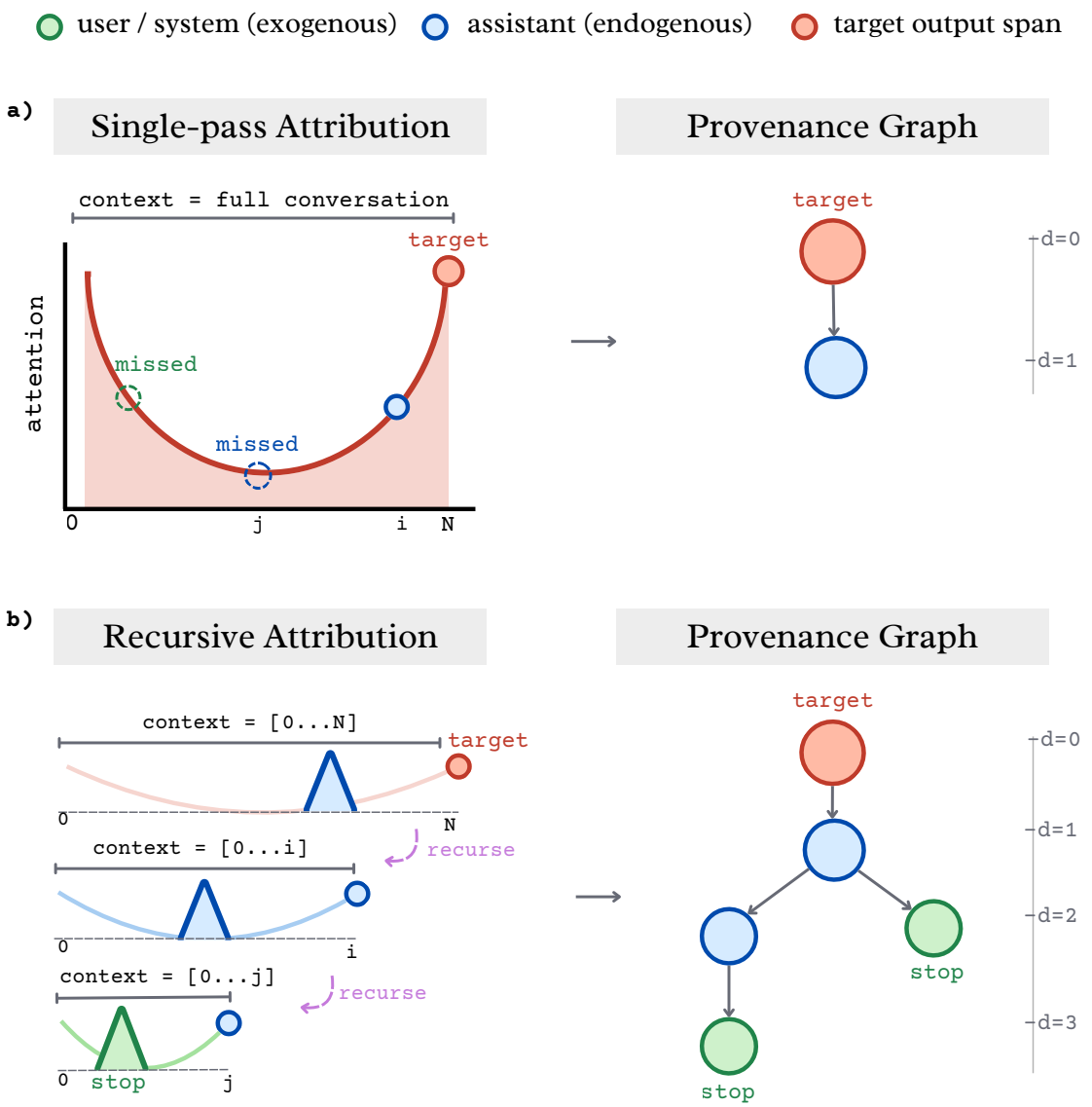

Figure 14: **Provenance collapse under flat attribution and recursive recovery.** *Top:* Single-pass (flat) attribution computes attention from the target output span over the entire conversation in a single step. Attention degrades for middle turns (Liu et al., 2024), causing intermediate sources (blue) to be missed or underweighted. The resulting provenance graph has depth 1 and loses the multi-hop structure. *Bottom:* Recursive attribution traces backward through intermediate assistant turns, re-running attribution on each one's context window until exogenous turns (green) are reached. The first call sees the full context $[0 \ldots N]$ but each recursive call starts from the position of the newly found intermediate, so the effective context window shrinks at every hop. This recovers the full depth-d provenance DAG, correctly attributing the target to its true sources.