# The Attribution Contract: Feature Attribution for Generative Language Models

Giang Nguyen
Guide Labs
nguyengiangbkhn@gmail.com

May 24, 2026

**Abstract**

Feature attribution methods promise to identify which input features matter for a model output. In generative language models, however, it is often unclear what should count as a feature in the first place. In autoregressive language models, earlier generated tokens are both outputs of the model and inputs to later predictions. In diffusion language models, generation proceeds through iterative denoising or unmasking rather than fixed left-to-right prediction, so local explanation may target a state of diffusion rather than a next token. We argue that this ambiguity is not merely an implementation detail, but a conceptual limitation of carrying classifier-era feature attribution directly into generative language modeling. We introduce the *Attribution Contract*, a specification for feature-attribution claims that names what output is being explained, which features are eligible to receive attribution, what generative process is assumed, what is held fixed, and what model score is being attributed. The contract clarifies why the same attribution method can answer different questions depending on how it is instantiated. We argue that many disagreements about feature attribution in generative language models are not disagreements about attribution algorithms, but about unstated explanatory contracts. Using autoregressive and diffusion language models as case studies, we show when attribution to earlier generated tokens, intermediate states, or denoising stages is informative, when it is misleading, and why feature-attribution methods in generative language models should be evaluated as method–contract pairs.

## 1 Introduction

Feature attribution methods are often described as answering a simple question: *which input features mattered for this output?* In the classical setting, this question has a natural shape. A model receives an input $x$, computes a prediction $f(x)$, and an attribution method assigns importance scores to components $x_i$ relative to a scalar target such as a class logit, probability, or loss. Integrated Gradients [13], for example, was introduced as an axiomatic method for attributing a deep network's prediction to its input features, motivated by desiderata such as sensitivity and invariance.

In generative language models (LMs), many real-world problems depend on feature attribution. A practitioner debugging a hallucinated summary needs to know whether the hallucinated content came from the source document or from the model's own prior tokens [24, 23]. A safety researcher auditing a harmful generation needs to know which part of the input triggered it [22, 26]. A scientist studying in-context learning needs to know which prompt features the model is actually using to make predictions [25].

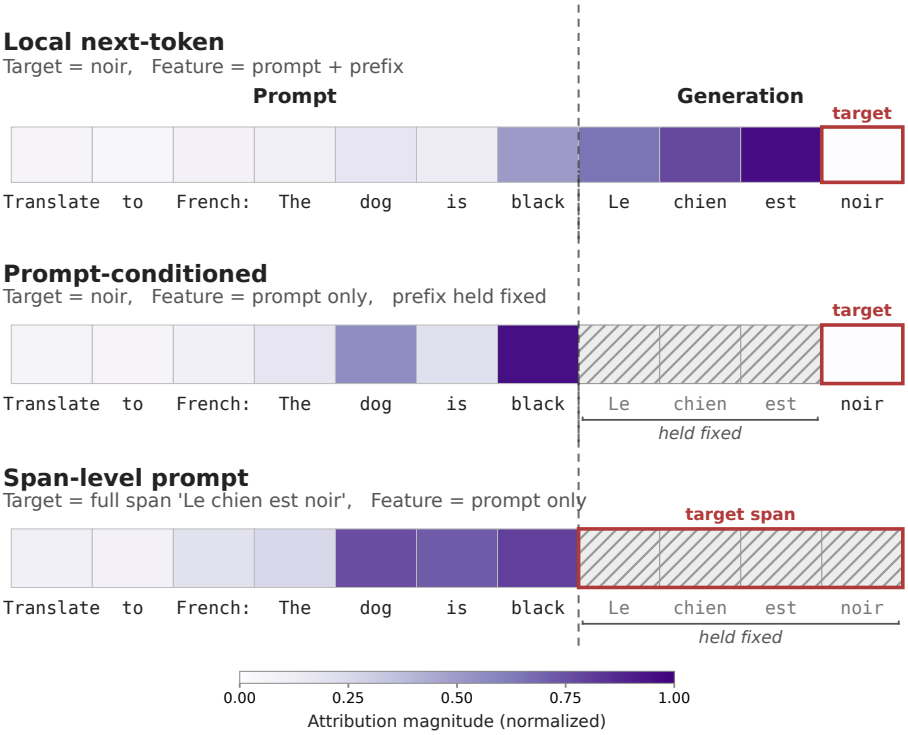


Figure 1: **The same attribution method produces different attribution maps under different settings.** All three rows use the same model, prompt, generation, and attribution method (Integrated Gradients [13]). Only the setting differs. *Top:* a local next-token setting attributes the prediction of `noir` over both prompt and prefix tokens, and mass concentrates on the generated prefix `Le chien est` because it is predictive of the next token. *Middle:* a prompt-conditioned setting holds the generated prefix fixed (gray, hatched) and attributes only over prompt tokens. "Held fixed" means the prefix remains in the forward pass at its actual value but is removed from the attribution path, so mass that would have gone to the prefix is now distributed only among prompt tokens. The result concentrates on `black`, the prompt token most predictive of `noir`. *Bottom:* a span-level setting targets the full generated span `Le chien est noir` and attributes only over prompt tokens; the prefix is again held fixed (gray, hatched).

Generative LMs break the clean setup of classical attribution. Consider an autoregressive LM with prompt $x$ and generated output $y_{1:T}$:

$$p(y_{1:T} \mid x) = \prod_{t=1}^{T} p(y_t \mid x, y_{<t}). \tag{1}$$

If we ask which tokens explain the next-token prediction $y_t$, the previously generated tokens $y_{<t}$ are genuine inputs to the model and may rightly receive high attribution. But those same tokens are also outputs of the model. As a result, feature attribution depends on what exactly we are trying to explain. An attribution to earlier-generated tokens may be appropriate if the goal is to explain the model's local prediction of the next token. **The same attribution answers a different question if we instead ask how the prompt influenced the overall response $y_{1:T}$.** We develop both questions in Section 4 and show that they correspond to different feature-attribution contracts.

Unlike in classification settings, the interpretation of a feature-attribution score in generative LMs depends on the explanatory goal. Suppose a model is prompted:

> Translate to French: "The dog is black."

and generates:

> Le chien est noir.

For the target token "noir," the generated prefix "Le chien est" is predictive: it helps determine which token is likely to come next. But if the explanatory question is which input token accounts for the meaning of "noir," the answer should point to "black," *not to the generated prefix*. These are different feature-attribution questions.

The issue is not that attribution to the generated prefix is wrong. Rather, the role of earlier generated tokens changes across explanatory settings, and feature-attribution methods for generative LMs often leave this distinction implicit [12, 5, 18]. As such, the same attribution score may be interpreted as answering a different feature-attribution question from the one it actually supports. We call this kind of interpretive error the *self-attribution fallacy*: treating attribution to generated-prefix tokens as if it answered a prompt-level explanatory question, without specifying the contract under which that interpretation is justified.

We argue that a feature-attribution score in a generative language model does not carry its own interpretation: that interpretation depends on the explanatory setting. We call such a setting an *Attribution Contract*, specifying what output is being explained, which features are eligible to receive attribution, what generative process is assumed, what is held fixed, and what model score is being attributed.

**Contributions.** To summarize, this paper makes four contributions. First, we identify *contract ambiguity* as a distinctive failure mode of feature attribution in generative language models. Second, we introduce the *Attribution Contract*, specifying the target, eligible feature set, generative process, conditioning regime, and attributed score of any feature-attribution claim. Third, we identify the *self-attribution fallacy*, in which attribution to generated-prefix tokens is misread as prompt-level explanation. Fourth, we propose that feature-attribution methods in generative language models should be evaluated as *method–contract pairs* rather than as context-free attribution algorithms.

## 2 From classifier attribution to generative attribution

Classical feature attribution inherits a static input-output schema:

$$x \longrightarrow f(x). \tag{2}$$

The candidate features are usually components of the input, such as pixels or tokens. The attribution target is usually a scalar, such as a class score, loss, or logit. Even in this setting, attribution has been criticized for being sensitive to the explanatory task and evaluation criterion [19, 20, 21]. Recent unification work argues that feature attribution, data attribution, and component attribution share methodological structure, even though they are often studied in fragmented literatures [17].

Generative language models replace this static schema with a *process* schema. For autoregressive LMs, each generated token is conditioned on the prompt and all previously generated tokens:

$$x, y_1, \ldots, y_{t-1} \longrightarrow y_t. \tag{3}$$

The shift from classifier to generative attribution is a shift in how many predictions a model makes. An image or text classifier produces one prediction, and attribution assigns importance to input features relative to that single output. An autoregressive language model produces a sequence of predictions, one per generated token, each conditioned on the prompt and all previously generated tokens. Each next-token prediction is itself a classification over the vocabulary, but the model makes many such predictions in sequence. Attribution must now choose which prediction to explain, and what counts as a feature when earlier predictions become inputs to later ones.

For diffusion language models, generation proceeds through an iterative denoising or unmasking process rather than a fixed left-to-right ordering. A simplified process view can be written as:

$$z_T \longrightarrow z_{T-1} \longrightarrow \cdots \longrightarrow z_0 \longrightarrow y, \tag{4}$$

where the intermediate states $z_t$ represent progressively refined states rather than a completed output sequence. In masked diffusion language models, generation may proceed through a forward masking process and a reverse process that predicts masked tokens [6].

These process structures make the notion of a "feature" significantly less stable than in the classical setting. In an autoregressive language model, feature attribution may target prompt tokens or generated-prefix tokens. In a diffusion language model, it may target prompt tokens, intermediate states, or denoising steps. As a result, the feature-attribution problem is no longer fully specified by the model and the output alone. This motivates the Attribution Contract introduced in the next section.

## 3 Attribution Contract

**Definition 1** (Attribution Contract)**.** *An Attribution Contract, i.e., the explanatory setting under which an attribution claim is made, is a tuple*

$$\mathcal{A} = (\mathcal{S}, \mathcal{C}, \mathcal{O}, \mathcal{P}, \mathcal{E}), \tag{5}$$

*where*

$$\begin{aligned}
\mathcal{S} &= \textit{model score},\\
\mathcal{C} &= \textit{what is held fixed},\\
\mathcal{O} &= \textit{output being explained},\\
\mathcal{P} &= \textit{generative process},\\
\mathcal{E} &= \textit{features eligible to receive attribution}.
\end{aligned}$$

*Read in order, the tuple spells* SCOPE*, capturing the idea that an Attribution Contract specifies the explanatory scope of a feature-attribution claim.*

Each element of the contract plays a distinct role. The score term $\mathcal{S}$ specifies the model quantity to which attribution is assigned, such as a logit, probability, loss, or log-likelihood. This matters because two explanations may target the same output while assigning attribution to different underlying scores.

The conditioning $\mathcal{C}$ specifies what is held fixed while attribution is computed or interpreted. Computationally, holding a variable fixed means that it remains in the forward pass at its actual value but is not part of the path or perturbation set over which attribution is computed (e.g., the fixed variable is not interpolated along the Integrated Gradients path). Two contracts that differ only in what is held fixed $\mathcal{C}$ therefore produce different attribution maps over the eligible features.

The target $\mathcal{O}$ specifies what is being explained: for example, a next-token prediction, an intermediate diffusion state, or a generated sequence. The feature set $\mathcal{E}$ specifies which features are eligible to receive attribution. These may include prompt tokens, generated-prefix tokens, intermediate generation states, or other process-related features, depending on the explanatory task. The process term $\mathcal{P}$ specifies the assumed structure of generation. This matters because feature influence in generative language models often propagates through a sequence of intermediate states rather than in a single input-output step. $\mathcal{P}$ is always part of the contract, but its computational role varies: under some contracts, attribution is computed by aggregating across the chain; under others, the chain serves only as the conditioning structure of a local prediction.

In the next section, we instantiate the contract framework in several concrete settings for autoregressive and diffusion language models. These settings show how the same feature-attribution method answers different questions under different contracts.

# 4 Feature-attribution settings in generative language models

The first row of Table 1 shows the classical classifier setting, where attribution has one canonical setup. Many existing feature-attribution methods for generative LMs are instantiated around local predictive targets, such as next-token probability $p(y_t \mid x, y_{<t})$ or sequence log-probability $\log p(y_{1:T} \mid x)$ decomposed token by token [12, 5, 18, 9]. The Attribution Contract makes clear that these are only some among many possible feature-attribution settings. In this section, we instantiate the contract framework in the settings most relevant to generative language models, summarized in Table 1. For each setting, we specify all five contract elements explicitly.

## 4.1 Autoregressive settings

Autoregressive language models provide the most direct extension of classical feature attribution. Even in this familiar setting, feature attribution can be posed in several different ways depending on whether the explanatory target is a next-token prediction, a prompt-conditioned token prediction, or a generated text span.

**Local next-token attribution.** We have

$$\mathcal{S} = \log p(y_t \mid x, y_{<t}), \tag{6}$$
$$\mathcal{C} = \text{nothing held fixed}, \tag{7}$$
$$\mathcal{O} = y_t, \tag{8}$$
$$\mathcal{P} = x \longrightarrow y_1 \longrightarrow \cdots \longrightarrow y_t, \tag{9}$$
$$\mathcal{E} = \{x_1, \ldots, x_n, y_1, \ldots, y_{t-1}\}. \tag{10}$$

Here, both prompt tokens and generated-prefix tokens are eligible features for attribution. The explanatory question is local: which available context features helped the model predict the next token?

**Prompt-conditioned token attribution.** This setting also explains a single target token, but restricts attribution to prompt features $x$ while treating the generated prefix as fixed context $\mathcal{C}$.

$$\mathcal{S} = \log p(y_t \mid x, y_{<t}), \tag{11}$$
$$\mathcal{C} = y_{<t} \text{ held fixed}, \tag{12}$$
$$\mathcal{O} = y_t, \tag{13}$$
$$\mathcal{P} = x \longrightarrow y_1 \longrightarrow \cdots \longrightarrow y_t, \tag{14}$$
$$\mathcal{E} = \{x_1, \ldots, x_n\}. \tag{15}$$

Here, the generated prefix still contributes to the model's prediction, but it is not treated as an eligible attribution feature. The explanatory question is narrower: which prompt tokens explain the target token given the already-generated context?

**Span-level prompt attribution.** This setting shifts the target from a single token to a larger generated span.

$$\mathcal{S} = \log p(y_{1:T} \mid x), \tag{16}$$
$$\mathcal{C} = y_{1:T} \text{ held fixed}, \tag{17}$$
$$\mathcal{O} = y_{1:T}, \tag{18}$$
$$\mathcal{P} = x \longrightarrow y_1 \longrightarrow \cdots \longrightarrow y_T, \tag{19}$$
$$\mathcal{E} = \{x_1, \ldots, x_n\}. \tag{20}$$

Here, prompt features $x$ remain the eligible attribution targets, but their influence propagates through the autoregressive chain $\mathcal{P}$: a prompt token can affect $y_T$ indirectly via its effect on $y_1$, $y_2$, and so on. The token-level contracts above also assume the same autoregressive process, but only as the conditioning structure of a single next-token prediction. Here, $\mathcal{P}$ does additional work: the score $\log p(y_{1:T} \mid x) = \sum_t \log p(y_t \mid x, y_{<t})$ aggregates attribution across the chain, so prompt influence is not localized to one step.

**A common interpretive mistake: self-attribution fallacy.** Attribution to generated-prefix tokens is not inherently wrong. Under a local next-token contract, it may be exactly what the method is supposed to recover. The mistake arises when such attribution is interpreted as though it answered a different question, such as how the prompt influenced the overall response.

A concrete scenario illustrates the failure mode. Consider a practitioner debugging an autoregressive language model that produced a hallucinated entity in a summary of a source document. The hallucinated token does not appear in the source. The practitioner runs Integrated Gradients to explain the hallucinated token, with the model's input consisting of the source document followed by the partially generated summary. The resulting attribution map shows that the immediately preceding generated tokens carry most of the mass, while the source document carries little. A natural reading is that the source document did not cause the hallucination, and that the model produced it from its own prior tokens.

This reading commits the *self-attribution fallacy*. The attribution was computed under a local next-token contract: $\mathcal{E}$ is the full input including the generated prefix, $\mathcal{O}$ is the hallucinated token, and $\mathcal{S}$ is the log probability of that token. The map correctly identifies which tokens were predictive of the next-token prediction. It does not answer the question the practitioner is actually asking: *which features of the source document explain the hallucinated content of the summary?*. Answering that question requires a different contract: $\mathcal{E}$ is the source document, $\mathcal{C}$ holds fixed the generated

prefix up to the hallucinated span, $\mathcal{O}$ is the hallucinated span itself, and $\mathcal{S}$ is the log-probability of the hallucinated span given the prompt and fixed prefix. This is the span-level analog of the prompt-conditioned setting, with the target shifted from a single token to the hallucinated content.

Recent work on attribution for autoregressive language models has identified related instances of contract confusion: CAGE [15] argues that current context-attribution methods give incomplete explanations because they ignore inter-generational influence, and HETA [9] argues that linear approximations from classifier-era attribution fail to capture autoregressive structure. These are specific instances of the more general problem the Attribution Contract is designed to make explicit.

### 4.2 Diffusion settings

Diffusion language models include three main families: masked diffusion, discrete diffusion, and continuous diffusion models [6, 7, 8]. We use the term *state* as a common abstraction across these families. In masked or discrete diffusion models, a state may correspond to a partially resolved token sequence. In continuous diffusion models, it may correspond to a continuous latent representation.

Diffusion LMs differ from autoregressive ones in a way that matters for feature attribution. There is no fixed left-to-right ordering and no canonical next token to predict. Generation proceeds through an iterative process, and attribution can target a state of refinement, a denoising stage, or a final output. These choices yield contracts that have no clean analog in the autoregressive case.

An example clarifies what differs across these contracts. Consider a masked diffusion language model prompted with `The capital of France is [MASK].` Suppose the model unmasks through a sequence of intermediate states. At an early state, the masked position resolves to a generic noun such as `city`. At a later state, it refines to `Paris`. The final output is `Paris`. Three feature-attribution questions arise:

- *State-level attribution* asks: what explains the model's choice of `city` at the intermediate state, given the prompt and the prior states?
- *Denoising-stage attribution* asks: which denoising stage was most responsible for committing the masked position to `Paris` rather than another capital?
- *Prompt-to-output attribution* asks: which prompt features (e.g., `France`, `capital`) explain the final output `Paris`?

We now make each of these contracts precise.

**State-level attribution.** This setting targets an intermediate state during generation rather than a final token.

$$\mathcal{S} = \log p(z_t \mid x, z_{>t}), \tag{21}$$
$$\mathcal{C} = \text{nothing held fixed}, \tag{22}$$
$$\mathcal{O} = z_t, \tag{23}$$
$$\mathcal{P} = z_T \longrightarrow z_{T-1} \longrightarrow \cdots \longrightarrow z_0 \longrightarrow y, \tag{24}$$
$$\mathcal{E} = \{x_1, \ldots, x_n\} \cup \{z_{t+1}, \ldots, z_T\}. \tag{25}$$

We write $z_{>t} = (z_{t+1}, \ldots, z_T)$ for the states generated before $z_t$ in the denoising chain. In our indexing, higher $t$ corresponds to earlier generation, which is the reverse of the autoregressive convention.

The explanatory question is: which prompt features or previously generated states influenced the current state of diffusion? This shifts the local attribution problem from predicting a next token to explaining how an intermediate state evolves.

Table 1: **Seven feature-attribution settings, organized by Attribution Contract.** The classifier row shows the classical setting where attribution is well-defined. Generative LMs have multiple contracts: three autoregressive and three diffusion. Columns follow the SCOPE ordering: $\mathcal{S}$, $\mathcal{C}$, $\mathcal{O}$, $\mathcal{P}$, $\mathcal{E}$. A dash in the conditioning column indicates that no variable is held fixed.

| **Setting** | $\mathcal{S}$ | $\mathcal{C}$ | $\mathcal{O}$ | $\mathcal{P}$ | $\mathcal{E}$ |
|---|---|---|---|---|---|
| *Classical settings* | | | | | |
| Classifier | $\log p(c \mid x)$ | — | $c$ | $x \to f(x)$ | input features |
| *Autoregressive settings* | | | | | |
| Local next-token | $\log p(y_t \mid x, y_{<t})$ | — | $y_t$ | $x \to y_1 \to \cdots \to y_t$ | prompt + prefix |
| Prompt-conditioned | $\log p(y_t \mid x, y_{<t})$ | $y_{<t}$ fixed | $y_t$ | $x \to y_1 \to \cdots \to y_t$ | prompt |
| Span-level prompt | $\log p(y_{1:T} \mid x)$ | $y_{1:T}$ fixed | $y_{1:T}$ | $x \to y_1 \to \cdots \to y_T$ | prompt |
| *Diffusion settings* | | | | | |
| State-level | $\log p(z_t \mid x, z_{>t})$ | — | $z_t$ | $z_T \to \cdots \to z_0 \to y$ | prompt + generated states |
| Denoising-stage | $\log p(y \mid x) - \log p(y \mid x; \mathrm{pert}(s_t))$ | — | $y$ | $z_T \to \cdots \to z_0 \to y$ | denoising stages |
| Prompt-to-output | $\log p(y \mid x)$ | — | $y$ | $z_T \to \cdots \to z_0 \to y$ | prompt |

**Denoising-stage attribution.** This setting asks which stages of the denoising process were most responsible for a property of the final output. The eligible features here are not tokens but stages of the generative process.

$$\mathcal{S} = \log p(y \mid x) - \log p(y \mid x; \mathrm{pert}(s_t)), \tag{26}$$
$$\mathcal{C} = \text{nothing held fixed}, \tag{27}$$
$$\mathcal{O} = y, \tag{28}$$
$$\mathcal{P} = z_T \longrightarrow z_{T-1} \longrightarrow \cdots \longrightarrow z_0 \longrightarrow y, \tag{29}$$
$$\mathcal{E} = \{1, \ldots, T\}. \tag{30}$$

The score is a difference rather than the marginal $\log p(y \mid x)$ because the marginal does not decompose over denoising stages: there is no analog of the autoregressive factorization $\log p(y_{1:T} \mid x) = \sum_t \log p(y_t \mid x, y_{<t})$. Attribution to a stage is therefore defined as the change in output likelihood caused by perturbing the stage $s_t$ to $\mathrm{pert}(s_t)$.

In a diffusion model, different stages typically commit to different aspects of the output: early stages settle the overall meaning, late stages settle the specific wording. In our running example, the early stage chose a generic noun (`city`), while a later stage committed to the specific word `Paris`. This unequal contribution is what makes the contract informative: a per-stage attribution map can identify which stage is responsible for a specific property of the output, such as a safety-relevant token or a factual error. Computing per-stage attribution requires methods that operate on stages directly, for example by ablating a stage or perturbing the noise schedule.

**Prompt-to-output attribution.** This setting targets the final output $y$, with attribution restricted to prompt features $x$.

$$\mathcal{S} = \log p(y \mid x), \tag{31}$$
$$\mathcal{C} = \text{nothing held fixed}, \tag{32}$$
$$\mathcal{O} = y, \tag{33}$$
$$\mathcal{P} = z_T \longrightarrow z_{T-1} \longrightarrow \cdots \longrightarrow z_0 \longrightarrow y, \tag{34}$$
$$\mathcal{E} = \{x_1, \ldots, x_n\}. \tag{35}$$

This is the diffusion analog of span-level prompt attribution in autoregressive models. The explanatory question is the same: which prompt features influenced the final output? The contract differs because prompt influence propagates through a denoising process rather than a left-to-right factorization.

These settings are not interchangeable. They answer different feature-attribution questions, so feature-attribution methods should not be evaluated against a single generic standard. The next section develops this point.

## 5 Evaluation must be contract-specific

If feature-attribution claims are contract-relative, then feature-attribution evaluation must also be contract-specific. There is no single generic notion of a "good attribution map" that applies uniformly across generative language models. A test that is appropriate for one feature-attribution setting may be inappropriate for another, because different contracts define different explanatory targets, different eligible feature sets, and different notions of what it means for an attribution to be faithful.

**Local next-token attribution.** For local next-token attribution, evaluation should focus on whether highly attributed features affect the score assigned to the target token. Suitable tests include deletion, insertion, or other counterfactual perturbations of prompt tokens and generated-prefix tokens, followed by measurement of the change in

$$\log p(y_t \mid x, y_{<t}). \tag{36}$$

This is the setting in which standard local faithfulness tests are most natural, because the explanatory target is itself local.

**Prompt-conditioned token attribution.** For prompt-conditioned token attribution, the generated prefix should remain fixed throughout evaluation. Accordingly, evaluation should perturb prompt features while holding the generated prefix constant, and should measure changes in the same token-level score:

$$\log p(y_t \mid x, y_{<t}). \tag{37}$$

**Span-level prompt attribution.** For span-level prompt attribution, the explanatory target is the generated span $y_{1:T}$, not a single token. Evaluation should perturb highly attributed prompt features while keeping the original generation $y_{1:T}$ fixed, and measure the change in

$$\log p(y_{1:T} \mid x). \tag{38}$$

This re-scores the original generation under the perturbed prompt; it does not regenerate a new sequence. An evaluation that regenerated would be testing a different contract.

**State-level attribution in diffusion language models.** The explanatory target is an intermediate state $z_t$ rather than a final token. Evaluation should perturb highly attributed prompt features (by deletion, masking, or replacement) or previously generated states (by replacing $z_{t'}$ with an alternative state for $t' > t$), then continue the denoising chain through to step $t$ and measure the change in

$$\log p(z_t \mid x, z_{>t}). \tag{39}$$

**Denoising-stage attribution.** For denoising-stage attribution, the eligible features are stages of the generative process rather than tokens, so token-level perturbation tests do not apply. Evaluation should perturb stages directly. Three natural perturbation types are: ablating a stage (replacing its update with a no-op), perturbing the noise schedule at that stage, and substituting an alternative denoising step. After perturbing a highly attributed stage, the rest of the denoising chain is run to completion, and evaluation measures the change in

$$\log p(y \mid x) - \log p(y \mid x; \mathrm{pert}(s_t)). \tag{40}$$

**Prompt-to-output attribution.** For prompt-to-output attribution, the target is the final output $y$ and the eligible features are prompt tokens, with the denoising chain treated as the assumed generative process. Evaluation should perturb prompt features and regenerate the output through the full denoising chain, measuring the change in

$$\log p(y \mid x). \tag{41}$$

This parallels span-level prompt attribution in the autoregressive case, but each perturbation requires re-running the denoising process, so faithfulness tests are computationally more expensive than their autoregressive counterparts.

These examples imply that in generative language models, an attribution method should be evaluated together with its contract, not on its own. When the contract is not stated, two methods that look like they disagree may just be answering different questions.

# 6 Related work

## 6.1 Classical feature attribution

Classical feature attribution methods assign importance scores to input features relative to a model output, with Integrated Gradients [13] as a canonical example motivated by axioms such as Sensitivity and Invariance. Subsequent work has shown that even in classical settings, attribution can be sensitive to the explanatory task and evaluation criterion [19, 20, 21]. For text models, the attribution path itself is a design choice. Integrated Gradients [13] interpolates in a straight line between embeddings, but this path passes through points that do not correspond to real tokens. Discretized Integrated Gradients [11] and Sequential Integrated Gradients [2] propose alternative paths and show that the choice changes attribution scores. We extend this line of work to generative language models, where the explanatory target, eligible feature set, conditioning regime, and assumed generation process may all vary, introducing a further source of under-specification.

## 6.2 Feature attribution for generative language models

Several recent works extend feature attribution to sequence generation and decoder-only language models. Inseq [12] provides an interpretability toolkit for sequence generation, Captum's generative-

LM interface [5] adapts attribution tooling to generated text, and ReAGent [18] proposes a model-agnostic method for generative LMs. More recent work such as HETA [9] and CAGE [15] targets autoregressive decoder-only generation, including higher-order effects and inter-generational influence among generated tokens. ContextCite [1] attributes generation to context features, and Jacobian Scopes [3] propose three gradient-based variants targeting different model predictive outputs.

These methods are often read as competing alternatives. **We argue they are not**. ContextCite and CAGE operate within prompt-to-output contracts: their target is the final generation, their eligible features are prompt tokens, and their score is the joint sequence probability. Inseq and Captum typically operate within local next-token contracts: their target is a single next-token prediction, their eligible features include the generated prefix, and their score is the next-token log-probability. HETA argues that linear attribution methods miss higher-order effects across generation steps, which is a within-contract critique of a particular method family rather than a competing alternative. Jacobian Scopes apply gradient-based attribution to three different scores: a specific logit (Semantic), the full predictive distribution (Fisher), and model confidence (Temperature). These variants instantiate the same local next-token contract and the same underlying attribution algorithm; they differ only in the score element $\mathcal{S}$. Once the contracts are named, the methods discussed above are no longer in conflict.

The Attribution Contract makes this classification explicit: methods that look comparable on the surface may answer different questions, and methods that look different may answer the same one.

# 7 Conclusion

Feature attribution in generative language models cannot be treated as a straightforward extension of classifier-era attribution. In classical settings, it is often acceptable to assume a stable input-output structure, a stable feature set, and a clear explanatory target. In generative language models, these assumptions break down.

This paper introduced the *Attribution Contract*, the explanatory setting under which a feature-attribution claim is made. We used this framework to distinguish several feature-attribution settings in generative language models, including local next-token attribution, prompt-conditioned token attribution, span-level prompt attribution, and three diffusion contracts: state-level, denoising-stage, and prompt-to-output attribution. We also identified a common interpretive error in autoregressive models, the *self-attribution fallacy*: reading attribution to generated-prefix tokens as if it answered a prompt-level question.

The broader implication is that feature-attribution methods in generative language models cannot be evaluated against a single standard. Different contracts ask different questions, and evaluation must match the contract. Methods should therefore be evaluated as method–contract pairs, not in isolation. We do not aim to replace existing methods; we aim to make their claims specific enough to compare.


# Acknowledgments

The author thanks Aya Abdelsalam Ismail, Anh Totti Nguyen, and Julius Adebayo for their comments on earlier drafts of this paper. The views expressed in this paper are those of the author and do not necessarily reflect the views of Guide Labs.

## References


[1] Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Mądry. ContextCite: Attributing Model Generation to Context. In *Advances in Neural Information Processing Systems*, volume 37, 2024. https://proceedings.neurips.cc/paper_files/paper/2024/hash/adbea136219b64db96a9941e4249a857-Abstract-Conference.html

[2] Joseph Enguehard. Sequential Integrated Gradients: a Simple but Effective Method for Explaining Language Models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7555–7565, 2023. https://aclanthology.org/2023.findings-acl.477/

[3] Toni J.B. Liu, Baran Zadeoğlu, Nicolas Boullé, Raphaël Sarfati, and Christopher J. Earls. Jacobian Scopes: Token-level Causal Attributions in LLMs. arXiv preprint arXiv:2601.16407, 2026. https://arxiv.org/abs/2601.16407

[4] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual Associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 2022. https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html

[5] Vivek Miglani, Aobo Yang, Aram H. Markosyan, Diego Garcia-Olano, and Narine Kokhlikyan. Using Captum to Explain Generative Language Models. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 165–173, 2023. https://aclanthology.org/2023.nlposs-1.19/

[6] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large Language Diffusion Models. In *Advances in Neural Information Processing Systems*, 2025. https://openreview.net/forum?id=KnqiC0znVF

[7] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32819–32848. PMLR, 2024. https://proceedings.mlr.press/v235/lou24a.html

[8] Jaehyeong Jo and Sung Ju Hwang. Continuous Diffusion Model for Language Modeling. arXiv preprint arXiv:2502.11564, 2025. https://arxiv.org/abs/2502.11564

[9] Vishal Pramanik, Maisha Maliha, Nathaniel D. Bastian, and Sumit Kumar Jha. Hessian-Enhanced Token Attribution (HETA): Interpreting Autoregressive LLMs. In *International Conference on Learning Representations*, 2026. https://openreview.net/forum?id=XsEZcigEjq

[10] Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. Measuring Attribution in Natural Language Generation Models. *Computational Linguistics*, 49(4):777–840, 2023. https://aclanthology.org/2023.cl-4.2/

[11] Soumya Sanyal and Xiang Ren. Discretized Integrated Gradients for Explaining Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10285–10299, 2021. https://aclanthology.org/2021.emnlp-main.805/

[12] Gabriele Sarti, Nils Feldhus, Ludwig Sickert, and Oskar van der Wal. Inseq: An Interpretability Toolkit for Sequence Generation Models. In *Proceedings of the 61st Annual Meeting of the*

*Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 421–435, 2023. https://aclanthology.org/2023.acl-demo.40/

[13] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017. https://proceedings.mlr.press/v70/sundararajan17a.html

[14] Raphael Tang, Linqing Liu, Akshat Pandey, Zhiying Jiang, Gefei Yang, Karun Kumar, Pontus Stenetorp, Jimmy Lin, and Ferhan Ture. What the DAAM: Interpreting Stable Diffusion Using Cross Attention. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5644–5659, 2023. https://aclanthology.org/2023.acl-long.310/

[15] Chase Walker and Rickard Ewetz. Explaining the Reasoning of Large Language Models Using Attribution Graphs. arXiv preprint arXiv:2512.15663, 2025. https://arxiv.org/abs/2512.15663

[16] Theodora Worledge, Judy Hanwen Shen, Nicole Meister, Caleb Winston, and Carlos Guestrin. Unifying Corroborative and Contributive Attributions in Large Language Models. arXiv preprint arXiv:2311.12233, 2023. https://arxiv.org/abs/2311.12233

[17] Shichang Zhang, Tessa Han, Usha Bhalla, and Himabindu Lakkaraju. Towards Unified Attribution in Explainable AI, Data-Centric AI, and Mechanistic Interpretability. arXiv preprint arXiv:2501.18887, 2025. https://arxiv.org/abs/2501.18887

[18] Zhixue Zhao and Boxuan Shan. ReAGent: A Model-Agnostic Feature Attribution Method for Generative Language Models. In *Proceedings of the AAAI Workshop on Responsible Language Models*, 2024. https://arxiv.org/abs/2402.00794

[19] Blair Bilodeau, Natasha Jaques, Pang Wei Koh, and Been Kim. Impossibility Theorems for Feature Attribution. *Proceedings of the National Academy of Sciences*, 121(2):e2304406120, 2024. https://www.pnas.org/doi/10.1073/pnas.2304406120

[20] Giang Nguyen, Daeyoung Kim, and Anh Nguyen. The Effectiveness of Feature Attribution Methods and Its Correlation with Automatic Evaluation Scores. In *Advances in Neural Information Processing Systems*, volume 34, pages 26422–26436, 2021. https://proceedings.neurips.cc/paper/2021/hash/de043a5e421240eb846da8effe472ff1-Abstract.html

[21] Julius Adebayo, Michael Muelly, Harold Abelson, and Been Kim. Post Hoc Explanations May Be Ineffective for Detecting Unknown Spurious Correlation. In *International Conference on Learning Representations*, 2022. https://openreview.net/forum?id=xNOVfCCvDpM

[22] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, Andy Jones, Sam Bowman, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Nelson Elhage, Sheer El-Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Danny Hernandez, Tristan Hume, Josh Jacobson, Scott Johnston, Shauna Kravec, Catherine Olsson, Sam Ringer, Eli Tran-Johnson, Dario Amodei, Tom Brown, Nicholas Joseph, Sam McCandlish, Chris Olah, Jared Kaplan, Jack Clark. Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned. arXiv preprint arXiv:2209.07858, 2022. https://arxiv.org/abs/2209.07858

[23] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Chen, Wenhao Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*, 2024. https://dl.acm.org/doi/10.1145/3703155

[24] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 55(12):1–38, 2023. https://dl.acm.org/doi/10.1145/3571730

[25] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, 2022. https://aclanthology.org/2022.emnlp-main.759/

[26] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red Teaming Language Models with Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022. https://arxiv.org/abs/2202.03286