ACL Anthology ID 2026.acl-long.1159 / revision 2 / 21 Oct 2026

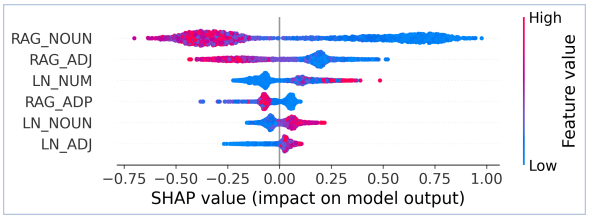
TPA: Next Token Probability Attribution for Detecting Hallucinations in RAG

Pengqian Lu, Jie Lu*, Anjin Liu, and Guangquan Zhang
Australian Artificial Intelligence Institute (AAII)
University of Technology Sydney
Ultimo, NSW 2007, Australia
{Pengqian.Lu@student., Jie.Lu@, Anjin.Liu@, Guangquan.Zhang@}uts.edu.au

Abstract

Detecting hallucinations in Retrieval-Augmented Generation (RAG) remains a critical reliability challenge, as ungrounded responses can have severe consequences in high-stakes applications such as clinical decision support, legal research assistants, and autonomous agents that act on retrieved evidence. Prior approaches attribute hallucinations to a binary conflict between internal knowledge stored in FFNs and the retrieved context. However, this perspective is incomplete, failing to account for the impact of other components of the LLM, such as the user query, previously generated tokens, the self token, and the Final LayerNorm adjustment. To comprehensively capture the impact of these components on hallucination detection, we propose TPA which mathematically attributes each token's probability to seven distinct sources: Query, RAG Context, Past Token, Self Token, FFN, Final LayerNorm, and Initial Embedding. This attribution quantifies how each source contributes to the generation of the next token. Specifically, we aggregate these attribution scores by Part-of-Speech (POS) tags to quantify the contribution of each model component to the generation of specific linguistic categories within a response. By leveraging these patterns, such as detecting anomalies where Nouns rely heavily on LayerNorm, TPA effectively identifies hallucinated responses. Extensive experiments on five LLMs (Llama2-7B/13B, Llama3-8B, Mistral-7B, and Qwen3-8B) demonstrate that TPA achieves state-of-the-art performance across diverse architectures.

(a) TPA attributes each next-token probability to seven sources, then aggregates source attributions by POS tags to form the features for detecting hallucination.



(b) Feature-importance analysis (SHAP) shows that the detector leverages source contributions conditioned on POS tags. For example, responses are more likely to be hallucinated when RAG contributes little to NOUN tokens or when LN contributes too much to NUM tokens.

Figure 1: Applying the TPA framework to a Llama2-7b response from RAGTruth dataset (Niu et al., 2024).

1 Introduction

Large Language Models (LLMs), despite their impressive capabilities, are prone to hallucinations (Huang et al., 2025). Consequently, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) is widely used to alleviate hallucinations by grounding models in external knowledge. However, RAG systems are not perfect. They can still hallucinate by ignoring or misinterpreting the retrieved information (Sun et al., 2025). Detecting such failures is therefore a critical challenge. Following Sun et al.

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 25273–25292
July 2-7, 2026 ©2026 Association for Computational Linguistics

(2025), we define a *hallucination* as a response containing content inconsistent with the retrieved RAG context (assuming the context is relevant and correct), excluding retrieval errors, outdated knowledge, and ambiguous evidence. The cost rises with stakes: hallucinated medication dosages in clinical decision support can harm patients, fabricated case citations in legal research assistants have led to sanctioned court filings, and ungrounded intermediate responses in autonomous agents propagate into downstream actions.

The prevailing paradigm for hallucination detection typically relies on hand-crafted proxy signals. For example, common approaches detect hallucination through consistency checks (Manakul et al., 2023) or scalar uncertainty metrics such as semantic entropy (Han et al., 2024). However, these methods only measure the symptoms of hallucination, such as output variance or surface confidence, rather than the underlying architectural causes. Consequently, they often fail when a model is confidently incorrect (Simhi et al., 2025).

To address the root cause of hallucination, recent research has shifted focus to the model's internal representations. Pioneering works such as ReDeEP (Sun et al., 2025) explicitly assume the RAG context is correct. They reveal that hallucinations in RAG typically stem from a disproportionate dominance of internal parametric knowledge (stored in FFNs) over the retrieved external context.

This insight inspires a fundamental question: **Is the binary conflict between FFNs and RAG the only cause of hallucination?** Critical components like LayerNorm and User Query are often overlooked. **Do contributions from these sources also drive hallucinations?** In this paper, we extend the analysis to cover all additive components along the transformer residual stream. This approach enables detection based on the model's full internal mechanics instead of relying on partial proxy signals.

To achieve this, we also assume the RAG context contains relevant information and introduce TPA (Next **T**oken **P**robability **A**ttribution for Detecting Hallucinations in RAG). This framework mathematically attributes the final probability of each token to seven distinct sources: Query, RAG, Past, Self Token, FFN, Final LayerNorm, and Initial Embedding. The attribution scores of these seven parts sum to the token's final probability, ensuring we capture the complete generation process.

To compute these attributions, we proposed a probe function similar with nostalgebraist (2020) that uses the model's unembedding matrix to read out the next-token probability directly from an intermediate residual-stream state. Concretely, for each component on the residual stream, we define its contribution as the change in the probed next-token probability *before* versus *after* applying that component. In this way, we can compute the contribution from Initial Embedding, attention block, FFN, and Final LayerNorm. For the attention block, we further distribute its contribution to Query, RAG, Past and Self Token according to their attention weights.

However, these attention scores are insufficient for detection. A high reliance on internal parametric knowledge (FFNs) does not necessarily imply a hallucination. This pattern is expected for function words like "the" or "of". Yet, it becomes highly suspicious when found in named entities. Therefore, treating all tokens equally fails to capture these critical distinctions.

To capture this distinction, we aggregate the attribution scores using Part-of-Speech (POS) tags. We employ POS tags to capture comprehensive syntactic patterns. Unlike Named Entity Recognition (NER), which is limited to specific entity types, POS tagging covers all tokens (including critical categories like Numerals and Adpositions) and maintains high computational efficiency.

Figure 1 illustrates how TPA turns a single response into detection features: we first compute token-level source attributions, then aggregate them by POS tags. The second step is critical since hallucination signals vary across distinct parts of speech. For example, low RAG contribution on nouns or high LN contribution on numerals is often indicative of hallucination. These patterns are harder to capture if we only use raw token-level attribution scores without POS information.

Our main contributions are:

1. We propose TPA, a novel framework that mathematically attributes each token's probability to seven distinct attribution sources. This provides a comprehensive mechanistic view of the token generation process.

2. We introduce a syntax-aware aggregation mechanism. By quantifying how attribution sources drive distinct parts of speech, this approach enables the detector to pinpoint anomalies in specific entities while ignoring benign grammatical patterns.

3. Extensive experiments demonstrate that TPA achieves state-of-the-art performance. Our framework also offers transparent interpretability, automatically uncovering novel mechanistic signatures, such as anomalous LayerNorm contributions, that extend beyond the traditional FFN-RAG binary conflict.

2 Related Work

Uncertainty and Proxy Metrics. Approaches in this category estimate hallucination via output inconsistency or proxy signals. Some methods quantify uncertainty using model ensembles (Malinin and Gales, 2021) or by measuring self-consistency across multiple sampled generations from a single model (Manakul et al., 2023). Others utilize lightweight proxy scores computable from a single generation pass, such as energy-based OOD proxy scores (Liu et al., 2020), embedding-based distance scores for conditional LMs (Ren et al., 2023), and token-level uncertainty heuristics for hallucination detection (Lee et al., 2024; Zhang et al., 2023). While efficient, these scores provide indirect signals (e.g., confidence or distribution shift) and therefore may be imperfect indicators of factual correctness.

LLM-based Evaluation. External LLMs are also employed as verifiers. In RAG settings, outputs can be checked against retrieved evidence (Friel and Sanyal, 2023) or through claim extraction and reference-based verification (Hu et al., 2024), and LLM-as-a-judge baselines are often instantiated using curated prompts (Niu et al., 2024). Automated evaluation suites (Es et al., 2024; TrueLens, 2024) have also been developed. Other strategies include cross-examination to expose inconsistencies (Cohen et al., 2023; Yehuda et al., 2024) or fine-tuning detectors for span-level localization (Su et al., 2025). However, many of these approaches require extra LLM calls or multi-step verification.

Probing Internal Activations. Recent work extracts factuality signals from internal representations, e.g., linear truthful directions or inference-time shifts (Burns et al., 2022; Li et al., 2023), and probe-based detectors trained on hidden states (Azaria and Mitchell, 2023; Han et al., 2024). Related studies show internal states remain predictive for hallucination detection (Chen et al., 2024). Beyond detection, mechanistic analyses conflicts between FFN and RAG context (Sun et al., 2025), and lightweight indicators use attention-head norms (Ho et al., 2025). Active approaches steer or edit activations (Park et al., 2025; Li et al., 2023), or adjust decoding probabilities for diagnosis (Chen et al., 2025). In contrast, we decompose the final token probability into fine-grained sources.

3 Methodology

As illustrated in Figure 2, TPA operates in three stages and can be implemented with a fully parallel teacher-forced pass. Given the generated response sequence $\mathbf{y}$ of length T, we can feed the entire sequence into the model with standard causal masking to extract hidden states and attention maps for all T tokens in a single teacher-forced pass. This avoids autoregressive resampling while enabling efficient attribution computation.

We first derive a complete decomposition of token probabilities (Sec. 3.2), then attribute attention contributions to specific attribution sources (Sec. 3.3). Finally, we aggregate these scores to quantify how sources drive distinct parts of speech (Sec. 3.4). The pseudo-code and complexity analysis are provided in the Appendix. We report complexity instead of wall-clock time since the latter varies in different implementation hardware. To provide the theoretical basis for our method, we first formalize the transformer's architecture.

3.1 Preliminaries: Transformer Architecture

3.1.1 Notations

We consider a standard decoder-only Transformer with L layers. We denote the query tokens as $\mathbf{x}_{\text{qry}}$, the retrieved context tokens as $\mathbf{x}_{\text{rag}}$, and the generated response as $\mathbf{y} = (y_1, \ldots, y_T)$, with prompt length $T_0 = |\mathbf{x}_{\text{qry}}| + |\mathbf{x}_{\text{rag}}|$. We analyze generation at step $t \in \{1, \ldots, T\}$, where the model observes the prefix $\mathbf{s}_t = [\mathbf{x}_{\text{qry}}, \mathbf{x}_{\text{rag}}, y_1, \ldots, y_{t-1}]$, and predicts the next token y_t. Let $n_t = |\mathbf{s}_t| = T_0 + t - 1$ denote the position index of the last token in $\mathbf{s}_t$ (the token whose embedding is used to predict y_t).

Unless stated otherwise, all hidden states and residual outputs (e.g., $\mathbf{h}^{(l)}$, $\mathbf{h}^{(l)}_{\text{mid}}$) refer to the vector at the last position n_t, and we omit the explicit index n_t and the step index t for clarity. We keep explicit indices only for attention weights (e.g., $\mathbf{A}^{(l)}_h[n_t, k]$). We use d for the hidden dimension, H for the number of attention heads, and $d_h = d/H$ for the head dimension.

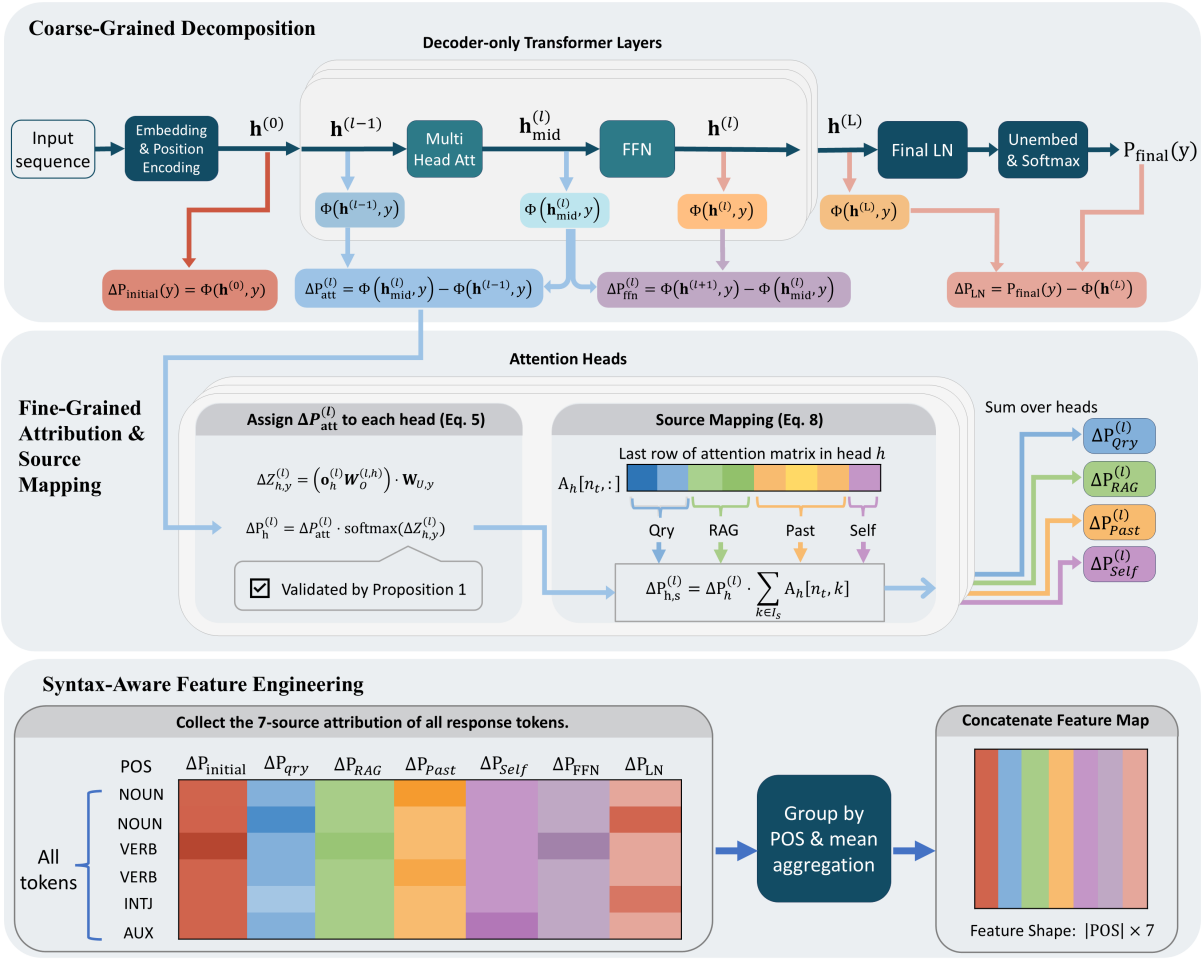


Figure 2: **Overview of the TPA framework.** (1) **Coarse-Grained Decomposition**: Complete decomposition of token probability into four components (Section 3.2). (2) **Fine-Grained Attribution**: Mapping attention contributions to four input sources via head-specific weights (Section 3.3). (3) **Syntax-Aware Feature Engineering**: Aggregating these attributions by POS tags to construct the final detection features (Section 3.3.4).

3.1.2 Residual Updates and Probing

The input tokens are mapped to continuous vectors via an embedding matrix $\mathbf{W}_e \in \mathbb{R}^{|\mathcal{V}| \times d}$ and summed with positional embeddings. The initial state at the target position is $\mathbf{h}^{(0)} = \mathbf{W}_e[\mathbf{s}_t[n_t]] + \mathbf{p}_{n_t}$, where $\mathbf{p}_{n_t}$ is the positional embedding at position n_t. We adopt the Pre-LN configuration. Crucially, each layer l updates the hidden state via additive residual connections:

$$\mathbf{h}_{\text{mid}}^{(l)} = \mathbf{h}^{(l-1)} + \text{Attn}(\text{LN}(\mathbf{h}^{(l-1)})) \quad (1)$$

$$\mathbf{h}^{(l)} = \mathbf{h}_{\text{mid}}^{(l)} + \text{FFN}(\text{LN}(\mathbf{h}_{\text{mid}}^{(l)})) \quad (2)$$

Here, $\text{Attn}(\cdot)$ denotes the attention output vector at position n_t under causal masking. This structure implies that the final representation is the sum of the initial embedding and all subsequent layer updates. To quantify these updates, we define a *Probe Function* $\Phi(\mathbf{h}, y)$ similar to the logit lens technique (nostalgebraist, 2020) that measures the hypothetical probability of the target token y given any intermediate state vector $\mathbf{h}$:

$$\Phi(\mathbf{h}, y) = [\text{Softmax}(\mathbf{h}\mathbf{W}_U)]_y \quad (3)$$

where $\mathbf{W}_U$ is the unembedding matrix.

Guiding Question: *Since the model is a stack of residual updates, can we mathematically decompose the final probability exactly into the sum of component contributions?*

3.2 Coarse-Grained Decomposition

We answer the preceding question affirmatively by leveraging the additive nature of the residual updates. Based on the probe function $\Phi(\mathbf{h}, y)$ defined in Eq. (3), we isolate the probability contribution of each model component as the distinct change it induces in the probe output.

We define the baseline contribution from input static embeddings ($\Delta P_{\text{initial}}$), the incremental

gains from Attention and FFN blocks in layer l ($\Delta P_{\text{att}}^{(l)}, \Delta P_{\text{ffn}}^{(l)}$), and the adjustment from the final LayerNorm (ΔP_{LN}) as follows:

$$\Delta P_{\text{initial}}(y) = \Phi(\mathbf{h}^{(0)}, y) \tag{4}$$

$$\Delta P_{\text{att}}^{(l)} = \Phi(\mathbf{h}_{\text{mid}}^{(l)}, y) - \Phi(\mathbf{h}^{(l-1)}, y) \tag{5}$$

$$\Delta P_{\text{ffn}}^{(l)} = \Phi(\mathbf{h}^{(l)}, y) - \Phi(\mathbf{h}_{\text{mid}}^{(l)}, y) \tag{6}$$

$$\Delta P_{\text{LN}} = P_{\text{final}}(y) - \Phi(\mathbf{h}^{(L)}, y) \tag{7}$$

We define $P_{\text{final}}(y)$ as the model output probability after applying the final LayerNorm at position n_t.

By summing these differences, we derive the complete decomposition of the model's output.

Theorem 1 (Complete Probability Decomposition)**.** *The final probability for a target token y is exactly the sum of the contribution from the initial embedding, the cumulative contributions from Attention and FFN blocks across all L layers, and the adjustment from the final LayerNorm:*

$$\begin{aligned} P_{final}(y) =& \Delta P_{initial}(y) + \Delta P_{LN} \\ &+ \sum_{l=1}^{L} \left(\Delta P_{att}^{(l)} + \Delta P_{ffn}^{(l)} \right) \end{aligned} \tag{8}$$

Proof. See Appendix. □

The Guiding Question: While Eq. (8) quantifies *how much* the model components contribute to the prediction probability, it treats the term $\Delta P_{\text{att}}^{(l)}$ as a black box. To effectively detect hallucinations, we must identify *where* this attention is focused.

3.3 Fine-Grained Attribution

To identify the focus of attention, we must decompose the attention contribution $\Delta P_{\text{att}}^{(l)}$ into contributions from individual attention heads.

3.3.1 The Challenge: Non-Linearity

Standard Multi-Head Attention concatenates the outputs of H independent heads and projects them via an output matrix $\mathbf{W}_O^{(l)}$. Mathematically, by partitioning $\mathbf{W}_O^{(l)}$ into head-specific sub-matrices, this operation is strictly equivalent to the sum of projected head outputs:

$$\mathbf{h}_{\text{att}}^{(l)} = \sum_{h=1}^{H} \underbrace{\left(\mathbf{A}_h^{(l)}[n_t, :] \mathbf{V}_h^{(l)} \right)}_{\mathbf{o}_h^{(l)}} \mathbf{W}_O^{(l,h)} \tag{9}$$

where $\mathbf{o}_h^{(l)}$ is the head output vector at the target position, derived from the attention row $\mathbf{A}_h^{(l)}[n_t, :]$ and the value matrix $\mathbf{V}_h^{(l)}$.

Eq. (9) establishes that the attention output is linear with respect to individual heads in the hidden state space. However, our goal is to attribute the probability change $\Delta P_{\text{att}}^{(l)}$ to each head h. Since the probe function $\Phi(\cdot)$ employs a non-linear Softmax operation, the sum of probability changes calculated by probing individual heads does not equal the attention block contribution. This inequality prevents us from calculating head contributions by simply probing each head individually, motivating our shift to the logit space.

3.3.2 Logit-Based Apportionment

To bypass the non-linearity of the Softmax, we analyze contributions in the logit space. Let $\Delta z_{h,y}^{(l)}$ denote the scalar contribution of head h to the logit of the **target token** y. This is calculated as the dot product between the projected head output and the target token's unembedding vector $\mathbf{w}_{U,y}$:

$$\Delta z_{h,y}^{(l)} = \left(\mathbf{o}_h^{(l)} \mathbf{W}_O^{(l,h)} \right) \cdot \mathbf{w}_{U,y} \tag{10}$$

We then apportion the complete probability contribution $\Delta P_{\text{att}}^{(l)}$ (derived in Section 3.2) to each head h proportional to its exponential logit contribution:

$$\Delta P_h^{(l)} = \Delta P_{\text{att}}^{(l)} \cdot \frac{\exp(\Delta z_{h,y}^{(l)})}{\sum_{j=1}^{H} \exp(\Delta z_{j,y}^{(l)})} \tag{11}$$

3.3.3 Theoretical Justification

We ground the logit-based apportionment using a first-order Taylor expansion similar to (Montavon et al., 2019). This approximates how logit changes affect the final probability.

Proposition 1 (Linear Decomposition)**.** *The total attention contribution $\Delta P_{att}^{(l)}$ is approximated by the sum of head logits $\Delta z_{h,y}^{(l)}$ scaled by a gradient $\mathcal{G}^{(l)}$:*

$$\Delta P_{att}^{(l)} \approx \mathcal{G}^{(l)} \sum_{h=1}^{H} \Delta z_{h,y}^{(l)} + \mathcal{E} \tag{12}$$

Proof. See Appendix. □

While Proposition 1 implies a linear relationship, direct attribution is unstable when head logits sum to zero. To resolve this, we employ the Softmax normalization in Eq. (11). This ensures numerical stability and constrains the sum of head scores to exactly match the layer total $\Delta P_{\text{att}}^{(l)}$. Thus, it preserves the conservation principle in Theorem 1.

Regarding the approximation error $\mathcal{E}$, we rely on the first-order term for efficiency. This is effective

because hallucinations are often high-confidence (Kadavath et al., 2022), which suppresses higher-order terms. For low-confidence scenarios, prior work identifies low probability itself as a strong hallucination signal (Guerreiro et al., 2023). Our framework naturally captures this feature because the attribution scores sum exactly to the final output probability (Theorem 1). Therefore, TPA inherently incorporates this critical probability signal and effectively detects such hallucinations.

3.3.4 Source Mapping and The 7-Source Split

Having isolated the head contribution $\Delta P_h^{(l)}$, we can now answer the guiding question by tracing attention back to input tokens using the attention matrix $\mathbf{A}_h^{(l)}$. We categorize inputs into four source types: $\mathcal{S} = \{\texttt{Qry}, \texttt{RAG}, \texttt{Past}, \texttt{Self}\}$. Let $T_q = |\mathbf{x}_{\text{qry}}|$ and $T_r = |\mathbf{x}_{\text{rag}}|$, so $T_0 = T_q + T_r$ and $n_t = T_0 + t - 1$. We partition the causal attention range $[1, n_t]$ into four disjoint index sets: $\mathcal{I}_{\texttt{Qry}} = [1, T_q]$, $\mathcal{I}_{\texttt{RAG}} = [T_q + 1, T_0]$, $\mathcal{I}_{\texttt{Past}} = [T_0 + 1, n_t - 1]$, and $\mathcal{I}_{\texttt{Self}} = \{n_t\}$. For a source type S containing token indices $\mathcal{I}_S$, the aggregated contribution is:

$$\Delta P_S^{(l)} = \sum_{h=1}^{H} \left(\Delta P_h^{(l)} \cdot \sum_{k \in \mathcal{I}_S} \mathbf{A}_h^{(l)}[n_t, k] \right) \quad (13)$$

The seven sources are not ad hoc. The Pre-LN residual stream contributes three non-attention sources ($\Delta P_{\text{initial}}, \Delta P_{\text{ffn}}, \Delta P_{\text{LN}}$), and the attention contribution is partitioned into four sources ($\{\texttt{Qry}, \texttt{RAG}, \texttt{Past}, \texttt{Self}\}$) by the causal attention range into disjoint, exhaustive index sets. Coarser groupings (e.g., merging Past and Self) would collapse the current-vs-prior asymmetry that prior work links to RAG hallucination signals.

By aggregating these components, we achieve a complete partition of the final probability $P_{\text{final}}(y)$ into seven distinct sources:

$$P_{\text{final}}(y) = \Delta P_{\text{initial}}(y) + \Delta P_{\text{LN}} + \sum_{l=1}^{L} \left(\Delta P_{\text{ffn}}^{(l)} + \sum_{S \in \mathcal{S}} \Delta P_S^{(l)} \right) \quad (14)$$

The Guiding Question: We have now derived a 7-dimensional attribution vector for every token. However, raw attribution scores lack context: a high FFN contribution might be normal for a function word but suspicious for a proper noun. How to contextualize these scores with syntactic priors?

3.4 Syntax-Aware Feature Engineering

To resolve this ambiguity, we employ Part-of-Speech (POS) tagging as a lightweight syntactic prior. Specifically, we assign a POS tag by Spacy (Honnibal et al., 2020) to each generated token and aggregate the attribution scores for each grammatical category. By profiling which attribution sources (e.g., RAG) the LLM relies on for different parts of speech, we detect hallucination effectively.

3.4.1 Tag Propagation Strategy

A mismatch problem arises because LLMs may split a single word into multiple tokens while standard POS taggers process whole words. We resolve this granularity issue via tag propagation: generated sub-word tokens inherit the POS tag of their parent word. For instance, if the noun "modification" is tokenized into [modi, fication], both sub-tokens are assigned the NOUN tag.

3.4.2 Aggregation

We first define the attribution vector $\mathbf{v}_t \in \mathbb{R}^7$ for each token y_t as the concatenation of its 7 source components derived in Section 3.3.4. Then we compute the mean attribution for each POS tag τ:

$$\bar{\mathbf{v}}_\tau = \frac{\sum_{t:\text{POS}(y_t)=\tau} \mathbf{v}_t}{|\{t \mid \text{POS}(y_t) = \tau\}|} \quad (15)$$

The final feature vector $\mathbf{f} \in \mathbb{R}^{7 \times |\text{POS}|}$ is the concatenation of these POS-specific vectors. This feature combines source attribution with linguistic structure, forming a basis for hallucination detection.

4 Experiments

4.1 Experimental Setup

We treat hallucination detection as a supervised binary classification task.[1] We employ an ensemble of 5 XGBoost (Chen and Guestrin, 2016) classifiers initialized with different random seeds. The input is a 126-dimensional syntax-aware feature vector, constructed by aggregating the 7-source attribution scores across 18 universal POS tags (e.g., NOUN, VERB) defined by SpaCy (Honnibal et al., 2020). For Qwen3-8B (Yang et al., 2025), we generate new responses on RAGTruth's queries and retrieved contexts and use Claude Opus 4.6 as an LLM-as-judge to determine whether each response is hallucinated. We validate this judge on RAGTruth's 450 human-annotated Llama2-7B

[1] Code is available at https://github.com/pengqian-lu/TPA.

Method	RAGTruth								
	LLaMA2-7B			LLaMA2-13B			LLaMA3-8B		
Metric	**AUC**	**Recall**	**F1**	**AUC**	**Recall**	**F1**	**AUC**	**Recall**	**F1**
ChainPoll (Friel and Sanyal, 2023)	0.6738	0.7832	0.7066	0.7414	0.7874	0.7342	0.6687	0.4486	0.5813
RAGAS (Es et al., 2024)	0.7290	0.6327	0.6667	0.7541	0.6763	0.6747	0.6776	0.3909	0.5094
Trulens (TrueLens, 2024)	0.6510	0.6814	0.6567	0.7073	0.7729	0.6867	0.6464	0.3909	0.5053
RefCheck (Hu et al., 2024)	0.6912	0.6280	0.6736	0.7857	0.6800	0.7023	0.6014	0.3580	0.4628
EigenScore (Chen et al., 2024)	0.6045	0.7469	0.6682	0.6640	0.6715	0.6637	0.6497	0.7078	0.6745
SEP (Han et al., 2024)	0.7143	0.7477	0.6627	0.8089	0.6580	0.7159	0.7004	0.7333	0.6915
ITI (Li et al., 2023)	0.7161	0.5416	0.6745	0.8051	0.5519	0.6838	0.6534	0.6850	0.6933
ReDeEP (Sun et al., 2025)	0.7458	0.8097	0.7190	0.8244	0.7198	0.7587	0.7285	0.7819	0.6947
TSV (Park et al., 2025)	0.6609	0.5526	0.6632	0.8123	**0.8068**	0.6987	0.7769	0.5546	0.6442
Novo (Ho et al., 2025)	0.7608	0.8274	0.7057	0.8506	0.7826	0.7733	**0.8258**	0.7737	0.7801
TPA	**0.7873**†	**0.8328**	**0.7238**†	**0.8681**†	0.7913	**0.7975**†	0.8211	**0.7860**	**0.7843**

Method	Dolly (AC)								
	LLaMA2-7B			LLaMA2-13B			LLaMA3-8B		
Metric	**AUC**	**Recall**	**F1**	**AUC**	**Recall**	**F1**	**AUC**	**Recall**	**F1**
ChainPoll (Friel and Sanyal, 2023)	0.6593	0.4138	0.5581	0.7070	0.4364	0.6000	0.6114	0.3415	0.4516
RAGAS (Es et al., 2024)	0.6648	0.5345	0.6392	0.6412	0.4182	0.5476	0.6870	0.8000	0.5246
Trulens (TrueLens, 2024)	0.7110	0.5517	0.6667	0.6521	0.3818	0.4941	0.7040	0.3659	0.5172
RefCheck (Hu et al., 2024)	0.6494	0.3966	0.5412	0.6626	0.2545	0.3944	0.5260	0.1951	0.2759
EigenScore (Chen et al., 2024)	0.6786	0.7500	0.7241	0.7214	0.8181	0.7200	0.6612	0.7142	0.5952
SEP (Han et al., 2024)	0.6067	0.6216	0.7023	0.7098	0.6545	0.6923	0.5159	0.6829	0.5385
ITI (Li et al., 2023)	0.5492	0.5816	0.6281	0.5511	0.5385	0.6712	0.5011	0.3091	0.4250
ReDeEP (Sun et al., 2025)	**0.7949**	0.8245	**0.7833**	**0.8420**	0.8518	0.7603	0.7354	**0.8392**	0.7100
TSV (Park et al., 2025)	0.7454	**0.8728**	0.7684	0.7552	0.5952	0.6043	0.7347	0.6467	0.6695
Novo (Ho et al., 2025)	0.6423	0.8070	0.7244	0.6909	0.7222	0.6903	0.7418	0.5854	0.6316
TPA	0.7134	0.7897	0.7527	0.8159	**0.9741**†	**0.8075**†	**0.7608**†	0.6561	**0.7529**†

Table 1: Results on RAGTruth and Dolly (AC). TPA results are averaged over 5 random seeds. The dagger symbol † indicates statistically significant improvement ($p < 0.05$) over the strongest baseline. Bold values indicate the best performance and underlined values indicate the second-best. Full results in Appendix.

Method	**F1**	**AUC**	**Recall**
TSV (Park et al., 2025)	0.6764	0.7972	0.5538
Novo (Ho et al., 2025)	0.8000	0.8419	0.8765
TPA	**0.8702**†	**0.9096**†	**0.9200**†

Table 2: Performance comparison on the RAGTruth dataset using the Mistral-7B model. **Bold** indicates the best performance, and † indicates statistically significant improvement over the strongest baseline ($p < 0.05$). The TPA results are averaged over 5 runs.

responses, where it attains 0.82 accuracy, 0.83 F1, and 0.64 Cohen's κ. See more implementation details in Appendix C.

4.2 Dataset and Baselines

To ensure a fair comparison, we utilize the public RAG hallucination benchmark established by Sun et al. (2025). This benchmark consists of the Dolly (AC) and RAGTruth datasets. The former includes responses from Llama2 (7B/13B) and Llama3 (8B), while the latter covers these models in addition to Mistral-7B. Implementation details are provided in Appendix. We compare our method against representative approaches from three categories introduced in Section 2. The introduction of baselines is provided in Appendix. For Mistral-7B, we compare against TSV and Novo. ReDeEP's Copying Heads were identified only on LLaMA-family backbones in the original release, with no head selection provided for Mistral. Other probing baselines such as SEP and ITI release training code but no Mistral-7B checkpoints, requiring full per-model probe or steering-direction re-fitting. Dolly does not include Mistral-7B responses, so Mistral is evaluated on RAGTruth only.

4.3 Comparison with Baselines

We evaluate TPA against baselines on RAGTruth (Llama2, Llama3, Mistral, Qwen3-8B) and Dolly (AC) (Llama2, Llama3). As shown in Table 1 and Table 2, TPA is competitive across benchmarks.

On **RAGTruth**, TPA achieves statistically significant Rank-1 results ($p < 0.05$) on Llama2-7B

Method	F1	AUC	Recall
TSV (Park et al., 2025)	0.5493	0.7710	0.6259
Novo (Ho et al., 2025)	0.5468	0.7919	0.6259
TPA	**0.6006**†	**0.8236**†	**0.7130**†

Table 3: Performance comparison on the RAGTruth dataset using the Qwen3-8B model. **Bold** indicates the best performance, and † indicates statistically significant improvement over the strongest baseline ($p < 0.05$). The TPA results are averaged over 5 runs.

and Llama2-13B for both F1 and AUC. The largest improvement appears on Mistral-7B, where TPA reaches 0.8702 F1, outperforming Novo by 7%, indicating good transfer to newer architectures with sliding-window attention. On Llama3-8B, TPA ranks first in F1 and Recall but is statistically comparable to the strongest baselines, suggesting a smaller margin on newer models. We also report results on Qwen3-8B (Yang et al., 2025) to verify generalization to a newer-generation backbone. As shown in Table 3, TPA again outperforms Novo and TSV on F1, AUC, and Recall.

On **Dolly (AC)**, TPA ranks third in AUC and F1 on Llama2-7B. On Llama2-13B, it secures significant Rank-1 Recall and F1, while its AUC is second to ReDeEP. On Llama3-8B, it achieves significant Rank-1 AUC and F1.

Runtime. TPA's attribution pipeline adds three post-hoc stages (probe decomposition, head-wise attribution, source mapping) with total complexity $\mathcal{O}(LT|\mathcal{V}|d + LTd^2 + LHT^2)$. On an A100-40GB, feature extraction takes about 20 seconds per response in our sequential implementation ($\sim$17 GPU-hours per LLM per dataset). Classifier inference is under one second per response. See Appendix E for the full derivation.

5 Ablation Study Analysis

We conduct an ablation study on RAGTruth to validate TPA (Figure 4). The full method generally achieves the best performance. Removing core components like RAG or FFN consistently degrades accuracy (e.g., a 3.01% F1 drop on Llama-2-7B), confirming the importance of the retrieval-parameter conflict. Crucially, previously overlooked sources are distinctively vital. For instance, removing LN causes a sharp 5.83% drop on Llama-3-8B. While excluding specific components yields marginal gains in several cases (e.g., Self on Mistral-7B), we retain the complete feature set to maintain a unified framework robust across diverse architectures. We exclude Dolly as its small sample size makes fine-grained feature evaluation unstable.

We further ablate the Part-of-Speech aggregation step in Section 3.4. **TPA-POS** denotes our default features that aggregate the seven attribution sources separately within each POS tag. We compare it against two simpler aggregations that pool over all tokens regardless of syntactic category: **TPA-Mean** takes the mean of each source over all tokens in a response, yielding a 7-dimensional feature, while **TPA-Stat** additionally appends the per-source standard deviation, yielding 14 dimensions. As shown in Table 5 in the Appendix, TPA-POS attains the best F1 in all seven backbone-dataset settings, with gains of 1.8–4.6% F1 over Mean on RAGTruth and up to 22.5% on Dolly (LLaMA2-13B). Mean and Stat erase the source-POS interactions that the syntax-conditioned aggregation preserves.

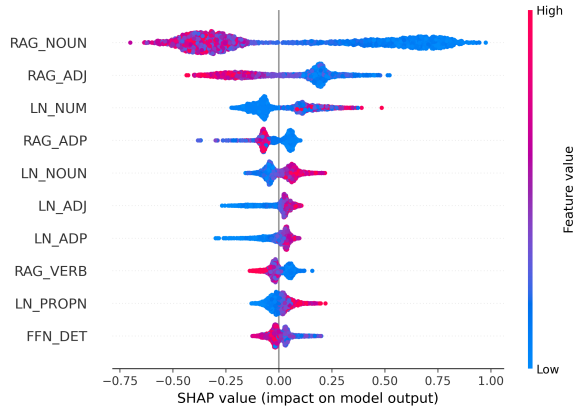
6 Interpretability Analysis

We apply SHAP analysis to classifiers trained on RAGTruth to validate our design principles. Results for Llama2 are shown in Figure 3, while results for Llama3-8B and Mistral-7B are detailed in Appendix due to space constraints. We obtain three observations from this analysis.

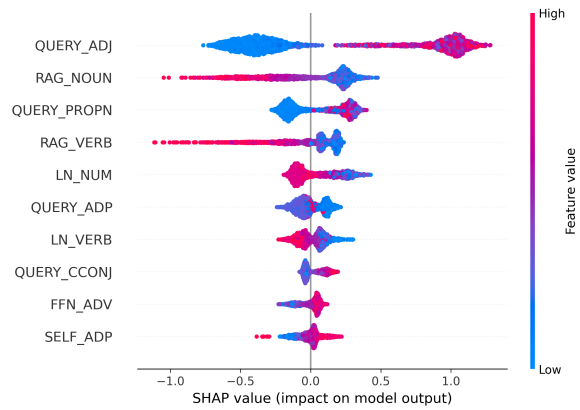
Observation 1: Fine-grained attribution is necessary. Relying solely on the binary conflict between internal FFN knowledge and external RAG context is insufficient for robust detection. As shown in Figure 3, the classifier frequently depends on features derived from other components. For instance, the feature LN_NUM plays a decisive role in Llama2-7B. This pattern indicates that the specific contribution from the Final LayerNorm to Numeral tokens serves as a critical signal. Similarly, query-based features like QUERY_NOUN appear as top predictors in Mistral-7B. These findings confirm that accurate hallucination detection requires a complete decomposition of the generative process.

Observation 2: Syntactic aggregation captures model-specific grounding logic. Different architectures ground information via distinct syntactic structures. While RAG_NOUN dominates in Llama2 (Figure 3) and Mistral, Llama3-8B relies heavily on relational structures, with RAG_ADP (Adpositions) ranking highest. TPA's POS-based aggregation is thus essential to capture these diverse patterns.

Observation 3: Hallucination fingerprints vary



(a) Llama2-7B



(b) Llama2-13B

Figure 3: SHAP summary plots illustrating the decision logic. We visualize the top-10 features for classifiers trained on the RAGTruth subsets corresponding to Llama2-7B and Llama2-13B. Plots for Llama3-8B and Mistral-7B are provided in Appendix. The x-axis represents the SHAP value. Positive values indicate a push towards classifying the response as a Hallucination. The color represents the feature value (Red = High attribution, Blue = Low).

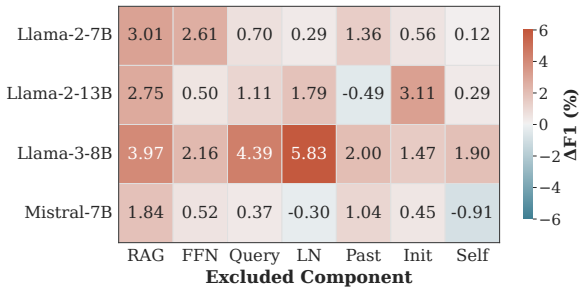


Figure 4: F1 Score Drop by Removing Components.

across architectures. Our analysis shows that hallucination signals are model-specific rather than universal. A clear example is LN_NUM: high attribution is a strong hallucination signal in Llama2-7B, but this pattern reverses in Llama2-13B (Figure 3b), where higher values correlate with factuality (negative SHAP values). This reversal suggests that larger models may use LayerNorm differently to regulate output distributions, motivating a learnable, syntax-aware detector.

7 Conclusion and Future Work

We introduced TPA to attribute token probability into seven distinct sources. By combining these with syntax-aware features, our framework effectively detects RAG hallucinations and outperforms baselines. Our results show that hallucination signals vary across models. This confirms the need for a learnable approach rather than static heuristics.

Future work will proceed in four directions. First, extending TPA to phrase- or span-level attribution for efficiency. Second, active mitigation via online monitoring and intervention on risky patterns (e.g., abnormal FFN or LN reliance). Third, studying TPA under noisy or adversarial retrieval, where expected source shifts may themselves serve as detection signals. Fourth, extending POS-based aggregation to non-English and specialized domains (code, scientific text) via multilingual spaCy or AST node types.

Limitations

Our framework presents three limitations. First, it relies on white-box access to model internals, preventing application to closed-source APIs. Second, the decomposition incurs higher computational overhead than simple scalar probes. Third, our feature engineering depends on external linguistic tools (POS taggers), which may limit generalization to specialized domains like code generation where standard syntax is less defined.

Acknowledgements

This work was supported by the Australian Research Council through the Laureate Fellow Project under Grant FL190100149.

References

Amos Azaria and Tom Mitchell. 2023. The internal state of an llm knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2022. Discovering latent knowledge in lan-

guage models without supervision. *arXiv preprint arXiv:2212.03827*.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. Inside: Llms' internal states retain the power of hallucination detection.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Yuyan Chen, Zehao Li, Shuangjie You, Zhengyu Chen, Jingwen Chang, Yi Zhang, Weinan Dai, Qingpei Guo, and Yanghua Xiao. 2025. Attributive reasoning for hallucination diagnosis of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23660–23668.

Roi Cohen, May Hamri, Mor Geva, and Amir Globerson. 2023. Lm vs lm: Detecting factual errors via cross examination. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12621–12640.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158.

Robert Friel and Atindriyo Sanyal. 2023. Chainpoll: A high efficacy method for llm hallucination detection. *arXiv preprint arXiv:2310.18344*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Nuno M Guerreiro, Elena Voita, and André FT Martins. 2023. Looking for a needle in a haystack: A comprehensive study of hallucinations in neural machine translation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1059–1075.

Jiatong Han, Jannik Kossen, Muhammed Razzak, Lisa Schut, Shreshth A Malik, and Yarin Gal. 2024. Semantic entropy probes: Robust and cheap hallucination detection in llms. In *ICML 2024 Workshop on Foundation Models in the Wild*.

Zheng Yi Ho, Siyuan Liang, Sen Zhang, Yibing Zhan, and Dacheng Tao. 2025. Novo: Norm voting off hallucinations with attention heads in large language models. In *The Thirteenth International Conference on Learning Representations*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, Adriane Boyd, and 1 others. 2020. spacy: Industrial-strength natural language processing in python.

Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Guo, Tianhang Zhang, Yang Xu, Yun Luo, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. Refchecker: Reference-based fine-grained hallucination checker and benchmark for large language models. *arXiv preprint arXiv:2405.14486*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Hakyung Lee, Keon-Hee Park, Hoyoon Byun, Jeyoon Yeom, Jihee Kim, Gyeong-Moon Park, and Kyungwoo Song. 2024. Ced: Comparing embedding differences for detecting out-of-distribution and hallucinated text. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14866–14882.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475.

Andrey Malinin and Mark Gales. 2021. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 9004–9017.

Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. 2019. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209.

Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Randy Zhong, Juntong Song, and Tong Zhang. 2024. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10862–10878.

nostalgebraist. 2020. interpreting gpt: the logit lens. LessWrong.

Seongheon Park, Xuefeng Du, Min-Hsuan Yeh, Haobo Wang, and Yixuan Li. 2025. Steer LLM latents for hallucination detection. In *Forty-second International Conference on Machine Learning*.

Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J Liu. 2023. Out-of-distribution detection and selective generation for conditional language models. In *The Eleventh International Conference on Learning Representations*.

Adi Simhi, Itay Itzhak, Fazl Barez, Gabriel Stanovsky, and Yonatan Belinkov. 2025. Trust me, i'm wrong: Llms hallucinate with certainty despite knowing the answer. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 14665–14688.

Hsuan Su, Ting-Yao Hu, Hema Swetha Koppula, Kundan Krishna, Hadi Pouransari, Cheng-Yu Hsieh, Cem Koc, Joseph Yitan Cheng, Oncel Tuzel, and Raviteja Vemulapalli. 2025. Learning to reason for hallucination span detection. *arXiv preprint arXiv:2510.02173*.

Zhongxiang Sun, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Weijie Yu, Yang Song, and Han Li. 2025. Redeep: Detecting hallucination in retrieval-augmented generation via mechanistic interpretability. In *ICLR*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

TrueLens. 2024. Truelens: Evaluate and track llm applications.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yakir Yehuda, Itzik Malkiel, Oren Barkan, Jonathan Weill, Royi Ronen, and Noam Koenigstein. 2024. Interrogatellm: Zero-resource hallucination detection in llm-generated answers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9333–9347.

Tianhang Zhang, Lin Qiu, Qipeng Guo, Cheng Deng, Yue Zhang, Zheng Zhang, Chenghu Zhou, Xinbing Wang, and Luoyi Fu. 2023. Enhancing uncertainty-based hallucination detection with stronger focus. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 915–932.

A Proof of Theorem 1

We expand the right-hand side (RHS) of Eq. 8 by substituting the definitions of each term.

First, consider the summation term. By substituting $\Delta P_{\text{att}}^{(l)}$ and $\Delta P_{\text{ffn}}^{(l)}$, the intermediate probe value $\Phi(\mathbf{h}_{\text{mid}}^{(l)})$ cancels out within each layer:

$$\begin{aligned}
&\sum_{l=1}^{L}\left(\Delta P_{\text{att}}^{(l)} + \Delta P_{\text{ffn}}^{(l)}\right) \\
&= \sum_{l=1}^{L}([\Phi(\mathbf{h}_{\text{mid}}^{(l)}) \\
&\quad - \Phi(\mathbf{h}^{(l-1)})] + [\Phi(\mathbf{h}^{(l)}) - \Phi(\mathbf{h}_{\text{mid}}^{(l)})]) \\
&= \sum_{l=1}^{L}\left(\Phi(\mathbf{h}^{(l)}) - \Phi(\mathbf{h}^{(l-1)})\right)
\end{aligned}$$

This summation forms a telescoping series where adjacent terms cancel:

$$\begin{aligned}
&\sum_{l=1}^{L}(\Phi(\mathbf{h}^{(l)}) - \Phi(\mathbf{h}^{(l-1)})) \\
&= (\Phi(\mathbf{h}^{(1)}) - \Phi(\mathbf{h}^{(0)})) + \dots \\
&+ (\Phi(\mathbf{h}^{(L)}) - \Phi(\mathbf{h}^{(L-1)})) \\
&= \Phi(\mathbf{h}^{(L)}) - \Phi(\mathbf{h}^{(0)})
\end{aligned}$$

Now, substituting this result, along with the definitions of $\Delta P_{\text{initial}}$ and ΔP_{LN}, back into the full RHS expression:

$$\begin{aligned}
\text{RHS} &= \underbrace{\Phi(\mathbf{h}^{(0)})}_{\Delta P_{\text{initial}}} + \underbrace{(P_{\text{final}} - \Phi(\mathbf{h}^{(L)}))}_{\Delta P_{\text{LN}}} \\
&+ \underbrace{(\Phi(\mathbf{h}^{(L)}) - \Phi(\mathbf{h}^{(0)}))}_{\text{Summation}} = P_{\text{final}}
\end{aligned}$$

The RHS simplifies exactly to $P_{\text{final}}(y)$, which completes the proof.

B Proof of Proposition 1

Proof. We consider the l-th layer of the Transformer. Let $\mathbf{h}^{(l-1)} \in \mathbb{R}^d$ be the input hidden state vector at the target position n_t. The Multi-Head

Attention mechanism computes a residual update $\Delta\mathbf{h}$ by summing the outputs of H heads:

$$\Delta\mathbf{h} = \sum_{h=1}^{H} \mathbf{u}_h \tag{16}$$

where $\mathbf{u}_h \in \mathbb{R}^d$ is the projected output of the h-th head.

The probe function $\Phi(\mathbf{h}, y)$ computes the probability of the target token y by projecting the hidden state onto the vocabulary logits $\mathbf{z} \in \mathbb{R}^{|\mathcal{V}|}$ and applying the Softmax function:

$$\Phi(\mathbf{h}, y) = \frac{\exp(\mathbf{z}_y)}{\sum_{v\in\mathcal{V}} \exp(\mathbf{z}_v)}, \text{where } \mathbf{z}_v = \mathbf{h} \cdot \mathbf{w}_{U,v} \tag{17}$$

Here, $\mathbf{w}_{U,v}$ is the unembedding vector for token v from matrix $\mathbf{W}_U$. For brevity, let $p_y = \Phi(\mathbf{h}^{(l-1)}, y)$ denote the probability of the target token at the current state.

We approximate the change in probability, $\Delta P_{\text{att}}^{(l)}$, using a first-order Taylor expansion of Φ with respect to $\mathbf{h}$ around $\mathbf{h}^{(l-1)}$:

$$\begin{aligned}\Delta P_{\text{att}}^{(l)} &= \Phi(\mathbf{h}^{(l-1)} + \Delta\mathbf{h}, y) - \Phi(\mathbf{h}^{(l-1)}, y) \\ &\approx \nabla_{\mathbf{h}}\Phi(\mathbf{h}^{(l-1)}, y)^\top \cdot \Delta\mathbf{h} \\ &= \sum_{h=1}^{H} \left(\nabla_{\mathbf{h}}\Phi(\mathbf{h}^{(l-1)}, y)^\top \cdot \mathbf{u}_h\right)\end{aligned} \tag{18}$$

To compute the gradient $\nabla_{\mathbf{h}}\Phi$, we apply the chain rule through the logits z_v. The partial derivative of the Softmax output p_y with respect to any logit z_v is given by $p_y(\delta_{yv} - p_v)$, where δ is the Kronecker delta. The gradient of the logit z_v with respect to $\mathbf{h}$ is simply $\mathbf{w}_{U,v}$. Thus:

$$\begin{aligned}\nabla_{\mathbf{h}}\Phi &= \sum_{v\in\mathcal{V}} \frac{\partial\Phi}{\partial z_v}\frac{\partial z_v}{\partial\mathbf{h}} \\ &= \sum_{v\in\mathcal{V}} p_y(\delta_{yv} - p_v)\mathbf{w}_{U,v} \\ &= p_y(1-p_y)\mathbf{w}_{U,y} - \sum_{v\neq y} p_y p_v \mathbf{w}_{U,v}\end{aligned} \tag{19}$$

Substituting this gradient back into Eq. (18) for a specific head contribution term (denoted as Term_h):

$$\begin{aligned}\text{Term}_h &= \nabla_{\mathbf{h}}\Phi^\top \cdot \mathbf{u}_h \\ &= \underbrace{p_y(1-p_y)}_{\mathcal{G}^{(l)}}(\mathbf{w}_{U,y}^\top \cdot \mathbf{u}_h) - \\ &\quad \underbrace{\sum_{v\neq y} p_y p_v (\mathbf{w}_{U,v}^\top \cdot \mathbf{u}_h)}_{\mathcal{E}_h}\end{aligned} \tag{20}$$

We observe that the dot product $\mathbf{w}_{U,y}^\top \cdot \mathbf{u}_h$ is strictly equivalent to the scalar logit contribution $\Delta z_{h,y}^{(l)}$ defined in Eq. 10. The factor $\mathcal{G}^{(l)} = p_y(1-p_y)$ represents the gradient common to all heads, depending only on the layer input $\mathbf{h}^{(l-1)}$. Therefore, the contribution of head h is dominated by the linear term $\mathcal{G}^{(l)} \cdot \Delta z_{h,y}^{(l)}$, subject to the off-target error term $\mathcal{E}_h$. □

C Implementation Details

Environment and Models. All experiments were conducted on a computational node equipped with a single NVIDIA A100 (40GB) GPU and 200GB of RAM. Our software stack uses CUDA 12.8, Python 3.10, PyTorch 2.x, and HuggingFace Transformers 4.56.1. We evaluate our framework using three Large Language Models: Llama2-7b-chat, Llama2-13b-chat (Touvron et al., 2023), and Llama3-8b-instruct (Grattafiori et al., 2024).

Due to GPU memory constraints (40GB), we implement TPA using a sequential prefix-replay procedure (token-by-token), rather than a fully parallel teacher-forced pass. On our hardware, generating the full TPA feature vector for one response takes approximately 20 seconds on average. Since each dataset contains fewer than 3,000 samples, the total feature-extraction cost is on the order of $\sim$17 GPU-hours per dataset (per evaluated LLM).

Hallucination detection is performed using an ensemble of five XGBoost classifiers; inference is typically well below one second per response, and the total classification cost per dataset is only a few CPU-hours, negligible compared to feature extraction.

For POS tagging, we use spaCy with `en_core_web_sm` and disable NER.

Feature Extraction and Classifier. For each response, we extract the 7-dimensional attribution vector for every token and aggregate them based on 18 universal POS tags (e.g., NOUN, VERB) defined by the SpaCy library (Honnibal et al., 2020). This results in a fixed-size feature vector ($7 \times 18 = 126$ dimensions) for each sample.

Training and Evaluation Protocols. To ensure fair comparison and statistical robustness, we tailor our training strategies to the data availability of each dataset. We implement strict data isolation to prevent leakage. Crucially, to mitigate the variance

Algorithm 1 TPA Part I: Token Probability Attribution (Teacher-Forced Pass / sequential process)

Require: Transformer $\mathcal{M}$, Query Tokens $\mathbf{x}_{\text{qry}}$, Retrieved Context Tokens $\mathbf{x}_{\text{rag}}$
Ensure: Sequence of 7-source attribution vectors $\mathcal{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_T)$

Initialize contexts $\{\mathbf{C}_t\}_{t=1}^{T}$ and attribution storage $\mathcal{V} \leftarrow \emptyset$
$\mathbf{C}_1 \leftarrow [\mathbf{x}_{\text{qry}}, \mathbf{x}_{\text{rag}}]$
for position $t = 2 \ldots T$ **do**
  $\mathbf{C}_t \leftarrow [\mathbf{C}_{t-1}, y_{t-1}]$
end for
for position $t = 1 \ldots T$ **do**
  1. Forward Pass & State Caching
  Run the model on $\mathbf{C}_t$ and cache states at the last position $n_t = |\mathbf{C}_t|$.
  Cache $\mathbf{h}^{(0)}$, $\mathbf{h}_{\text{mid}}^{(l)}$, $\mathbf{h}^{(l)}$, and Attention Maps $\mathbf{A}^{(l)}$ for all layers $l \in [1, L]$.
  2. Coarse Decomposition (Residual Stream)
  Initialize $\mathbf{v}_t \in \mathbb{R}^7$ with zeros.
  $\mathbf{v}_t[\texttt{Init}] \leftarrow \text{Probe}(\mathbf{h}^{(0)}, y_t)$ {Contribution from initial embedding}
  $\mathbf{v}_t[\texttt{LN}] \leftarrow P_{\text{final}}(y_t) - \text{Probe}(\mathbf{h}^{(L)}, y_t)$ {Contribution from final LayerNorm}
  3. Layer-wise Attribution
  for layer $l = 1$ to L **do**
    $\mathbf{v}_t[\texttt{FFN}] \mathrel{+}= \text{Probe}(\mathbf{h}^{(l)}, y_t) - \text{Probe}(\mathbf{h}_{\text{mid}}^{(l)}, y_t)$
    $\Delta P_{\text{att}} \leftarrow \text{Probe}(\mathbf{h}_{\text{mid}}^{(l)}, y_t) - \text{Probe}(\mathbf{h}^{(l-1)}, y_t)$
    4. Fine-Grained Decomposition (Head & Source)
    for head $h = 1$ to H **do**
      Let $\mathbf{o}_h$ be the output vector of head h.
      Compute logit update: $\Delta z_h \leftarrow \left(\mathbf{o}_h \mathbf{W}_O^{(l,h)}\right) \cdot \mathbf{w}_{U,y_t}$
      Compute ratio $\omega_h \leftarrow \frac{\exp(\Delta z_h)}{\sum_j \exp(\Delta z_j)}$ {Logit-based apportionment}
      $\Delta P_h \leftarrow \Delta P_{\text{att}} \times \omega_h$
      for source $S \in \{\texttt{Qry}, \texttt{RAG}, \texttt{Past}, \texttt{Self}\}$ **do**
        Sum attention weights on indices of S: $a_{h,S} \leftarrow \sum_{k \in \mathcal{I}_S} \mathbf{A}_h^{(l)}[n_t, k]$
        $\mathbf{v}_t[S] \mathrel{+}= \Delta P_h \cdot a_{h,S}$
      end for
    end for
  end for
  Store $\mathbf{v}_t$ in attribution matrix $\mathcal{V}$.
end for
return $\mathcal{V}$, Generated Tokens $\mathbf{y}$

inherent in small-data scenarios, we adopt a Multi-Seed Ensemble Strategy. For every experiment, we repeat the entire evaluation process using 5 distinct outer random seeds. For each seed, we construct an ensemble of 5 XGBoost classifiers. The final prediction is derived via Hard Voting (majority rule) for binary classification metrics (F1, Recall) and Soft Voting (probability averaging) for AUC.

Protocol I: Standard Split (RAGTruth Llama2-7b/13b/Mistral). For datasets with official splits, we utilize the standard training and test sets.

- **Optimization:** We employ **Optuna** with a TPE sampler to optimize hyperparameters. We run 50 trials maximizing the F1-score using 5-fold Stratified Cross-Validation on the training set.
- **Training:** For each of the 5 outer seeds, we train an ensemble of 5 models. Each ensemble member is trained on a distinct 85%/15% split of the training data to facilitate diversity and Early Stopping (patience=50).
- **Evaluation:** Predictions are aggregated via voting on the held-out test set.

Protocol II: Stratified 20-Fold CV (RAGTruth Llama3-8b). As the Llama3-8b subset lacks a

Algorithm 2 TPA Part II: Syntax-Aware Feature Aggregation with Sub-word Tag Propagation

Require: Generated Tokens $\mathbf{y} = (y_1, \ldots, y_T)$, Attribution Vectors $\mathcal{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_T)$
Ensure: Syntax-Aware Feature Vector $\mathbf{f} \in \mathbb{R}^{126}$

1. String Reconstruction & Alignment Map
Decode tokens $\mathbf{y}$ into a complete string `text`.
Construct an alignment map M where $M[i]$ contains the list of token indices corresponding to the i-th word in `text`.
2. POS Tagging & Propagation
Initialize tag list $\mathcal{T}$ of length T.
Run POS tagger (e.g., SpaCy) on string `text` to obtain words $W_1, \ldots, W_K$ and tags $\text{tag}_1, \ldots, \text{tag}_K$.
for each word index $k = 1$ to K **do**
  Get corresponding token indices: $\mathcal{I}_{\text{tokens}} \leftarrow M[k]$
  Get POS tag for the word: $c \leftarrow \text{tag}_k$
  for each token index $t \in \mathcal{I}_{\text{tokens}}$ **do**
    $\tau_t \leftarrow c$ {Propagate parent word's tag to all sub-word tokens}
  end for
end for
3. Syntax-Aware Aggregation
Initialize feature vector $\mathbf{f} \leftarrow \emptyset$.
Define set of Universal POS tags $\mathcal{P}_{\text{univ}}$.
for each POS category $c \in \mathcal{P}_{\text{univ}}$ **do**
  Identify tokens belonging to this category: $\mathcal{I}_c = \{t \mid \tau_t = c\}$
  if $\mathcal{I}_c \neq \emptyset$ **then**
    // Compute mean attribution profile for this syntactic category
    $\bar{\mathbf{v}}_c \leftarrow \frac{1}{|\mathcal{I}_c|} \sum_{t \in \mathcal{I}_c} \mathbf{v}_t$
  else
    $\bar{\mathbf{v}}_c \leftarrow \mathbf{0}_7$ {Fill with zeros if category is absent in response}
  end if
  $\mathbf{f} \leftarrow \text{Concatenate}(\mathbf{f}, \bar{\mathbf{v}}_c)$
end for
return $\mathbf{f}$

training split, we adopt a Stratified 20-Fold Cross-Validation.

- **Optimization:** Hyperparameters are optimized via Optuna on the available data prior to the cross-validation loop.
- **Training:** We iterate through 20 folds. Within each fold, we train the 5-member XGBoost ensemble on the training partition (using diverse internal splits for early stopping).
- **Aggregation:** Predictions from all folds are concatenated to compute the final performance metrics for each outer seed.

Protocol III: Nested Leave-One-Out CV (Dolly). Given the limited size of the Dolly dataset ($N = 100$), we implement a rigorous Nested Leave-One-Out (LOO) Cross-Validation.

- **Outer Loop:** We iterate 100 times, isolating a single sample for testing in each iteration.
- **Inner Loop (Optimization):** On the remaining 99 samples, we conduct independent hyperparameter searches using Optuna (50 trials).
- **Ensemble Training:** For each LOO step, we train 5 XGBoost models on the 99 training samples. To handle class imbalance, we dynamically adjust the `scale_pos_weight` parameter.
- **Inference:** The final verdict for the single test sample is determined by the hard vote of the 5 ensemble members. This process is repeated for all 5 outer seeds to verify statistical significance.

Implementation Note regarding Memory Constraints. While TPA is theoretically designed for single-pass parallel execution via teacher forcing, storing the full attention matrices $\mathcal{O}(T^2)$ and computational graphs for long sequences can be memory-intensive. In our specific experiments, due to GPU memory limitations (NVIDIA A100 40G), we implemented the attribution process sequentially (token-by-token). We emphasize that this implementation is mathematically equivalent to the parallel version due to the causal masking mechanism of Transformer decoders. The choice between serial and parallel implementation represents a trade-off between efficiency and memory usage, without affecting the attribution values or detection performance reported in this paper.

Hyperparameter Search and Best Value Discussion. We utilize the Optuna framework with a Tree-structured Parzen Estimator (TPE) sampler to perform automated hyperparameter tuning. For each model and data split, we run 50 trials to maximize the F1-score. The comprehensive search space is presented in Table 4. Regarding the best-found values, our analysis reveals that the optimal configuration is highly dependent on the specific LLM and dataset size. We observed a consistent preference for moderate tree depths ($4 \leq$ max_depth ≤ 6) and stronger regularization ($\lambda \geq 1.5, \gamma \geq 0.2$) across most experiments, indicating that preventing overfitting is critical given the high dimensionality of our feature space relative to the dataset size. Conversely, the optimal learning rate varied significantly (0.01 to 0.1) depending on the base model (e.g., Llama-2 vs. Llama-3). Therefore, rather than fixing a single set of hyperparameters, we adopt a dynamic optimization strategy where the best parameters are re-evaluated for each fold and seed. This approach ensures that our reported results reflect the robust performance of the method rather than a specific tuning artifact.

Qwen3-8B Setup. Responses are generated with greedy decoding (`temperature=0.0`, `enable_thinking=False`) using Qwen3's native chat template without a system prompt, matching RAGTruth's decoding configuration. TPA attribution is computed with a separate fork of HuggingFace Transformers 4.55.0 (the version with native Qwen3 support) that applies the Pre-LN probe and per-head source decomposition to `Qwen3Attention` and `Qwen3DecoderLayer`.

Table 4: Hyperparameter search space for the XGBoost classifier in TPA.

Hyperparameter	**Search Values**
Learning Rate	{0.01, 0.02, 0.05, 0.1}
Max Depth	{4, 5, 6, 7}
Subsample	{0.6, 0.7, 0.8}
Colsample By Tree	{0.7, 0.8, 0.9}
Gamma (γ)	{0.1, 0.2, 0.5}
Reg Alpha (α)	{0.01, 0.1, 0.5}
Reg Lambda (λ)	{1, 1.5, 2}
Fixed Parameters	n_estimators=1000, patience=50

We use the same 126-dimensional feature vector (18 POS $\times$ 7 contributions) and 5-seed $\times$ 5-ensemble XGBoost protocol as for Llama. Because the Qwen3-8B response set on RAGTruth is class-imbalanced toward the negative (no-hallucination) label, we set `scale_pos_weight` to the negative/positive ratio on the training set to maintain comparable class balance during optimization.

Backbone	**TPA-POS**	**TPA-Mean**	**TPA-Stat**
RAGTruth			
LLaMA2-7B	**0.7238**	0.7055	0.7162
LLaMA2-13B	**0.7975**	0.7513	0.7645
LLaMA3-8B	**0.7843**	0.7458	0.7541
Mistral-7B	**0.8702**	0.8490	0.8521
Dolly (AC)			
LLaMA2-7B	**0.7527**	0.6769	0.7206
LLaMA2-13B	**0.8075**	0.5828	0.6009
LLaMA3-8B	**0.7529**	0.5947	0.5869

Table 5: F1 score of TPA with different aggregation strategies (5-seed average). TPA-POS wins in all seven settings.

Artifacts and intended use. We use publicly available benchmarks (RAGTruth and Dolly (AC)) and open-access LLM checkpoints strictly for research evaluation, consistent with their intended research use. We do not redistribute any original datasets or model weights; our released artifact is research code and documentation for reproducing the experiments, and it instructs users to obtain the datasets/models from their official sources.

D Baselines Introduction

1. **EigenScore/INSIDE**(Chen et al., 2024) Focuses on detecting hallucination by evaluating the response's semantic consistency, defined as the log-determinant of the covariance matrix of the LLM's internal states during re-

sponse generation.

2. **SEP**(Han et al., 2024) Proposes a linear model to detect hallucination based on semantic entropy at test time without requiring multiple responses.

3. **SAPLMA**(Azaria and Mitchell, 2023) Detecting hallucination based on the hidden layer activations of LLMs.

4. **ITI**(Li et al., 2023) Detecting hallucination based on the hidden layer activations of LLMs.

5. **Ragtruth Prompt**(Niu et al., 2024) Provides prompts for an LLM-as-judge to detect hallucination in the RAG setting.

6. **ChainPoll**(Friel and Sanyal, 2023) Provides prompts for an LLM-as-judge to detect hallucination in the RAG setting.

7. **RAGAS**(Es et al., 2024) Uses an LLM to split the response into a set of statements and verify whether each statement is supported by the retrieved documents. If any statement is not supported, the response is considered hallucinated.

8. **Trulens**(TrueLens, 2024) Evaluating the overlap between the retrieved documents and the generated response to detect hallucination by a LLM.

9. **P(True)**(Kadavath et al., 2022) The paper detects hallucinations by having the model estimate the probability that its own generated answer is correct, based on the key assumption that it is often easier for a model to recognize a correct answer than to generate one.

10. **SelfCheckGPT**(Manakul et al., 2023) SelfCheckGPT detects hallucinations by checking for informational consistency across multiple stochastically sampled responses, based on the assumption that factual knowledge leads to consistent statements while hallucinations lead to divergent and contradictory ones.

11. **LN-Entropy**(Malinin and Gales, 2021) This paper detects hallucinations by quantifying knowledge uncertainty, which it measures primarily with a novel metric called Reverse Mutual Information that captures the disagreement across an ensemble's predictions, with high RMI indicating a likely hallucination.

12. **Energy**(Liu et al., 2020) This paper detects hallucinations by using an energy score, derived directly from the model's logits, as a more reliable uncertainty measure than softmax confidence to identify out-of-distribution inputs that cause the model to hallucinate.

13. **Focus**(Zhang et al., 2023) This paper detects hallucinations by calculating an uncertainty score focused on keywords, and then refines it by propagating penalties from unreliable context via attention and correcting token probabilities using entity types and inverse document frequency to mitigate both overconfidence and underconfidence.

14. **Perplexity**(Ren et al., 2023) This paper detects hallucinations by separately measuring the Relative Mahalanobis Distance for both input and output embeddings, based on the assumption that in-domain examples will have embeddings closer to their respective foreground (in-domain) distributions than to a generic background distribution.

15. **REFCHECKER**(Hu et al., 2024) Uses an LLM to extract claim triplets from a response and verifies them with another LLM to detect hallucination.

16. **REDEEP**(Sun et al., 2025) Detects hallucination by analyzing the balance between contributions from Copying Heads that process external context and Knowledge FFNs that inject internal knowledge, based on the finding that RAG hallucinations often arise from conflicts between these two sources. This method has two versions: token level and chunk level. We compare with the latter since it generally has better performance.

17. **NoVo**(Ho et al., 2025) It leverages the L2 norms of specific attention heads as reliable indicators of truthfulness. By identifying a subset of truth-correlated heads from a small reference set, it employs a voting mechanism based on these head norms to detect hallucinations without requiring model parameter updates.

18. **TSV**(Park et al., 2025) It introduces a lightweight steering vector to reshape the

LLM's latent space during inference. By actively intervening to enhance the linear separability between truthful and hallucinated representations in the hidden states, it enables effective detection using a simple classifier on the steered embeddings.

E Complexity Analysis of the Attribution Process

In this section, we rigorously analyze the computational overhead of our attribution framework. We report complexity in terms of analysis passes and asymptotic costs, as wall-clock varies substantially with caching strategies and kernel implementations. We focus strictly on the attribution extraction process for a generated response of length T. Let L, d, $|\mathcal{V}|$, and H denote the number of layers, hidden dimension, vocabulary size, and attention heads (per layer), respectively. The standard inference complexity for a Transformer is $\mathcal{O}(L \cdot T \cdot d^2 + L \cdot H \cdot T^2)$. Our attribution process introduces post-hoc computations, decomposed into three specific stages:

1. Complete Probability Decomposition. To satisfy Theorem 1, we must compute the complete probability changes using the probe function $\Phi(\mathbf{h}, y)$. The bottleneck is the calculation of the global partition function (denominator) in Softmax.

- **Mechanism:** The probe function $\Phi(\mathbf{h}, y) = \text{Softmax}(\mathbf{h}\mathbf{W}_U)_y = \frac{\exp(\mathbf{w}_{U,y}^\top \mathbf{h})}{\sum_{v \in \mathcal{V}} \exp(\mathbf{w}_{U,v}^\top \mathbf{h})}$ requires projecting the hidden state $\mathbf{h}$ to the full vocabulary logits $\mathbf{z} = \mathbf{h}\mathbf{W}_U$.
- **Single Probe Complexity:** For a single hidden state $\mathbf{h} \in \mathbb{R}^d$, the matrix-vector multiplication with the unembedding matrix $\mathbf{W}_U \in \mathbb{R}^{d \times |\mathcal{V}|}$ costs $\mathcal{O}(|\mathcal{V}| \cdot d)$.
- **Total Calculation:** We must apply this probe at multiple points:
  1. **Global Components:** For $\Delta P_{\text{initial}}$ and ΔP_{LN}, the probe is called once per generation step. Cost: $\mathcal{O}(T \cdot |\mathcal{V}| \cdot d)$.
  2. **Layer Components:** For $\Delta P_{\text{att}}^{(l)}$ and $\Delta P_{\text{ffn}}^{(l)}$, the probe is invoked twice per layer (before and after the residual update). Summing over L layers, this costs $\mathcal{O}(L \cdot T \cdot |\mathcal{V}| \cdot d)$.
- **Stage Complexity:** Combining these terms, the dominant complexity is $\mathcal{O}(L \cdot T \cdot |\mathcal{V}| \cdot d)$.

2. Head-wise Attribution. Once $\Delta P_{\text{att}}^{(l)}$ is obtained, we apportion it to individual heads based on their contribution to the target logit.

- **Mechanism:** This attribution requires projecting the target token vector $\mathbf{w}_{U,y}$ back into the hidden state space using the layer's output projection matrix $\mathbf{W}_O^{(l)} \in \mathbb{R}^{d \times d}$.
- **Step Complexity:** The calculation proceeds in two sub-steps:
  1. **Projection:** We compute the projected target vector $\mathbf{g} = (\mathbf{W}_O^{(l)})^\top \mathbf{w}_{U,y}$. Since $\mathbf{W}_O^{(l)}$ is a $d \times d$ matrix, this matrix-vector multiplication costs $\mathcal{O}(d^2)$.
  2. **Assignment:** We distribute the contribution to H heads by performing dot products between the head outputs $\mathbf{o}_h$ and the corresponding segments of $\mathbf{g}$. For H heads, this sums to $\mathcal{O}(d)$.
- **Stage Complexity:** The projection step ($\mathcal{O}(d^2)$) dominates the assignment step ($\mathcal{O}(d)$). Integrating over L layers and T tokens, the total complexity is $\mathcal{O}(L \cdot T \cdot d^2)$.

3. Mapping Attention to Input Sources. Finally, we map head contributions to the four sources by aggregating attention weights $\mathbf{A} \in \mathbb{R}^{H \times |\mathbf{s}| \times |\mathbf{s}|}$. This involves two distinct sub-steps for each generated token at step t within a single layer:

- **Step 1: Summation.** For each head h, we sum the attention weights corresponding to specific source indices (e.g., $\mathcal{I}_{\texttt{RAG}}$):

$$w_{h,S} = \sum_{k \in \mathcal{I}_S} \mathbf{A}_h[n_t, k]$$

  This requires iterating over the causal range up to n_t. For H heads, the cost is $\mathcal{O}(H \cdot n_t)$.
- **Step 2: Normalization & Weighting.** We calculate the final source contribution by weighting the head contributions:

$$\Delta P_S = \sum_{h=1}^{H} \Delta P_h \cdot \frac{w_{h,S}}{\sum_{\text{all sources}} w_{h,\cdot}}$$

  This involves scalar operations proportional to the number of heads H. Cost: $\mathcal{O}(H)$.

- **Stage Complexity:** The summation step ($\mathcal{O}(H \cdot n_t)$) dominates. We sum this cost across all L layers, and then accumulate over the generation steps $t = 1$ to T. The calculation is $\sum_{t=1}^{T}(L \cdot H \cdot n_t) \approx \mathcal{O}(L \cdot H \cdot T^2)$.

Overall Efficiency. The total computational cost is the sum of these three stages:

$$\mathcal{C}_{\text{total}} = \mathcal{O}(\underbrace{L \cdot T \cdot |\mathcal{V}| \cdot d}_{\text{Prob. Decomp.}} + \underbrace{L \cdot T \cdot d^2}_{\text{Head Attr.}} + \underbrace{L \cdot H \cdot T^2}_{\text{Source Map}})$$

Runtime Efficiency. It is worth noting that theoretical complexity does not directly equate to wall-clock latency. Standard text generation is *serial* (token-by-token), which limits GPU parallelization. In contrast, our framework can process the full sequence of length T in a single parallel teacher-forced pass, enabling efficient GPU matrix operations. When implemented this way, it avoids the K sequential generation passes required by baselines like SelfCheckGPT.

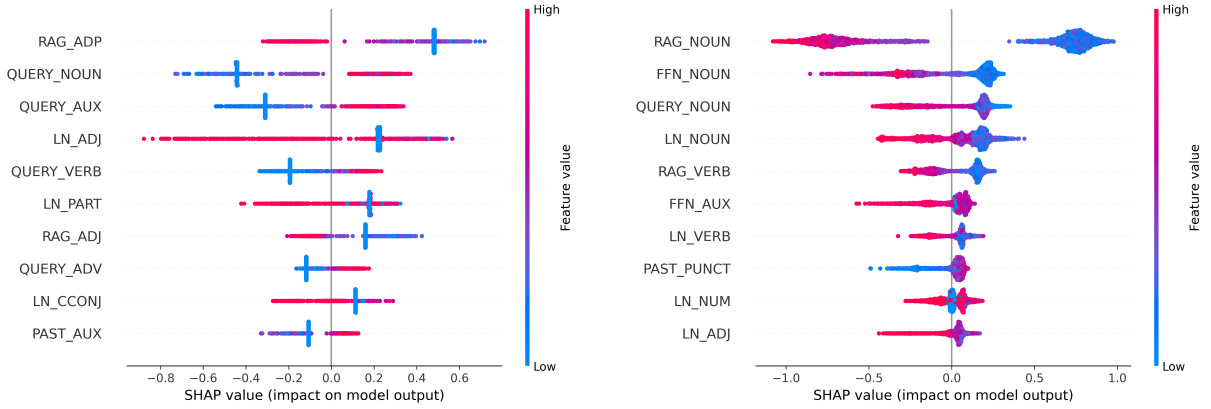
F AI Assistance Disclosure

We used AI-based tools to assist with language editing and draft refinement. All technical content, experiments, and conclusions were produced and verified by the authors.

	RAGTruth								
Model	LLaMA2-7B			LLaMA2-13B			LLaMA3-8B		
Metric	**AUC**	**Recall**	**F1**	**AUC**	**Recall**	**F1**	**AUC**	**Recall**	**F1**
SelfCheckGPT (Manakul et al., 2023)	—	0.3584	0.4642	—	0.3584	0.4642	—	0.4111	0.5111
Perplexity (Ren et al., 2023)	0.5091	0.5190	0.6749	0.5091	0.5190	0.6749	0.6235	0.6537	0.6778
LN-Entropy (Malinin and Gales, 2021)	0.5912	0.5383	0.6655	0.5912	0.5383	0.6655	0.7021	0.5596	0.6282
Energy (Liu et al., 2020)	0.5619	0.5057	0.6657	0.5619	0.5057	0.6657	0.5959	0.5514	0.6720
Focus (Zhang et al., 2023)	0.6233	0.5309	0.6622	0.7888	0.6173	0.6977	0.6378	0.6688	0.6879
Prompt (Niu et al., 2024)	—	0.7200	0.6720	—	0.7000	0.6899	—	0.4403	0.5691
ChainPoll (Friel and Sanyal, 2023)	0.6738	0.7832	0.7066	0.7414	0.7874	0.7342	0.6687	0.4486	0.5813
RAGAS (Es et al., 2024)	0.7290	0.6327	0.6667	0.7541	0.6763	0.6747	0.6776	0.3909	0.5094
Trulens (TrueLens, 2024)	0.6510	0.6814	0.6567	0.7073	0.7729	0.6867	0.6464	0.3909	0.5053
RefCheck (Hu et al., 2024)	0.6912	0.6280	0.6736	0.7857	0.6800	0.7023	0.6014	0.3580	0.4628
P(True) (Kadavath et al., 2022)	0.7093	0.5194	0.5313	0.7998	0.5980	0.7032	0.6323	0.7083	0.6835
EigenScore (Chen et al., 2024)	0.6045	0.7469	0.6682	0.6640	0.6715	0.6637	0.6497	0.7078	0.6745
SEP (Han et al., 2024)	0.7143	0.7477	0.6627	0.8089	0.6580	0.7159	0.7004	0.7333	0.6915
SAPLMA (Azaria and Mitchell, 2023)	0.7037	0.5091	0.6726	0.8029	0.5053	0.6529	0.7092	0.5432	0.6718
ITI (Li et al., 2023)	0.7161	0.5416	0.6745	0.8051	0.5519	0.6838	0.6534	0.6850	0.6933
ReDeEP (Sun et al., 2025)	0.7458	0.8097	0.7190	0.8244	0.7198	0.7587	0.7285	0.7819	0.6947
TSV (Park et al., 2025)	0.6609	0.5526	0.6632	0.8123	**0.8068**	0.6987	0.7769	0.5546	0.6442
Novo (Ho et al., 2025)	0.7608	0.8274	0.7057	0.8506	0.7826	0.7733	**0.8258**	0.7737	0.7801
TPA	**0.7873**†	**0.8328**	**0.7238**†	**0.8681**†	0.7913	**0.7975**†	0.8211	**0.7860**	**0.7843**
Std	0.0007	0.0145	0.0039	0.0075	0.0086	0.0076	0.0025	0.0068	0.0053
P-val	<0.001	$=0.227$	$=0.025$	$=0.003$	-	$=0.001$	-	$=0.125$	$=0.074$

	Dolly (AC)								
Model	LLaMA2-7B			LLaMA2-13B			LLaMA3-8B		
Metric	**AUC**	**Recall**	**F1**	**AUC**	**Recall**	**F1**	**AUC**	**Recall**	**F1**
SelfCheckGPT (Manakul et al., 2023)	—	0.1897	0.3188	—	0.1897	0.3188	—	0.2195	0.3600
Perplexity (Ren et al., 2023)	0.6825	0.7719	0.7097	0.6825	0.7719	0.7097	0.5924	0.3902	0.4571
LN-Entropy (Malinin and Gales, 2021)	0.7001	0.7368	0.6772	0.7001	0.7368	0.6772	0.6011	0.5365	0.5301
Energy (Liu et al., 2020)	0.6074	0.6316	0.6261	0.6074	0.6316	0.6261	0.5014	0.4047	0.5440
Focus (Zhang et al., 2023)	0.6783	0.5593	0.6534	0.7067	0.7333	0.6168	0.6177	0.6918	0.6874
Prompt (Niu et al., 2024)	—	0.3965	0.5476	—	0.4182	0.5823	—	0.3902	0.5000
ChainPoll (Friel and Sanyal, 2023)	0.6593	0.4138	0.5581	0.7070	0.4364	0.6000	0.6114	0.3415	0.4516
RAGAS (Es et al., 2024)	0.6648	0.5345	0.6392	0.6412	0.4182	0.5476	0.6870	0.8000	0.5246
Trulens (TrueLens, 2024)	0.7110	0.5517	0.6667	0.6521	0.3818	0.4941	0.7040	0.3659	0.5172
RefCheck (Hu et al., 2024)	0.6494	0.3966	0.5412	0.6626	0.2545	0.3944	0.5260	0.1951	0.2759
P(True) (Kadavath et al., 2022)	0.6011	0.6350	0.6509	0.6396	0.6180	0.5739	0.6871	0.5707	0.6573
EigenScore (Chen et al., 2024)	0.6786	0.7500	0.7241	0.7214	0.8181	0.7200	0.6612	0.7142	0.5952
SEP (Han et al., 2024)	0.6067	0.6216	0.7023	0.7098	0.6545	0.6923	0.5159	0.6829	0.5385
SAPLMA (Azaria and Mitchell, 2023)	0.5365	0.5714	0.7179	0.6053	0.6000	0.6923	0.5019	0.4040	0.5714
ITI (Li et al., 2023)	0.5492	0.5816	0.6281	0.5511	0.5385	0.6712	0.5011	0.3091	0.4250
ReDeEP (Sun et al., 2025)	**0.7949**	0.8245	**0.7833**	**0.8420**	0.8518	0.7603	0.7354	**0.8392**	0.7100
TSV (Park et al., 2025)	0.7454	**0.8728**	0.7684	0.7552	0.5952	0.6043	0.7347	0.6467	0.6695
Novo (Ho et al., 2025)	0.6423	0.8070	0.7244	0.6909	0.7222	0.6903	0.7418	0.5854	0.6316
TPA	0.7134	0.7897	0.7527	0.8159	**0.9741**†	**0.8075**†	**0.7608**†	0.6561	**0.7529**†
Std	0.0215	0.0227	0.0199	0.0210	0.0096	0.0137	0.0164	0.0452	0.0337
P-val	-	-	-	-	<0.001	<0.001	0.0025	-	$=0.001$

Table 6: Full Results on RAGTruth and Dolly (AC) datasets. TPA results are reported as the Mean and Standard Deviation over 5 random seeds, obtained using an ensemble of 5 XGBoost models. † indicates that the improvement over the strongest baseline is statistically significant ($p < 0.05$) under a one-sample t-test. P-values are reported for metrics where TPA achieves Rank 1; otherwise, a dash (-) is shown. Bold values indicate the best performance and underlined values indicate the second-best.



(a) Llama3-8B

(b) Mistral-7B

Figure 5: SHAP summary plots illustrating the decision logic. We visualize the top-10 features for classifiers trained on the RAGTruth subsets corresponding to Llama3-8B and Mistral-7B. The x-axis represents the SHAP value. Positive values indicate a push towards classifying the response as a Hallucination. The color represents the feature value (Red = High attribution, Blue = Low).