Hypergraph Modeling of Transformer Attention for Hallucination Detection

Mark Junjie Li [1,2,3] **Zhishun Liu** [1] **Wei Wang** [1*] **Yuanshan Lu** [1]

Jingxi Yang [1] **Xianyu Bao** [4] **Jun Li** [4] **Sunjie Huang** [4]

[1]Shenzhen University, Shenzhen, China
[2]Guangdong Provincial Key Laboratory of Visual Media and Multidimensional Intelligence, CSSE, Shenzhen University
[3]Intelligent Computing Center, Shenzhen University, Shenzhen, China
[4]Shenzhen Academy of Inspection and Quarantine, Shenzhen, China

Abstract

Large Language Models (LLMs) often generate factually unsupported content known as hallucination. However, existing detection approaches rely on handcrafted heuristics or isolated attention signals, limiting their ability to capture higher-order dependencies. In this work, we propose **AttnHyper**, a framework that represents a Transformer's attention patterns as a hypergraph for hallucination detection. Tokens are treated as nodes, while hyperedges connect groups of tokens that co-occur across heads and layers, with features derived from attention weights to preserve higher-order interactions beyond pairwise graphs. Based on this representation, the method formulates hallucination detection as a hypergraph learning problem and employs a hypergraph neural network (HGNN) to extract structural cues. It consistently outperforms state-of-the-art methods on the RAGTruth benchmark, achieving an average gain of +5.08% AUROC over the strongest baseline across diverse settings and architectures, while also demonstrating strong zero-shot transfer. Importantly, it incurs minimal overhead at inference time, enabling efficient deployment. Our results demonstrate that hypergraph-based attention modeling provides a more expressive and reliable signal for hallucination detection.The source code and implementation details are available at `https://github.com/liuzhishun/NeurIPS-2026-AttnHyper`.

1 Introduction

Large language models (LLMs) have achieved remarkable success across a wide range of applications, from question answering and summarization to code generation and scientific reasoning [1–3]. Despite this success, LLMs are known to occasionally generate factual inconsistencies or unsupported hallucinated content. Such hallucinations can have serious potential impacts, especially in high-risk application scenarios [4]. Ensuring the reliability of LLMs' outputs has thus made hallucination detection an indispensable component for trustworthy deployment.

However, existing hallucination detection approaches have significant limitations. One line of work uses response consistency, generating multiple answers for the same query and checking their agreement [5, 6]. This method requires multiple model runs, incurring high computational cost and latency. Another line of research trains classifiers with the hidden representations on the model's

*Corresponding author: `steveweiwang@szu.edu.cn`.

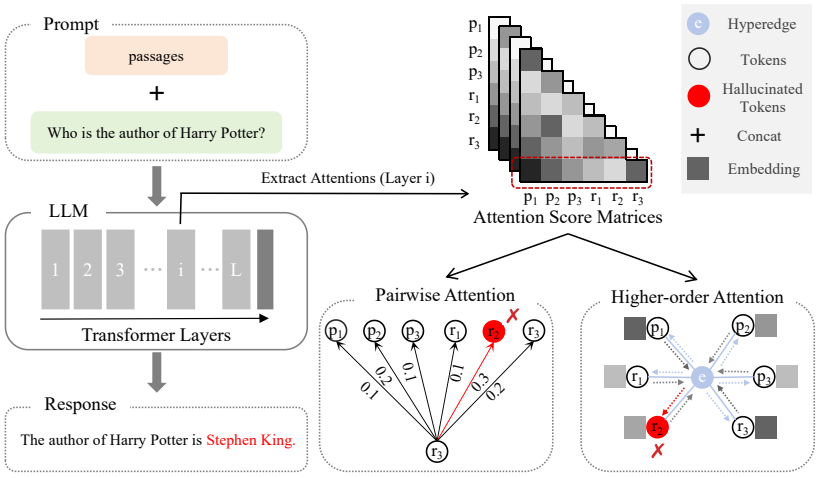


Figure 1. Pairwise vs. Higher-order Attention for Hallucination Detection

internal states obtained during a single generation [7–9]. This approach avoids repeated sampling but offers limited interpretability, and it often fails to catch subtle fact-consistent hallucinations that do not directly contradict known facts.

Recently, attention-based methods have been proposed to examine the Transformer's attention patterns between the input context and the generated content as signals for hallucination [10–12]. These methods use various heuristics, like attention weight distribution ratios or graph spectral metrics, as proxies for factual grounding. However, these methods rely heavily on handcrafted features and model only pairwise token-to-token attention relationships. As shown in Fig. 1, such pairwise modeling cannot capture the higher-order structural dependencies inherent in the attention mechanism, limiting effectiveness on long-context and multi-hop reasoning scenarios.

Furthermore, practical applications often require identifying hallucinated spans at the token-level to provide more interpretable explanations, while prior works have mainly focused on response-level hallucination detection [9]. As shown in Table 1, token-level hallucination detection can pinpoint 'Stephen King.' as hallucinated tokens, whereas response-level methods can only mark the entire response as hallucinated. Token-level is therefore more precise but also more challenging [4, 13], as it requires stronger fine-grained semantic modeling and careful contextual alignment [14, 15]. Although current methods have made some progress in token-level hallucination detection, their detection capability still leaves considerable room for improvement.

Table 1. Response-level vs. Token-level Hallucination Detection

Prompt: Who is the author of Harry Potter? **LLM Response**: The author of Harry Potter is Stephen King. **Ground-Truth**: The author of Harry Potter is J. K. Rowling. (✓)	
Response HD	The author of Harry Potter is Stephen King. (✗)
Token HD	The author of Harry Potter is Stephen King. (✗)

To address these limitations, we propose **AttnHyper**, a novel supervised learning framework for hallucination detection that is the first to represent Transformer attention patterns as a hypergraph. **AttnHyper** explicitly captures higher-order interactions in the attention mechanism via a hypergraph structure, learning token-level dependency patterns beyond pairwise relationships. In our hypergraph representation, each token serves as a node, and we introduce a single hyperedge connecting a given query token to the set of key tokens it attends to, annotated with features derived from the attention weights. This formulation naturally aligns with the multi-way nature of Transformer attention. When an LLM's attention head attends strongly to multiple tokens, those tokens are jointly linked via a hyperedge, preserving a higher-order dependency that pairwise attention graphs would lose. We cast hallucination detection as a hypergraph learning problem, enabling us to apply the Hypergraph Neural Networks (HGNN) [16, 17] with message passing over attention-induced hyperedges [18]. Through

this learnable hypergraph modeling of attention, **AttnHyper** can effectively aggregate signals across all layers and heads to capture intrinsic token relationships indicative of hallucinations, without relying on handcrafted heuristics.

Extensive experiments demonstrate the advantages of our method. **AttnHyper** achieves state-of-the-art performance on all three tasks of the RAGTruth hallucination benchmark [14], significantly outperforming prior methods. It shows outstanding performance on zero-shot task transfer. These results underscore the benefit of hypergraph's higher-order attention structures for hallucination detection. Our key contributions are summarized as follows:

- We introduce a principled **hypergraph** representation for LLMs' attention patterns, where tokens form nodes and multi-token attention groups form hyperedges. This formulation preserves the multi-way nature of attention that pairwise token graphs fundamentally lose.
- We propose **AttnHyper**, a hypergraph neural networks (HGNN) based framework that learns to detect token-level hallucinations via message passing over attention-induced hypergraphs. This unified approach directly leverages the Transformer's attention signals within a trainable architecture, rather than relying on handcrafted heuristics.
- We empirically demonstrate that **AttnHyper** consistently outperforms state-of-the-art detectors across multiple hallucination detection tasks. It also exhibits robust zero-shot transfer to new tasks and evaluation settings, highlighting strong generalization without any task-specific fine-tuning.

2 Related Work

Hallucination detection (HD) in LLMs is critical for reliable deployment [4, 13, 19]. Existing methods can be categorized by the signals they exploit. Sampling-based approaches estimate consistency across multiple generations via lexical overlap or semantic clustering [5, 6], but require multiple forward passes. Logit-based methods instead rely on single-pass uncertainty measures such as token confidence or perplexity [20], yet reduce sequences to scalar statistics and ignore structural dependencies.

To capture richer information, representation-based methods analyze internal model states [21, 7, 8, 22, 23], showing that hidden activations encode truthfulness signals. Complementary attention-based approaches exploit attention distributions, including context–generation ratios [10] and spectral properties of attention matrices [12]. Despite these advances, most existing methods still rely on point-wise or pairwise summaries, leaving higher-order structural interactions underexplored.

Attention-based Hallucination Detection. Recent work explores whether Transformer attention distributions can reveal hallucinations, under the hypothesis that factual grounding is reflected in how models attend to context versus generated tokens. For example, Lookback Lens [10] uses per-head attention ratios as a grounding signal, while other methods analyze structural or spectral properties of attention graphs [24, 12]. Some approaches further learn discriminative patterns from attention maps via supervised classification [11, 25].

However, these methods typically rely on handcrafted statistics or pairwise graph abstractions, which fail to fully capture higher-order interactions across heads and layers. In contrast, **AttnHyper** preserves token-level attention structure and leverages learnable hypergraph message passing to aggregate information across the entire attention landscape.

Hypergraph Neural Networks (HGNN) extend graph neural networks by modeling higher-order relations via hyperedges that connect arbitrary node subsets [16, 26]. This formulation has been unified under general message-passing frameworks such as UniGNN [27] and further formalized through expressive multiset functions in AllSet [28]. Integrating attention with hypergraph structures has been shown to enhance representation learning [29, 30].

In NLP, hypergraphs have been applied to tasks such as document classification, summarization, and reasoning [29, 31, 32], demonstrating their effectiveness in modeling heterogeneous dependencies. However, their application to hallucination detection remains unexplored. Our method fills this gap by modeling Transformer attention patterns as hypergraphs and applying message passing to capture higher-order interactions across layers and heads.

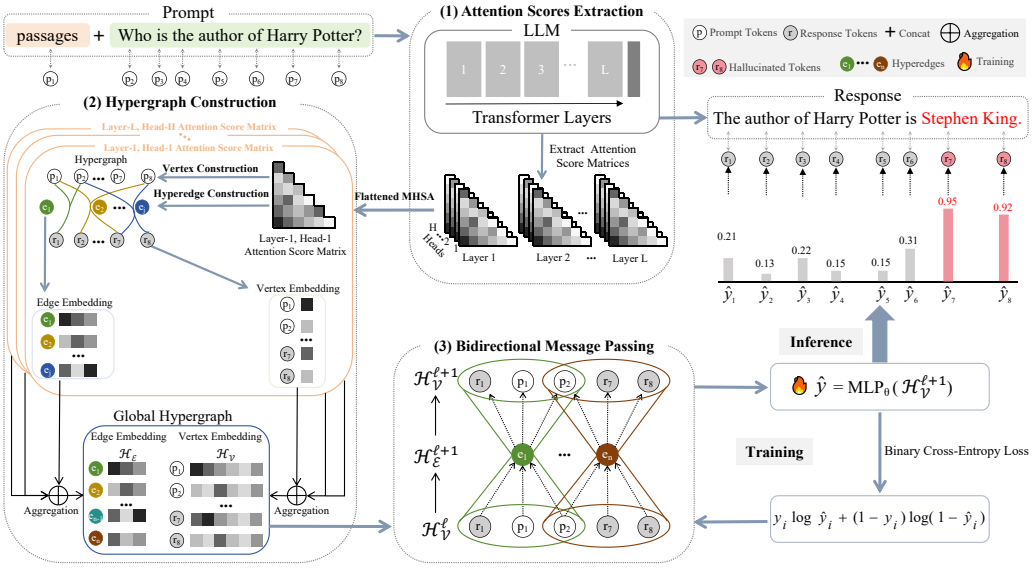


Figure 2. The overview framework of **AttnHyper**.

3 Problem Setup

Definition 3.1 (Hallucination Detection). We define the truthful distribution $\mathbb{P}_{\text{true}}$ as the joint distribution over pairs of input prompts and their corresponding factually supported responses. Let Σ denote the vocabulary space of a causal LLM. Define the prompt tokens as $\mathbf{P} = \{p_1, p_2, \ldots, p_m\} \subseteq \Sigma$ and the response tokens as $\mathbf{R} = \{r_1, r_2, \ldots, r_n\} \subseteq \Sigma$. The task of hallucination detection aims to learn a binary predictor $G : (\mathbf{P}, \mathbf{R}) \to \{0, 1\}$:

$$G(\mathbf{P}, \mathbf{R}) = \begin{cases} 1, & \text{if } (\mathbf{P} \oplus \mathbf{R}) \not\sim \mathbb{P}_{\text{true}}, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where $\mathbf{P} \oplus \mathbf{R}$ represents the ordered concatenation of the prompt tokens $\mathbf{P}$ and the response tokens $\mathbf{R}$.

Definition 3.2 (Hypergraph Neural Networks). A hypergraph is defined as $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where each hyperedge connects an arbitrary subset of nodes. The incidence matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$ encodes node–hyperedge relations, with $\mathbf{H}_{v,e} = 1$ if $v \in e$.

Let $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ denote node features. HGNNs generalize GNNs by propagating messages through hyperedges to model higher-order interactions. A standard hypergraph convolution layer is:

$$\mathbf{X}^{(l+1)} = \sigma\left(\mathbf{D}_v^{-\frac{1}{2}} \mathbf{H} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-\frac{1}{2}} \mathbf{X}^{(l)} \mathbf{W}^{(l)}\right), \tag{2}$$

where $\mathbf{D}_v$ and $\mathbf{D}_e$ are node and hyperedge degree matrices, respectively, and $\sigma(\cdot)$ is a nonlinear activation. Stacking layers enables modeling higher-order dependencies beyond pairwise message passing.

4 Methodology

Overview. The overall pipeline of **AttnHyper** is illustrated in Figure 2. At the attention processing stage, we introduce a hypergraph-based formulation to construct higher-order attention, which enables modeling complex high-order dependencies within attention patterns. Finally, a hypergraph neural network is trained via a bidirectional message-passing mechanism to perform hallucination detection. Overall, our framework consists of three main components. (1) Attention Scores Extraction, (2) Hypergraph Construction, and (3) Bidirectional Message Passing.

4.1 Attention Scores Extraction

LLMs are composed of multiple transformer layers, which include both Attention and FNN components. Our current goal is to obtain the attention scores for every token across all Attention layers. Define the prompt tokens as $\mathbf{P} = \{p_1, p_2, \ldots, p_m\}$,and the response tokens as $\mathbf{R} = \{r_1, r_2, \ldots, r_n\}$. By concatenating the prompt tokens and the response tokens as the input to the LLMs, a single forward pass produces the attention score matrix for all tokens:

$$\text{Attn} = \text{LLMs}(p_1, p_2, \ldots, p_m, r_1, r_2, \ldots, r_n), \tag{3}$$

where Attn denotes the attention score matrix, and the resulting attention matrix has shape $(L,\ H,\ m+n,\ m+n)$, where L denotes the number of layers, H denotes the number of attention heads per layer, m denotes the number of prompt tokens, and n denotes the number of response tokens.

4.2 Hypergraph Construction

Flattened Multi-Head Self-Attention (MHSA). Since large language models consist of multiple attention heads at each layer, we assign a unique global index to every head across all layers. This unified indexing scheme facilitates the construction of the hypergraph by enabling consistent representation and aggregation of attention interactions across different layers and heads(See more details in Appendix D). All flattened attention heads are indexed sequentially as:

$$\text{head}_j, \quad j = 0, 1, \ldots, (L \cdot H) - 1, \tag{4}$$

Under this indexing scheme, the attention matrix becomes

$$A \in \mathbb{R}^{(L \cdot H) \times (m+n) \times (m+n)}, \tag{5}$$

where the first dimension corresponds to the flattened head index j.

Vertex Construction. Since hallucinations often manifest as weak or misaligned dependencies between the prompt and the generated response, we include tokens from both segments as hypergraph nodes. This design allows the hypergraph to explicitly model prompt–response interactions through attention-based hyperedges, providing critical structural cues for hallucination detection. The hypergraph vertex set corresponds to all input tokens:

$$\mathcal{V} = \{\, v_i \mid i \in P \cup R \,\}, \tag{6}$$

where each vertex v_i represents a unique input token. The attribute of vertex v_i is defined by concatenating its self-attention values across all layers and heads:

$$\text{attr}(v_i) = \text{Concat}\left(\text{A}[j, i, i] \;\middle|\; 0 \le j < L \cdot H\right), \tag{7}$$

where $\text{A}[j, i, i]$ denotes the diagonal element of the attention matrix corresponding to the flattened head index j and token i. The concatenation is taken over all layers and all attention heads, resulting in a feature vector of length $L \times H$ for each vertex v_i.

Finally, we stack the attributes of all vertices to form the node feature matrix:

$$\mathcal{H}_{\mathcal{V}} = \{\, \text{attr}(v_i) \mid v_i \in \mathcal{V} \,\}. \tag{8}$$

Hyperedge Construction. We define the hyperedge set as:

$$\mathcal{E} = \left\{\, e_{i,j} \mid i \in R,\ j \in \{0, \ldots, L \cdot H - 1\} \,\right\}, \tag{9}$$

where each hyperedge $e_{i,j}$ associated with attention head j is defined as:

$$e_{i,j} = \left\{\, \text{token}_u \;\middle|\; u < i,\ \text{A}[j, i, u] \ > \ \tau \right\} \ \cup\ \{\text{token}_i\}, \tag{10}$$

which collects all preceding tokens $u \in [0, i-1]$ receiving an attention score greater than τ from token i under head j, with token i itself included as the center node of the hyperedge. For each hyperedge $e_{i,j}$, we define its attribute vector as:

$$\text{attr}(e_{i,j}) = \big(\text{mean}(e_{i,j}),\ \max(e_{i,j}),\ \text{head_id}_\text{j}\big), \tag{11}$$

where mean value is defined as $\text{mean}(e_{i,j}) = \frac{1}{|e_{i,j}|} \sum_{u \in e_{i,j}} A[j, i, u]$, maximum value is defined as $\max(e_{i,j}) = \max_{u \in e_{i,j}} A[j, i, u]$, and head_id_j denotes the index normalized identifier of attention

head j, defined as $\text{head_id}_j = \frac{j}{L \cdot H - 1}$. Based on the above attribute definition, we organize the attributes of all hyperedges to form the hyperedge feature matrix:

$$\mathcal{H}_{\mathcal{E}} = \{\, \text{attr}(e_i) \mid e_i \in \mathcal{E} \,\}. \tag{12}$$

With the vertex set, hyperedge set, and their corresponding embeddings defined above, the resulting global hypergraph is represented as:

$$\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{H}_{\mathcal{V}}, \mathcal{H}_{\mathcal{E}}), \tag{13}$$

Considering the labels associated with each node token in the hypergraph $\mathcal{H}$, the labeled hypergraph is defined as $(\mathcal{H}, y)$.

For token-level supervision, we define the label vector as:

$$y = (y_1, y_2, \ldots, y_{|\mathcal{V}|}) \in \{0, 1\}^{|\mathcal{V}|}, \tag{14}$$

Here, $y_i = 1$ if the token $r_i \in R$ belongs to a hallucinated span, and $y_i = 0$ otherwise.

4.3 Bidirectional Message Passing

Given a hypergraph input $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \text{attr}(\mathcal{V}), \text{attr}(\mathcal{E}))$, each node $v \in \mathcal{V}$ is associated with an input feature $x_v \in \mathbb{R}^{L \cdot H}$. We first project node features into a hidden representation space:

$$h_v^{(0)} = \phi(x_v) = \text{ReLU}(W_{\text{in}} x_v), \tag{15}$$

where W_{in} is a learnable projection matrix and $h_v^{(0)} \in \mathbb{R}^d$ denotes the initial hidden representation of node i.

The node-to-edge message passing stage aggregates node-level information into hyperedge representations. At each message-passing layer l, the model first aggregates information from incident nodes to update hyperedge representations. Specifically, for each hyperedge $e \in \mathcal{E}$, we compute:

$$h_e^{(l)} = \frac{1}{|e|} \sum_{v \in e} \text{MLP}_{n \to e}\left(h_v^{(l)}\right), \tag{16}$$

where $|e|$ denotes the number of nodes incident to hyperedge e, $h_v^{(l)}$ denotes the node feature representation of node v at layer l, and $\text{MLP}_{n \to e}(\cdot)$ is a learnable multilayer perceptron that transforms node features into the hyperedge representation space. The resulting $h_e^{(l)}$ captures high-order semantic information shared among nodes connected by hyperedge e.

We then propagate hyperedge representations back to nodes via edge-to-node message passing. The updated hyperedge representations are then propagated back to nodes through an edge-to-node aggregation. For each node $v \in \mathcal{V}$, we compute:

$$\begin{aligned} m_v^{(l)} &= \frac{1}{\deg(v)} \sum_{v \in e} \text{MLP}_{e \to n}\Big(\text{Concat}(\mathcal{H}_e, h_e^{(l)})\Big), \\ h_v^{(l+1)} &= h_v^{(l)} + \text{LN}(m_v^{(l)}), \end{aligned} \tag{17}$$

where $\deg(v)$ denotes the number of hyperedges incident to node v, $\mathcal{H}_e$ represents the attribute features associated with hyperedge e, $\text{Concat}(\cdot)$ denotes vector concatenation, and $\text{MLP}_{e \to n}(\cdot)$ is a learnable multilayer perceptron that maps hyperedge-level information into the node message space. The intermediate message $m_v^{(l)}$ aggregates information from all hyperedges connected to node v at layer l. $\text{LN}(\cdot)$ denotes layer normalization, which is applied to stabilize training and improve convergence. This message-passing procedure is repeated for L layers, yielding the final node embedding $h_v^{(L)}$. We provide theoretical proof in Appendix D.1.

The positions of response tokens are determined by the starting index r_1:

$$\mathcal{R} = \{\, i \mid \text{position}(i) \geq r_1 \,\}. \tag{18}$$

A lightweight MLP classifier is then applied to the token-level representation h_i to produce the final prediction:

$$\hat{y}_i = \text{MLP}_\theta(h_i), \qquad i \in \mathcal{R}. \tag{19}$$

where the **MLP** consists of two linear layers with a ReLU activation and dropout in between. Since hallucinations arise from the content generated by LLMs rather than the given prompt, we restrict hallucination prediction to tokens in the response segment. This design focuses the model on assessing the factual reliability of generated outputs, while avoiding unnecessary supervision on prompt tokens that are assumed to be factual inputs.

Binary Cross-Entropy Loss. We adopt the binary cross-entropy (BCE) loss to supervise hallucination prediction at the token level. Specifically, the loss is computed only over response tokens:

$$\mathcal{L} = -\sum_{i \in \mathcal{R}} \Big[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \Big], \tag{20}$$

where $y_i \in \{0, 1\}$ denotes the ground-truth hallucination label of the i-th response token, and $\hat{y}_i$ represents the predicted probability that the token is hallucinated. This design aligns with our objective of detecting hallucinations in LLM-generated responses and enables fine-grained token-level supervision.

The overall training objective of the model is:

$$\theta^* = \arg\min_{\theta} \sum_{(\mathcal{H}, y) \in \mathcal{D}} \mathcal{L}(\textbf{AttnHyper}(\mathcal{H}; \theta), y), \tag{21}$$

where θ denotes the set of trainable parameters of the **AttnHyper** model, and θ^* represents the optimal parameters obtained by minimizing the empirical risk over the training set $\mathcal{D}$.

5 Experiments

In this section, we present experiments to evaluate the effectiveness of the proposed HyperGraph-based hallucination detection method for LLMs, addressing the following research questions. **RQ1:** How effective is our hypergraph-based method compared to state-of-the-art hallucination detection approaches? **RQ2:** How transferable is our method across different datasets? **RQ3:** How do hyperparameters and hypergraph structure design influence the performance of our method? **RQ4:** How does layer-wise hypergraph construction influence the performance of **AttnHyper**?

5.1 Experimental Setups

Datasets. We evaluate on RAGTruth [14], a hallucination corpus with fine-grained span-level annotations. It covers three representative tasks: QA (question answering), Summary (news summarization), and Data2txt (data-to-text generation), enabling evaluation of token-level detection. See Appendix B for detailed statistics.

Models. We conduct experiments on LLaMA-2-7B-chat and Mistral-7B-Instruct, as these are the generation models used to construct the RAGTruth benchmark. This alignment ensures that internal-state extraction and response generation are consistent, avoiding confounds due to model mismatch.

Baselines. We compare **AttnHyper** against a diverse set of baseline methods spanning different detection paradigms and granularities. To ensure a rigorous evaluation, we categorize baselines along two dimensions: *method type* and *detection granularity*. **(1) Logits-based** —*Perplexity* [33], **(2) Hidden-state-based**— *TSV* [9], and *ICR Probe* [23]. **(3) Attention-based**—*Lookback Lens* [10], *LapEigvals* [12]. All baselines are evaluated using their official implementations with default hyperparameters as specified in the original papers. We report AUROC as evaluation metrics.

5.2 Effectiveness Comparison (RQ1)

We compare our hypergraph-based hallucination detection method with five baselines on LLaMA-2-7B-chat and Mistral-7B-Instruct, as shown in Table 2. Results show that token-level hallucination detection remains challenging for existing methods, particularly on RAGTruth tasks with long contexts and sparse labels (see Appendix B). Despite these difficulties, our method consistently achieves state-of-the-art performance across both models and all tasks.

Table 2. Performance comparison with state-of-the-art baselines on **AUROC** (%) for RAGTruth. Higher values indicate better performance. The best and second-best results are in **bold** and underlined, respectively. For our method, mean and standard deviation are computed across five random seeds.

Model	Type	Method	QA	Summary	Data2txt
Mistral-7B	Logits	Perplexity	59.14	64.66	33.91
	Hidden	ICR	60.96	55.66	53.33
		TSV	79.43	66.07	68.86
	Attention	LapEig	76.31	68.39	66.65
		Lookback	91.35	77.75	89.96
		AttnHyper	**96.18**$\pm$0.1	**87.89**$\pm$0.8	**94.04**$\pm$0.4
LLaMA-2-7B	Logits	Perplexity	54.29	60.81	36.85
	Hidden	ICR	58.86	53.86	60.18
		TSV	78.05	63.57	66.52
	Attention	LapEig	79.63	63.34	65.44
		Lookback	85.09	69.65	85.05
		AttnHyper	**90.18**$\pm$0.1	**71.54**$\pm$0.4	**89.51**$\pm$0.3

In particular, **AttnHyper** outperforms Lookback Lens by 13.04% on the Summary task with Mistral-7B-Instruct. This gain comes from explicitly modeling higher-order attention interactions via hypergraphs, which better captures hallucination patterns than pairwise approaches.

5.3 Transferability Across Datasets (RQ2)

We assess the cross-task transferability of **AttnHyper** by training the model on a single source task and directly evaluating it on other target tasks without any additional fine-tuning. Table 3 reports the cross-task transfer results in terms of AUROC, including comparisons with several baseline methods.

Table 3. Cross-dataset transferability (AUROC, %) on the Mistral-7B-Instruct.

Train	Method	QA	Summary	Data2txt
QA	Lookback Lens	91.35	74.43	69.25
	ICR Probe	60.96	57.03	52.37
	AttnHyper	**96.18**	**78.41**	**77.53**
Summary	Lookback Lens	84.06	77.75	68.87
	ICR Probe	62.68	55.66	52.77
	AttnHyper	**94.41**	**87.89**	**88.94**
Data2txt	Lookback Lens	83.80	71.73	89.96
	ICR Probe	62.38	50.51	53.33
	AttnHyper	**92.24**	**84.15**	**94.04**

Although transfer performance dropped across all tasks, our method consistently remained state-of-the-art. For example, a model trained on Data2txt achieved an AUROC of 92.24% on QA tests, which is close to the 94.04% AUROC obtained when trained directly on Data2txt. This strong transferability demonstrates AttnHyper's potential in LLMs, effectively detecting hallucinations even under task transfer. More detailed analysis of cross-task transfer behaviors and the observed asymmetries is provided in Appendix F.1.

5.4 Impact of Hyperparameters and Hypergraph Structure (RQ3)

Hyperparameters. Regarding the choice of the threshold τ, we empirically determine its value using the Mistral-7B-Instruct model by evaluating its impact on QA performance under different settings. Table 4 shows that $\tau = 0.05$ achieves a good trade-off between detection performance and resource usage. On the QA task, it yields an AUROC of 96.18%, outperforming both $\tau = 0.1$ (95.83%) and $\tau = 0.01$ (96.01%), while maintaining a moderate hypergraph size of 14.34 MB. Larger thresholds produce smaller hypergraphs but lower performance, whereas $\tau = 0.01$ nearly doubles the graph size with negligible AUROC gain. Additional hyperparameter choices are provided in Appendix E.1.

Table 4. Effect of threshold τ on the average number of hyperedges, hypergraph size, and QA performance.

τ	Num. Edges	Size (MB)	AUROC (%)
0.5	2582.54	2.38	94.47 ± 0.2
0.1	130547.66	11.76	95.83 ± 0.1
0.05	138967.71	14.34	96.18 ± 0.1
0.01	140476.09	26.49	96.01 ± 0.1

Impact of Hypergraph Structure. To verify the effectiveness of the hypergraph structure, we conduct an ablation study on the QA task, Summary task and Data2txt task by comparing model variants with and without the message passing module. The results are reported in Table 5. When message passing is removed, the model exhibits a clear degradation in AUROC, consistently underperforming its counterpart equipped with message passing. These results demonstrate that hypergraph-based message passing plays a crucial role in capturing relational information and improving hallucination detection performance. Additional analyses on the effects of the number of message passing layers and hidden dimensions are provided in Appendix G.1.

Table 5. Effect of the message passing module (**MP**) on detection performance on the QA, Summary, and Data2txt tasks using Mistral-7B-Instruct.

Dataset	Module MP	AUROC (%)
QA	✗	89.56 ± 0.1
	✓	$\mathbf{96.18 \pm 0.1}$
Summary	✗	79.53 ± 0.1
	✓	$\mathbf{87.89 \pm 0.8}$
Data2txt	✗	89.13 ± 0.2
	✓	$\mathbf{94.04 \pm 0.4}$

5.5 Layer-wise Hypergraph Analysis (RQ4)

To understand how attention patterns are encoded into hypergraph structures across Transformer layers, we analyze layer-wise hypergraphs for hallucination detection. As shown in Figure 3, performance is generally stable across intermediate layers, with a noticeable drop at layer 3, while using a single layer performs comparably to aggregating all layers. This suggests that each layer contains sufficient information for hallucination detection, which can be effectively captured via hypergraph construction.

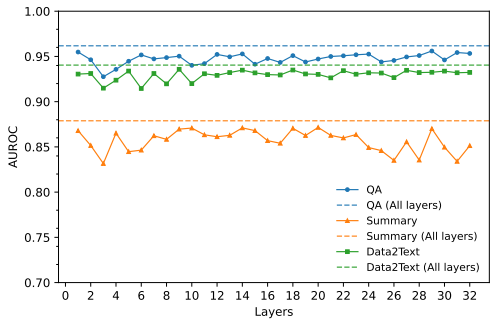


Figure 3. Average AUROC across transformer layers for different tasks (QA, Summary, and Data2txt) on Mistral-7B-Instruct.

Importantly, this also implies that hypergraphs can be constructed from a single selected layer without significant performance degradation, thereby reducing inference cost and enabling more efficient deployment. Detailed results are provided in Appendix F.4 and Appendix F.5.

5.6 Limitations

Our method focuses on leveraging internal attention scores of LLMs for hallucination detection. Therefore, it relies on open-source LLMs to extract attention scores and requires additional training of an HGNN to obtain a hallucination detection model, making it not immediately plug-and-play.

6 Conclusion

This work introduces **AttnHyper**, the first framework to model Transformer attention patterns as hypergraphs for hallucination detection in LLMs. By constructing hyperedges that connect multiple

co-attended tokens, **AttnHyper** captures higher-order dependencies and employs bidirectional message passing to aggregate information across attention heads and layers. Experiments show that it outperforms existing methods on the RAGTruth benchmark and exhibits strong zero-shot generalization. The approach reveals structural patterns in attention, linking low-level token interactions to higher-level reasoning, providing new insights into the internal mechanisms of LLMs and laying a foundation for building safer and more trustworthy models.

7 Acknowledgments

This work is supported by the Intelligent Computing Center of Shenzhen University. We gratefully acknowledge the center for providing the computational resources and technical infrastructure necessary for conducting the experiments and evaluations presented in this work. This work was supported by the Joint Funds of the National Natural Science Foundation of China (No. U24A20322), the Shenzhen Science and Technology Major Project (KJZD20230923114605011), Guangdong Provincial Key Laboratory of Visual Media and Multidimensional Intelligence (2026B1212010012), the Research Center for FinTech and Digital-Intelligent Management, Shenzhen University, and the Shenzhen Humanities Social Sciences Key Research Bases.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

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

[3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[4] 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.

[5] Potsawee Manakul, Adian Liusie, and Mark Gales. 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, 2023.

[6] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

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

[8] Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. INSIDE: LLMs' internal states retain the power of hallucination detection. In *International Conference on Learning Representations*, 2024.

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

[10] Yung-Sung Chuang, Linlu Qiu, Cheng-Yu Hsieh, Ranjay Krishna, Yoon Kim, and James R. Glass. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1419–1436, 2024.

[11] Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. LLM-check: Investigating detection of hallucinations in large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[12] Jakub Binkowski, Denis Janiak, Albert Sawczyn, Bogdan Gabrys, and Tomasz Jan Kajdanowicz. Hallucination detection in LLMs using spectral features of attention maps. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

[13] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua 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*, 43(2), 2024.

[14] Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, KaShun Shum, Randy Zhong, Juntong Song, and Tong Zhang. RAGTruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 10862–10878, 2024.

[15] Abhika Mishra, Akari Asai, Vidhisha Balachandran, Yizhong Wang, Graham Neubig, Yulia Tsvetkov, and Hannaneh Hajishirzi. Fine-grained hallucination detection and editing for language models. In *First Conference on Language Modeling*, 2024.

[16] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3558–3565, 2019.

[17] Sunwoo Kim, Soo Yong Lee, Yue Gao, Alessia Antelmi, Mirko Polato, and Kijung Shin. A survey on hypergraph neural networks: An in-depth and step-by-step guide. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 6534–6544, 2024.

[18] Lev Telyatnikov, Maria Sofia Bucarelli, Guillermo Bernardez, Olga Zaghen, Simone Scardapane, and Pietro Lio. Hypergraph neural networks through the lens of message passing: A common perspective to homophily and architecture design. *Transactions on Machine Learning Research*, 2025.

[19] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the ai ocean: A survey on hallucination in large language models. *Computational Linguistics*, pages 1–45, 2025.

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

[21] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *International Conference on Learning Representations*, 2023.

[22] Xuefeng Du, Chaowei Xiao, and Yixuan Li. HaloScope: Harnessing unlabeled LLM generations for hallucination detection. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[23] Zhenliang Zhang, Xinyu Hu, Huixuan Zhang, Junzhe Zhang, and Xiaojun Wan. ICR probe: Tracking hidden state dynamics for reliable hallucination detection in LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17986–18002, 2025.

[24] Alexandra Bazarova, Andrei Volodichev, Aleksandr Yugay, Andrey Shulga, Alina Ermilova, Konstantin Polev, Julia Belikova, Rauf Parchiev, Dmitry Simakov, Maxim Savchenko, Andrey Savchenko, Serguei Barannikov, and Alexey Zaytsev. Hallucination detection in LLMs with topological divergence on attention graphs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15449–15470, 2026.

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

[26] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems*, 2019.

[27] Jing Huang and Jie Yang. UniGNN: A unified framework for graph and hypergraph neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 2563–2569, 2021.

[28] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are AllSet: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations*, 2022.

[29] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. Be more with less: Hypergraph attention networks for inductive text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4927–4936, 2020.

[30] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.

[31] Haopeng Zhang, Xiao Liu, and Jiawei Zhang. HEGEL: Hypergraph transformer for long document summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10167–10176, 2022.

[32] Yu-Jung Heo, Eun-Sol Kim, Woo Suk Choi, and Byoung-Tak Zhang. Hypergraph Transformer: Weakly-supervised multi-hop reasoning for knowledge-based visual question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 373–390, 2022.

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

[34] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. pages 1601–1611, 2017.

A Mathematical Notation and Formal Definitions

This appendix provides a comprehensive reference for all mathematical notation employed throughout the main paper. We organize the symbols into semantically coherent categories to facilitate reader comprehension and ensure consistency across theoretical derivations and empirical analyses.

Table 6 presents the complete set of mathematical symbols used in this work. The notation follows standard conventions in the hypergraph neural networks and natural language processing literature, with extensions specific to our hypergraph-based formulation.

Table 6. Complete mathematical notation used throughout this paper. Symbols are organized by category for ease of reference.

Type	Symbol	Description
LLMs Architecture Parameters	L	Number of transformer layers in the LLM.
	H	Number of attention heads per transformer layer.
Attention Representations	$A \in \mathbb{R}^{(L \cdot H) \times (m+n) \times (m+n)}$	Flattened attention tensor aggregating all heads across all layers.
	$A[j, i, u]$	Attention score from query token i to key token u at flattened head index j.
	τ	Attention threshold hyperparameter for hyperedge construction.
Hypergraph Structure	$\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{H}_{\mathcal{V}}, \mathcal{H}_{\mathcal{E}})$	Attributed hypergraph representation.
	$\mathcal{V} = \{\, v_i \mid i \in P \cup R \,\}$	Set of hypergraph vertices corresponding to tokens.
	$\mathcal{E} = \{e_{i,j}\}$	Set of hyperedges indexed by query token i and head index j.
	$e_{i,j}$	Hyperedge containing tokens attended by query i under head j with attention $> \tau$.
	$\mathrm{attr}(v_i) \in \mathbb{R}^{L \cdot H}$	Vertex attribute vector for token v_i (diagonal attention values).
	$\mathrm{attr}(e_{i,j})$	Hyperedge attribute tuple (mean attention, max attention, head identifier).
	$\deg(i)$	Hypergraph degree of node i (number of incident hyperedges).
	$\lvert e \rvert$	Cardinality of hyperedge e (number of member nodes).
Neural Network Components	l	Number of message-passing layers in the HyperGNN.
	$h_v^{(l)} \in \mathbb{R}^d$	Node embedding of vertex v at layer l.
	$h_e^{(l)} \in \mathbb{R}^d$	Hyperedge representation at layer l.
	$W_{\text{in}} \in \mathbb{R}^{d \times (L \cdot H)}$	Input projection matrix.
	$\mathrm{MLP}_{n \to e}(\cdot)$	Multi-layer perceptron for node-to-edge message transformation.
	$\mathrm{MLP}_{e \to n}(\cdot)$	Multi-layer perceptron for edge-to-node message transformation.
	$\mathrm{LN}(\cdot)$	Layer normalization operation.
Labels and Predictions	$y_i \in \{0, 1\}$	Token-level binary hallucination label for token i.
	$\hat{y}_i \in [0, 1]$	Predicted hallucination probability for token i.
	$\mathcal{R}$	Set of response token indices (positions $\geq r_1$).
	$\mathcal{L}$	Binary cross-entropy loss function.

B Dataset Statistics

This section presents comprehensive statistics of the datasets used in our experimental evaluation. All experiments are conducted on RAGTruth, a benchmark specifically designed for hallucination detection in retrieval-augmented generation (RAG) settings. The dataset covers multiple application domains, including question answering (QA), summarization (Summary), and data-to-text generation (Data2txt), enabling a systematic evaluation of the generalization ability of **AttnHyper** across diverse generation tasks. Dataset statistics for the three tasks are summarized in Table 7.

Table 7. Detailed statistics for the RAGTruth benchmark tasks. Statistics are computed over the combined training and test splits. Hallucination rates indicate the proportion of samples that contain at least one hallucinated span.

Statistic	**QA**	**Summary**	**Data2txt**
Total Samples	5,934	5,658	6,198
Train / Test Split	5,034 / 900	4,758 / 900	5,298 / 900
Hallucination rates (Sample-level)	29.05%	29.80%	68.64%
Avg. Prompt Length(characters)	1671.00	3372.29	2758.99
Avg. Response Length(characters)	686.54	693.30	999.36
Source Document Types	MARCO	CNN/DM	Yelp

Table 8. Token-level Hallucination ratios (%) for different models on RAGTruth benchmark tasks. Ratios are computed separately for the training, validation, and test splits.

Model	**Dataset**	**Train**	**Val**	**Test**
Mistral-7B-Instruct	QA	11.47	10.71	6.68
	Summary	7.70	8.05	6.96
	Data2txt	8.17	8.43	6.91
LLaMA-2-7B-chat	QA	11.70	13.47	7.27
	Summary	5.17	4.63	4.45
	Data2txt	7.08	7.04	6.16

Data Generation Models. For each task in the RAGTruth benchmark, the dataset is generated using six different large language models: `gpt-4-0613`, `gpt-3.5-turbo-0613`, `mistral-7B-instruct`, `llama-2-7b-chat`, `llama-2-13b-chat`, and `llama-2-70b-chat`. This diversity of generation models introduces substantial variability in hallucination patterns, ranging from minor factual deviations to severe numerical and relational errors. In our method, we primarily focus on samples generated by `mistral-7B-instruct` and `llama-2-7b-chat`. These two models are chosen as representative mid-scale open-source generators, enabling a controlled and fair evaluation of hallucination detection performance while avoiding reliance on proprietary models.

Data split. For each task, we randomly split 20% of the original training set into a validation set, using a fixed random seed of 42 to ensure reproducibility.

Hallucination Ratio. Table 8 presents the token-level hallucination ratios (%) for different RAGTruth benchmark tasks across two model tokenizers: Mistral-7B-Instruct and LLaMA-2-7B-chat. The ratios are computed separately for the training, validation, and test splits.

Overall, the hallucination ratios are consistently low across all tasks and models. For instance, even the highest value, 13.47% for LLaMA-2-7B-chat on the QA validation split, remains relatively modest. This indicates that the majority of tokens in the generated responses are faithful, and hallucinated content constitutes only a small fraction of the total output. Consequently, all tasks in the benchmark can be considered challenging for hallucination detection, as the low prevalence of hallucinated tokens implies a high signal-to-noise ratio where noisy or spurious information is prevalent. Detecting hallucinations under these conditions requires highly sensitive methods capable of identifying subtle

deviations, since the sparse hallucinated tokens are easily obscured by the overwhelming presence of correct tokens.

In addition, comparing the two tokenizers shows that the differences in hallucination ratios are generally small, suggesting that the choice of tokenizer has limited impact on the overall frequency of hallucinations. This emphasizes that the intrinsic difficulty of the tasks, rather than tokenization artifacts, is the primary factor affecting hallucination detection performance.

C Detailed Algorithmic Framework

This section provides complete algorithmic specifications for all components of **AttnHyper**, enabling full reproducibility of our approach. We present pseudocode for the hyperedge construction, message passing, training pipeline, and inference procedures.

C.1 Hyperedge Construction

This subsection describes the hyperedge construction process, which is a critical component of our framework. Algorithm 1 provides detailed pseudocode for this procedure, incorporating threshold-based strategies to ensure robust hypergraph construction across varying attention distributions.

Algorithm 1 Hyperedge Construction from Attention Patterns

Input: Attention tensor $\mathbf{A} \in \mathbb{R}^{L\times H\times N\times N}$, response start index r, attention threshold τ, minimum hyperedge size $s_{\min}$.
Output: Hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathrm{attr}(\mathcal{V}), \mathrm{attr}(\mathcal{E}))$.

// Flatten multi-head attention across all layers.
Reshape $\mathbf{A} \leftarrow \mathbf{A}.\mathrm{view}((L \cdot H), N, N)$
Initialize vertex set $\mathcal{V} \leftarrow \{v_0, v_1, \ldots, v_{N-1}\}$
Initialize hyperedge set $\mathcal{E} \leftarrow \emptyset$, hyperedge counter $k \leftarrow 0$
// Compute vertex attributes via Eq. (7).
for $i = 0$ **to** $N - 1$ **do**
    $\mathrm{attr}(v_i) \leftarrow \mathrm{Concat}(\mathbf{A}[j, i, i] \mid 0 \le j < L \cdot H)$
end for
// Construct causal mask and disable prompt-prompt attention.
$\mathbf{M} \leftarrow \mathrm{tril}(\mathbf{1}_{N\times N}, \mathrm{diagonal} = -1)$
$\mathbf{M}[0\!:\!r,\ 0\!:\!r] \leftarrow \mathbf{0}$
// Construct hyperedges for each attention head.
for $j = 0$ **to** $L \cdot H - 1$ **do**
    $\mathbf{A}^{(j)} \leftarrow \mathbf{A}[j] \odot \mathbf{M}$
    for $i = r$ **to** $N - 1$ **do**
        Compute hyperedge members $e_{i,j} \leftarrow \{u \mid \mathbf{A}^{(j)}_{i,u} > \tau\} \cup \{i\}$ via Eq. (10)
        if $|e_{i,j}| < s_{\min}$ **then**
            continue
        end if
        Compute $\mu_k \leftarrow \frac{1}{|e_{i,j}|} \sum_{u\in e_{i,j}} \mathbf{A}^{(j)}_{i,u}$, $\max_k \leftarrow \max_{u\in e_{i,j}} \mathbf{A}^{(j)}_{i,u}$
        Compute hyperedge attributes $\mathrm{attr}(e_k) \leftarrow (\mu_k, \max_k, j/(L \cdot H - 1))$ via Eq. (11)
        Assign cross-segment mark: $\mathrm{mark}(e_k) \leftarrow [1, 0]$ if $\exists\, u \in e_{i,j} : u < r$, else $[0, 1]$
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_k\}$, $k \leftarrow k + 1$
    end for
end for
Assemble hypergraph $\mathcal{H} \leftarrow (\mathcal{V}, \mathcal{E}, \mathrm{attr}(\mathcal{V}), \mathrm{attr}(\mathcal{E}))$ =0

C.2 HyperGNN Message Passing

This subsection details the message passing procedure within the HyperGNN, consisting of alternating node-to-edge and edge-to-node aggregation steps. This two-phase design enables information propagation through hyperedges while preserving the higher-order structural relationships encoded in

the hypergraph. The overall procedure is formally summarized in Algorithm 2, which specifies the exact computation order and aggregation operations at each propagation layer.

Algorithm 2 HyperGNN Message Passing

Input: Node features $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{in}}}$, hyperedge attributes $\text{attr}(\mathcal{E})$, incidence structure $\mathcal{I}$, number of layers G.

Output: Updated node representations $\mathbf{H}^{(G)} \in \mathbb{R}^{N \times d}$.

Project input node features into hidden space $h_i^{(0)} \leftarrow \text{ReLU}(W_{\text{in}} \cdot x_i)$ for all $i \in \mathcal{V}$ via Eq. (15).
for $l = 0$ **to** $G - 1$ **do**
  Node-to-Edge Aggregation:
  for each hyperedge $e \in \mathcal{E}$ **do**
    Aggregate incident node features to compute hyperedge representation $h_e^{(l)} \leftarrow \frac{1}{|e|} \sum_{v \in e} \text{MLP}_{n \to e}(h_v^{(l)})$ via Eq. (16).
  end for
  Edge-to-Node Aggregation:
  for each node $v \in \mathcal{V}$ **do**
    Aggregate messages from incident hyperedges $m_v^{(l)} \leftarrow \frac{1}{\deg(v)} \sum_{e \ni v} \text{MLP}_{e \to n}\big(\text{Concat}(\text{attr}(e), h_e^{(l)})\big)$ via Eq. (17).
    Update node representation with residual connection $h_v^{(l+1)} \leftarrow h_v^{(l)} + \text{LayerNorm}(m_v^{(l)})$ via Eq. (17).
  end for
end for
Return $\mathbf{H}^{(G)} = \{h_0^{(G)}, h_1^{(G)}, \ldots, h_{N-1}^{(G)}\}$ =0

C.3 Training Pipeline

Algorithm 3 illustrates the training procedure of **AttnHyper**. We first construct hypergraph-structured datasets and load them using mini-batch data loaders, where each hypergraph encodes token-level dependencies via attention-induced hyperedges. The model consists of G stacked hypergraph message-passing layers and is trained in an end-to-end fashion.

To address class imbalance in hallucination detection, the positive class weight w^+ is computed solely on response-region tokens and incorporated into a weighted binary cross-entropy loss. Training is performed using the AdamW optimizer with a cosine learning-rate schedule and linear warm-up, and gradients are clipped to stabilize optimization.

The loss is computed only on response-region nodes using a dynamic masking strategy to prevent supervision leakage from prompt tokens. Model selection is based on validation AUPR, with early stopping applied. Finally, the model with the best validation performance is evaluated on the test set and used for inference.

C.4 Inference Pipeline

During inference, the trained **AttnHyper** model predicts token-level hallucination probabilities for unseen hypergraph samples. The hypergraph is loaded to the device, and the model computes logits for all nodes, which are converted to probabilities via the sigmoid function. Only tokens in the response region, determined by the response index, are selected for the final output. This ensures that the predictions focus exclusively on generated tokens and can be directly used for evaluation metrics such as AUROC and AUPR. The detailed steps of the inference procedure are summarized in Algorithm 4.

Algorithm 3 The Proposed **AttnHyper** Algorithm

Input: Training set $\mathcal{D}_{\text{tr}}$, validation set $\mathcal{D}_{\text{val}}$, test set $\mathcal{D}_{\text{te}}$, max epochs T, early stopping patience P, learning rate η, number of HyperGNN layers G.
Output: Optimal model parameters θ^*, token-level hallucination predictions.

Training Phase:
Initialize **AttnHyper** model $\mathcal{M}_\theta$ with G message passing layers.
Compute positive class weight $w^+ \leftarrow \frac{n_{\text{neg}}}{\max(n_{\text{pos}},1)}$ on response tokens to handle class imbalance.
Initialize AdamW optimizer and cosine learning rate scheduler.
Set best validation score $s^* \leftarrow -\infty$ and patience counter $c \leftarrow 0$.
for $\text{epoch} = 1$ **to** T **do**
  for each mini-batch $\mathcal{B} \subseteq \mathcal{D}_{\text{tr}}$ **do**
    Construct hypergraph $\mathcal{H}$ for each sample via Algorithm 1.
    Calculate node embeddings $\mathbf{H}^{(G)} \leftarrow$ Algorithm 2$(\mathcal{H})$.
    Construct response token mask $\mathcal{R} \leftarrow \{i \mid \text{position}(i) \geq r_1\}$ via Eq. (18).
    Calculate token-level predictions $\hat{y}_i \leftarrow \text{MLP}(h_i^{(G)})$ for all $i \in \mathcal{R}$ via Eq. (19).
    Calculate weighted binary cross-entropy loss $\mathcal{L} \leftarrow -\sum_{i\in\mathcal{R}}[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)]$ via Eq. (20).
    Back-propagate $\mathcal{L}$ and update parameters θ with gradient clipping.
  end for
  Evaluate $\mathcal{M}_\theta$ on $\mathcal{D}_{\text{val}}$ and compute AUPR score s.
  if $s > s^*$ **then**
    Update best score $s^* \leftarrow s$, save parameters $\theta^* \leftarrow \theta$, reset counter $c \leftarrow 0$.
  else
    Increment patience counter $c \leftarrow c + 1$.
    if $c \geq P$ **then**
      break (early stopping)
    end if
  end if
end for
Inference Phase:
Load optimal parameters θ^* into model $\mathcal{M}$.
for each hypergraph $\mathcal{H} \in \mathcal{D}_{\text{te}}$ **do**
  Compute node embeddings $\mathbf{H}^{(G)} \leftarrow$ Algorithm 2$(\mathcal{H})$.
  Compute hallucination probabilities $\hat{y}_i \leftarrow \sigma(\text{MLP}(h_i^{(G)}))$ for all $i \in \mathcal{R}$ via Eq. (19).
end for
Return optimal parameters θ^* and predictions $\{\hat{y}_i\}_{i\in\mathcal{R}}$. =0

Algorithm 4 Inference Pipeline of **AttnHyper**

Input: Trained model $\mathcal{M}^*$ with optimal parameters θ^*, test set $\mathcal{D}_{\text{te}}$, response start index r_1.
Output: Token-level hallucination probabilities $\{\hat{y}_i\}_{i\in\mathcal{R}}$ for all test samples.

Set model $\mathcal{M}^*$ to evaluation mode and disable gradient computation.
for each sample $(\mathcal{H}, y) \in \mathcal{D}_{\text{te}}$ **do**
  Load hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \text{attr}(\mathcal{V}), \text{attr}(\mathcal{E}))$ to computation device.
  Calculate node embeddings $\mathbf{H}^{(G)} \leftarrow$ Algorithm 2$(\mathcal{H}; \theta^*)$.
  Construct response token set $\mathcal{R} \leftarrow \{i \mid \text{position}(i) \geq r_1\}$ via Eq. (18).
  Calculate prediction logits $z_i \leftarrow \text{MLP}(h_i^{(G)})$ for all $i \in \mathcal{R}$ via Eq. (19).
  Convert logits to hallucination probabilities $\hat{y}_i \leftarrow \sigma(z_i)$ for all $i \in \mathcal{R}$.
  Store predictions $\{\hat{y}_i\}_{i\in\mathcal{R}}$ for evaluation.
end for
Compute evaluation metrics (AUROC, AUPR) by comparing $\{\hat{y}_i\}$ with ground-truth labels $\{y_i\}$.
Return token-level hallucination probabilities $\{\hat{y}_i\}_{i\in\mathcal{R}}$. =0

D Theoretical Analysis

We begin by formally establishing the validity and mathematical properties of constructing hypergraphs from transformer attention patterns. The key insight is that the mathematical structure of self-attention naturally defines hyperedges connecting each query token to its attended keys, thereby encoding higher-order relational information that is lost in pairwise graph representations.

Consider a transformer model with L layers and H attention heads processing a sequence of tokens. Let $\mathrm{A}[j, i, u]$ denote the attention weight from query token i to key token u at flattened head index j, where we flatten the layer-head structure by assigning a unique global index $j = h + l \cdot H$ to each attention head across all layers. We construct an **attention-induced hypergraph** $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{H}_{\mathcal{V}}, \mathcal{H}_{\mathcal{E}})$ that directly encodes these attention patterns through the following structure.

The **vertex set** $\mathcal{V} = \{\, v_i \mid i \in P \cup R \,\}$ consists of all tokens in the sequence, including both prompt tokens P and response tokens R. The **hyperedge set** is defined as in Eq. (9):

$$\mathcal{E} = \big\{\, e_{i,j} \mid i \in R,\ j \in \{0, \ldots, L \times H - 1\} \,\big\}, \tag{9}$$

where each hyperedge is indexed by response token positions and flattened head indices. For each query token i and flattened head index j, the corresponding hyperedge $e_{i,j}$ connects the query token to all key tokens receiving significant attention, as defined in Eq. (10):

$$e_{i,j} = \Big\{\, \text{token}_u \;\Big|\; u < i,\ \mathrm{A}[j, i, u] \ > \ \tau \Big\} \ \cup\ \{\text{token}_i\}, \tag{10}$$

where $\tau \in (0, 1)$ is a threshold parameter controlling hyperedge sparsity, and the query token i itself is always included as the center node of the hyperedge. The constraint $u < i$ enforces the causal masking structure inherent in autoregressive language modeling, ensuring that each token only attends to preceding tokens. This construction naturally captures the multi-way relationships encoded in attention patterns: each hyperedge represents a simultaneous dependence of the query token on multiple key tokens, preserving the higher-order structure that would be fragmented into separate pairwise edges in a graph-based representation.

The softmax normalization creates joint dependencies among all keys that cannot be factorized into independent pairwise terms, naturally inducing hypergraph structure. Specifically, the attention weight $\mathrm{A}[j, i, u]$ from query i to key u depends functionally on *all* keys through the partition function. To see this formally, consider the partial derivative with respect to a different key $\mathbf{k}_m$ where $m \neq u$:

$$\frac{\partial \mathrm{A}[j, i, u]}{\partial \mathbf{k}_m} = -\mathrm{A}[j, i, u] \cdot \mathrm{A}[j, i, m] \cdot \frac{\mathbf{q}_i}{\sqrt{d_k}} \neq \mathbf{0}. \tag{22}$$

This non-zero gradient demonstrates that changing key $\mathbf{k}_m$ affects the attention weight to key u, even though $m \neq u$. The softmax constraint $\sum_u \mathrm{A}[j, i, u] = 1$ creates competition among all keys for the query's attention budget—a joint dependency structure that is precisely what hyperedges capture. When query i attends to keys $\{u_1, \ldots, u_k\}$ with weights exceeding threshold τ, these tokens form a coherent attention group that should be modeled as a single hyperedge rather than k independent pairwise edges.

D.1 Bidirectional Message Passing

We now analyze that the bidirectional message passing in **AttnHyper** possesses universal approximation guarantees. The theoretical foundation relies on the Deep Sets theorem.

[Deep Sets] A function $f : 2^{\mathcal{X}} \to \mathbb{R}$ on finite sets is permutation-invariant if and only if it can be decomposed as:

$$f(S) = \rho\left(\sum_{x \in S} \phi(x)\right), \tag{23}$$

for some functions $\phi : \mathcal{X} \to \mathbb{R}^d$ and $\rho : \mathbb{R}^d \to \mathbb{R}$.

By the universal approximation theorem for MLPs, both ϕ and ρ can be approximated to arbitrary precision by neural networks with sufficient width. This provides the theoretical basis for our aggregation operations.

Node-to-Edge Aggregation. The hyperedge representation at layer l is computed as:

$$h_e^{(l)} = \frac{1}{|e|} \sum_{v \in e} \mathrm{MLP}_{n \to e}(h_v^{(l)}). \tag{16}$$

The node-to-edge aggregation defined in Eq. (16) can approximate any continuous permutation-invariant function over the node feature multiset $\{h_i^{(l)}\}_{i \in e}$.

We show that Eq. (16) instantiates the Deep Sets decomposition. Define $\phi(\cdot) := \mathrm{MLP}_{n \to e}(\cdot)$. The sum $\sum_{v \in e} \phi(h_v^{(l)})$ is permutation-invariant by commutativity of addition. Mean pooling is equivalent to composing with $\rho(z) = z/|e|$, which preserves approximation capability since $|e|$ is known at computation time. By Theorem D.1 and MLP universal approximation, the result follows.

Edge-to-Node Aggregation. Node representations are updated by aggregating incident hyperedge information:

$$\begin{aligned} m_v^{(l)} &= \frac{1}{\deg(v)} \sum_{v \in e} \mathrm{MLP}_{e \to n}\Big(\mathrm{Concat}(\mathcal{H}_e, h_e^{(l)})\Big), \\ h_v^{(l+1)} &= h_v^{(l)} + \mathrm{LN}(m_v^{(l)}), \end{aligned} \tag{17}$$

where $\mathcal{H}_e$ denotes hyperedge attributes and $\deg(v) = |\{e : v \in e\}|$ is the node degree. The concatenation conditions the message on hyperedge attributes, enabling head-specific transformations. The degree normalization bounds the aggregated message magnitude, ensuring stable gradient flow.

The concatenation of hyperedge attributes $\mathcal{H}_e = (\mathrm{mean}, \max, \mathrm{head_id})$ enables attention-conditioned message passing: the MLP learns head-specific transformations weighted by attention strength. The degree normalization $1/\deg(i)$ ensures bounded spectral radius and prevents high-degree nodes from dominating.

Residual Update. The residual connection $h_v^{(l+1)} = h_v^{(l)} + \mathrm{LN}(m_v^{(l)})$ yields:

$$\frac{\partial \mathcal{L}}{\partial h_v^{(l)}} = \frac{\partial \mathcal{L}}{\partial h_v^{(l+1)}} \cdot \left(\mathbf{I} + \frac{\partial \mathrm{LN}(m_v^{(l)})}{\partial h_v^{(l)}} \right). \tag{24}$$

This provides gradient highways through identity connections, enabling deep network training while preserving input attention features at all layers. The identity matrix $\mathbf{I}$ ensures non-vanishing gradients across layers.

D.2 Approximation of Prior Spectral Methods

We prove that **AttnHyper** subsumes prior attention-based detectors as special cases. Let $\mathbf{A}^{(j)} \in [0,1]^{n \times n}$ denote the attention matrix for head j. Define the Laplacian diagonal feature:

$$\lambda_i^{(j)} := d_i^{(j)} - \mathbf{A}^{(j)}[i,i], \quad \text{where} \quad d_i^{(j)} = \sum_{u=1}^{n} \mathbf{A}^{(j)}[i,u]. \tag{25}$$

For any $\varepsilon > 0$, there exists a single-layer **AttnHyper** such that $|h_i^{(1)}[j] - \lambda_i^{(j)}| < \varepsilon$ for all response tokens i and heads j.

The proof follows from three observations: (1) vertex features encode diagonal attention $\mathbf{A}[j,i,i]$; (2) hyperedge aggregation computes column-wise attention sums; (3) MLPs approximate the difference to arbitrary precision. This implies **AttnHyper** encompasses spectral features, and self-attention patterns, while learning novel discriminative features end-to-end.

E Implementation Details

This section provides comprehensive implementation details to ensure full reproducibility of our experimental results.

E.1 Training Configuration

All training hyperparameter settings used in our experiments are summarized in Table 9. All experiments are conducted on a single NVIDIA H100 GPU (80GB, 22C, 240G).

Table 9. Training hyperparameters.

Parameter	Value	Description
Optimizer	AdamW	Adam with decoupled weight decay
Learning rate	$\{5 \times 10^{-4}, 1 \times 10^{-3}\}$	Initial learning rate
Weight decay	10^{-3}	L2 regularization coefficient
Batch size	{1,2}	Graphs per batch (variable sizes)
HGNN Layers	{1,2,3,4}	Number of hypergraph message passing layers
Hidden dims	{32,64,128}	Hidden dimension size of node embeddings
Max epochs	{20,50}	Maximum training epochs
Early stopping patience	{3,5}	Epochs without validation improvement
LR scheduler	Cosine annealing	Learning rate decay schedule
Loss function	BCE	Binary cross-entropy loss
Warmup epochs ratio	{0.02,0.05}	Linear warm-up period ratio
Gradient clipping	1.0	Maximum gradient norm
Random seed	{1, 2, 3, 4, 5}	training random seeds

E.2 Data Preprocessing

E.2.1 Tokenization

We use the native tokenizer of each LLM (i.e., LlamaTokenizer for LLaMA-2 and MistralTokenizer for Mistral) to measure the token lengths on the three RAGTruth tasks. The statistics are summarized in Table 10.As shown in the table, the QA task consistently exhibits the smallest overall number of tokens across both prompts and responses. We therefore treat QA as a relatively simpler task in terms of sequence complexity. This assumption is further validated by our subsequent experimental results, as shown in Table 11. In contrast, tasks involving longer input sequences pose greater challenges for hallucination detection.

Table 10. Numbers of prompt (*Pro.*) and response (*Res.*) tokens across tasks, measured using different tokenizers.

Tokenizer		QA		Summary		Data2txt	
	Stat.	**Pro.**	**Res.**	**Pro.**	**Res.**	**Pro.**	**Res.**
Mistral-7B-Instruct	Mean	435.19	137.38	822.21	155.85	876.90	194.78
	Max	799	512	2530	512	2174	481
	Min	218	6	189	11	448	54
LLaMA-2-7B-chat	Mean	467.86	275.80	871.94	152.56	918.90	201.42
	Max	817	512	2634	511	2257	512
	Min	239	23	212	29	473	98

E.2.2 Attention Extraction

We extract attention matrices from the backbone LLM by enabling the `output_attentions=True` option during the forward pass in the Hugging Face `transformers` framework. Specifically, for each input sequence, we collect attention weights from all layers and heads, resulting in a set of attention tensors of shape $L \times H \times N \times N$, where L denotes the number of transformer layers, H the number of attention heads, and N the sequence length.

To improve training efficiency and reduce GPU memory consumption, the extracted attention tensors are cached to disk in `float16` precision and reused during subsequent training epochs. This design decouples attention extraction from model optimization, enabling scalable training on long-sequence tasks without repeatedly recomputing attention maps.

E.2.3 Label Alignment

We align character-level hallucination annotations to subword-level tokens according to the tokenizer of the backbone LLM. Given a token t_i spanning a character interval $[s_i, e_i)$, we assign its token-level label y_i using a conservative aggregation rule:

$$y_i = \begin{cases} 1, & \text{if } \exists\, c \in [s_i, e_i) \text{ such that } y(c) = 1, \\ 0, & \text{otherwise,} \end{cases} \tag{26}$$

where $y(c)$ denotes the character-level hallucination label.

This strategy ensures that any hallucinated content appearing within a token is fully preserved at the token level, avoiding false negatives caused by subword segmentation.

Special tokens (e.g., `[CLS]`, `[SEP]`) are excluded from loss computation, as they do not correspond to natural language content. However, they are retained during hypergraph construction to maintain global structural connectivity and information propagation across tokens.

F Extended Experimental Results

F.1 Cross-Dataset Transferability

To evaluate the transferability of our method across different datasets, we conduct cross-task transfer experiments, where the model is trained on one task and directly evaluated on other tasks without fine-tuning. The results are reported in Table 11.

Table 11. Cross-dataset transfer evaluation. Models are trained on the source task and evaluated on target tasks without fine-tuning. Results are reported in AUROC (%).

Model	Train	Method	QA	Summary	Data2txt
Mistral-7B-Instruct	QA	Lookback Lens	91.35	74.43	69.25
		ICR Probe	60.96	57.03	52.37
		AttnHyper	**96.18**	**78.41**	**77.53**
	Summary	Lookback Lens	84.06	77.75	68.87
		ICR Probe	62.68	55.66	52.77
		AttnHyper	**94.41**	**87.89**	**88.94**
	Data2txt	Lookback Lens	83.80	71.73	89.96
		ICR Probe	62.38	50.51	53.33
		AttnHyper	**92.24**	**84.15**	**94.04**
LLaMA-2-7B-chat	QA	Lookback Lens	85.09	64.98	65.31
		ICR Probe	58.86	51.21	51.20
		AttnHyper	**90.18**	**70.36**	**72.63**
	Summary	Lookback Lens	79.40	69.65	67.17
		ICR Probe	60.65	53.86	51.06
		AttnHyper	**85.62**	**71.54**	**82.67**
	Data2txt	Lookback Lens	79.00	65.26	85.05
		ICR Probe	64.35	57.61	60.18
		AttnHyper	**83.80**	**72.51**	**89.51**

We observe that models trained on short-sequence tasks exhibit limited performance when transferred to long-sequence tasks, whereas models trained on long-sequence tasks generalize substantially better to short-sequence tasks. This asymmetry further supports our hypothesis that long-sequence tasks pose greater challenges for hallucination detection compared to short-sequence tasks.Statistics of token lengths for each task are shown in Table 10.

F.2 Model Recency

We further evaluate on recent models, Llama-3.1-8B-Instruct, Qwen3-8B and Qwen3-14B. The results, shown in Table 12, demonstrate consistent superiority over pairwise edges and stable performance across modern architectures.

Table 12. Results on recent models (AUROC, %).

Model	Method	QA	Summary	Data2txt
Llama-3.1-8B-Instruct	TSV	59.05	60.62	68.82
	ICR Probe	65.70	64.49	66.43
	Lookback Lens	87.23	76.39	87.18
	AttnHyper	**91.31**	**78.33**	**90.57**
Qwen3-8B	TSV	60.25	52.28	53.30
	ICR Probe	65.09	53.42	67.18
	Lookback Lens	86.36	74.55	87.57
	AttnHyper	**91.61**	**76.23**	**90.44**
Qwen3-14B	TSV	61.10	53.85	55.63
	ICR Probe	66.43	55.18	68.72
	Lookback Lens	87.42	74.69	87.77
	AttnHyper	**90.95**	**77.63**	**91.71**

F.3 Evaluation on TriviaQA

To further demonstrate the robustness of our method beyond the original benchmark, we conduct additional experiments on TriviaQA [34], a dataset that differs substantially in data source and QA distribution. This setting provides a complementary test bed for evaluating the generalization ability of our approach under a distinct QA scenario.

We report AUROC consistently across all methods under the same experimental protocol. As shown in Table 13, AttnHyper continues to achieve the best performance on TriviaQA under two recent backbones, Llama-3.1-8B-Instruct and Qwen3-8B. These results indicate that the performance gains of AttnHyper are not specific to RAGTruth, but generalize to substantially different QA data distributions.

Table 13. Token-level results on TriviaQA (AUROC, %).

Model	Method	TriviaQA
Llama-3.1-8B-Instruct	Perplexity	76.30
	TSV	84.50
	ICR Probe	73.25
	LapEigvals	81.25
	Lookback Lens	71.55
	AttnHyper (Ours)	**86.55 $\pm$ 0.1**
Qwen3-8B	Perplexity	81.20
	TSV	81.62
	ICR Probe	75.84
	LapEigvals	75.46
	Lookback Lens	76.21
	AttnHyper (Ours)	**82.12 $\pm$ 0.2**

F.4 Computational efficiency

We include strong non-graph baselines in the efficiency analysis for a more comprehensive comparison, and evaluate detection performance together with inference latency, FLOPs, and peak GPU memory under the same backbone, precision, and input-length settings.

As summarized in Table 14, AttnHyper operates in a similar computational regime to methods that explicitly access attention pairs (e.g., LapEigvals and ICR Probe). Consequently, it is less lightweight than Lookback Lens, which avoids pairwise attention processing. We acknowledge this overhead. Importantly, the additional cost primarily stems from attention-pair access and threshold-based filtering, rather than from the parameter size of the HGNN itself.

Table 14. Efficiency comparison under the same backbone and input setting.

Metric	TSV	ICR Probe	LapEigvals	Lookback Lens	AttnHyper
Trainable Params	4,096	15,041	51,201	1,025	341,889
AUROC	59.05	65.70	72.54	87.18	**91.31**
Training Time	1,137 s	9,674 s	324.3 s	1,915.7 s	10,450 s
Per-sample Inference	79 ms	50 ms	111.1 ms	40 ms	26 ms
Total FLOPs	5.94×10^{16}	5.51×10^{15}	1.17×10^{16}	3.5×10^{12}	7.06×10^{15}
GPU Peak Memory	24.6 GB	37.1 GB	17.5 GB	20.01 GB	24.8 GB
GPU Utilization	99%	19%	89%	20%	99%
Memory w.r.t. N	$O(LNd)$	$O(LHN^2)$	$O(LHN^2)$	$O(LHN)$	$O(LHN^2)$

In return for this cost, AttnHyper achieves the best detection performance (AUROC 91.31), outperforming Lookback Lens (87.18), while maintaining peak GPU memory within a comparable range (24.8 GB vs. 20.01 GB). Moreover, AttnHyper is faster than ICR Probe in our measurements and remains feasible to run on a single GPU.

Finally, AttnHyper provides explicit cost–accuracy trade-offs through controllable design choices, such as the sparsification threshold τ and selecting a subset of layers for hypergraph construction. We include additional analysis of these options in the revised version.

F.5 Scalability to Large Models

To quantify the end-to-end cost of our method, we benchmark AttnHyper on RAGTruth-QA (long-context) and TriviaQA (short-context) using two backbones: Llama-3.1-8B-Instruct and Llama-3.3-70B-Instruct. We report a detailed breakdown of (i) attention extraction, (ii) hypergraph construction, (iii) HGNN training and inference, as well as (iv) GPU peak memory and trainable parameter count, as summarized in Table 15. All measurements are obtained from the same pipeline implementation.

Table 15. End-to-end efficiency breakdown on long- and short-context datasets.

Metric	RAGTruth-QA		TriviaQA	
	8B	70B	8B	70B
Layers $\times$ Heads	32×32	80×64	32×32	80×64
AUROC(%)	91.31	90.89	86.55	86.73
Extract attention time (s)	3308	8876	395.2	1124.8
Hypergraph construction time (s)	2618	4767	65.3	222.7
Training time (s)	4524	8412	57.8	370
Per-sample inference (ms)	26	561	0.78	3.24
GPU Peak training (GB)	24.8	82	11.5	41.19
Trainable Params (M)	0.34	0.34	0.34	0.34
Memory w.r.t. N	$O(LHN^2)$	$O(LHN^2)$	$O(LHN^2)$	$O(LHN^2)$
Total FLOPs	7.06×10^{15}	2.82×10^{16}	3.50×10^{14}	2.01×10^{15}

These results highlight two practical properties.

Constant parameter scale. The number of trainable parameters in AttnHyper remains 0.34M for both 8B and 70B backbones. The detector does not scale its parameterization with the backbone size, as it operates as a lightweight HGNN on top of frozen attention signals.

Computational scalability governed by attention size. The dominant scalability factor is attention materialization, which is shared by all attention-based methods. Extracting attention requires constructing a tensor of shape $L \times H \times N \times N$. As the configuration increases from 32×32 to 80×64, both time and memory increase accordingly, as reflected by the extraction time and peak memory. Hypergraph construction introduces additional cost, but this step is a deterministic post-processing of already materialized attention maps rather than an extra quadratic factor beyond attention itself.

AttnHyper also provides practical controls to reduce overhead: (i) sparsification through the threshold τ to reduce hyperedge size, and (ii) selective layer/head construction. Our layer-wise analysis shows that using a single informative layer can achieve performance close to aggregating all layers, enabling a flexible trade-off between accuracy and runtime when deploying on large backbones.

Overall, the results on the 70B model demonstrate that AttnHyper remains feasible at this scale, with stable detection performance, constant detector size, and end-to-end latency that scales proportionally with the backbone attention size rather than any hypergraph-specific parameter growth.

G Comprehensive Ablation Studies

This section presents comprehensive ablation experiments analyzing the contribution of individual components within the **AttnHyper** framework. All ablation experiments are conducted on the RAGTruth using Mistral-7B-Instruct, with results averaged over five random seeds.

G.1 Impact of Number of HGNN Layers and hidden dimensions

To investigate the effect of HGNN architectural design on model performance, we conduct a systematic ablation study on two key factors: the hidden dimension size and the number of HGNN layers. Specifically, we evaluate the model under Dims $\in \{32, 64, 128\}$ and Layers $\in \{0, 1, 2, 3, 4\}$, with the results summarized in Table 16.

Experimental results show that the model consistently achieves the best performance across all tasks when the hidden dimension is set to 128 and the number of HGNN layers is 3. Consequently, we adopt this configuration for all subsequent experiments. In contrast, when Layers $= 0$, corresponding to the absence of message passing, the model exhibits the worst performance on all datasets. This observation clearly demonstrates the critical importance of the message passing mechanism in HGNN, highlighting its essential role in enabling effective structured information propagation.

Table 16. Ablation study on HGNN layers and hidden dimensions using Mistral-7B-Instruct. Performance is measured by AUROC (%).

Task	Dims	Layers=0	1	2	3	4
QA	32	0.8935	0.9389	0.9483	0.9570	0.9580
	64	0.8960	0.9483	0.9603	0.9591	0.9592
	128	0.8973	0.9488	0.9594	**0.9618**	0.9612
Summary	32	0.7933	0.8634	0.8720	0.8730	0.8687
	64	0.7958	0.8739	0.8688	0.8775	0.8777
	128	0.7968	0.8694	0.8548	**0.8789**	0.8769
Data2txt	32	0.8875	0.9257	0.9272	0.9378	0.9400
	64	0.8920	0.9332	0.9308	0.9278	0.9399
	128	0.8944	0.9318	0.9370	**0.9404**	0.9396

G.2 Impact of Attention Threshold

To determine an appropriate value for the threshold parameter τ, we conduct an ablation study to analyze its impact on hypergraph construction, training efficiency, and detection performance.

Specifically, we evaluate $\tau \in \{0.01, 0.05, 0.1, 0.5\}$, with the results summarized in Table 17 with Mistral-7B-Instruct.

As shown in the table, decreasing τ leads to denser hypergraphs, as reflected by a substantial increase in the average number of hyperedges and the overall hypergraph size. This increased structural richness consistently improves detection performance in terms of AUROC. However, excessively small values of τ incur significant computational and storage overhead, resulting in sharply increased training time. Balancing performance gains against efficiency considerations, we adopt $\tau = 0.05$ as the default setting for all subsequent experiments.

Table 17. Effect of threshold τ on the average number of hyperedges, the average hypergraph size, the average training time, and detection performance on the QA task, averaged over five different random seeds.

τ	**Num. Edges**	**Size. (MB)**	**Time. (s)**	**AUROC**
0.5	2582.54	2.38	390.03	94.47 ± 0.2
0.1	130547.66	11.76	3246.50	95.83 ± 0.1
0.05	138967.71	14.34	3748.76	96.18 ± 0.1
0.01	140476.09	26.49	7040.93	96.01 ± 0.1

G.3 Pairwise Ablation

By tuning the threshold τ, the hypergraph adaptively controls the number of nodes per hyperedge. When $\tau = 0.5$, each hyperedge collapses to a node pair, reducing the hypergraph to a standard graph trained with a GNN. The results, shown in Table 18, demonstrate the effectiveness of modeling higher-order relations beyond pairwise connections.

Table 18. Pairwise-edge ablation results (AUROC, %).

Model	**Method**	**QA**	**Summary**	**Data2txt**
Llama-3.1-8B-Instruct	Pairwise Graph (τ=0.5)	85.61	74.68	86.70
	AttnHyper	**91.31**	**78.33**	**90.57**
Qwen3-8B	Pairwise Graph (τ=0.5)	81.69	69.04	85.45
	AttnHyper	**91.61**	**76.23**	**90.44**

H Preserving Attention-Group Identity: Why Hypergraphs Are Necessary Beyond Pairwise Modeling

Key intuition for representation-level necessity. For each response token i and attention head j, self-attention naturally defines a *set* of key tokens that are jointly attended by the same query, forming a hyperedge $e_{i,j}$. This *attention group identity*—which head the group originates from and which tokens co-occur within the same group—constitutes the high-order structure we aim to preserve.

A common way to convert a hypergraph into a pairwise graph is *clique expansion*, which connects all node pairs within each hyperedge. However, clique expansion is **not** an information-preserving transformation: different hyperedge groupings can collapse into the same set of pairwise edges, fundamentally losing group identity. This is not merely a visualization issue. Once hyperedge identity disappears, the model can no longer condition message passing on *which* attention group (i.e., head-specific set) a token belongs to.

Concrete example via visualization. In our example visualization (`https://github.com/liuzhishun/NeurIPS-2026-AttnHyper/blob/main/images/sample_15607_comparison.png`), anonymized link provided in the supplement), the clique-expanded view becomes extremely dense and entangles attention groups from different heads, making it difficult to isolate head-specific "evidence sets". In contrast, the hypergraph view preserves each head-induced attention group as a distinct hyperedge, clearly showing how a hallucinated span participates in multiple head-specific groups. This structure becomes indistinguishable after clique expansion.

Why attention is intrinsically high-order. In self-attention, softmax normalization couples all keys. The weight assigned to one key changes when another key changes:

$$\frac{\partial \mathbf{A}[j,i,u]}{\partial \mathbf{k}_m} = -\mathbf{A}[j,i,u] \cdot \mathbf{A}[j,i,m] \cdot \frac{\mathbf{q}_i}{\sqrt{d_k}} \neq \mathbf{0}, \quad \text{for } m \neq u.$$

This shows that keys do not form independent pairwise relations but jointly compete for a shared attention budget. Such behavior is naturally captured by a hyperedge rather than fragmented pairwise edges.

Empirical support from pairwise ablation. Consistent with the above analysis, our pairwise ablation—where a high threshold causes hyperedges to largely collapse into pairwise edges and HGNN is replaced with a standard GNN—results in substantial performance drops compared to AttnHyper. This confirms that preserving attention-group structure provides measurable gains beyond pairwise modeling.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the core claims of the paper, including the motivation for modeling attention as hyperedges, the proposed AttnHyper framework, and its empirical advantages over pairwise baselines.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the work are discussed in Section 5.6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: See Section D for the full set of assumptions and complete proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Experimental settings, model configurations, datasets, preprocessing steps, evaluation protocols, and hyperparameters are fully described in Sec. 5 and the Appendix to enable faithful reproduction of the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will release the anonymized code, data processing scripts, and detailed reproduction instructions upon acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Experimental settings, model configurations, datasets, preprocessing steps, evaluation protocols, and hyperparameters are fully described in Sec. 5 and the Appendix to enable faithful reproduction of the reported results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Error bars are reported as mean ± standard deviation over multiple runs with different random seeds, reflecting the stability and statistical reliability of the results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details of the hardware environment, GPU memory usage, and execution time for major components are reported in Appendix. E.1 and Appendix. F.4, enabling others to estimate the required computational resources for reproduction.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research complies with the NeurIPS Code of Ethics. It involves standard publicly available datasets and models, does not introduce harmful applications, and follows appropriate research practices regarding data usage, evaluation, and reporting.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: This work is primarily methodological and does not target any specific real-world application or deployment scenario. As a result, it does not have a direct or identifiable societal impact. We therefore do not include a dedicated societal impact discussion.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The work does not involve releasing high-risk models or scraped datasets with dual-use or misuse concerns. Therefore, no specific safeguards for controlled release are required in this paper.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets, models, and external code used in this work are properly credited to their original authors, and their usage complies with the corresponding licenses and terms of use as described in the experimental section and references.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not introduce or release any new datasets, models, or code artifacts. Therefore, no additional documentation of new assets is applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing experiments or any studies with human subjects. Therefore, no participant instructions, screenshots, or compensation details are applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This work does not involve human subjects or any crowdsourcing experiments. Therefore, IRB approval, risk disclosure, and related ethical review procedures are not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: This work studies hallucination in LLMs and explicitly analyzes internal attention mechanisms as part of the proposed method. However, LLMs are not used as an external tool or generative component within the methodology; instead, they serve as the target models under analysis. Therefore, the usage of LLMs is fully described in terms of model selection, inference setting, and experimental evaluation.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.