Detecting Hallucinations for Large Language Model-Based Knowledge Graph Reasoning

Xinyan Zhu[1], Yaoqi Liu[1,2], Yue Gao[3], Huadong Ma[1], Cheng Yang[1]*, Chuan Shi[1]*

[1]Beijing University of Posts and Telecommunications
[2]Peng Cheng Laboratory [3]Tsinghua University
{zhuxinyan, yaoqiliu, mhd, yangcheng, shichuan}@bupt.edu.cn
gaoyue@tsinghua.edu.cn

Abstract

Knowledge graph (KG) reasoning infers new knowledge from existing facts and is widely applied in question answering, recommendation, and decision support. With the rapid development of large language models (LLMs), LLM-based KG reasoning frameworks have become increasingly popular by leveraging retrieved KG information. However, hallucinations in LLMs remain a critical issue. Even when relevant KG knowledge is incorporated, models may still generate incorrect outputs, leading to misinformation and unreliable decisions. Existing hallucination detection methods either focus on LLM internal states or verify consistency with retrieved contexts, but both overlook the structural information in KGs, resulting in suboptimal performance. To address this gap, we propose **LUCID**, the first hal**LU**cination dete**C**t**I**on method for LLM-based knowle**D**ge graph reasoning frameworks. LUCID extracts node and edge features from attention scores and semantic similarities, and integrates them with KG structure using a graph neural network (GNN). Moreover, we also construct manually annotated benchmark datasets for evaluation. Experiments on nine datasets show that LUCID achieves state-of-the-art performance compared to 15 baselines. The code and data are available at: https://github.com/BUPT-GAMMA/LUCID.

1 Introduction

Knowledge graph (KG) reasoning (Chen et al., 2020) focuses on analyzing and deducing existing factual information in KGs to infer implicit knowledge. It supports downstream tasks such as question answering, recommendation systems, and decision-making (Guo et al., 2020; Ji et al., 2021). Integrating large language models (LLMs) (Ouyang et al., 2022; Qwen et al., 2025; OpenAI et al., 2024) with KGs, particularly through LLM-based KG reasoning frameworks (Hu et al., 2023; Pan et al., 2024), has become popular. These frameworks retrieve relevant triples from KGs, incorporate them into the prompt, and guide LLMs to generate more accurate answers, enhancing performance in complex tasks.

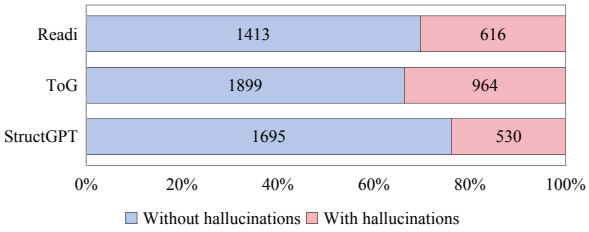

Figure 1: Distribution of responses with / without hallucinations across 3 representative frameworks. The marked numbers indicate the sample counts.

Despite their effectiveness, LLM-based KG reasoning frameworks still suffer from severe hallucinations (Ji et al., 2023; Huang et al., 2025), generating factually incorrect responses even when correct KG evidence has already been retrieved. To investigate this issue, we manually annotate responses generated by three representative frameworks, including Readi (Cheng et al., 2024), ToG (Sun et al., 2024), and StructGPT (Jiang et al., 2023), on three KBQA datasets. As shown in Figure 1, the average hallucination rate reaches 29.65%. Figure 2 further illustrates a typical failure case where the LLM ignores the correct KG evidence and produces a hallucinated answer. Such hallucinations can severely compromise the accuracy of KG reasoning, leading to incorrect conclusions and misguided decision-making in downstream applications.

These observations suggest that hallucinations in KG reasoning differ from conventional text hallucinations in several important aspects. The issue is often caused not by missing knowledge, but by the incorrect utilization of retrieved KG evidence during reasoning. Hallucinations typically arise when the reasoning process deviates from the semantic and structural evidence supported by the

*Corresponding Author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 3254–3271
October 24-29, 2026 ©2026 Association for Computational Linguistics

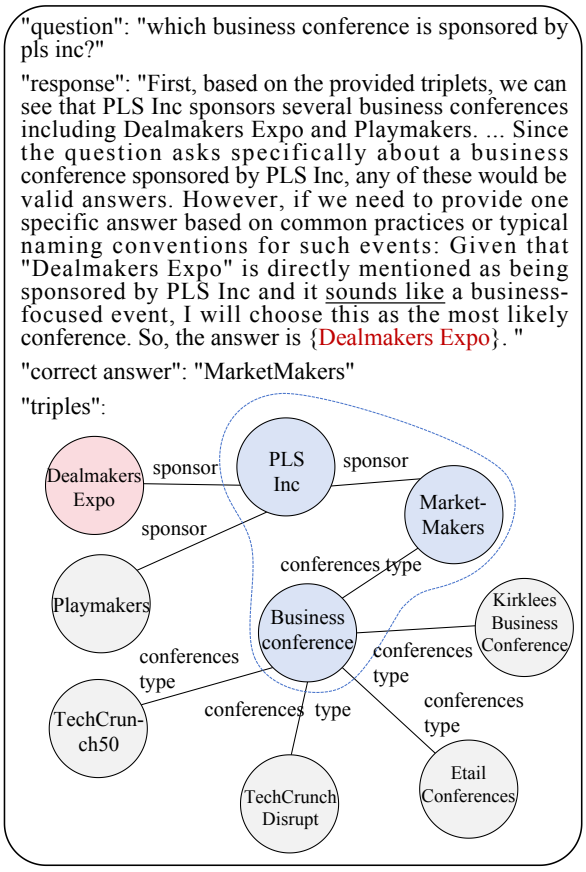

Figure 2: A hallucination example in LLM-based KG reasoning. The nodes and edges that can help LLMs make correct reasoning are circled in blue, while the incorrect final answers are marked in red.

KG. Therefore, hallucination detection in KG reasoning should be formulated as a reasoning consistency modeling problem rather than simple response verification.

However, existing hallucination detection methods are not designed for this setting. Methods based on LLM internal states (e.g., EigenScore (Chen et al., 2024)) ignore external KG evidence, while RAG-specific methods (e.g., RAGAs Faithfulness (Es et al., 2024)) mainly focus on response-context consistency without modeling graph structures. Consequently, they cannot effectively characterize whether the reasoning process remains faithful to KG-supported reasoning paths.

To address this gap, we propose the first hal**LU**cination dete**CtI**on framework for LLM-based knowle**D**ge graph reasoning, termed **LUCID**. Instead of treating hallucination detection as pure response verification, LUCID models hallucinations as inconsistencies between LLM reasoning behaviors and KG-supported reasoning structures.

Our key insight is that faithful reasoning exhibits aligned attention allocation, semantic relevance, and structural propagation patterns, whereas hallucinated reasoning breaks this alignment. Based on this insight, LUCID jointly models: (1) LLM attention over KG elements, reflecting evidence utilization during generation; (2) semantic relevance between KG relations and queries; and (3) structural reasoning patterns within KG subgraphs.

Specifically, we first extract attention scores between generated responses and KG elements from the LLM generation process. We then compute semantic relevance scores between KG relations and queries using a pre-trained language model. These signals are injected into KG subgraphs as node and edge features and further processed by a graph neural network (GNN) to model reasoning consistency and predict hallucination probabilities.

Extensive experiments show that LUCID consistently outperforms 15 baseline methods and achieves state-of-the-art performance. Further analysis demonstrates that the hallucination probabilities predicted by LUCID can serve as effective signals for downstream QA refinement.

The main contributions are as follows:

(1) We firstly point out the hallucinations in LLM-based KG reasoning frameworks and construct benchmark datasets through manual annotation, with the subsequent analysis uncovering an average hallucination rate of 29.65%.

(2) We propose LUCID, the first hallucination detection method specifically designed for LLM-based KG reasoning frameworks, which integrates LLM internal state information, KG semantic and structural information.

(3) Extensive experiments demonstrate that LUCID consistently outperforms all baselines and exhibits strong generalization ability across frameworks and datasets. Moreover, the hallucination probabilities predicted by LUCID can effectively improve downstream QA performance.

2 Related Work

2.1 General Hallucination Detection Methods

General hallucination detection methods can be broadly categorized into uncertainty-based, sampling-based, and LLM-based methods.

Uncertainty-based methods estimate hallucination risk from the uncertainty of LLM generations. Representative methods include LN-Entropy (Malinin and Gales, 2020), Energy (Liu et al., 2020), and Perplexity (Ren et al., 2022), which measure

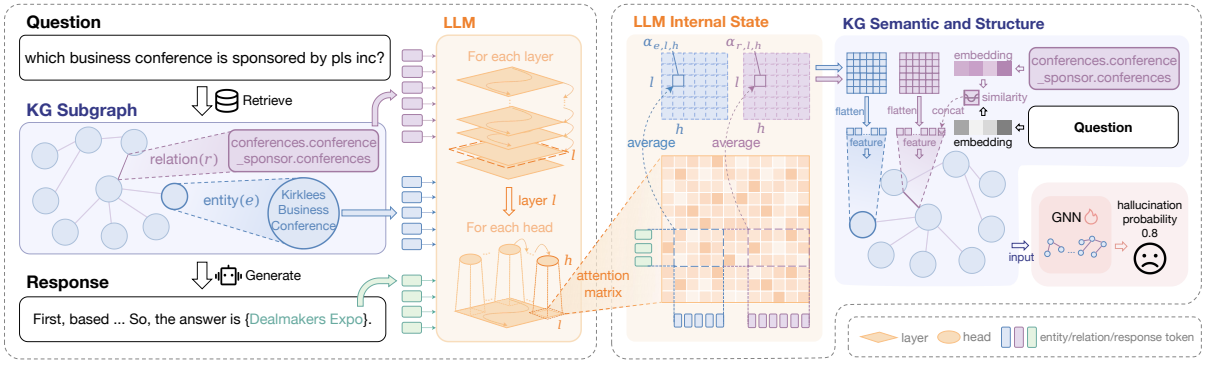

Figure 3: The architecture of LUCID. The diagram uses one representative node and edge from the graph as an example. The left portion depicts the typical workflow of an LLM-based KG reasoning framework and the relationships between the various layers and heads inside the LLMs, while the right portion shows how the method constructs a KG subgraph by integrating three information components: LLM internal state, KG semantic and structure information. This enriched subgraph is then processed by a GNN model to generate the predicted hallucination probability.

uncertainty from different perspectives. Focus (Zhang et al., 2023) further improves factuality assessment by emphasizing uncertain tokens and keywords.

Sampling-based methods detect hallucinations by generating multiple responses and measuring their consistency. Lexical Similarity (Lin et al., 2022) evaluates semantic consistency across sampled outputs, while SelfCheckGPT (Manakul et al., 2023) compares generated responses using NLI and BertScore metrics.

LLM-based methods leverage prompting strategies, fine-tuning, or internal model representations for hallucination detection. EigenScore (Chen et al., 2024) analyzes the covariance structure of LLM embeddings to measure semantic consistency, while MetaQA (Yang et al., 2025) detects hallucinations through metamorphic prompt mutations without requiring external knowledge.

2.2 RAG-specific Hallucination Detection Methods

RAG-specific hallucination detection methods mainly evaluate the consistency between generated responses and retrieved contexts.

RAGAs Faithfulness (Es et al., 2024) verifies whether response statements are supported by retrieved evidence, while Trulens Groundedness[1] measures response-context alignment through overlap-based groundedness scores. LettuceDetect (Kovács and Recski, 2025) performs hallucination detection via token classification over context-question-answer triples. ReDeEP (Sun et al., 2025) predicts hallucinations using token-level or chunk-level scores that model the interaction between external context and parametric knowledge.

2.3 LLM-based KG Reasoning Frameworks

LLM-based KG reasoning frameworks integrate structured KGs with LLMs to enhance complex reasoning and question answering. These frameworks typically retrieve KG triples and incorporate them into the reasoning process to support multi-hop inference and knowledge-grounded generation.

Representative frameworks adopt different KG integration strategies. Readi (Cheng et al., 2024) performs reasoning path editing over KG-guided reasoning paths. ToG (Sun et al., 2024) iteratively explores KGs to collect evidence for multi-hop reasoning. StructGPT (Jiang et al., 2023) interacts with structured KG interfaces through iterative reading-and-reasoning cycles.

3 Methodology

3.1 Problem Formulation and Overview

Given a query Q, LLM-based KG reasoning frameworks retrieve a KG subgraph $G = (V, E)$, where V and E denote entities and relations, respectively. The retrieved subgraph is serialized into text S_Q and concatenated with Q as the input to the LLM, which then generates the response R_Q.

In this work, a response is considered hallucinated if it contains factual statements unsupported by the retrieved KG evidence or contradictory to objective facts. The hallucination detection task is

[1] https://www.trulens.org/

formulated as learning a function:

$$D(Q, S_Q, R_Q) = \begin{cases} 1, & \text{if } R_Q \text{ has hallucinations,} \\ 0, & \text{otherwise.} \end{cases}$$

Rather than treating hallucination detection as pure response verification, we model it as a reasoning consistency modeling problem. Specifically, hallucinations in KG reasoning are often associated with: (1) inconsistent attention allocation over KG evidence; (2) reasoning through semantically irrelevant relations; and (3) structurally inconsistent reasoning paths within the retrieved subgraph.

Based on this observation, LUCID jointly models three complementary signals: attention-based evidence utilization from LLM internal states, semantic relevance between KG relations and queries, and structural dependencies within KG subgraphs. As illustrated in Figure 3, LUCID first extracts node and edge attention features from LLM internal states, then computes semantic similarity scores between relations and queries, and finally performs graph-based hallucination prediction over the retrieved KG subgraph.

3.2 LLM Internal State Information Processing

The internal states of LLMs contain rich semantic information, where the attention mechanism is critical for modeling contextual dependencies and token relationships (Vaswani et al., 2017). To leverage this capability, we analyze the attention scores generated during response generation.

In Transformer architectures, multi-layer, multi-head attention jointly determines how the model processes and reasons over input information. Different layers progressively refine representations, while attention heads capture diverse token dependencies (Clark et al., 2019).

Formally, given an input context $C = Q \parallel S_Q$, where $\parallel$ denotes concatenation and S_Q is the textually serialized subgraph retrieved for query Q, let $T_{S_Q} = \{t_1, t_2, \ldots, t_i\}$ denote the tokens corresponding to elements in S_Q. During response generation, the model produces a token sequence A. For each response token $a \in A$, the LLM computes attention scores $\alpha_{a,t} \in [0, 1]$ over tokens $t \in T_{S_Q}$, indicating the degree of focus on the retrieved KG subgraph. For simplicity, KG nodes (entities) and edges (relations) are denoted as e and r, respectively.

To obtain graph-level representations, we aggregate token-level attention across all response tokens. For a node e with token set $T_e \subseteq T_{S_Q}$, the attention score at layer l and head h is:

$$\alpha_{e,l,h} = \frac{1}{|T_e| \cdot |A|} \sum_{a \in A} \sum_{t \in T_e} \alpha_{a,t,l,h} \tag{1}$$

Here, $\alpha_{a,t,l,h}$ denotes the attention weight between response token a and token t.

Similarly, for an edge r with token set $T_r \subseteq T_{S_Q}$, the attention score is:

$$\alpha_{r,l,h} = \frac{1}{|T_r| \cdot |A|} \sum_{a \in A} \sum_{t \in T_r} \alpha_{a,t,l,h} \tag{2}$$

The resulting scores across all layers $l \in \{1, \ldots, L\}$ and heads $h \in \{1, \ldots, H\}$ are aggregated into two mean attention matrices: $\mathbf{M}_e \in \mathbb{R}^{L \times H}$ for nodes and $\mathbf{M}_r \in \mathbb{R}^{L \times H}$ for edges.

3.3 KG Semantic and Structural Information Processing

The relevance of KG elements to the query directly affects reasoning reliability. To capture this relevance, we compute semantic similarity between KG relations and the query, distinguishing task-related from task-irrelevant edges. Since entity surface forms are often ambiguous and provide limited semantic information, semantic relevance is computed only for relations.

For each edge in the retrieved KG subgraph, we first obtain the textual description of its relation. We then employ the lightweight Transformer-based model all-MiniLM-L6-v2[2] to generate embeddings for both relations and queries. Given the relation embedding $\text{emb}(r) \in \mathbb{R}^d$ and query embedding $\text{emb}(Q) \in \mathbb{R}^d$, we compute the semantic similarity score $s(r, Q)$ using cosine similarity.

Rather than directly using serialized triples for hallucination detection, we construct a graph from the retrieved KG subgraph. Let V and E denote the sets of nodes and edges, respectively, and represent the graph as $G = (V, E)$. Node and edge features are then constructed based on the processed information above.

Specifically, node features are obtained by flattening node attention matrices: $\mathbf{x}_e = \text{flatten}(\mathbf{M}_e)$. Edge features further incorporate semantic similarity by concatenating the flattened edge attention matrices with the similarity score: $\mathbf{x}_r = \text{concat}(\text{flatten}(\mathbf{M}_r), s(r, Q))$.

[2]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

Method	GrailQA				WebQSP				QALD-10			
	ACC	AUC	PCC	AVG	ACC	AUC	PCC	AVG	ACC	AUC	PCC	AVG
LLM	0.7705	0.6301	0.3545	0.5850	0.7299	0.6289	0.2970	0.5519	0.6923	0.6779	0.3480	0.5727
Perplexity	0.6410	0.7150	0.3160	0.5573	0.6299	0.6688	0.2368	0.5118	0.6390	0.6624	0.2462	0.5159
Energy	0.5254	0.5543	0.0368	0.3722	0.4729	0.4555	-0.0549	0.2912	0.5024	0.4416	-0.0186	0.3085
LN-Entropy	0.6933	0.7315	0.3168	0.5805	0.6337	0.6581	0.2268	0.5062	0.6000	0.6582	0.2117	0.4900
Lexical Similarity	0.6479	0.7036	0.2782	0.5432	0.5861	0.5746	0.1159	0.4255	0.6098	0.6299	0.1898	0.4765
SelfCk-Nli	0.6575	0.7527	0.3787	0.5963	0.6765	0.6954	0.3353	0.5691	0.6878	0.7246	0.3469	0.5864
SelfCk-BERTScore	0.6176	0.6775	0.2186	0.5046	0.6089	0.6477	0.2436	0.5001	0.6098	0.6691	0.2934	0.5241
Focus	0.5395	0.5253	0.0475	0.3708	0.5395	0.5253	0.0778	0.3809	0.5610	0.4837	0.0618	0.3688
EigenScore	0.6052	0.6327	0.2051	0.4810	0.5005	0.5326	0.0726	0.3686	0.5463	0.5447	0.1412	0.4107
MetaQA	0.5392	0.5522	0.1043	0.3986	0.6175	0.5578	0.1121	0.4291	0.6000	0.6558	0.2117	0.4892
RAGAs	0.7015	0.7341	0.3578	0.5978	0.6394	0.6949	0.3080	0.5474	0.5610	0.6179	0.2013	0.4601
Trulens	0.7538	**0.8202**	0.4800	0.6847	0.6969	0.7636	0.4010	0.6205	0.6404	0.6778	0.2630	0.5271
LettuceDetect	0.7455	0.5222	0.4611	0.5763	0.6108	0.4399	0.3337	0.4615	0.4829	0.4000	0.2724	0.3851
ReDeEP (token)	0.7125	0.7850	0.3931	0.6302	0.6546	0.6487	0.2481	0.5171	0.6098	0.6583	0.3072	0.5251
ReDeEP (chunk)	0.7262	0.7914	0.4122	0.6433	0.6841	0.7488	0.3125	0.5818	0.6976	0.7529	0.3466	0.5990
LUCID	**0.7882***	0.8045	**0.5171***	**0.7033***	**0.7307**	**0.7910***	**0.4043**	**0.6420***	**0.7171***	**0.7616***	**0.3629***	**0.6139***

Table 1: Main results of the hallucination detection performance of LUCID and 15 baseline methods on three datasets GrailQA, WebQSP and QALD-10 under the **Readi** framework. The evaluation metrics include ACC, AUC, PCC and the average of the three (AVG). Bold and underline fonts denote the best and second-best among all methods. * indicates that LUCID significantly outperforms the optimal baseline at the 99% significance level.

3.4 GNN for Hallucination Detection

After constructing the graph $G = (V, E)$ and the corresponding node and edge features $\{\mathbf{x}_e\}_{e \in V}$ and $\{\mathbf{x}_r\}_{r \in E}$, we employ a graph isomorphism network with edge features (GINE) (Hu et al., 2019) to predict hallucination probabilities. Compared with graph isomorphism networks (GIN) (Xu et al., 2018), GINE explicitly incorporates edge features into message passing, making it more suitable for KG-based reasoning scenarios with rich relational information.

Formally, node features are iteratively updated over K layers. For node e, the feature update at layer k is defined as

$$\mathbf{h}_e^{(k)} = \mathrm{MLP}^{(k)}\Bigg(\mathbf{h}_e^{(k-1)} + \sum_{(u,e)\in E} \mathrm{ReLU}\left(\mathbf{h}_u^{(k-1)} + \mathbf{x}_{(u,e)}\right)\Bigg) \tag{3}$$

Here, $\mathbf{h}_e^{(0)} = \mathbf{x}_e$, $\mathrm{MLP}^{(k)}$ denotes the multilayer perceptron at layer k, and $\mathbf{x}_{(u,e)}$ is the edge feature between nodes u and e.

After K layers, a graph-level representation is obtained through sum pooling: $\mathbf{h}_G = \mathrm{sum}\left(\{\mathbf{h}_e^{(K)}\}_{e\in V}\right)$. The hallucination probability is then predicted by

$$p = \sigma\left(\mathbf{W}\mathbf{h}_G + \mathbf{b}\right) \tag{4}$$

Here, σ denotes the sigmoid function, and $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters.

Using manually annotated datasets, we assign label 1 to hallucinated responses and label 0 otherwise, and train the model using binary cross-entropy loss. During inference, the detection threshold is determined by maximizing the geometric mean of sensitivity and specificity on the ROC curve (Davis and Goadrich, 2006).

4 Experiments

4.1 Experimental Setting

4.1.1 Benchmark

We construct the training set using responses generated by Readi on CWQ (Talmor and Berant, 2018), while the benchmark test set contains responses from Readi, ToG, and StructGPT on WebQSP (Yih et al., 2016), GrailQA (Gu et al., 2021), and QALD-10 (Usbeck et al., 2024). These frameworks represent substantially different KG reasoning paradigms: ToG performs search-based reasoning through iterative KG exploration, StructGPT adopts rule-based reasoning via structured KG interactions, and Readi employs reflective reasoning by refining reasoning paths during inference.

	Readi						ToG						StructGPT					
	GrailQA		WebQSP		QALD-10		GrailQA		WebQSP		QALD-10		GrailQA		WebQSP		QALD-10	
	EM	Cost	EM	Cost	EM	Cost	EM	Cost	EM	Cost	EM	Cost	EM	Cost	EM	Cost	EM	Cost
Qwen2.5-7B	69.18	-	69.83	-	50.24	-	60.47	-	67.25	-	46.43	-	61.02	-	71.76	-	54.10	-
Qwen3-235B	77.03	31.13	75.17	58.45	62.93	7.13	67.91	30.86	74.27	64.25	61.92	11.02	68.17	23.57	76.49	22.84	60.45	7.85
Mixed	76.34	11.00	74.31	25.97	62.93	3.06	66.67	14.11	73.20	26.77	60.06	5.15	67.79	10.04	75.80	11.79	59.33	3.95
Δ	-0.9%	-64.7%	-1.1%	-55.6%	0.0%	-57.1%	-0.8%	-54.3%	-1.4%	-58.3%	-3.0%	-53.2%	-0.6%	-57.4%	-0.9%	-48.4%	-1.9%	-49.7%

Table 2: Comparison of QA strategies with and without LUCID-guided refinement. The EM column reports accuracy in percentage, whereas the Cost column is measured in dollars ($). The first two strategies represent using Qwen2.5-7B and Qwen3-235B for QA tasks on 9 datasets. Mixed refers to the process where, when LUCID detects a high hallucination probability in Qwen2.5-7B outputs, these cases are reprocessed by Qwen3-235B. The Δ row represents the percentage difference between the Mixed and Qwen3-235B model.

Consequently, they exhibit diverse hallucination patterns and graph structures. All responses are manually annotated (dataset details in Appendix A, guidelines in Appendix D).

To evaluate generalization capability, we train LUCID exclusively on Readi outputs from CWQ and test on unseen reasoning frameworks (ToG, StructGPT) and out-of-distribution datasets (WebQSP, GrailQA, QALD-10), simulating realistic deployment scenarios with diverse reasoning behaviors and KG environments.

4.1.2 Baselines

We compare LUCID with 15 baselines, which can be divided into two categories. One is general hallucination detection methods: LLM (Leveraging the LLM's own capabilities for detection), LN-Entropy, Energy, Perplexity, Lexical Similarity, SelfCk-Nli, SelfCk-BERTScore, Focus, EigenScore, MetaQA; the other is RAG-specific hallucination detection methods: RAGAs Faithfulness, Trulens Groundedness, LettuceDetect, ReDeEP (token), ReDeEP (chunk). See Appendix B for a detailed introduction.

For methods that require open-source LLMs to be deployed, we use the Qwen2.5-7B-Instruct model; for methods that need to use closed-source LLMs, we use the GPT-4o-mini model.

4.1.3 Model Setup

We train GNN on the annotated CWQ dataset (as described in Section 4.1.1). For model parameters, we use two layers of GINEConv, set the number of channels in the hidden layer to 512, set the learning rate to 1×10^{-3}, and train 300 epochs.

4.1.4 Evaluation Metrics

In order to comprehensively evaluate the performance of all methods in hallucination detection, we use the evaluation metrics: ACC (Accuracy), AUC (Area Under the ROC Curve), PCC (Pearson Correlation Coefficient), and AVG, where AVG is the average of ACC, AUC, and PCC.

	GrailQA	WebQSP	QALD-10	Avg
Readi	66.85%	87.25%	68.77%	74.29%
ToG	81.49%	93.16%	98.45%	91.03%
StructGPT	98.12%	99.30%	93.68%	97.03%

Table 3: Percentage of sparse graphs among KG subgraphs retrieved by three frameworks on three datasets. The last column reports the average percentage across datasets for each framework.

4.2 Main Results

The performance of LUCID and all baselines across three frameworks and three datasets is shown in Tables 1, 5, and 6. The results of the latter two experiments are placed in the Appendix E. Overall, LUCID achieves state-of-the-art (SOTA) performance across all settings. On the AVG metric, it improves over SelfCk and REDEEP (chunk) by 6.76% and 5.48%, respectively, while maintaining high efficiency with an average inference time of 0.04 ms per sample.

Among the three frameworks, LUCID achieves the largest average improvement on Readi (+2.81%). This is likely because Readi's reasoning-path editing mechanism incorporates more valid triples into retrieved subgraphs, producing denser local graph structures (as shown in Table 3) that are particularly suitable for LUCID's message-passing mechanism. In contrast, ToG and StructGPT tend to retrieve sparser subgraphs, providing weaker structural evidence for hallucination detection and thus yielding comparatively smaller, though still consistent, gains.

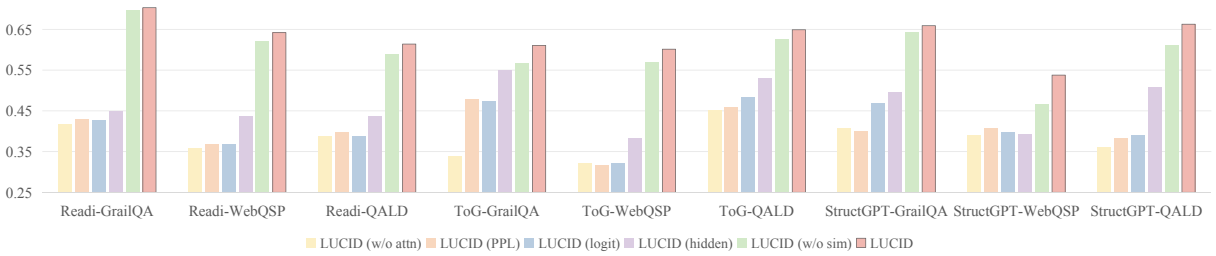

Figure 4: AVG performance of the method on 3 frameworks and 3 datasets when using different features.

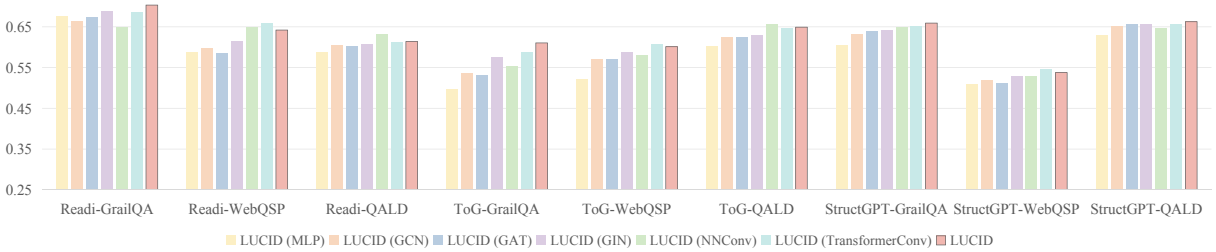

Figure 5: AVG performance of the method on 3 frameworks and 3 datasets when using different trained models.

From the dataset perspective, LUCID achieves the largest average improvement on WebQSP (+2.57%), where hallucinations often arise from structurally unsupported relation selection in single- and two-hop reasoning. On QALD-10, which contains more diverse and complex query formulations, LUCID still maintains stable gains (+2.37%), demonstrating robustness under higher linguistic and structural variability.

Across all settings, RAG-specific methods outperform general-purpose baselines by approximately 11.46% on AVG, highlighting the importance of fine-grained evidence alignment in KG reasoning hallucination detection. Unlike prior methods focusing mainly on surface-level response-context consistency, LUCID further aligns internal attention patterns with KG structure.

Utilize Hallucination Probabilities for QA Refinement. In order to show that the probability of hallucinations detected by LUCID can be applied to improving the accuracy of QA, we evaluate the effectiveness as a signal for response refinement.

Specifically, we compare the following three QA strategies. The first two apply a single model to the entire dataset: Qwen2.5-7B and Qwen3-235B. Qwen2.5-7B is deployed locally on an NVIDIA RTX 4060 Ti (16GB) GPU with no monetary cost, whereas Qwen3-235B is accessed via API calls and incurs monetary cost. The third strategy, denoted as *Mixed*, performs selective refinement based on hallucination detections of LUCID. Responses are first generated by Qwen2.5-7B, and instances with high hallucination probabilities are refined using Qwen3-235B, while the remaining answers are retained. For all strategies, we report exact-match (EM) accuracy and monetary cost ($).

The results are shown in Table 2. The EM accuracy of Mixed is highly similar to that of the Qwen3-235B model across all datasets, with an average difference of 1.18%. This indicates that refining certain outputs using the hallucination probabilities provided by LUCID can achieve performance close to that of refining all outputs. Besides, the cost of QA is substantially reduced in Mixed. Compared to Qwen3-235B, Mixed saves an average of 55.4% in monetary costs. These findings demonstrate that the hallucination detection mechanism of LUCID can significantly reduce costs while improving the accuracy of the model.

For additional experimental results, including detailed hyperparameter studies and representative case studies, please refer to Appendix G and I.

4.3 Cross-dataset and Cross-framework Generalization

To evaluate the generalization ability of LUCID, we conduct cross-dataset and cross-framework transfer experiments, where the model is trained on one framework or dataset and directly tested on unseen settings. Results are shown in Table 4.

For cross-framework transfer, LUCID maintains competitive performance when transferred across different reasoning frameworks. For example, the model trained on Readi achieves stable results on

	Readi			ToG			StructGPT		
	GrailQA	**WebQSP**	**QALD-10**	**GrailQA**	**WebQSP**	**QALD-10**	**GrailQA**	**WebQSP**	**QALD-10**
LUCID (Readi)	-	-	-	0.6210	0.6110	0.6457	0.6687	0.5340	0.6827
LUCID (ToG)	0.7041	0.6363	0.6011	-	-	-	0.6646	0.5425	0.6762
LUCID (StructGPT)	0.6939	0.6295	0.6080	0.6154	0.6264	0.6525	-	-	-
LUCID (GrailQA)	-	0.6486	0.6080	-	0.6108	0.6561	-	0.5237	0.6696
LUCID (WebQSP)	0.7089	-	0.6293	0.6296	-	0.6576	0.6696	-	0.6759
LUCID (QALD-10)	0.6896	0.6327	-	0.6250	0.6090	-	0.6492	0.5429	-
LUCID	0.7033	0.6420	0.6139	0.6105	0.6013	0.6491	0.6591	0.5377	0.6638

Table 4: Cross-dataset and cross-framework transfer performance of LUCID on KBQA tasks. "-" indicates the in-domain setting, which was trained on the corresponding dataset and is therefore excluded from transfer evaluation.

both ToG and StructGPT. Similar trends can also be observed for models trained on ToG and StructGPT, indicating that hallucination graphs contain transferable structural patterns across different reasoning frameworks.

For cross-dataset transfer, models trained on one dataset also generalize well to other datasets under the same framework. Interestingly, in several settings, the transferred models even outperform the corresponding in-domain models. We conjecture that this phenomenon is caused by the diversity of hallucination patterns across different datasets and frameworks. Training on external settings may expose the model to more varied structural hallucination behaviors, reducing overfitting to dataset-specific reasoning patterns and improving the robustness of learned graph representations.

4.4 Ablation Studies

We conduct ablation studies to evaluate the effectiveness of different feature components and graph models in LUCID. Experiments are performed on nine settings across three datasets and three frameworks: Readi-GrailQA, ToG-GrailQA, StructGPT-GrailQA, Readi-WebQSP, ToG-WebQSP, StructGPT-WebQSP, Readi-QALD, ToG-QALD, and StructGPT-QALD.

4.4.1 Does the content of the feature matter?

We first investigate the contribution of different feature components by modifying node and edge features. Specifically, we remove semantic similarity scores from edge features, and replace attention matrices with either hidden layer scores (Dai et al., 2022) or randomly initialized matrices while keeping the feature dimensions unchanged.

We denote the full method as "LUCID", the variant without semantic similarity as "LUCID (w/o sim)", the variant replacing attention matrices with random initialization as "LUCID (w/o attn)", and the variant using hidden layer scores as "LUCID (hidden)". To further compare with other API-friendly signals, we additionally consider "LUCID (logit)" and "LUCID (PPL)", which derive features from logits and perplexity (PPL), respectively. Since logits and PPL are defined over output tokens and cannot be directly mapped to input KG elements, we obtain their features through input-level perturbation. Specifically, for each entity or relation, we mask its input token span, compute the resulting logit or PPL, and use its difference from the baseline value as the corresponding feature. Results are shown in Figure 4.

Across all experimental settings, LUCID achieves the best overall performance. Removing semantic similarity yields a moderate performance drop, while replacing attention matrices with hidden-layer scores or random initialization leads to substantial performance degradation; logit- and PPL-based variants also underperform LUCID noticeably. These results demonstrate the effectiveness of integrating attention information with KG semantic similarity, show that KG semantics supply complementary information beyond attention signals, and highlight the critical role of attention in hallucination detection. They further suggest that output-level signals such as logits and PPL are less informative for hallucination detection than attention, and support attention as an effective indicator of evidence utilization in KG reasoning.

4.4.2 Does the choice of GNN model matter?

To validate the importance of graph-aware modeling, we replace GINE with MLP, GCN (Kipf and Welling, 2016), GAT (Veličković et al., 2017), GIN (Xu et al., 2018), NNConv (Gilmer et al., 2017), and TransformerConv (Yunsheng et al., 2021). Since GCN, GAT, and GIN cannot directly utilize edge features, we concatenate adjacent edge features with node features as their input; by con-

trast, NNConv and TransformerConv natively support edge features during message passing and can incorporate edge features directly without such concatenation. All models adopt identical training settings to ensure a fair comparison.

As shown in Figure 5, LUCID obtains the best overall performance. GINE outperforms GIN by better leveraging edge features for message-passing under KG reasoning scenarios; GCN and GAT perform worse than GINE across most settings, and the MLP variant yields consistently inferior results compared with GNN-based models. Among edge-feature-aware GNNs, GINE achieves the strongest overall performance, with NNConv and TransformerConv delivering competitive performance. These observations highlight the necessity of explicit KG topology modeling via GNNs, validate the efficacy of graph-based modeling, and demonstrate that LUCID's superiority is not restricted to any single GNN architecture.

5 Conclusion

This paper addresses hallucination detection in LLM-based KG reasoning frameworks by proposing LUCID, a novel method that integrates LLM internal state information with KG semantic and structural information.

Future work will focus on improving fine-grained hallucination attribution and extending evaluation to broader KG environments, such as multilingual and cross-domain settings, to further enhance the robustness and reliability of LLM-based KG reasoning systems.

Limitations

Although our method achieves strong hallucination detection performance, several limitations remain. The current framework predicts only a graph-level hallucination probability and cannot provide fine-grained attribution for specific tokens, which limits interpretability and actionable error correction. In addition, the method relies on extracting layer- and head-level attention information during generation, making direct deployment challenging for closed-source LLM APIs where internal attention weights are inaccessible. Practical deployment may therefore require approximate alternatives based on behavioral or uncertainty signals.

The detector is also somewhat coupled with the underlying base model distribution. When the backbone LLM changes, shifts in reasoning patterns and attention behaviors may reduce detection performance, requiring retraining or adaptation. Finally, while the framework introduces graph-based structural modeling into hallucination detection, some components still follow relatively standard graph classification paradigms, leaving room for further methodological innovation.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. 62550138, 62572064, 62472329) and the Beijing Natural Science Foundation (No.253004). And we used AI to assist with language editing and limited text drafting.

References

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

Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. *Expert systems with applications*, 141:112948.

Sitao Cheng, Ziyuan Zhuang, Yong Xu, Fangkai Yang, Chaoyun Zhang, Xiaoting Qin, Xiang Huang, Ling Chen, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. 2024. Call me when necessary: LLMs can efficiently and faithfully reason over structured environments. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4275–4295, Bangkok, Thailand. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL workshop BlackboxNLP: analyzing and interpreting neural networks for NLP*, pages 276–286.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.

Jesse Davis and Mark Goadrich. 2006. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. Ragas: Automated evaluation of retrieval augmented generation. In *Proceed-*

ings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, pages 150–158.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. Pmlr.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568.

Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2023. A survey of knowledge enhanced pre-trained language models. *IEEE Transactions on Knowledge and Data Engineering*, 36(4):1413–1430.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2).

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 conference on empirical methods in natural language processing*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Ádám Kovács and Gábor Recski. 2025. Lettucedetect: A hallucination detection framework for rag applications. *arXiv preprint arXiv:2502.17125*.

Zi Lin, Jeremiah Zhe Liu, and Jingbo Shang. 2022. Towards collaborative neural-symbolic graph semantic parsing via uncertainty. *Findings of the Association for Computational Linguistics: ACL 2022*.

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

Andrey Malinin and Mark Gales. 2020. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *EMNLP*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J Liu. 2022. Out-of-distribution detection and selective generation for conditional language models. *arXiv preprint arXiv:2209.15558*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

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

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.

Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, Muhammad Saleem, and Andreas Both. 2024. Qald-10 – the 10th challenge on question answering over linked data: Shifting from dbpedia to wikidata as a kg for kgqa. *Semantic Web*, 15(6):2193–2207.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Borui Yang, Md Afif Al Mamun, Jie M Zhang, and Gias Uddin. 2025. Hallucination detection in large language models with metamorphic relations. *Proceedings of the ACM on Software Engineering*, 2(FSE):425–445.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32.

Shi Yunsheng, Huang Zhengjie, Feng Shikun, Zhong Hui, Wang Wenjing, and Sun Yu. 2021. Masked label prediction: Unified message passing model for semi-supervised classification. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1548–1554.

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

A Datasets

The KBQA datasets used in the experiment are as follows:

CWQ (Talmor and Berant, 2018) is a dataset that features complex, broad questions requiring reasoning over multiple information pieces and diverse paraphrasing, designed to test models' ability to decompose questions and integrate answers for complex QA tasks.

WebQSP (Yih et al., 2016) is a single-hop and multi-hop QA dataset with annotated SPARQL queries, commonly used to evaluate semantic parsing and structured reasoning on complex questions.

GrailQA (Gu et al., 2021) is a large-scale KBQA dataset containing diverse questions and corresponding logical forms over Freebase. It evaluates three levels of generalization—i.i.d., compositional, and zero-shot—and includes questions involving multi-hop reasoning, aggregation, and other complex operations.

QALD-10 (Usbeck et al., 2024) is a multilingual KGQA benchmark that features complex questions. It primarily evaluates model performance in cross-lingual and cross-domain reasoning tasks. In our experiments we selected the English subset of this dataset.

B Baseline Details

LN-Entropy (Malinin and Gales, 2020) quantifies generation uncertainty at the sequence level by computing length-normalized entropy, which serves as an effective indicator for identifying hallucinated outputs

Energy (Liu et al., 2020) identifies hallucinations by analyzing the uncertainty in the generated response using energy functions. Higher energy suggests a higher likelihood of hallucinations.

Perplexity (Ren et al., 2022) employs the perplexity metric for LLM-generated responses, with elevated perplexity scores signaling higher uncertainty and a greater tendency toward hallucinatory content.

Lexical Similarity (Lin et al., 2022) is a straightforward semantic consistency metric that computes the mean similarity across multiple generated responses to quantify answer agreement.

SelfCk-Nli (Manakul et al., 2023) extends the SelfCheckGPT framework by employing natural

language inference (NLI) to quantify the semantic consistency between generated responses and their sampled variations.

SelfCk-BERTScore (Manakul et al., 2023) also modifies the SelfCheckGPT framework by employing BERTScore to evaluate semantic alignment between the primary generated response and its sampled counterparts.

Focus (Zhang et al., 2023) is a reference-free, uncertainty-based method for LLM hallucination detection, mimicking human fact-checking by focusing on key keywords, uncertain historical tokens, and token properties via proxy models.

EigenScore (Chen et al., 2024) measures semantic consistency of responses via the average logarithm of eigenvalues of the covariance matrix of their embeddings from LLMs' internal states.

MetaQA (Yang et al., 2025) is a self-contained hallucination detection method for LLMs that leverages metamorphic relations and prompt mutation, operating without external resources.

RAGAs Faithfulness (Es et al., 2024) evaluates the faithfulness of generated responses by decomposing them into fine-grained assertions and verifying each against the source context. The faithfulness score is computed as the ratio of supported statements.

Trulens Groundedness measures the alignment between the generated response and the source context by quantifying their informational overlap. This assessment generates a groundedness score to the value reflects the extent of overlap between the two.

LettuceDetect (Kovács and Recski, 2025) is a token-classification framework based on ModernBERT, trained on the RAGTruth dataset, which processes context-question-answer triples to detect hallucinated tokens in RAG applications.

ReDeEP (token) (Sun et al., 2025) is a token-level method that predicts hallucinations by regressing decoupled External Context Scores and Parametric Knowledge Scores, leveraging the high Pearson correlation between these scores and hallucination labels.

ReDeEP (chunk) (Sun et al., 2025) is a chunk-level method that divides retrieved context and responses into chunks, calculating chunk-level External Context Scores and Parametric Knowledge Scores, and then regressing these scores to predict hallucinations.

C Prompt Templates

The prompts used to instruct the LLM to generate responses based on triples are as follows. {question + triples} represents the request that needs to be processed by LLMs. The LLM we used to generate the corresponding response is Qwen2.5-7B-Instruct.

```
Given a question and the
   associated retrieved knowledge
    graph triples (entity,
   relation, entity), you are
   asked to answer the question
   with these triples. If the
   given knowledge triples is not
    enough or missing, you can
   use your own knowledge. After
   determining the final answer,
   please write: "So, the answer
   is {...}". Use {} to enclose
   the answer! Please think step
   by step.

Q: Find the person who said "
   Taste cannot be controlled by
   law", where did this person
   die?
Knowledge Triples: (Taste cannot
   be controlled by law.,
   media_common.quotation.author,
    Thomas Jefferson)
A: First, based on (Taste cannot
   be controlled by law.,
   media_common.quotation.author,
    Thomas Jefferson), the person
    who said "Taste cannot be
   controlled by law" is Thomas
   Jefferson. Second, no Triple
   provided can answer where
   Thomas Jefferson's dead,
   however, based on my own
   knowledge, Thomas Jefferson
   died in Charlottesville. So,
   the answer is {
   Charlottesville }.

Q: The artist nominated for The
   Long Winter lived where?
Knowledge Triples: (The Long
   Winter, book.written_work.
   author, Laura Ingalls Wilder)
(Laura Ingalls Wilder, people.
   person.places_lived, m.28e5697
```

```
   )
(m.28e5697, people.place_lived.
   location, De Smet)
A: First, based on (The Long
   Winter, book.written_work.
   author, Laura Ingalls Wilder),
    the author of The Long Winter
    is Laura Ingalls Wilder.
   Second, based on (Laura
   Ingalls Wilder, people.person.
   places_lived, m.28e5697), (m
   .28e5697, people.place_lived.
   location, De Smet), Laura
   Ingalls Wilder lived in De
   Smet. So, the answer is {De
   Smet}.

Q: Who is the coach of the team
   owned by Steve Bisciotti?
Knowledge Triples: (Steve
   Bisciotti, sports.
   professional_sports_team.
   owner_s, Baltimore Ravens)
(Steve Bisciotti, sports.
   sports_team_owner.teams_owned,
    Baltimore Ravens)
(Steve Bisciotti, organization.
   organization_founder.
   organizations_founded, Allegis
    Group)
A: First, based on (Steve
   Bisciotti, sports.
   professional_sports_team.
   owner_s, Baltimore Ravens), (
   Steve Bisciotti, sports.
   sports_team_owner.teams_owned,
    Baltimore Ravens), the team
   owned by Steve Bisciotti is
   the Baltimore Ravens. Second,
   no Triple provided can answer
   who is the coach of the
   Baltimore Ravens, however,
   based on my own knowledge, the
    coach of the Baltimore Ravens
   , is John Harbaugh. So, the
   answer is {John Harbaugh}.

Q: {question + triples}
```

D Annotation Guidelines

A response is labeled as hallucinated if it contains at least one factual statement that is not supported by, or contradicts, the retrieved knowledge graph (KG) subgraph or objective facts. Annotation focuses strictly on factual correctness, not fluency, style, or minor paraphrasing differences.

D.1 Annotation Procedure

1. For each instance, annotators examine the question, the retrieved KG subgraph, and the generated response.

2. All factual claims (entities, relations, attributes, numbers, temporal/spatial information) are extracted and verified against the retrieved subgraph.

3. If unsupported by the subgraph, claims are cross-checked against authoritative sources (e.g., Wikidata). When retrieved KG evidence conflicts with authoritative real-world facts, the response is evaluated based on objective factual correctness rather than strict consistency with the retrieved subgraph.

4. A response is marked as hallucinated if any claim is fabricated, contradictory, or unsupported; otherwise, it is marked as faithful.

D.2 Inter-Annotator Agreement and Quality Control

Two independent annotators labeled all instances. Disagreements were resolved by a third expert. Inter-annotator agreement is reported using Cohen's kappa coefficient (κ). Across all frameworks and datasets, we achieve an overall kappa score of $\mathbf{0.87}$ (substantial agreement). Breakdown per framework:

- Readi: $\kappa = 0.89$
- ToG: $\kappa = 0.85$
- StructGPT: $\kappa = 0.86$

D.3 Dataset Statistics and Exclusion Details

We manually annotated responses from three frameworks (**Readi, ToG, StructGPT**) over three test datasets (**GrailQA, WebQSP, QALD-10**). Total annotated instances: $\mathbf{7117}$. Label distribution (hallucinated / faithful):

- Readi: 616 / 1413 (hallucination rate: 30.36%)

- ToG: 964 / 1899 (hallucination rate: 33.67%)
- StructGPT: 530 / 1695 (hallucination rate: 23.82%)

D.4 Edge Cases

- Subjective opinions or vague statements are not marked as hallucinations unless factually contradictory.
- Omitted information is not hallucination, unless it changes factual meaning.
- Minor wording variations are allowed if factual content remains consistent.

E Other Main Results

Due to space limitations, the main results of the two frameworks ToG and StructGPT are shown in the following Table 5 and 6.

F Other Ablation Studies

To further investigate the contribution and complementarity of the components in LUCID, we conduct two complementary analyses: a component combination ablation study and a permutation test.

F.1 Component Combination Ablation

We evaluate selected combinations of the three components: Attention (A), Semantic Similarity (S), and Graph Structure (G). The results are reported in Table 7 across three QA frameworks and three datasets.

The complete A+S+G configuration achieves the best performance across all nine framework–dataset combinations. Removing any component leads to performance degradation, while A+G consistently provides the strongest incomplete configuration. Adding Semantic Similarity to A+G further improves performance in every setting, demonstrating the complementarity of the three signals.

F.2 Permutation Test

To further examine whether the specific information encoded in Attention and Semantic Similarity contributes to LUCID's performance, we randomly shuffle these two signals while keeping the remaining inputs unchanged. We consider three settings: Shuffled Attention, Shuffled Similarity, and Both Shuffled. The results are shown in Table 8.

The results show that shuffling attention information, semantic similarity signals, or both persistently impairs the performance of LUCID on all datasets and baseline models. All shuffled variants produce apparent performance declines compared with the original LUCID model. These observations verify that the explicit information encoded in attention distribution and semantic similarity serves as indispensable cues, demonstrating that both signals contribute complementary meaningful knowledge and are critical to the overall superiority of our method.

G Hyper-parameter Experiments

To further explore the impact of key hyper-parameters on the performance of the GINE model in hallucination detection, we conduct experiments on hyper-parameters such as the number of GINE layers, hidden layer channel size, and learning rate, under the same 9 sets of data as in Section 4.4. The specific results can be found in Figure 6, 7 and 8.

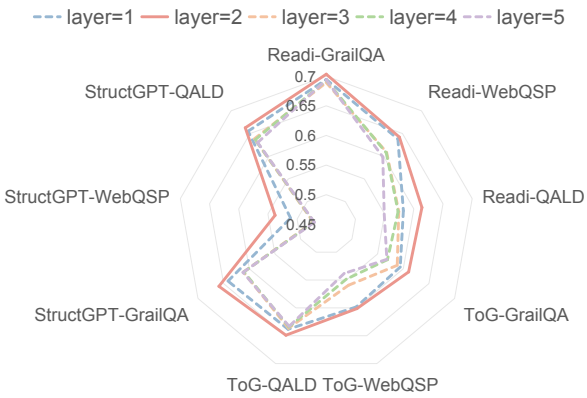

Figure 6: AVG comparison of methods on three frameworks and three datasets when using different numbers of GINE layers. The optimal result is shown as a solid line, while others are shown as dotted lines.

G.1 Does the number of layers matter?

We evaluate the model with different numbers of GINE layers (1–5). As shown in Figure 6, the model achieves the best overall performance with 2 layers. Increasing the depth beyond 2 leads to marginal gains or performance degradation, and a sharp drop is observed with 4 or 5 layers, likely due to overfitting caused by excessive depth.

G.2 Does the channel of hidden layers matter?

We evaluate the model performance with different hidden layer channel sizes (128, 256, 512, 1024).

Method	GrailQA				WebQSP				QALD-10			
	ACC	AUC	PCC	AVG	ACC	AUC	PCC	AVG	ACC	AUC	PCC	AVG
LLM	0.7018	0.6026	0.2806	0.5283	0.6637	0.6518	0.3306	0.5487	0.7089	0.6697	0.3837	0.5874
Perplexity	0.6704	0.7091	0.2368	0.5388	0.6228	0.6310	0.1940	0.4826	0.6174	0.6421	0.2500	0.5032
Energy	0.4439	0.3830	-0.0549	0.2573	0.3928	0.3845	-0.1765	0.2003	0.5370	0.5135	0.0748	0.3751
LN-Entropy	0.6480	0.6772	0.2268	0.5173	0.6020	0.6337	0.1867	0.4741	0.6431	0.6492	0.2884	0.5269
Lexical Similarity	0.6054	0.6590	0.1159	0.4601	0.5772	0.5900	0.1377	0.4350	0.5820	0.5963	0.1571	0.4451
SelfCk-Nli	0.6297	0.7166	0.3119	0.5527	0.6758	0.7143	0.3753	0.5885	0.6603	0.6964	0.3439	0.5669
SelfCk-BERTScore	0.7115	0.7596	0.3228	0.5980	0.6452	0.6818	0.3002	0.5424	0.6635	0.7246	0.3792	0.5891
Focus	0.6491	0.5476	0.1335	0.4434	0.5065	0.5166	0.0748	0.3660	0.5673	0.5615	0.1353	0.4214
EigenScore	0.5191	0.5814	0.1264	0.4090	0.5303	0.5351	0.0405	0.3686	0.5466	0.5451	0.0951	0.3956
MetaQA	0.4790	0.5803	0.0867	0.3820	0.4707	0.5397	0.0133	0.3412	0.4968	0.4969	0.0052	0.3330
RAGAs	0.6588	0.6839	0.2795	0.5407	0.6087	0.6589	0.2602	0.5093	0.6506	0.6870	0.3536	0.5637
Trulens	0.6573	0.7245	0.3113	0.5644	0.6562	0.6769	0.2826	0.5386	0.6977	0.7340	0.4146	0.6154
LettuceDetect	0.6868	0.4582	0.3239	0.4896	0.5872	0.4481	0.2792	0.4382	0.4739	0.4034	0.4121	0.4298
ReDeEP (token)	0.7191	0.7355	0.3103	0.5883	0.6699	0.7144	0.2433	0.5425	0.6731	0.6802	0.3790	0.5774
ReDeEP (chunk)	0.7061	0.7540	0.3232	0.5944	0.6367	0.7185	0.3580	0.5711	0.7019	0.7636	0.4147	0.6267
LUCID	**0.7287***	**0.7712***	**0.3316***	**0.6105***	**0.6882***	**0.7368***	**0.3790**	**0.6013***	**0.7340***	**0.7832***	**0.4302***	**0.6491***

Table 5: Main results of the hallucination detection performance of LUCID and 15 baseline methods on three datasets GrailQA, WebQSP and QALD-10 under the **ToG** framework. The evaluation metrics include ACC, AUC, PCC and the average of the three (AVG). Bold and underline fonts denote the best and second-best among all methods. * indicates that LUCID significantly outperforms the optimal baseline at the 99% significance level.

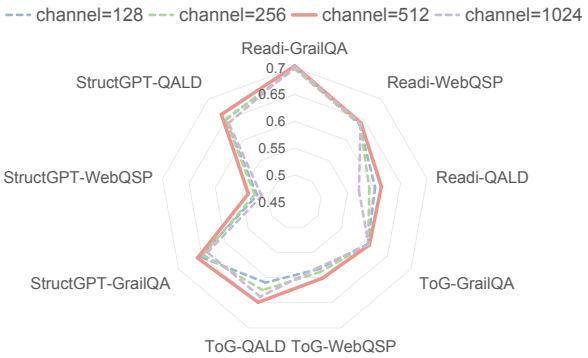

Figure 7: AVG comparison of methods on three frameworks and three datasets when using different numbers of GINE hidden channels. The optimal result is shown as a solid line, while others are shown as dotted lines.

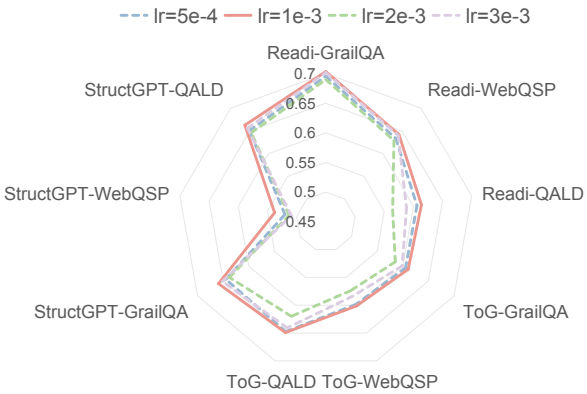

Figure 8: AVG comparison of methods on three frameworks and three datasets when using different learning rate. The optimal result is shown as a solid line, while others are shown as dotted lines.

Figure 7 shows that 512 channels yield the best performance. Smaller channel sizes limit the model's capacity to capture complex graph features, while excessively large channels degrade performance, likely due to overfitting.

G.3 Does the learning rate matter?

We test the model performance with different learning rates (5×10^{-4}, 1×10^{-3}, 2×10^{-3}, 3×10^{-3}). As shown in Figure 8, a learning rate of 1×10^{-3} performs best. Lower rates slow convergence, whereas higher rates lead to unstable optimization and reduced performance.

H Computational Overhead

To assess the computational cost of extracting attention features, we measure the end-to-end processing time and GPU memory footprint under different subgraph sizes. The average time per sample, including both LLM generation and attention extraction, is 3.45s, 4.31s, and 6.72s for subgraphs containing 5, 10, and 20 triples, respectively. The total GPU memory footprint is 13,223.64 MB on GPU0 and 16,941.66 MB on GPU1. These measurements include the memory required by the un-

Method	GrailQA				WebQSP				QALD-10			
	ACC	AUC	PCC	AVG	ACC	AUC	PCC	AVG	ACC	AUC	PCC	AVG
LLM	0.7163	0.5729	0.2263	0.5051	0.6398	**0.7263**	0.1803	0.5155	0.7433	0.5987	0.3159	0.5526
Perplexity	0.6754	0.7613	0.3921	0.6096	0.6175	0.6314	0.1851	0.4780	0.6866	0.7364	0.3805	0.6012
Energy	0.3208	0.5781	0.0419	0.3136	0.4437	0.4936	0.0070	0.3148	0.4963	0.4658	-0.0411	0.3070
LN-Entropy	0.7080	0.7626	0.4053	0.6253	0.5675	0.6554	0.2063	0.4764	0.6866	0.7204	0.3417	0.5829
Lexical Similarity	0.6579	0.7326	0.3423	0.5776	0.5939	0.6191	0.1588	0.4573	0.6903	0.6936	0.3344	0.5728
SelfCk-Nli	0.7252	0.7863	0.4365	0.6493	0.6245	0.7062	**0.2464**	0.5257	0.7201	**0.8129**	0.4218	0.6516
SelfCk-BERTScore	0.7030	0.7952	0.4477	0.6486	0.5869	0.6544	0.2275	0.4896	0.7015	0.7527	0.4106	0.6216
Focus	0.5451	0.4627	0.0155	0.3411	0.2197	0.4564	-0.0180	0.2194	0.6455	0.5432	0.1599	0.4495
EigenScore	0.5539	0.6220	0.1855	0.4538	0.5758	0.6042	0.1618	0.4473	0.6231	0.6378	0.2211	0.4940
MetaQA	0.5727	0.5764	0.1396	0.4296	0.5216	0.5691	0.0740	0.3882	0.5858	0.6293	0.2598	0.4916
RAGAs Faithfulness	0.6688	0.6989	0.3210	0.5629	0.5633	0.6253	0.2255	0.4714	0.5933	0.6584	0.2741	0.5086
Trulens Groundedness	0.6876	0.7410	0.3741	0.6009	0.6134	0.6854	0.1683	0.4890	0.6217	0.7225	0.3973	0.5805
LettuceDetect	0.5301	0.3891	0.2306	0.3833	0.3561	0.2416	0.1212	0.2396	0.6731	0.6035	0.2434	0.5067
ReDeEP (token)	0.6867	0.7527	0.3203	0.5866	0.4840	0.5262	0.1943	0.4015	0.6045	0.6463	0.3989	0.5499
ReDeEP (chunk)	**0.7419**	0.7865	0.4164	0.6483	0.5522	0.6134	0.2047	0.4568	0.7463	0.8027	0.4178	0.6556
LUCID	0.7331	**0.7958**	**0.4485**	**0.6591***	**0.6634***	0.7206	0.2292	**0.5377***	**0.7537***	0.8122	**0.4255**	**0.6638***

Table 6: Main results of the hallucination detection performance of LUCID and 15 baseline methods on three datasets GrailQA, WebQSP and QALD-10 under the **StructGPT** framework. The evaluation metrics include ACC, AUC, PCC and the average of the three (AVG). Bold and underline fonts denote the best and second-best among all methods. * indicates that LUCID significantly outperforms the optimal baseline at the 99% significance level.

			Readi			ToG			StructGPT		
A	S	G	GrailQA	WebQSP	QALD-10	GrailQA	WebQSP	QALD-10	GrailQA	WebQSP	QALD-10
✓			0.6614	0.5696	0.5831	0.5059	0.5130	0.5837	0.5820	0.4582	0.5946
	✓		0.4699	0.4638	0.4718	0.4835	0.4646	0.5476	0.4227	0.3911	0.4579
✓	✓		0.6751	0.5882	0.5883	0.4957	0.5214	0.6013	0.6034	0.5080	0.6299
✓		✓	0.6864	0.6203	0.5888	0.5660	0.5691	0.6257	0.6432	0.4659	0.6113
	✓	✓	0.4887	0.4984	0.4889	0.5096	0.5013	0.5573	0.4688	0.4395	0.4614
✓	✓	✓	**0.7033**	**0.6420**	**0.6139**	**0.6105**	**0.6013**	**0.6491**	**0.6591**	**0.5377**	**0.6638**

Table 7: Component combination ablation of LUCID. The full model consistently achieves the best performance across all framework–dataset combinations.

derlying LLM, while the additional memory introduced specifically by attention extraction is relatively modest compared with the model's overall memory requirements.

I Case Studies

To better understand how LUCID distinguishes hallucinated and non-hallucinated reasoning behaviors, we conduct a case study using GNNExplainer (Ying et al., 2019) visualizations and feature mean distributions. Figure 9 presents hallucinated cases, while Figure 10 shows non-hallucinated cases. In each figure, the left column visualizes the important subgraphs identified by GNNExplainer, where node color and edge thickness indicate the importance scores learned by the model. The right column presents the corresponding mean feature distributions of nodes, reflecting the aggregated feature activations used during prediction.

The visualization results reveal clear structural differences between hallucinated and non-hallucinated reasoning patterns.

For hallucinated cases (Figure 9), the feature mean distributions exhibit strong imbalance across the graph, where only a few nodes dominate the activation values while most surrounding nodes contribute minimally. Such uneven feature aggregation suggests that the model relies excessively on isolated semantic signals instead of integrating evidence from multiple structurally related entities. Correspondingly, the GNNExplainer results show that the model attention is distributed across multiple disconnected or weakly related graph regions. Although several nodes receive relatively

	Readi			ToG			StructGPT		
	GrailQA	**WebQSP**	**QALD-10**	**GrailQA**	**WebQSP**	**QALD-10**	**GrailQA**	**WebQSP**	**QALD-10**
Shuffled Attention	0.6829	0.6110	0.6037	0.5810	0.5727	0.6335	0.6382	0.5201	0.6483
Shuffled Similarity	0.6910	0.6284	0.6070	0.6047	0.5896	0.6432	0.6516	0.5290	0.6554
Both Shuffled	0.6740	0.5999	0.5943	0.5829	0.5615	0.6245	0.6257	0.5186	0.6410
LUCID	**0.7033**	**0.6420**	**0.6139**	**0.6105**	**0.6013**	**0.6491**	**0.6591**	**0.5377**	**0.6638**

Table 8: Permutation test for Attention and Semantic Similarity.

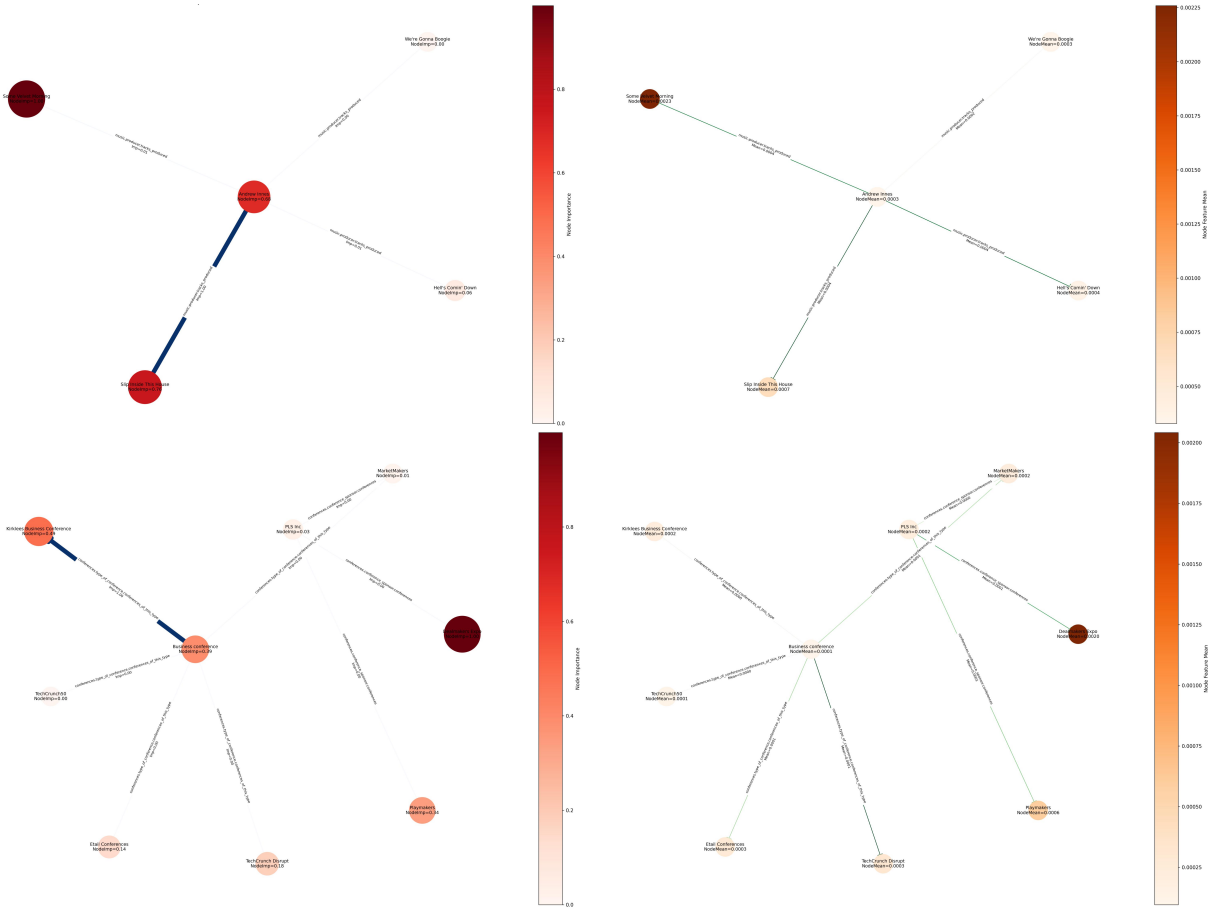

Figure 9: Visualization of hallucinated reasoning cases. The left column shows the important subgraphs identified by GNNExplainer, where darker node colors and thicker edges indicate higher importance scores. The upper and lower subfigures correspond to two different hallucinated examples. The right column presents the corresponding node feature mean distributions.

high importance scores, these important nodes are scattered and connected through sparse or low-confidence edges. This fragmented explanation pattern indicates that the reasoning process fails to establish coherent relational paths within the KG. As a result, the model tends to overemphasize locally salient entities or relations while overlooking globally consistent KG semantics, ultimately increasing the risk of hallucinated reasoning propagation.

In contrast, the non-hallucinated cases (Figure 10) exhibit noticeably smoother and more balanced feature mean distributions across connected nodes. This balanced aggregation behavior suggests that the model effectively integrates information from multiple supporting entities rather than relying on a single dominant feature source. Consistent with this observation, the GNNExplainer results reveal significantly more concentrated and coherent reasoning structures, where important nodes are organized around a compact subgraph with strong local connectivity. Instead of depending on multiple unrelated regions, the prediction mainly relies on a focused relational path or a small set of semantically

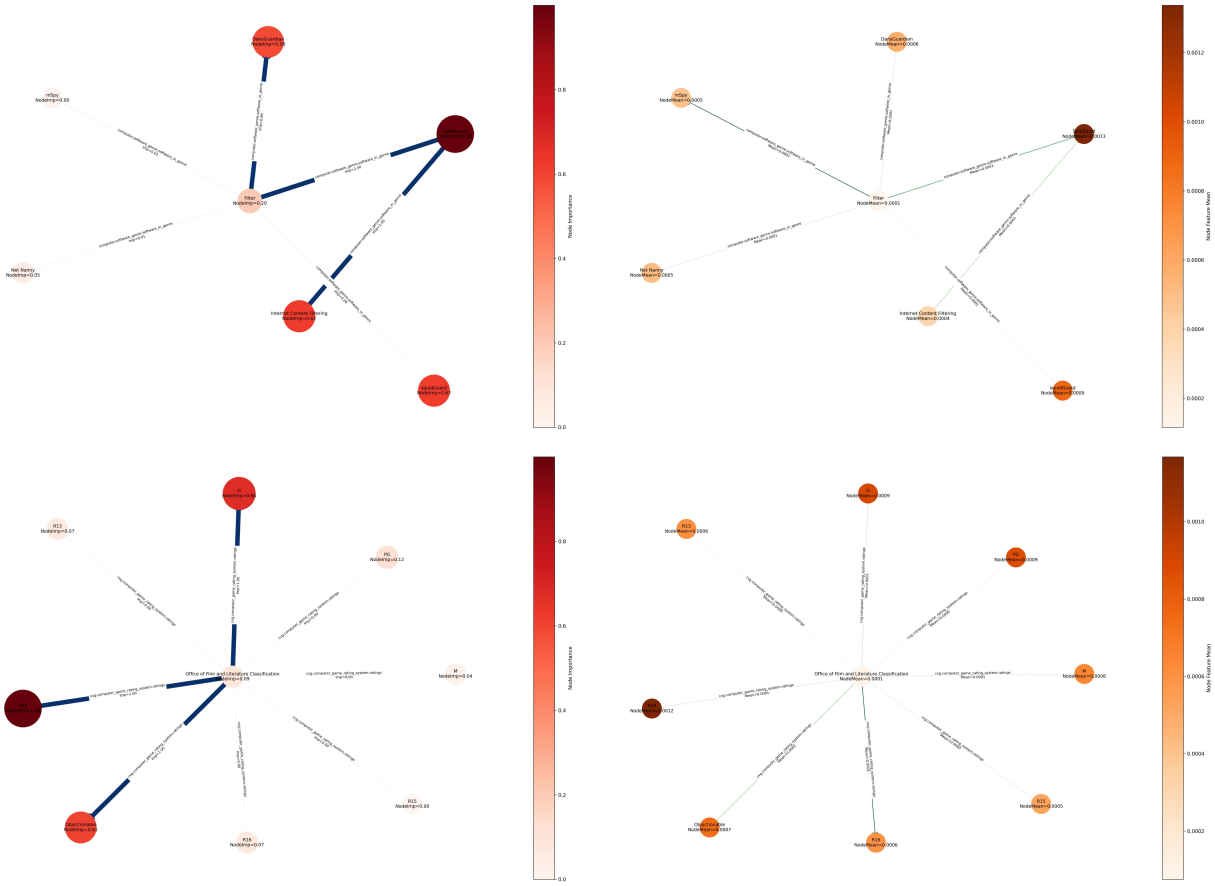

Figure 10: Visualization of non-hallucinated reasoning cases. The left column illustrates the important subgraphs extracted by GNNExplainer, while the right column shows the corresponding node feature mean distributions. The upper and lower subfigures correspond to two faithful reasoning examples.

aligned neighbors. Such structurally consistent evidence aggregation enables the model to capture coherent KG semantics and perform more reliable reasoning.

Another notable observation is that hallucinated cases often contain stronger feature polarization. Specifically, several nodes receive extremely high importance scores while many others remain nearly inactive. This phenomenon suggests unstable evidence selection during reasoning. By comparison, non-hallucinated cases demonstrate more gradual importance transitions between neighboring nodes and edges, implying more robust information propagation within the graph. These observations further validate the motivation of LUCID: reliable hallucination detection requires jointly modeling semantic relevance, structural consistency, and internal reasoning dynamics instead of relying solely on isolated confidence signals.

Overall, the case study demonstrates that hallucinations in LLM-based KG reasoning are closely associated with dispersed structural attention and imbalanced feature activations, whereas faithful reasoning tends to rely on compact and semantically coherent subgraph structures. The visualization results provide intuitive evidence that LUCID effectively captures these discriminative structural patterns for hallucination detection.

J Potential Risks

This work focuses on improving hallucination detection and reasoning reliability in large language models. We expect the proposed method to contribute to the development of more trustworthy AI systems.

All experiments are conducted using publicly available datasets, and no private or sensitive user information is involved. The current work does not include high-risk application scenarios.

As with other large language model technologies, responsible deployment and appropriate human oversight remain important considerations.