KG-Guard: Graph-Based Hallucination Detection for Knowledge Base Question Answering

Albert Sawczyn[1,†] **Piotr Bielak**[1] **Tomasz Kajdanowicz**[1]

[1]Department of Artificial Intelligence
Wrocław University of Science and Technology
Wrocław, Poland
[†]albert.sawczyn@pwr.edu.pl

Abstract

Large language models (LLMs) are increasingly used for knowledge base question answering (KBQA), also referred to as knowledge graph question answering (KGQA), where answering requires selecting entities from a question-specific knowledge graph subgraph. Yet LLMs are known to hallucinate across tasks, and KBQA is no exception: even when we provide a graph as the knowledge source, the model may rely on parametric knowledge instead of graph evidence or perform invalid reasoning over the given relations. Such hallucinated answer nodes can limit the practical deployment of KBQA systems, especially in high-stakes domains such as healthcare. We formulate hallucination detection in KBQA as an answer-node classification problem and propose a lightweight graph-based framework that treats the answering LLM as a black box. `KG-Guard` represents each KBQA instance as an augmented graph. It initializes node features with semantic representations of KG entities, marks topic entities and LLM-proposed answer nodes with learned vectors, and connects a virtual question node to the topic entities. A graph encoder then produces verification-oriented node representations, and a small MLP classifies each proposed answer node using its graph representation together with the question embedding. Experiments on WebQSP, ComplexWebQuestions, and PUGG show that our detector achieves the best average rank across all three benchmarks (F1 82.0, 87.4, and 84.3), while using $\sim 305\times$ fewer parameters than LLM-based baselines. It also retains the best average rank when jointly evaluated across an extended set of pipelines and settings. Beyond detection, the node-level feedback is actionable: when flagged answers are fed back to the KBQA system for iterative refinement, downstream KBQA F1 improves by 13.0–14.5 points and Exact Match by 16.9–17.6 points.

1 Introduction

Large language models (LLMs) are increasingly used as reasoning components in knowledge-intensive applications. One important setting is knowledge-base question answering (KBQA), also referred to as knowledge graph question answering (KGQA), where the model must return the entity nodes from a knowledge graph that answer a given question [Lan et al., 2023]. Earlier approaches queried the full graph symbolically; most LLM-based pipelines instead retrieve a question-specific subgraph and reason over it. For example, given "What is the capital of Australia?", the system should select *Canberra* from a subgraph built around the topic entity *Australia*. This graph structure provides an explicit grounding space whose entities and relations are easier to curate, update, and inspect than unstructured documents. Earlier KBQA systems often relied on graph-centric architectures, whereas

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

recent pipelines increasingly use LLMs to select answer entities from retrieved subgraphs [Ma et al., 2025, Baek et al., 2023, He et al., 2024].

LLM-based pipelines improve language understanding but also introduce hallucinations [Huang et al., 2025]. In KBQA, the LLM may rely on parametric knowledge instead of the retrieved subgraph or reason incorrectly over graph facts, leading to wrong answer nodes. To the best of our knowledge, hallucination detection for LLM-based KBQA outputs remains underexplored: prior work focuses mainly on answering KBQA or on hallucination detection in other QA settings.

Existing detectors typically treat hallucination as a text-level problem: they ignore the KBQA subgraph, do not classify individual answer nodes, or require white-box access to internal LLM signals unavailable through closed APIs. We therefore treat hallucination detection in KBQA as a graph learning problem. Returned nodes leave structured signals in the retrieved subgraph: hallucinated and factual nodes may differ in their connections to topic entities and in their question-relevant local neighborhoods.

We propose `KG-Guard`, a graph-based hallucination detector for KBQA that treats the answering LLM as a black box. Given a question, retrieved subgraph, and LLM-proposed answer nodes, `KG-Guard` builds semantic node and question representations, marks topic entities and answer nodes, adds a virtual question node connected to topic entities, and runs a lightweight graph encoder. Each returned node is classified from its graph representation and the question embedding using a small MLP. Thus, `KG-Guard` uses only the retrieved graph and LLM outputs, without internal states or activations [Chen et al., 2024, Binkowski et al., 2025], and avoids the extra LLM calls required by judge-based [Zheng et al., 2023] or sampling-based detectors [Manakul et al., 2023]. Its node-level feedback also supports iterative answer refinement, following evidence that fine-grained hallucination feedback can improve factuality correction [Sawczyn et al., 2026].

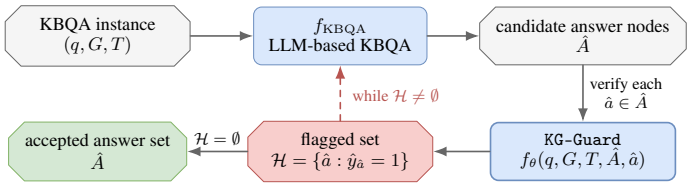

Figure 1: Role of `KG-Guard` in the KBQA loop. The LLM-based KBQA method maps (q, G, T) to candidate answer nodes $\hat{A}$. `KG-Guard` labels returned nodes and feeds flagged hallucinations $\mathcal{H}$ back for targeted refinement until $\mathcal{H} = \emptyset$ or the iteration cap is reached (see Section 4.4).

Our contributions can be summarized as follows:

- We formulate KBQA hallucination detection as answer-node classification on retrieved KG subgraphs – to our knowledge, the first dedicated method for this problem.
- We propose `KG-Guard`, a lightweight black-box graph-based detector that achieves the best average rank across all three benchmarks, using $\sim$305$\times$ fewer parameters than LLM-based baselines. It also retains the best average rank when jointly evaluated across an extended set of pipelines and settings.
- We show that node-level feedback enables iterative refinement (Fig. 1), improving KBQA F1 by 13.0–14.5 points and Exact Match by 16.9–17.6 points.
- We evaluate on WebQSP, CWQ, and PUGG against LLM-judge, sampling-based, and fine-tuned text-encoder baselines, ablate key design choices, and test robustness across answering LLMs, retrieval methods, and KBQA pipelines.
- We release all code, intermediate data, and a KBQA hallucination detection dataset. [1]

2 Related Work

Recent KBQA systems increasingly combine language models with retrieved graph evidence. KAPING augments prompts with relevant KG facts for zero-shot question answering [Baek et al., 2023].

[1] Code: github.com/graphml-lab-pwr/KG-Guard (License: CC BY-SA 4.0); Dataset: huggingface.co/collections/graphml-lab-pwr/kg-guard-datasets (License: CC BY-SA 4.0 (WebQSP and PUGG subsets); GPL v3 (CWQ subset)).

Earlier graph QA models, such as EmbedKGQA, GRAFT-Net, QA-GNN, and GreaseLM, rank candidates via KG embeddings, question-aware subgraph reasoning, or joint language–graph representations, without an LLM generator [Saxena et al., 2020, Sun et al., 2018, Yasunaga et al., 2021, Zhang et al., 2021]. A related approach, G-Retriever, targets open-ended QA over KG, applying retrieval-augmented generation (RAG) to construct query-relevant subgraphs via prize-collecting Steiner tree (PCST) optimization [He et al., 2024]. More recent agentic approaches traverse or plan over KG reasoning paths iteratively to produce grounded answers [Sun et al., 2024, Luo et al., 2024]. GNN-RAG feeds GNN-retrieved KG reasoning paths to an LLM for answer generation [Mavromatis and Karypis, 2025]. These methods use graph evidence for question answering. Our goal is orthogonal: given candidate answer nodes from an external LLM-based KBQA system, we ask whether they are hallucinated – a problem, to our knowledge, not previously studied.

Hallucination detection has largely been studied in free-text settings. Black-box sampling-based methods such as SelfCheckGPT estimate factuality by sampling multiple generations [Manakul et al., 2023]. Fine-grained methods verify outputs at the level of individual facts or claims [Min et al., 2023, Sawczyn et al., 2026]; our work shares this spirit by classifying individual answer nodes rather than whole generations. Another line of work exploits internal model signals such as hidden states [Azaria and Mitchell, 2023, Chen et al., 2024, Kossen et al., 2024, Farquhar et al., 2024] or attention maps [Chuang et al., 2024, Sriramanan et al., 2024, Binkowski et al., 2025]. These methods use internal LLM signals to detect hallucinations, but they are not designed to verify answer nodes against a retrieved KBQA subgraph. Their access requirements also limit applicability to closed models.

Several recent approaches apply the KG structure to hallucination detection. GraphEval [Sansford et al., 2024] extracts atomic claims from LLM output as KG triples and verifies each against the provided textual context via natural language inference (NLI) models; FactAlign [Rashad et al., 2024] and knowledge-centric detection [Hu et al., 2024] similarly extract triples from generated text and align them with a textual reference. GraphCheck constructs KGs from both the claim and source document, then applies GNNs as a soft prompt to an LLM verifier [Chen et al., 2025]. These approaches verify free-form generated text using external textual evidence, whereas we classify answer nodes directly on an existing structured KG.

Iterative refinement is a general strategy for correcting LLM outputs [Madaan et al., 2023, Dhuliawala et al., 2024, Sawczyn et al., 2026]. KGR uses direct triple lookup in a KG to guide revision [Guan et al., 2024]. Our correction procedure identifies hallucinated answer nodes via a trained graph-encoder classifier rather than triple-level conflict resolution.

Message-passing GNNs compute node representations by repeatedly aggregating information from local neighborhoods, as in GCN, GraphSAGE, and GIN [Kipf and Welling, 2017, Hamilton et al., 2017, Xu et al., 2019], with GINE extending GIN to incorporate edge features [Hu et al., 2020]. For our task, attention-based graph encoders are especially natural because only part of the retrieved graph may be relevant. Graph Attention Networks (GAT) learn neighbor-specific attention weights [Veličković et al., 2018], while GraphTransformer extends this with multi-head dot-product attention for expressive message passing [Dwivedi and Bresson, 2021, Shi et al., 2021].

3 Method

3.1 Problem Formulation

We consider a KBQA task in which each example is associated with a natural-language question and a question-specific subgraph extracted from a knowledge graph. Formally, each instance is represented as (q, G, T, A^*), where q is the question and $G = (V, E)$ is the retrieved subgraph. Here, V is the set of nodes and each edge $e = (u, r, v) \in E$ corresponds to a directed KG triple from node u to node v with relation label $r \in \mathcal{R}$, where $\mathcal{R}$ denotes the set of relation labels. We treat these relations as edge attributes rather than as a fixed heterogeneous graph schema, which allows relation labels to be encoded from their text. The set $T \subseteq V$ contains the topic entities, i.e., nodes mentioned in the question. Let V_{KG} denote the full KG entity set; $A^* \subseteq V_{\mathrm{KG}}$ contains the gold answer nodes, which need not all occur in the retrieved node set V.

Given (q, G, T), an LLM-based KBQA method f_{KBQA} is applied to answer the question:

$$\hat{A} = f_{\mathrm{KBQA}}(q, G, T), \qquad \hat{A} \subseteq V.$$

If the method abstains with a special *unknown* response, we set $\hat{A} = \emptyset$. In this work, we only consider examples for which the method returns at least one node, i.e., $\hat{A} \neq \emptyset$. This restriction matches the goal of our detector, which is to verify concrete answer nodes rather than abstentions.

The hallucination detection task is then formulated as a binary classification problem over individual returned nodes. If the LLM returns multiple nodes for a single question, we decompose this output into separate detection instances, one per predicted node. Thus, for each retained KBQA example $(q, G, T, A^*, \hat{A})$ and each $\hat{a} \in \hat{A}$, we create one classification instance $(q, G, T, \hat{A}, A^*, \hat{a})$ and define the target label $y \in \{0, 1\}$ as

$$y = \begin{cases} 0, & \text{if } \hat{a} \in A^*, \\ 1, & \text{otherwise,} \end{cases}$$

where $y = 1$ denotes a hallucinated answer node and $y = 0$ denotes a factual one. This definition naturally handles questions with multiple valid answers: a returned node is treated as factual if it matches any gold answer node, and hallucinated otherwise. Consequently, a single KBQA example may yield multiple hallucination-detection instances, corresponding to the different nodes proposed by the LLM. In a deployed system, however, the detector can process the graph once (single forward pass) and mark which of the returned nodes are hallucinated.

Our goal is to learn a detector $f_\theta(q, G, T, \hat{A}, \hat{a}) \in \{0, 1\}$ that predicts whether an individual node selected by f_{KBQA} is factual or hallucinated, where θ collectively denotes all trainable parameters of the detector.

3.2 `KG-Guard` (Hallucination Detector)

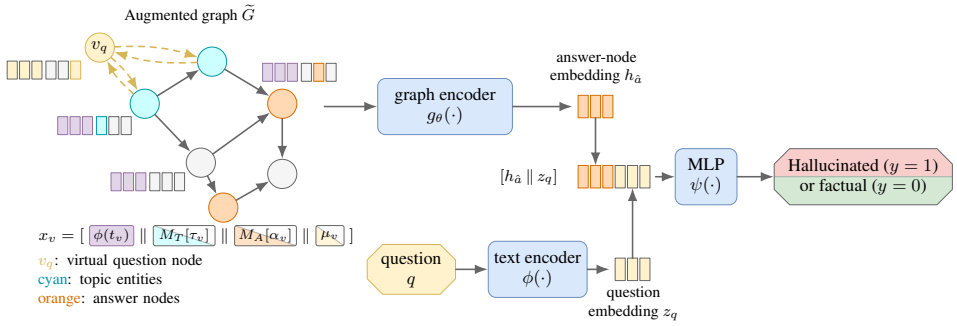

Figure 2: `KG-Guard` architecture for classifying answer nodes. Node features combine semantic node representations with topic-entity marks M_T and answer-node marks M_A. A virtual question node v_q is connected to the topic entities with edges in both directions. The graph encoder g_θ computes answer-node representations $h_{\hat{a}}$, which are concatenated with the question embedding z_q and passed to an MLP to predict whether the returned node is hallucinated ($y = 1$) or factual ($y = 0$).

Our method (see Figure 2) operates on top of the answer returned by the LLM-based KBQA method for a given KBQA instance (assuming $\hat{A} \neq \emptyset$).

We start from the retrieved subgraph and construct node features by combining semantic node information with task-specific marks. Knowledge-graph nodes typically do not come with informative numeric features, so we initialize them from text. Let t_v denote the textual representation of node v, such as its label or name, and let $\phi(\cdot)$ be a semantic text encoder. For each KG edge (u, r, v), the same encoder maps the relation label t_r to the edge feature $\phi(t_r)$.

We then augment these semantic features with trainable mark embeddings. Let $\tau_v = \mathbb{I}[v \in T]$ indicate whether node v is a topic entity, and let $\alpha_v = \mathbb{I}[v \in \hat{A}]$ indicate whether it was returned by the LLM-based KBQA method. We use two trainable lookup tables, $M_T \in \mathbb{R}^{2 \times d_T}$ and $M_A \in \mathbb{R}^{2 \times d_A}$, for topic-entity and answer-node marks, respectively. Each table contains one embedding for the positive case and one embedding for the negative case.

The initial feature vector of node v is then:

$$x_v = [\phi(t_v) \,\|\, M_T[\tau_v] \,\|\, M_A[\alpha_v] \,\|\, \mu_v]\,,$$

where $\|$ denotes concatenation and $\mu_v = \mathbb{I}[v = v_q]$ flags the virtual question node introduced below. Thus, each node representation contains four parts: a semantic representation of the node text, a learnable mark indicating whether the node is a topic entity, a learnable mark indicating whether the node was proposed as an answer, and a flag for the virtual question node. We use the same text encoder to obtain the question embedding $z_q = \phi(q)$.

To condition message passing on the question, we augment the graph with a virtual question node v_q, built from the same feature template x_v whose semantic part is z_q instead of $\phi(t_v)$. This node is connected only to the topic entities T, in both directions, rather than to every node in the retrieved subgraph. These edges have no relation label, so each direction uses a learnable edge embedding. Let $\widetilde{G}$ and $\widetilde{X}$ denote the resulting augmented graph and its node features.

The marked and augmented graph is then processed by a graph encoder g_θ to obtain node-level representations $H = g_\theta(\widetilde{G}, \widetilde{X})$, where $H = \{h_v\}_{v \in V}$. We instantiate g_θ as a GraphTransformer, GAT, or GINE, with the GraphTransformer used as the main configuration in our experiments. For a particular returned answer node $\hat{a}$, we extract its final representation $h_{\hat{a}}$ and concatenate it with the question embedding z_q. The resulting vector is passed through an MLP classifier, with parameters $\psi \subset \theta$, to obtain a scalar logit:

$$s_{\hat{a}} = \mathrm{MLP}_\psi\left([h_{\hat{a}} \,\|\, z_q]\right),$$

where $s_{\hat{a}}$ is a scalar logit. The final hallucination score is obtained as

$$\hat{p}(y = 1 \mid q, G, T, \hat{A}, \hat{a}) = \sigma(s_{\hat{a}}),$$

with $y = 1$ denoting a hallucinated answer node. Thus, the detector combines graph structure around the returned node with the semantics of the question, while remaining independent of the internal activations of the answering LLM.

4 Experimental Setup

The extended implementation details are provided in Appendix A.

4.1 Evaluation Data

Training and evaluation of our detector require tuples $(q, G, T, A^*, \hat{A})$ containing a question, a retrieved subgraph, topic entities, gold answer nodes, and LLM-predicted answer nodes. The KBQA benchmarks used here do not directly provide these complete tuples. Therefore, we construct the evaluation data by running a full KBQA pipeline on three established benchmarks, while recording both gold and predicted answer nodes as detection instances. In practical deployment, directly annotating the correctness of proposed answer nodes is another way to obtain detector labels.

4.1.1 KBQA Benchmarks

We evaluate on three KBQA benchmarks that vary in question complexity, underlying knowledge graph, and language. We inherit each benchmark's official question-level splits, assigning every answer-node instance from a question to the same split to prevent leakage between training and evaluation.

WebQuestionsSP (WebQSP) [Yih et al., 2016] (license: CC BY 4.0) is a standard factoid KBQA benchmark grounded in the Freebase knowledge graph [Bollacker et al., 2007]. Questions are naturally phrased real-world queries collected from Google Suggest [Berant et al., 2013]. Because the validation split provided with this dataset is small (246 questions), we supplement it with questions sampled from the training set to reach a validation set size of 500 questions.

ComplexWebQuestions (CWQ) [Talmor and Berant, 2018] (license: GPL v2+) extends WebQSP by programmatically appending compositional constraints to WebQSP SPARQL queries, then paraphrasing the resulting questions via crowdworkers. This procedure produces questions requiring logical reasoning, making CWQ substantially harder than WebQSP. It is also grounded in Freebase and is the largest of our three benchmarks.

PUGG [Sawczyn et al., 2024] (license: CC BY-SA 4.0) is a Polish KBQA dataset grounded in the Wikidata knowledge graph [Vrandečić and Krötzsch, 2014], combining naturally collected (Google Suggest) and template-based (SPARQL-paired templates and paraphrasing) questions. We use PUGG to evaluate KBQA on a non-English language and a different KG. Because Wikidata hub nodes inflate subgraph sizes, we exclude nodes connected to more than 1000 other entities as a preprocessing step before subgraph retrieval.

4.1.2 Hallucination Dataset Construction

Table 1: Dataset statistics for hallucination detection. Each KBQA question may yield multiple proposed answer nodes. #Answers counts these nodes plus abstention entries. %Hallu., %Correct, and %Abstained use this combined count as their denominator. Abstention entries are excluded from node classification.

Dataset	Split	#Questions	#Answers	%Hallu.	%Correct	%Abstained
WebQSP	train	2826	9179	50.0	44.5	5.5
	val	246	807	54.0	41.0	5.0
	test	1628	4842	46.2	47.9	5.8
CWQ	train	27639	43795	47.6	28.3	24.2
	val	3519	6001	55.1	24.1	20.8
	test	3531	5840	55.1	21.7	23.1
PUGG	train	3589	6372	63.6	24.3	12.1
	val	700	1078	58.6	28.9	12.4
	test	1081	1884	62.9	24.9	12.2

Subgraph retrieval Both KBQA and hallucination detection require a question-relevant subgraph from the KG; we apply the same retrieval procedure for both tasks. For all three datasets, we follow the G-Retriever pipeline [He et al., 2024]: candidate neighbors are first collected by BFS expansion from topic entities, and a Prize-Collecting Steiner Tree (PCST) formulation is then solved to select a compact, question-relevant subgraph. We use the same PCST hyperparameters as in the original G-Retriever pipeline. Subgraph statistics are reported in Appendix E.1 (Table 18).

As an alternative retrieval strategy, we also evaluate top-K=100 embedding retrieval to assess the robustness of `KG-Guard` to the choice of subgraph construction method; details and results are in Appendix D.1.2.

KBQA method We use KAPING [Baek et al., 2023] as the LLM-based KBQA method. Beyond the answer node, KAPING is extended to also return a textual reasoning summary and a list of reasoning triples (the KG triples cited as evidence). Our approach does not use these elements, and they are used only as additional context for the LLM-based baselines (Section 4.2). The answering LLM is constrained to return only nodes present in the retrieved subgraph via structured generation; it may also abstain with an *unknown* response. The full prompt template is provided in the code repository. As discussed in Section 3.1, abstaining examples are excluded from the hallucination-detection task. The statistics, including the fraction of unknown responses, are reported in Table 1. Additional statistics of KBQA answers are reported in Appendices E.2 and E.3.

In addition to KAPING, we evaluate `KG-Guard` on the G-Retriever pipeline [He et al., 2024], which uses the same PCST subgraph retrieval and a fine-tuned LLM to select answer nodes from the subgraph, to assess robustness to the choice of KBQA pipeline. Since its answering LLM returns free text rather than structured node identifiers, we parse G-Retriever's output into a set of answer nodes using GPT-5.4-nano as an extraction LLM. Results are reported in Appendix D.1.3. Hallucination labels are specific to the answering pipeline, generator, and retriever, so we train and evaluate a separate `KG-Guard` model for each configuration rather than reusing a detector in a zero-shot manner.

4.2 Baselines

We compare against four groups of baselines covering trivial predictors, a fine-tuned text encoder, LLM-based detection, and sampling-based consistency analysis. All baselines operate without access

to the answering model's internals, mirroring the common real-world deployment setting where the model is accessed as a service.

Trivial baselines *Random* assigns the hallucinated label uniformly at random; *Most Frequent* always predicts the majority class in the validation split. These provide reference points and illustrate the effect of class imbalance on each metric.

Fine-tuned text encoder We fine-tune RoBERTa-large [Liu et al., 2019] separately on WebQSP and CWQ, and polish-RoBERTa-large-v2 [Dadas et al., 2020] on PUGG. Unlike the prompted LLM judges, it is a task-specific discriminative detector trained on the same per-answer-node labels as `KG-Guard`. Each example concatenates the question, the predicted answer, and a subgraph serialized as a triple list. Unlike `KG-Guard`, it treats the retrieved graph as a flat text sequence rather than performing message passing over its nodes and edges.

Llama-4-Scout We use Llama-4-Scout [AI, 2025] as an LLM-as-judge baseline [Zheng et al., 2023, Janiak et al., 2025]. We evaluate three prompt variants supplying progressively richer context: **(1)** *Graph* (G): question, subgraph serialized as a triple list, and the predicted answers to verify; **(2)** *Graph + Reasoning Summary* (G+RS): the above plus the answering LLM's reasoning summary; **(3)** *Graph + Reasoning Summary + Reasoning Triples* (G+RS+RT): the above plus the answering LLM's reasoning triples. While straightforward to deploy, this approach is computationally expensive: Llama-4-Scout has 109B total parameters (17B active in its MoE architecture), requiring substantial GPU memory per detection call.

GPT-5-mini We evaluate GPT-5-mini [Singh et al., 2026] as a judge using two configurations: G and G+RS+RT. Both outperformed G+RS on average with Llama-4-Scout. Due to commercial API costs, we restrict evaluation to these two configurations.

SelfCheckGPT We adapt the sampling-based method of Manakul et al. [2023] to structured KBQA outputs. We draw N independent predictions from the LLM-based KBQA method with temperature 1.0. We then compute the fraction of predictions that do *not* include a given node as the hallucination score. We evaluate with $N \in \{5, 10\}$ to assess sensitivity to sample count. This baseline represents a widely used black-box hallucination detection method and is the most expensive computationally, requiring N forward passes of the LLM-based KBQA method per example.

4.3 Evaluation

For hallucination detection, we report F1 as the primary metric, computed over predicted answer nodes only; Precision, Recall, Accuracy, and AUC-PR are reported in Appendix D.1.4. Trainable detectors are trained with 3 random seeds; we report the mean and standard deviation across seeds. We also report Random over 3 seeds. For methods that rely on LLM calls, we report single-run results due to the high cost of inference.

4.4 Hallucination Correction

While our primary focus is hallucination detection, we also examine its potential for answer correction. `KG-Guard` can guide answer revision within a larger system (Fig. 1). We investigate whether feeding `KG-Guard` predictions back to the LLM enables it to revise its own hallucinated responses.

Starting from the initial KAPING answers, we iterate the following steps: **(1)** `KG-Guard` scores each predicted answer node. **(2)** Examples with no nodes flagged as hallucinated are marked resolved, and their answers remain unchanged. **(3)** For each active example (i.e., with at least one flagged answer node), the LLM is re-prompted in a chat-style conversation consisting of a system instruction, the original question and subgraph, the LLM's most recent response, and a follow-up message listing the flagged answer node names. Steps 1–3 repeat until all examples are resolved or the iteration cap (max. 5) is reached. We use Llama-4-Scout as the refinement LLM.

We report downstream KBQA metrics (F1, Precision, Recall, and Exact Match) at the initial step and after refinement on all three datasets, together with the distribution of refinement iterations.

5 Results

Ablation studies on graph encoder components, input modalities, virtual question-node connectivity, and graph encoder architecture, together with training on KBQA-derived labels and calibration, are presented in Appendix B.

5.1 Hallucination Detection

Hallucination detection results with Llama-4-Scout as the answering LLM are summarized in Table 2. Full per-metric results are in Appendix D.1.4.

`KG-Guard` (GraphTransformer) has the best average rank (1.7) and achieves the highest F1 on WebQSP (82.0). On CWQ, it reaches 87.4, 0.3 points below the fine-tuned RoBERTa baseline; on PUGG, it reaches 84.3, 1.2 points below SelfCheckGPT (N=10, 85.5). SelfCheckGPT (N=10) is the strongest baseline on average, but requires 10 generations from the answering LLM per example, which is extremely expensive, versus a single feed-forward pass for `KG-Guard`.

Table 2: Hallucination detection F1 comparing `KG-Guard` against baselines, with Llama-4-Scout as the KAPING answering LLM. Bold and underlined values indicate the best and second-best result per column, respectively. Avg. Rank is the average rank of each method across the three datasets (lower is better). Values with ± denote mean ± std over 3 runs. G, RS, and RT denote the retrieved subgraph, reasoning summary, and reasoning triples provided to the LLM judge, respectively.

Method	WebQSP	CWQ	PUGG	Avg. Rank
`KG-Guard` (GraphTransformer)	**82.0 ± 0.7**	<u>87.4 ± 0.2</u>	<u>84.3 ± 1.0</u>	1.7
`KG-Guard` (GINE)	<u>81.2 ± 0.3</u>	87.1 ± 0.3	83.1 ± 1.0	3.3
`KG-Guard` (GAT)	79.8 ± 1.2	86.3 ± 0.7	82.7 ± 2.7	4.3
SelfCheckGPT (N=10)	61.2	84.0	**85.5**	4.3
Most Frequent	65.9	83.5	83.5	5.0
RoBERTa (fine-tuned)	63.7 ± 5.2	**87.7 ± 0.7**	76.7 ± 2.6	5.0
SelfCheckGPT (N=5)	58.9	82.3	83.9	6.7
GPT-5-mini (G+RS+RT)	59.2	84.2	81.2	6.7
GPT-5-mini (G)	55.9	82.6	82.6	8.0
Llama-4-Scout (G+RS)	36.5	61.3	61.3	11.3
Llama-4-Scout (G)	36.7	60.2	68.8	11.3
Llama-4-Scout (G+RS+RT)	41.0	61.2	60.8	11.3
Random	49.6 ± 1.0	58.3 ± 0.5	58.8 ± 1.8	12.0

The other leading baselines are dataset-dependent: RoBERTa narrowly leads on CWQ but trails `KG-Guard` by 18.3 F1 points on WebQSP. Despite this dataset-specific lead, RoBERTa's average rank across the three datasets is only 5.0, placing it fifth overall (tied with the trivial Most Frequent baseline), which indicates that its performance varies substantially across datasets. Similarly, SelfCheckGPT (N=10) leads on PUGG but trails by 20.8 points on WebQSP and 3.4 points on CWQ. GPT-5-mini is the strongest LLM-judge baseline on PUGG but trails `KG-Guard` by 22.8 points on WebQSP. In contrast, `KG-Guard` ranks first on WebQSP and is within 0.3–1.2 F1 points of the leader on CWQ and PUGG, consistently placing among the top methods across fine-tuned text-encoder, sampling-based, and LLM-judge baselines. This highlights the advantage of graph-based representation.

LLM-based judges are less competitive overall. On WebQSP, all Llama-4-Scout variants fall below the Random baseline (F1 36.5–41.0 vs. 49.6), and GPT-5-mini reaches 59.2, compared with 82.0 for `KG-Guard`. GPT-5-mini is stronger on CWQ and PUGG but remains below `KG-Guard` on all three datasets. Across LLM judges, richer context does not consistently improve detection. On CWQ and PUGG, over 71% of answers are hallucinated (Table 1), which inflates F1 for trivial strategies: Most Frequent reaches 83.5 F1 on both datasets, but its AUC-PR equals the class frequency (71.7 on CWQ, 71.6 on PUGG), compared to 94.2 and 90.0 for `KG-Guard` (Tables 15, 16).

WebQSP consists of naturally formulated user queries, whereas CWQ questions are derived programmatically from logical reasoning templates , and a portion of PUGG questions share a similar structure. We hypothesize that judge-based methods benefit from the regular reasoning patterns inherent to template-derived questions; naturally phrased queries may provide weaker structural

regularities. Other differences, including class balance and retrieval coverage, may also contribute. KG-Guard performs well across all three datasets regardless of this distinction.

In the main comparison, SelfCheckGPT improves with sample count (N=10 over N=5) and outperforms all Llama-4-Scout judge variants, but requires the most LLM calls per example.

LLM judges are strongly precision-skewed (GPT-5-mini: 85.3 precision at 45.3 recall on WebQSP), missing many hallucinated nodes. Contrastingly, KG-Guard is more balanced, and the F1 gap is driven primarily by higher recall. Moreover, KG-Guard's precision–recall trade-off can be adjusted via the classification threshold, an option unavailable to LLM judges (Appendix D.1.4).

Results with Gemma 4 E4B and GPT OSS 20B as the KAPING answering LLM are reported in Appendix D.1.1 (Table 11). Consistent with Llama-4-Scout results, KG-Guard ranks first on 5 of 6 dataset–LLM combinations; on the remaining one, the best KG-Guard variant trails the best baseline by 0.5 F1. Results under top-K retrieval are in Appendix D.1.2 (Table 12): all three KG-Guard variants occupy the top three positions by average rank. Results on the G-Retriever pipeline are reported in Appendix D.1.3 (Table 13): KG-Guard (GINE) and KG-Guard (GraphTransformer) tie for the second-best average rank (3.0), behind only expensive GPT-5-mini. Averaged over the main benchmark, top-K retrieval, additional answering LLMs, and G-Retriever, KG-Guard (GINE) has the best average rank (2.4).[2]

5.2 Hallucination Correction

KBQA performance before and after hallucination-aware iterative refinement is summarized in Table 3 and full per-dataset results are in Appendix D.2. We observe consistent gains across all three datasets: mean F1 improves by +13.8 points and Exact Match by +17.4 points on average. Exact Match requires the predicted answer set to match the gold set exactly, so any missing or spurious node counts as a failure. Its increase shows that refinement produces fully correct answer sets for more evaluated questions. These results demonstrate that KG-Guard's node-level feedback can support targeted answer correction.

Table 3: KBQA answer quality before (Initial) and after (Refined) iterative refinement guided by KG-Guard, measured by F1 and Exact Match (EM). Δ is the score improvement.

	WebQSP		**CWQ**		**PUGG**	
	F1	**EM**	**F1**	**EM**	**F1**	**EM**
Initial	59.5	42.8	57.2	51.1	53.1	45.3
Refined	72.5	60.4	71.7	68.8	67.2	62.3
Δ	+13.0	+17.6	+14.5	+17.6	+14.0	+16.9

The metrics per refinement step are shown in Figure 3. The largest gains occur after the first refinement step, after which the improvement curve flattens. The distribution of iterations per example (Figure 4) reveals how many corrections each example needed. Step 0 denotes examples in which KG-Guard flags no hallucinated answer node at the initial step. Among examples resolved within the iteration cap, the majority resolve at the first refinement iteration, and the count drops quickly at later steps. A substantial fraction of examples reach the iteration cap without converging, indicating examples for which the detector continues to flag at least one answer. Here, *resolved* means that no answer is flagged. Among examples satisfying this stopping criterion within the cap, most stop after one refinement iteration. Practitioners can consider a single refinement step to reduce inference costs; the quality trade-off is shown in Figure 3.

6 Conclusion

We formulated hallucination detection in KBQA as answer-node classification on retrieved KG subgraphs and proposed KG-Guard, a lightweight black-box graph-based detector. Our method achieves the best average rank (1.7) across all three benchmarks, ranking first on WebQSP and within 0.3–1.2 F1 points of the best on CWQ and PUGG. The second-best method on average, SelfCheckGPT (N=10), is a sampling-based baseline that requires 10 LLM calls per example. In

[2]Each method's own rank, averaged with equal weight over Tables 2, 12, 11, and 13.

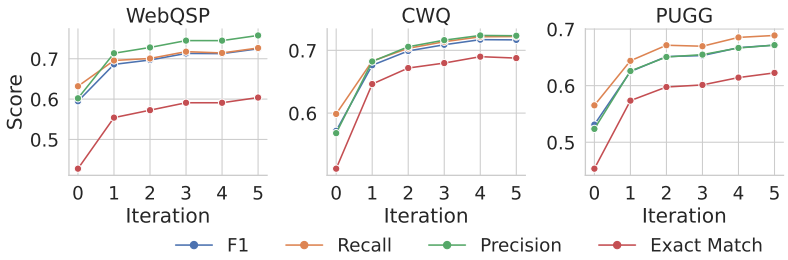

Figure 3: KBQA metrics per refinement step across three datasets. Step 0 is the initial unrefined prediction; the largest gains occur at step 1, after which the curve flattens.

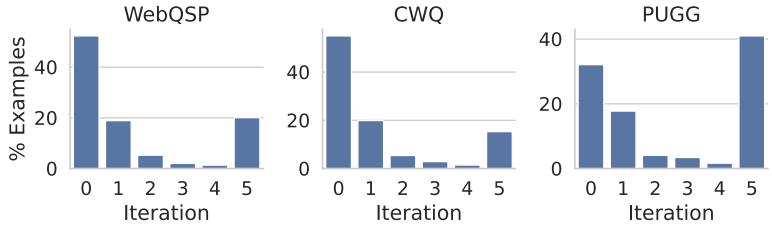

Figure 4: Distribution of refinement iterations per example across three datasets. Step 0 denotes examples in which `KG-Guard` flags no hallucinated answer node before any refinement; Step 5 denotes examples that reached the iteration cap.

contrast, `KG-Guard` needs only a single feed-forward pass and uses $\sim$305$\times$ fewer parameters than LLM-based baselines, requiring zero LLM calls at detection time. The detector generalizes across question types and languages.

Beyond detection, the node-level output of `KG-Guard` is actionable: feeding flagged answer nodes back to the KBQA LLM as targeted correction signals yields consistent downstream KBQA gains across all three datasets, with most of the improvement captured in a single refinement step. Across separately trained configurations using different answering LLMs, retrieval strategies, and KBQA pipelines, `KG-Guard` (GINE) achieves the best average rank (2.4). Together, these results establish explicit graph-based structural verification as a practical and efficient path toward more reliable LLM-based KBQA systems.

Limitations and Future Work

`KG-Guard` requires training examples with hallucination labels. These can be derived from any KBQA dataset by comparing answer nodes against gold answers or by labeling hallucinations directly. Because we perform node-level classification on the retrieved subgraph, it cannot handle cases where the LLM abstains with an *unknown* prediction. Such responses are not represented as nodes in the graph. However, defining what counts as a hallucinated abstention is non-trivial: LLMs are generally expected to abstain when the answer is genuinely uncertain. If a gold answer is never retrieved into the subgraph, the refinement loop (Section 4.4) cannot recover it either, since it only re-queries the same subgraph; extending it with subgraph expansion or re-retrieval could address this and is left for future work. The answering LLM also produces a free-form reasoning summary and supporting triples, used by the LLM-as-judge baselines. We experimented with marking reasoning triples and incorporating the summary embedding, but neither approach yielded consistent gains. We hypothesize that the reasoning signal is already recoverable from the subgraph. Whether more sophisticated fusion methods could exploit these signals remains open for future work.

Broader impacts `KG-Guard` aims to improve reliability of LLM-based KBQA systems in high-stakes domains. As any ML system can produce incorrect predictions, it should augment rather than replace human factual verification.

Acknowledgments and Disclosure of Funding

The research was supported by the Department of Artificial Intelligence and the Wrocław Centre for Networking and Supercomputing at Wrocław University of Science and Technology. AS was funded by the National Science Centre, Poland, under CHIST-ERA Open & Re-usable Research Data & Software (grant number 2022/04/Y/ST6/00183). TK was supported by the AITAX project under grant FENG.02.02-IP.05-0314/23, Action 2.2 FIRST TEAM, European Funds for a Modern Economy Programme 2021–2027 (FENG).

References

Meta AI. Introducing LLaMA 4: Advancing multimodal intelligence. Technical report, 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.68. URL https://aclanthology.org/2023.findings-emnlp.68/.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In Bhavana Dalvi Mishra, Greg Durrett, Peter Jansen, Danilo Neves Ribeiro, and Jason Wei, editors, *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 78–106, Toronto, Canada, June 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.nlrse-1.7. URL https://aclanthology.org/2023.nlrse-1.7/.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1160/.

Jakub Binkowski, Denis Janiak, Albert Sawczyn, Bogdan Gabrys, and Tomasz Jan Kajdanowicz. Hallucination detection in LLMs using spectral features of attention maps. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24354–24385, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1239. URL https://aclanthology.org/2025.emnlp-main.1239/.

Kurt Bollacker, Robert Cook, and Patrick Tufts. Freebase: a shared database of structured general human knowledge. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, page 1962–1963. AAAI Press, 2007. ISBN 9781577353232.

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, 2024. URL https://openreview.net/forum?id=Zj12nzlQbz.

Yingjian Chen, Haoran Liu, Yinhong Liu, Jinxiang Xie, Rui Yang, Han Yuan, Yanran Fu, Peng Yuan Zhou, Qingyu Chen, James Caverlee, and Irene Li. GraphCheck: Breaking long-term text barriers with extracted knowledge graph-powered fact-checking. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14976–14995, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.729. URL https://aclanthology.org/2025.acl-long.729/.

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 Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings*

of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 1419–1436, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.84. URL https://aclanthology.org/2024.emnlp-main.84/.

Sławomir Dadas, Michał Perełkiewicz, and Rafał Poświata. Pre-training polish transformer-based language models at scale. In *International Conference on Artificial Intelligence and Soft Computing*, pages 301–314. Springer, 2020.

Slawomir Dadas, Michał Perełkiewicz, and Rafał Poświata. PIRB: A comprehensive benchmark of Polish dense and hybrid text retrieval methods. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 12761–12774, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.1117/.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.212. URL https://aclanthology.org/2024.findings-acl.212/.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. In *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

William Falcon and The PyTorch Lightning team. PyTorch Lightning, March 2019. URL https://github.com/Lightning-AI/lightning.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, Jun 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07421-0. URL https://doi.org/10.1038/s41586-024-07421-0.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Xinyan Guan, Yanjiang Liu, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i16.29770. URL https://doi.org/10.1609/aaai.v38i16.29770.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 132876–132907. Curran Associates, Inc., 2024. doi: 10.52202/079017-4224. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/efaf1c9726648c8ba363a5c927440529-Paper-Conference.pdf.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH.

Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Guo, Tianhang Zhang, Yang Xu, Yun Luo, Pengfei Liu, Yue Zhang, and Zheng Zhang. Knowledge-centric hallucination detection. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical*

Methods in Natural Language Processing, pages 6953–6975, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.395. URL https://aclanthology.org/2024.emnlp-main.395/.

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 Trans. Inf. Syst.*, 43 (2), January 2025. ISSN 1046-8188. doi: 10.1145/3703155. URL https://doi.org/10.1145/3703155.

Denis Janiak, Jakub Binkowski, Albert Sawczyn, Bogdan Gabrys, Ravid Shwartz-Ziv, and Tomasz Jan Kajdanowicz. The illusion of progress: Re-evaluating hallucination detection in LLMs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 34728–34745, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1761. URL https://aclanthology.org/2025.emnlp-main.1761/.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms, 2024. URL https://arxiv.org/abs/2406.15927.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11196–11215, 2023. doi: 10.1109/TKDE.2022.3223858.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, 2024.

Chuangtao Ma, Yongrui Chen, Tianxing Wu, Arijit Khan, and Haofen Wang. Large language models meet knowledge graphs for question answering: Synthesis and opportunities. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24578–24597, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1249. URL https://aclanthology.org/2025.emnlp-main.1249/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: iterative refinement with self-feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

Costas Mavromatis and George Karypis. GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In Wanxiang Che, Joyce Nabende, Ekaterina

Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16682–16699, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.856. URL https://aclanthology.org/2025.findings-acl.856/.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.741. URL https://aclanthology.org/2023.emnlp-main.741/.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Mohamed Rashad, Ahmed Zahran, Abanoub Amin, Amr Abdelaal, and Mohamed Altantawy. FactAlign: Fact-level hallucination detection and classification through knowledge graph alignment. In Anaelia Ovalle, Kai-Wei Chang, Yang Trista Cao, Ninareh Mehrabi, Jieyu Zhao, Aram Galstyan, Jwala Dhamala, Anoop Kumar, and Rahul Gupta, editors, *Proceedings of the 4th Workshop on Trustworthy Natural Language Processing (TrustNLP 2024)*, pages 79–84, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.trustnlp-1.8. URL https://aclanthology.org/2024.trustnlp-1.8/.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Hannah Sansford, Nicholas Richardson, Hermina Petric Maretic, and Juba Nait Saada. Grapheval: A knowledge-graph based LLM hallucination evaluation framework. In Manas Gaur, Efthymia Tsamoura, Edward Raff, Nikhita Vedula, and Srinivasan Parthasarathy, editors, *Proceedings of the Workshop on Knowledge-infused Learning co-located with 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Barcelona, Spain, August 26, 2024*, CEUR Workshop Proceedings, pages 20–31. CEUR-WS.org, 2024. URL https://ceur-ws.org/Vol-3894/paper5.pdf.

Albert Sawczyn, Katsiaryna Viarenich, Konrad Wojtasik, Aleksandra Domogała, Marcin Oleksy, Maciej Piasecki, and Tomasz Kajdanowicz. Developing PUGG for Polish: A modern approach to KBQA, MRC, and IR dataset construction. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar,

editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10978–10996, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.652. URL https://aclanthology.org/2024.findings-acl.652/.

Albert Sawczyn, Jakub Binkowski, Denis Janiak, Bogdan Gabrys, and Tomasz Jan Kajdanowicz. FactSelfCheck: Fact-level black-box hallucination detection for LLMs. In Vera Demberg, Kentaro Inui, and Lluís Marquez, editors, *Findings of the Association for Computational Linguistics: EACL 2026*, pages 5603–5621, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi: 10.18653/v1/2026.findings-eacl.296. URL https://aclanthology.org/2026.findings-eacl.296/.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.412. URL https://aclanthology.org/2020.acl-main.412/.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1548–1554. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/214. URL https://doi.org/10.24963/ijcai.2021/214. Main Track.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, Alex Makelov, Alex Neitz, Alex Wei, Alexandra Barr, Alexandre Kirchmeyer, Alexey Ivanov, Alexi Christakis, Alistair Gillespie, Allison Tam, Ally Bennett, Alvin Wan, Alyssa Huang, Amy McDonald Sandjideh, Amy Yang, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrei Gheorghe, Andres Garcia Garcia, Andrew Braunstein, Andrew Liu, Andrew Schmidt, Andrey Mereskin, Andrey Mishchenko, Andy Applebaum, Andy Rogerson, Ann Rajan, Annie Wei, Anoop Kotha, Anubha Srivastava, Anushree Agrawal, Arun Vijayvergiya, Ashley Tyra, Ashvin Nair, Avi Nayak, Ben Eggers, Bessie Ji, Beth Hoover, Bill Chen, Blair Chen, Boaz Barak, Borys Minaiev, Botao Hao, Bowen Baker, Brad Lightcap, Brandon McKinzie, Brandon Wang, Brendan Quinn, Brian Fioca, Brian Hsu, Brian Yang, Brian Yu, Brian Zhang, Brittany Brenner, Callie Riggins Zetino, Cameron Raymond, Camillo Lugaresi, Carolina Paz, Cary Hudson, Cedric Whitney, Chak Li, Charles Chen, Charlotte Cole, Chelsea Voss, Chen Ding, Chen Shen, Chengdu Huang, Chris Colby, Chris Hallacy, Chris Koch, Chris Lu, Christina Kaplan, Christina Kim, CJ Minott-Henriques, Cliff Frey, Cody Yu, Coley Czarnecki, Colin Reid, Colin Wei, Cory Decareaux, Cristina Scheau, Cyril Zhang, Cyrus Forbes, Da Tang, Dakota Goldberg, Dan Roberts, Dana Palmie, Daniel Kappler, Daniel Levine, Daniel Wright, Dave Leo, David Lin, David Robinson, Declan Grabb, Derek Chen, Derek Lim, Derek Salama, Dibya Bhattacharjee, Dimitris Tsipras, Dinghua Li, Dingli Yu, DJ Strouse, Drew Williams, Dylan Hunn, Ed Bayes, Edwin Arbus, Ekin Akyurek, Elaine Ya Le, Elana Widmann, Eli Yani, Elizabeth Proehl, Enis Sert, Enoch Cheung, Eri Schwartz, Eric Han, Eric Jiang, Eric Mitchell, Eric Sigler, Eric Wallace, Erik Ritter, Erin Kavanaugh, Evan Mays, Evgenii Nikishin, Fangyuan Li, Felipe Petroski Such, Filipe de Avila Belbute Peres, Filippo Raso, Florent Bekerman, Foivos Tsimpourlas, Fotis Chantzis, Francis Song, Francis Zhang, Gaby Raila, Garrett McGrath, Gary Briggs, Gary Yang, Giambattista Parascandolo, Gildas Chabot, Grace Kim, Grace Zhao, Gregory Valiant, Guillaume Leclerc, Hadi Salman, Hanson Wang, Hao Sheng, Haoming Jiang, Haoyu Wang, Haozhun Jin, Harshit Sikchi, Heather Schmidt, Henry Aspegren, Honglin Chen, Huida Qiu, Hunter Lightman, Ian Covert, Ian Kivlichan, Ian Silber, Ian Sohl, Ibrahim Hammoud, Ignasi Clavera, Ikai Lan, Ilge Akkaya, Ilya Kostrikov, Irina Kofman, Isak Etinger, Ishaan Singal, Jackie Hehir, Jacob Huh, Jacqueline Pan, Jake Wilczynski, Jakub Pachocki, James Lee, James Quinn, Jamie Kiros, Janvi Kalra, Jasmyn Samaroo, Jason Wang, Jason Wolfe, Jay Chen, Jay Wang, Jean Harb, Jeffrey Han, Jeffrey Wang, Jennifer Zhao, Jeremy Chen, Jerene Yang, Jerry Tworek, Jesse Chand, Jessica Landon, Jessica Liang, Ji Lin, Jiancheng Liu, Jianfeng Wang, Jie Tang, Jihan Yin, Joanne Jang, Joel Morris, Joey Flynn, Johannes Ferstad, Johannes Heidecke, John Fishbein, John Hallman, Jonah Grant, Jonathan Chien, Jonathan Gordon, Jongsoo Park, Jordan

Liss, Jos Kraaijeveld, Joseph Guay, Joseph Mo, Josh Lawson, Josh McGrath, Joshua Vendrow, Joy Jiao, Julian Lee, Julie Steele, Julie Wang, Junhua Mao, Kai Chen, Kai Hayashi, Kai Xiao, Kamyar Salahi, Kan Wu, Karan Sekhri, Karan Sharma, Karan Singhal, Karen Li, Kenny Nguyen, Keren Gu-Lemberg, Kevin King, Kevin Liu, Kevin Stone, Kevin Yu, Kristen Ying, Kristian Georgiev, Kristie Lim, Kushal Tirumala, Kyle Miller, Lama Ahmad, Larry Lv, Laura Clare, Laurance Fauconnet, Lauren Itow, Lauren Yang, Laurentia Romaniuk, Leah Anise, Lee Byron, Leher Pathak, Leon Maksin, Leyan Lo, Leyton Ho, Li Jing, Liang Wu, Liang Xiong, Lien Mamitsuka, Lin Yang, Lindsay McCallum, Lindsey Held, Liz Bourgeois, Logan Engstrom, Lorenz Kuhn, Louis Feuvrier, Lu Zhang, Lucas Switzer, Lukas Kondraciuk, Lukasz Kaiser, Manas Joglekar, Mandeep Singh, Mandip Shah, Manuka Stratta, Marcus Williams, Mark Chen, Mark Sun, Marselus Cayton, Martin Li, Marvin Zhang, Marwan Aljubeh, Matt Nichols, Matthew Haines, Max Schwarzer, Mayank Gupta, Meghan Shah, Melody Y. Guan, Melody Huang, Meng Dong, Mengqing Wang, Mia Glaese, Micah Carroll, Michael Lampe, Michael Malek, Michael Sharman, Michael Zhang, Michele Wang, Michelle Pokrass, Mihai Florian, Mikhail Pavlov, Miles Wang, Ming Chen, Mingxuan Wang, Minnia Feng, Mo Bavarian, Molly Lin, Moose Abdool, Mostafa Rohaninejad, Nacho Soto, Natalie Staudacher, Natan LaFontaine, Nathan Marwell, Nelson Liu, Nick Preston, Nick Turley, Nicklas Ansman, Nicole Blades, Nikil Pancha, Nikita Mikhaylin, Niko Felix, Nikunj Handa, Nishant Rai, Nitish Keskar, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Oona Gleeson, Pamela Mishkin, Patryk Lesiewicz, Paul Baltescu, Pavel Belov, Peter Zhokhov, Philip Pronin, Phillip Guo, Phoebe Thacker, Qi Liu, Qiming Yuan, Qinghua Liu, Rachel Dias, Rachel Puckett, Rahul Arora, Ravi Teja Mullapudi, Raz Gaon, Reah Miyara, Rennie Song, Rishabh Aggarwal, RJ Marsan, Robel Yemiru, Robert Xiong, Rohan Kshirsagar, Rohan Nuttall, Roman Tsiupa, Ronen Eldan, Rose Wang, Roshan James, Roy Ziv, Rui Shu, Ruslan Nigmatullin, Saachi Jain, Saam Talaie, Sam Altman, Sam Arnesen, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Sarah Yoo, Savannah Heon, Scott Ethersmith, Sean Grove, Sean Taylor, Sebastien Bubeck, Sever Banesiu, Shaokyi Amdo, Shengjia Zhao, Sherwin Wu, Shibani Santurkar, Shiyu Zhao, Shraman Ray Chaudhuri, Shreyas Krishnaswamy, Shuaiqi, Xia, Shuyang Cheng, Shyamal Anadkat, Simón Posada Fishman, Simon Tobin, Siyuan Fu, Somay Jain, Song Mei, Sonya Egoian, Spencer Kim, Spug Golden, SQ Mah, Steph Lin, Stephen Imm, Steve Sharpe, Steve Yadlowsky, Sulman Choudhry, Sungwon Eum, Suvansh Sanjeev, Tabarak Khan, Tal Stramer, Tao Wang, Tao Xin, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Degry, Thomas Shadwell, Tianfu Fu, Tianshi Gao, Timur Garipov, Tina Sriskandarajah, Toki Sherbakov, Tomek Korbak, Tomer Kaftan, Tomo Hiratsuka, Tongzhou Wang, Tony Song, Tony Zhao, Troy Peterson, Val Kharitonov, Victoria Chernova, Vineet Kosaraju, Vishal Kuo, Vitchyr Pong, Vivek Verma, Vlad Petrov, Wanning Jiang, Weixing Zhang, Wenda Zhou, Wenlei Xie, Wenting Zhan, Wes McCabe, Will DePue, Will Ellsworth, Wulfie Bain, Wyatt Thompson, Xiangning Chen, Xiangyu Qi, Xin Xiang, Xinwei Shi, Yann Dubois, Yaodong Yu, Yara Khakbaz, Yifan Wu, Yilei Qian, Yin Tat Lee, Yinbo Chen, Yizhen Zhang, Yizhong Xiong, Yonglong Tian, Young Cha, Yu Bai, Yu Yang, Yuan Yuan, Yuanzhi Li, Yufeng Zhang, Yuguang Yang, Yujia Jin, Yun Jiang, Yunyun Wang, Yushi Wang, Yutian Liu, Zach Stubenvoll, Zehao Dou, Zheng Wu, and Zhigang Wang. Openai gpt-5 system card. Technical report, 2026. URL https://arxiv.org/abs/2601.03267.

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 A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 34188–34216. Curran Associates, Inc., 2024. doi: 10.52202/079017-1077. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/3c1e1fdf305195cd620c118aaa9717ad-Paper-Conference.pdf.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. Open domain question answering using early fusion of knowledge bases and text. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1455. URL https://aclanthology.org/D18-1455/.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large

language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=nnVO1PvbTv.

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *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, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1059. URL https://aclanthology.org/N18-1059/.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux, Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco

Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, September 2014. ISSN 0001-0782. doi: 10.1145/2629489. URL http://doi.acm.org/10.1145/2629489.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.45. URL https://aclanthology.org/2021.naacl-main.45/.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2033. URL https://aclanthology.org/P16-2033/.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. Greaselm: Graph reasoning enhanced language models. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2201.08860.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Appendix

This appendix provides supplementary material organized as follows:

Appendix A: Implementation details.

Appendix B: Ablation studies on graph encoder components, input modalities, virtual question-node connectivity, and graph encoder architecture, the effect of training with KBQA-derived labels, and classifier calibration.

Appendix C: Model size comparison and computation demands across methods.

Appendix D: Hallucination detection results with additional answering LLMs (Gemma 4 E4B, GPT OSS 20B), top-K retrieval, the G-Retriever pipeline, full detection metrics, and KBQA correction metrics including Exact Match.

Appendix E: Dataset and subgraph statistics, including subgraph structure, answer count distributions, and LLM abstention and hallucination patterns.

A Implementation Details

LLM settings For KAPING, the answering LLM is one of Llama-4-Scout [AI, 2025], Gemma 4 E4B [Team et al., 2026], or GPT OSS 20B [OpenAI et al., 2025], each at temperature 0.1 to favor the most probable outputs while allowing retries on output-parsing errors. Llama-4-Scout is the primary answering LLM in all main-paper experiments; Gemma 4 E4B and GPT OSS 20B are used as additional answering LLMs in Appendix D.1.1. Llama-4-Scout is also used as the LLM-as-judge baseline and for answer refinement. SelfCheckGPT draws its N samples from the answering LLM of that setting, at temperature 1.0. GPT-5-mini is queried with temperature 1 and default reasoning effort (`medium`); temperature 1 is the only value permitted by the API when reasoning is enabled.

`KG-Guard` **training** Node, edge, and question embeddings are produced by a frozen Sentence Transformer [Reimers and Gurevych, 2019]: `all-roberta-large-v1` [Reimers and Gurevych, 2019] for WebQSP and CWQ, and multilingual `mmlw-retrieval-roberta-large` [Dadas et al., 2024] for PUGG (in both cases $d = 1024$). The same encoders are used for top-K embedding retrieval. Node features are obtained by encoding the node label, and edge features by encoding the relation label. Dimensions of marking embeddings for topic entities and answer nodes are set to $d_T = d_A = 20$, plus a 1-dimensional virtual-question-node indicator, giving node features of size 1065. The learnable embeddings for edges between the virtual question node and the topic entities are sized to 1024. The graph encoder is instantiated as a GraphTransformer [Dwivedi and Bresson, 2021, Shi et al., 2021], a GAT [Veličković et al., 2018], or GINE [Hu et al., 2020]. All variants use 2 layers and 256-dimensional hidden representations; GraphTransformer and GAT additionally use 8 attention heads. We train with Adam (lr=10^{-3}, weight decay 10^{-4}), batch size 32, for up to 300 epochs with early stopping (patience 50), minimizing cross-entropy loss over the factual/hallucinated logits. Checkpoint selection and early stopping both monitor the validation F1 score. All hyperparameter values were chosen empirically. The implementation uses PyTorch Geometric [Fey and Lenssen, 2019] and PyTorch Lightning [Falcon and The PyTorch Lightning team, 2019].

Compute resources LLM inference (Llama-4-Scout, Gemma 4 E4B, and GPT OSS 20B) was performed using vLLM on 4 $\times$ H100 GPUs. `KG-Guard` training was conducted on an NVIDIA A40 GPU (see Appendix C for details).

B Additional Experiments

B.1 Effect of Graph Encoder Components

To justify the proposed graph-encoder design, we ablate node marking and the virtual question node independently and in combination. Table 4 presents the results. Removing node marking (topic entity and answer node trainable embeddings) reduces F1 by 0.8 points on average, while removing the virtual question node reduces it by 0.5 points; their combined removal reduces F1 by 2.2 points,

Table 4: Ablation of graph encoder components on hallucination detection F1. Δ is the F1 change in percentage points relative to the full KG-Guard model; negative values indicate performance loss from removing the component. Bold and underlined values indicate the best and second-best F1 per dataset, respectively. Mean $\pm$ std over 3 runs.

Variant	WebQSP	Δ	CWQ	Δ	PUGG	Δ	Avg. Δ
full	**82.0 ± 0.7**		<u>87.4 ± 0.2</u>		**84.3 ± 1.0**		+0.0
w/o marking	80.1 ± 0.9	-1.9	**87.9 ± 0.3**	+0.5	83.4 ± 1.3	-0.9	-0.8
w/o virtual q. node	<u>81.6 ± 0.2</u>	-0.5	86.6 ± 1.0	-0.8	<u>83.9 ± 0.4</u>	-0.4	-0.5
w/o marking & virtual q. node	77.8 ± 0.9	-4.2	86.5 ± 0.5	-0.9	82.7 ± 0.8	-1.6	-2.2

exceeding the sum of the individual effects, indicating the two are mutually reinforcing. CWQ is notably more robust across variants, whereas WebQSP shows the largest sensitivity to ablation.

B.2 Effect of Input Modalities

Table 5: Ablation of input modalities on hallucination detection F1. Δ is the F1 change in percentage points relative to the full KG-Guard model (graph + question text); negative values indicate performance loss from removing the modality. The reference row is shown in bold. RS is the reasoning summary produced by the answering LLM and was not used in KG-Guard. Mean $\pm$ std over 3 runs.

Variant	WebQSP	Δ	CWQ	Δ	PUGG	Δ	Avg. Δ
full (graph + text)	**82.1 ± 0.4**		**87.6 ± 0.5**		**83.7 ± 0.6**		+0.0
graph only	80.6 ± 0.6	-1.5	86.3 ± 0.1	-1.3	83.1 ± 0.6	-0.6	-1.1
question only	70.8 ± 2.3	-11.4	83.0 ± 0.5	-4.6	81.7 ± 0.3	-2.0	-6.0
question+RS	70.6 ± 2.0	-11.5	83.3 ± 0.6	-4.3	81.2 ± 0.3	-2.5	-6.1

We investigate how much each input modality contributes to detection and whether the LLM's reasoning summary (RS) (not used by KG-Guard) can substitute for graph structure. Table 5 reports the results. Ablating text features (graph only) reduces F1 by 1.1 points on average; removing the graph encoder (question only) reduces F1 by 6.0 points. These ablations support the contribution of the graph component. Adding the reasoning summary to the text input (question+RS) yields no improvement over question text alone (-6.1 vs. -6.0 points). Thus, adding the reasoning summary does not improve this text-only variant in our experiments. WebQSP is the most sensitive dataset for this ablation.

B.3 Virtual Question-Node Connectivity

Table 6: Virtual question-node connectivity ablation on hallucination detection F1. Bold and underlined values indicate the best and second-best result per column, respectively. Mean $\pm$ std over 3 runs.

Variant	WebQSP	CWQ	PUGG
Topic entities	<u>82.0 ± 0.7</u>	87.4 ± 0.2	84.3 ± 1.0
Answer nodes	**82.1 ± 0.6**	<u>87.5 ± 0.2</u>	<u>84.4 ± 0.6</u>
All nodes	81.4 ± 1.3	**88.4 ± 0.2**	**84.8 ± 0.5**

To inject question semantics, KG-Guard connects the virtual question node to topic entities only. We compare this design against connecting the virtual question node to all retrieved nodes or to the proposed answer nodes, using the GraphTransformer encoder. Table 6 presents the results. The answer-node and topic-entity variants have similar F1 scores. Connecting to all nodes improves CWQ by 1.0 F1 point and PUGG by 0.5 points but reduces WebQSP by 0.6 points. Connecting to all retrieved nodes also adds $2|V|$ directed edges instead of $2|T|$, raising compute and memory cost on heavy-tailed graphs such as PUGG's. Topic-entity and answer-node connections yield similar F1 scores; we use topic entities throughout KG-Guard.

B.4 Graph Encoder Architecture

Table 7: Graph encoder architecture ablation on hallucination detection F1. Bold and underlined values indicate the best and second-best result per column, respectively. Avg. Rank is the average rank of each variant across the three datasets (lower is better). Mean ± std over 3 runs.

Variant	WebQSP	CWQ	PUGG	Avg. Rank
GraphTransformer	**82.0 ± 0.7**	**87.4 ± 0.2**	**84.3 ± 1.0**	1.0
GINE	<u>81.2 ± 0.3</u>	<u>87.1 ± 0.3</u>	83.1 ± 1.0	2.7
GraphSAGE	79.2 ± 0.7	86.5 ± 0.2	83.6 ± 0.5	3.3
GAT	79.8 ± 1.2	86.3 ± 0.7	82.7 ± 2.7	4.0
GCN	75.2 ± 1.3	84.9 ± 0.7	<u>83.7 ± 0.7</u>	4.0

We compare the GraphTransformer encoder used in `KG-Guard` against four alternatives that vary along two axes: GAT, like GraphTransformer, aggregates via attention, while GINE, GraphSAGE, and GCN aggregate without attention; and GAT and GINE, like GraphTransformer, condition message passing on edge features, while GraphSAGE and GCN do not. Table 7 presents the results. GraphTransformer achieves the best F1 on all three datasets, followed on average by GINE, GraphSAGE, and GAT, with GCN weakest overall. Notably, GCN is the second-best variant on PUGG despite lacking edge features, and GraphSAGE and GCN together degrade only moderately rather than collapsing. This suggests that node topology and node-level marking already carry most of the discriminative signal. However, these architectures also differ in other ways beyond their use of edge features, so this comparison does not isolate edge features' individual contribution. GraphTransformer's consistent lead across datasets motivates its use as the main graph encoder in `KG-Guard`.

B.5 Training with KBQA-Derived Labels

A natural alternative to our approach is to train the same model to predict correct answer nodes using KBQA gold labels, then flag LLM-proposed answers as hallucinated whenever the model judges them to be incorrect. Since the KBQA datasets are considerably larger than our hallucination detection dataset, we subsample each KBQA split to match its size and class balance. We repeat the procedure with three random seeds for each training run (Table 9).

Table 8: Hallucination detection when training `KG-Guard` on hallucination labels (Hall. Det.) versus KBQA-derived labels (KBQA). Bold marks the better value in each metric. Mean ± std over 3 runs.

Dataset	Train. Data	F1	Precision	Recall	Accuracy	AUC-PR
WebQSP	Hall. Det.	**82.0 ± 0.7**	**84.9 ± 2.2**	81.3 ± 2.9	**85.9 ± 0.1**	**91.4 ± 0.1**
	KBQA	81.4 ± 0.9	81.6 ± 1.9	**83.2 ± 1.0**	85.1 ± 0.8	90.6 ± 0.2
CWQ	Hall. Det.	**87.4 ± 0.2**	85.8 ± 0.2	**90.2 ± 0.5**	**83.5 ± 0.2**	**94.2 ± 0.3**
	KBQA	81.7 ± 0.7	**88.1 ± 0.2**	77.7 ± 1.0	78.8 ± 0.5	92.5 ± 0.1
PUGG	Hall. Det.	**84.3 ± 1.0**	78.1 ± 3.4	**93.5 ± 4.8**	**78.8 ± 1.9**	**90.0 ± 0.6**
	KBQA	71.3 ± 5.7	**84.1 ± 2.0**	65.3 ± 9.3	69.8 ± 3.7	85.4 ± 0.2

Table 8 shows our method outperforms the KBQA-supervised model on all three datasets despite identical training set sizes, indicating that hallucination-specific labels provide a stronger training signal than KBQA gold labels for this task.

Annotating proposed answers may reduce the scope of annotation: an annotator judges a candidate answer rather than identifying every correct answer in the KG. However, this can still require domain expertise and access to evidence beyond the retrieved subgraph.

B.6 Calibration

Expected Calibration Error (ECE) is the average gap between the classifier's confidence and the fraction of correct predictions. With Llama-4-Scout as the answering LLM, pooling the 3 seeds, test ECE of `KG-Guard` on confidence in 10 bins is 0.045 on PUGG and 0.023 on CWQ, despite over 71% hallucinated answers among non-abstaining nodes (Table 1), and 0.084 on the more

Table 9: Training set statistics for hallucination-specific (Hall. Det.) and KBQA-derived supervision (KBQA-original: original split; KBQA-sampled: subsampled to match Hall. Det. size for fair comparison). #Instances: total labeled nodes; #Pos./#Neg.: hallucinated/correct instances; %Pos.: class imbalance; #Graphs: unique subgraphs. Mean $\pm$ std over sampling runs.

Dataset	Variant	#Instances	#Pos.	#Neg.	%Pos.	#Graphs
WebQSP	Hall. Det.	8677	4592	4085	52.9	2332
	KBQA-original	43768	38115	5653	87.1	2826
	KBQA-sampled	8677	4592	4085	52.9	2387$\pm$8.7
CWQ	Hall. Det.	33206	20833	12373	62.7	17223
	KBQA-original	456141	438020	18121	96.0	27631
	KBQA-sampled	33206	20833	12373	62.7	16154$\pm$46.9
PUGG	Hall. Det.	5467	3916	1551	71.6	2777
	KBQA-original	350208	348000	2208	99.4	3589
	KBQA-sampled	5467	3916	1551	71.6	1683$\pm$5.9

balanced WebQSP. Figure 5 shows bin accuracy tracking the diagonal on PUGG and CWQ, with mild overconfidence on WebQSP. No post-hoc calibration is applied. The decision threshold can be tuned on the validation split to match the precision–recall trade-off required in deployment.

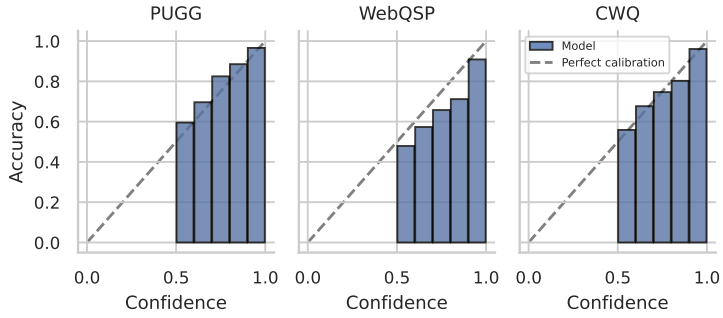

Figure 5: Reliability diagrams of `KG-Guard` on the test splits (10 bins, 3 seeds pooled). The dashed line denotes perfect calibration.

C Model Size and Computation Demands

Table 10 compares model sizes and query-time LLM call counts across methods. `KG-Guard` (Graph-Transformer) adds only 2.52M trainable parameters on top of a frozen sentence encoder ($\sim$358M total for WebQSP/CWQ and $\sim$437M for PUGG). The fine-tuned RoBERTa baseline is comparable in total size ($\sim$355M), but all 355M parameters are updated during training, requiring full back-propagation through the encoder for each target dataset. `KG-Guard` keeps this encoder frozen and trains only the graph-encoder and MLP parameters. WebQSP and CWQ reuse the same frozen English encoder; PUGG uses a separate frozen multilingual encoder. LLM-based baselines are far larger: Llama-4-Scout operates at 109B parameters—305$\times$ larger than our full inference stack using the English-language encoder ($\sim$249$\times$ using Polish-language encoder); at `fp16` precision this corresponds to $\sim$0.7 GB for `KG-Guard` with the English encoder ($\sim$0.87 GB with the Polish-language encoder) versus $\sim$218 GB for Llama-4-Scout, exceeding the capacity of a single high-end GPU. Each LLM call requires autoregressive generation: while the final answer spans only several tokens, *thinking* enabled in GPT-5-mini can consume hundreds of additional tokens per query. Self-CheckGPT further multiplies this cost by querying the answering LLM N=5 or N=10 times per query. `KG-Guard` makes zero LLM calls at detection time: classification is a single feed-forward pass through the graph encoder and a small MLP.

In practical deployment over a fixed knowledge graph, node and relation embeddings can be pre-computed and stored offline. During inference, only the question requires encoding, after which

one graph-encoder forward pass and a batched MLP evaluation produce hallucination scores for the proposed answer nodes. This lightweight design makes KG-Guard deployable on commodity hardware, with no multi-GPU infrastructure or external API access required.

Table 10: Model size and per-query LLM calls at detection time. Encoder: frozen Sentence-Transformer. Trainable: task-specific trainable parameters (KG-Guard: graph encoder + MLP; RoBERTa: full model). Total: Encoder + Trainable. $\times$ ours: total parameter ratio relative to KG-Guard. Llama-4-Scout has 109B total, but 17B active parameters (MoE); SelfCheckGPT also uses Llama-4-Scout but in a sampling-based paradigm (N calls per query). GPT-5-mini parameter count is undisclosed.

Method	Encoder	LLM calls	Trainable	Total	$\times$ ours
KG-Guard (GraphTransformer)	all-roberta-large-v1	0	2.52M	358M	—
KG-Guard (GraphTransformer)	mmlw-retrieval-roberta-large	0	2.52M	437M	—
KG-Guard (GINE)	all-roberta-large-v1	0	2.47M	358M	—
KG-Guard (GINE)	mmlw-retrieval-roberta-large	0	2.47M	437M	—
KG-Guard (GAT)	all-roberta-large-v1	0	1.51M	357M	—
KG-Guard (GAT)	mmlw-retrieval-roberta-large	0	1.51M	436M	—
RoBERTa (fine-tuned)	—	0	355M	355M	$\sim 1\times$
GPT-5-mini	—	1	—	n/a	n/a
SelfCheckGPT (N=10)	—	10	—	109B	$305\times$
SelfCheckGPT (N=5)	—	5	—	109B	$305\times$
Llama-4-Scout	—	1	—	109B	$305\times$

D Additional Results

D.1 Hallucination Detection

D.1.1 Additional Answering LLMs

Table 11: Hallucination detection F1 with Gemma 4 E4B and GPT OSS 20B as the KAPING answering LLM, sorted by average rank. Bold and underlined values indicate the best and second-best result per column, respectively. Avg. Rank is the average rank across all six dataset–LLM combinations (lower is better). Values with $\pm$ denote mean $\pm$ std over 3 runs. G, RS, and RT denote the retrieved subgraph, reasoning summary, and reasoning triples provided to the LLM judge.

Method	WebQSP		CWQ		PUGG		Avg. Rank
	Gemma 4 E4B	GPT OSS 20B	Gemma 4 E4B	GPT OSS 20B	Gemma 4 E4B	GPT OSS 20B	
KG-Guard (GINE)	**74.6 ± 1.6**	**75.8 ± 0.6**	**77.8 ± 1.0**	**79.2 ± 1.3**	73.8 ± 0.4	**81.0 ± 0.6**	1.5
KG-Guard (GraphTransformer)	73.1 ± 0.9	73.9 ± 0.4	77.1 ± 0.1	79.1 ± 0.3	74.2 ± 0.5	78.7 ± 0.5	2.8
KG-Guard (GAT)	73.1 ± 1.1	74.1 ± 0.7	76.9 ± 0.9	77.3 ± 0.5	73.6 ± 1.2	78.9 ± 0.3	3.2
SelfCheckGPT (N=10)	34.4	63.9	67.5	74.6	**74.7**	79.3	4.7
SelfCheckGPT (N=5)	58.4	65.3	61.8	74.7	71.9	77.3	5.3
RoBERTa (fine-tuned)	58.1 ± 1.8	59.4 ± 0.7	76.9 ± 0.8	78.3 ± 0.4	70.2 ± 2.1	76.0 ± 0.2	5.7
GPT-5-mini (G+RS+RT)	57.9	53.2	61.9	65.3	73.3	76.5	6.7
GPT-5-mini (G)	45.9	45.5	57.3	66.3	59.0	73.9	8.7
Most Frequent	0.0	0.0	0.0	71.0	74.6	76.2	9.2
Llama-4-Scout (G)	31.1	48.9	39.5	56.7	29.7	59.6	9.8
Random	43.5 ± 1.2	47.3 ± 0.5	53.0 ± 1.2	51.3 ± 0.6	53.3 ± 1.3	54.9 ± 0.2	10.0
Llama-4-Scout (G+RS+RT)	28.4	45.9	21.4	47.2	23.5	59.2	11.3
Llama-4-Scout (G+RS)	23.9	42.1	21.1	48.0	22.7	58.6	12.2

Table 11 reports hallucination detection results when KAPING is driven by Gemma 4 E4B or GPT OSS 20B instead of Llama-4-Scout. Consistent with Llama-4-Scout results, KG-Guard ranks first on 5 of 6 dataset–LLM combinations. On the remaining one (PUGG / Gemma 4 E4B), it trails SelfCheckGPT (N=10) by 0.5 F1. All three KG-Guard variants occupy the top three positions by average rank, demonstrating robustness to the choice of answering LLM. Different answering LLMs produce qualitatively different error profiles, including shifts in class balance across datasets. These shifts change the difficulty profile for trivial predictors yet leave KG-Guard's relative advantage intact.

Table 12: Hallucination detection F1 comparing KG-Guard against baselines under top-K=100 embedding retrieval with KAPING and Llama-4-Scout. Bold and underlined values indicate the best and second-best result per column, respectively. Avg. Rank is the average rank of each method across the three datasets (lower is better). Values with ± denote mean ± std over 3 runs. G, RS, and RT denote the retrieved subgraph, reasoning summary, and reasoning triples provided to the LLM judge, respectively.

Method	WebQSP	CWQ	PUGG	Avg. Rank
KG-Guard (GINE)	<u>82.0 ± 0.5</u>	**89.4 ± 0.1**	<u>85.8 ± 1.2</u>	1.7
KG-Guard (GAT)	81.9 ± 0.2	<u>88.9 ± 0.4</u>	85.4 ± 0.1	2.7
KG-Guard (GraphTransformer)	**82.6 ± 0.9**	88.4 ± 0.5	84.7 ± 1.0	3.3
SelfCheckGPT (N=10)	63.5	87.0	**86.8**	3.7
SelfCheckGPT (N=5)	61.1	86.1	85.3	5.7
Most Frequent	68.7	85.5	85.1	5.7
RoBERTa (fine-tuned)	65.7 ± 0.3	86.1 ± 2.4	82.4 ± 0.1	6.3
GPT-5-mini (G+RS+RT)	54.3	86.5	75.9	7.0
GPT-5-mini (G)	48.9	83.6	67.2	9.3
Random	50.9 ± 0.3	59.8 ± 0.3	60.0 ± 0.9	9.7
Llama-4-Scout (G+RS+RT)	37.0	49.7	50.1	11.0
Llama-4-Scout (G)	28.9	42.7	42.5	12.0

D.1.2 Top-K Retrieval

The top-K retriever keeps the K=100 triples whose embeddings are most similar to the question embedding, selecting each triple independently rather than as a jointly optimized connected subtree as in PCST. This changes the connectivity, sparsity, and noise level of the retrieved subgraphs. We regenerate the subgraphs, rerun KAPING with Llama-4-Scout as the answering LLM, and retrain and evaluate all methods on the resulting labels. Table 12 reports hallucination detection results under top-K=100 embedding retrieval (F1 only). All three KG-Guard variants occupy the top three positions by average rank, consistent with PCST results, showing that KG-Guard's advantage is not retrieval-specific. SelfCheckGPT (N=10) is the best-performing baseline (average rank 3.7), even outperforming all KG-Guard variants on PUGG, but it still trails KG-Guard (GINE) by 18.5 F1 points on WebQSP and requires 10 answering-LLM calls per question at detection time. Notably, LLM-based judges degrade more substantially under top-K retrieval than KG-Guard: all Llama-4-Scout and GPT-5-mini variants fall below KG-Guard by clear margins, and Llama-4-Scout (G) trails even the Random baseline. This suggests that graph topology provides a more stable verification signal than prompt-level context when the retrieved subgraph is a flat top-K neighborhood rather than a PCST-selected compact tree.

D.1.3 G-Retriever Pipeline

Table 13: Hallucination detection F1 and AUC-PR comparing KG-Guard against baselines under the G-Retriever KBQA pipeline. Bold and underlined values indicate the best and second-best result per column, respectively. Avg. Rank is the average rank of each method across the three datasets, calculated over F1 (lower is better). Values with ± denote mean ± std over 3 runs. G denotes the retrieved subgraph provided to the LLM judge.

Method	WebQSP		CWQ		PUGG		Avg. Rank
	F1	AUC-PR	F1	AUC-PR	F1	AUC-PR	
GPT-5-mini (G)	**85.9**	82.9	**88.4**	86.9	**90.5**	92.3	1.0
KG-Guard (GINE)	77.3 ± 0.1	<u>83.0 ± 0.8</u>	85.9 ± 0.7	91.2 ± 0.9	<u>87.3 ± 0.2</u>	<u>90.7 ± 0.8</u>	3.0
KG-Guard (GraphTransformer)	<u>77.4 ± 0.6</u>	**85.7 ± 0.7**	86.4 ± 0.4	<u>93.1 ± 0.4</u>	86.9 ± 0.4	90.4 ± 1.0	3.0
RoBERTa (fine-tuned)	71.1 ± 4.6	78.3 ± 13.3	<u>86.6 ± 0.7</u>	**95.2 ± 0.4**	86.9 ± 0.3	**93.5 ± 0.5**	3.7
KG-Guard (GAT)	76.2 ± 0.4	<u>83.0 ± 0.3</u>	85.3 ± 0.3	91.1 ± 0.0	86.5 ± 0.0	88.4 ± 1.4	4.7
Llama-4-Scout (G)	73.4	70.2	71.4	81.2	69.0	83.0	6.3
Most Frequent	68.6	52.2	82.4	70.1	85.5	74.7	6.3
Random	51.0 ± 0.3	52.2 ± 0.0	58.2 ± 0.2	70.1 ± 0.2	58.7 ± 1.3	74.4 ± 0.5	8.0

Table 13 reports hallucination detection results when G-Retriever is used as the KBQA pipeline. Unlike KAPING, G-Retriever fine-tunes an LLM on the KBQA dataset to select answer nodes from the PCST subgraph; the same subgraphs are shared across both pipelines. SelfCheckGPT is excluded from this comparison: beyond enabling stochastic sampling in G-Retriever's decoding, it would require N additional GPT-5.4-nano extraction calls per example, which we avoid to limit API costs. This experiment targets `KG-Guard`'s robustness to the choice of KBQA pipeline, not a full rerun of the baseline suite. `KG-Guard` (GINE) and `KG-Guard` (GraphTransformer) tie for the second-best average rank (3.0), behind GPT-5-mini which requires an expensive API call per query. On AUC-PR, `KG-Guard` (GraphTransformer) leads GPT-5-mini on WebQSP (+2.8 pp) and CWQ (+6.2 pp). Llama-4-Scout trails every `KG-Guard` variant in both F1 and AUC-PR on all three datasets. The pattern confirms that `KG-Guard` generalizes beyond the KAPING pipeline.

D.1.4 Full Results (Llama-4-Scout)

Table 14: Full hallucination detection results on WebQSP (F1, Precision, Recall, Accuracy, AUC-PR). Bold and underlined values indicate the best and second-best result per column, respectively. G, RS, and RT denote the retrieved subgraph, reasoning summary, and reasoning triples provided to the LLM judge. Values with ± denote mean ± std over 3 runs.

Method	F1	Precision	Recall	Accuracy	AUC-PR
`KG-Guard` (GraphTransformer)	**82.0 ± 0.7**	84.9 ± 2.2	81.3 ± 2.9	**85.9 ± 0.1**	**91.4 ± 0.1**
`KG-Guard` (GINE)	81.2 ± 0.3	81.5 ± 0.2	83.0 ± 0.3	84.4 ± 0.4	90.2 ± 0.2
`KG-Guard` (GAT)	79.8 ± 1.2	82.1 ± 1.3	79.8 ± 1.6	83.9 ± 1.0	90.7 ± 0.2
Most Frequent	65.9	49.1	**100.0**	49.1	49.1
RoBERTa (fine-tuned)	63.7 ± 5.2	61.5 ± 10.7	74.7 ± 8.2	72.8 ± 20.5	70.2 ± 12.8
SelfCheckGPT (N=10)	61.2	54.4	69.9	56.5	52.8
GPT-5-mini (G+RS+RT)	59.2	85.3	45.3	69.3	65.5
SelfCheckGPT (N=5)	58.9	55.3	63.1	56.8	53.0
GPT-5-mini (G)	55.9	**85.5**	41.6	67.9	64.2
Random	49.6 ± 1.0	49.4 ± 0.9	49.9 ± 1.1	50.3 ± 0.9	49.2 ± 0.5
Llama-4-Scout (G+RS+RT)	41.0	79.6	27.6	61.0	57.5
Llama-4-Scout (G)	36.7	66.7	25.3	57.1	53.5
Llama-4-Scout (G+RS)	36.5	73.4	24.3	58.5	55.0

Table 15: Full hallucination detection results on CWQ (F1, Precision, Recall, Accuracy, AUC-PR). Bold and underlined values indicate the best and second-best result per column, respectively. G, RS, and RT denote the retrieved subgraph, reasoning summary, and reasoning triples provided to the LLM judge. Values with ± denote mean ± std over 3 runs.

Method	F1	Precision	Recall	Accuracy	AUC-PR
RoBERTa (fine-tuned)	**87.7 ± 0.7**	87.9 ± 1.4	90.2 ± 2.2	**86.1 ± 0.5**	**95.5 ± 0.1**
`KG-Guard` (GraphTransformer)	87.4 ± 0.2	85.8 ± 0.2	90.2 ± 0.5	83.5 ± 0.2	94.2 ± 0.3
`KG-Guard` (GINE)	87.1 ± 0.3	84.7 ± 0.8	90.8 ± 0.8	82.8 ± 0.4	93.6 ± 0.4
`KG-Guard` (GAT)	86.3 ± 0.7	84.5 ± 0.3	89.6 ± 1.4	81.7 ± 0.9	93.1 ± 0.2
GPT-5-mini (G+RS+RT)	84.2	93.5	76.7	79.4	88.4
SelfCheckGPT (N=10)	84.0	81.0	87.2	76.2	79.8
Most Frequent	83.5	71.7	**100.0**	71.7	71.7
GPT-5-mini (G)	82.6	90.2	76.2	77.0	85.8
SelfCheckGPT (N=5)	82.3	82.5	82.1	74.7	80.6
Llama-4-Scout (G+RS)	61.3	94.0	45.4	58.8	81.8
Llama-4-Scout (G+RS+RT)	61.2	**94.4**	45.3	58.8	82.0
Llama-4-Scout (G)	60.2	91.6	44.8	57.5	80.7
Random	58.3 ± 0.5	71.4 ± 0.3	49.3 ± 0.5	49.4 ± 0.4	71.6 ± 0.1

Tables 14, 15, and 16 extend the results reported in Section 5.1 with the full precision, recall, accuracy, and AUC-PR breakdown. A consistent pattern emerges: LLM-based judges are strongly precision-skewed. GPT-5-mini (G+RS+RT) reaches 85.3% precision on WebQSP but only 45.3% recall; Llama-4-Scout variants reach up to 96.7% precision on PUGG at recalls near 44%. `KG-Guard` achieves a more balanced trade-off: 84.9% precision and 81.3% recall on WebQSP (F1 $= 82.0$),

Table 16: Full hallucination detection results on PUGG (F1, Precision, Recall, Accuracy, AUC-PR). Bold and underlined values indicate the best and second-best result per column, respectively. G, RS, and RT denote the retrieved subgraph, reasoning summary, and reasoning triples provided to the LLM judge. Values with ± denote mean ± std over 3 runs.

Method	F1	Precision	Recall	Accuracy	AUC-PR
SelfCheckGPT (N=10)	**85.5**	84.9	86.0	<u>79.0</u>	83.1
KG-Guard (GraphTransformer)	<u>84.3 ± 1.0</u>	78.1 ± 3.4	<u>93.5 ± 4.8</u>	78.8 ± 1.9	**90.0 ± 0.6**
SelfCheckGPT (N=5)	83.9	86.5	81.4	77.6	83.7
Most Frequent	83.5	71.6	**100.0**	71.6	71.6
KG-Guard (GINE)	83.1 ± 1.0	81.9 ± 1.0	86.0 ± 2.3	**79.1 ± 0.9**	**90.0 ± 0.5**
KG-Guard (GAT)	82.7 ± 2.7	79.6 ± 2.0	88.1 ± 7.7	77.4 ± 3.2	88.7 ± 0.7
GPT-5-mini (G)	82.6	<u>96.3</u>	72.3	78.2	<u>89.5</u>
GPT-5-mini (G+RS+RT)	81.2	95.1	70.9	76.6	88.3
RoBERTa (fine-tuned)	76.7 ± 2.6	75.1 ± 1.3	83.0 ± 5.5	76.3 ± 0.9	86.0 ± 0.2
Llama-4-Scout (G)	68.8	93.5	54.4	64.7	83.5
Llama-4-Scout (G+RS)	61.3	96.0	45.1	59.3	82.6
Llama-4-Scout (G+RS+RT)	60.8	**96.7**	44.3	59.0	82.7
Random	58.8 ± 1.8	71.8 ± 1.6	49.8 ± 1.8	50.1 ± 1.9	71.7 ± 0.8

showing that the F1 advantage over LLM judges is driven primarily by higher recall. KG-Guard leads in AUC-PR on WebQSP and PUGG (91.4 vs. RoBERTa's 70.2, and 90.0 vs. GPT-5-mini's 89.5), and is a close second on CWQ (94.2 vs. RoBERTa's 95.5).

D.2 Hallucination Correction

Table 17: KBQA answer quality before (<u>Initial</u>) and after (<u>Refined</u>) iterative refinement guided by KG-Guard hallucination flags. Δ is the score improvement.

Dataset		F1	Precision	Recall	Accuracy	Exact Match
WebQSP	Initial	59.5	60.2	63.2	68.6	42.8
	Refined	72.5	75.8	72.7	79.1	60.4
	Δ	+13.0	+15.6	+9.5	+10.5	+17.6
CWQ	Initial	57.2	56.8	59.9	61.3	51.1
	Refined	71.7	72.3	72.2	73.7	68.8
	Δ	+14.5	+15.5	+12.3	+12.4	+17.6
PUGG	Initial	53.1	52.4	56.5	58.0	45.3
	Refined	67.2	67.2	68.9	70.2	62.3
	Δ	+14.0	+14.8	+12.4	+12.2	+16.9
	Mean Δ	+13.8	+15.3	+11.4	+11.7	+17.4

Table 17 extends the F1 and Exact Match results reported in Section 5.2 with Precision, Recall, and Accuracy. Gains are consistent across all five metrics and all three datasets. Precision improves by 14.8–15.6 pp, while recall improves by 9.5–12.4 pp; precision gains exceeding recall gains indicate that refinement primarily eliminates incorrect answer nodes rather than recovering missing ones.

E Additional Statistics

E.1 Knowledge Graph Subgraphs

Table 18 reports subgraph sizes across datasets. WebQSP and CWQ subgraphs are compact (avg. 15–17 nodes, 17–21 triples), owing to Freebase's sparser entity neighborhood compared to Wikidata. PUGG subgraphs are several times larger (avg. 72–98 nodes, 83–121 triples) due to Wikidata's denser structure with many hub nodes, even after filtering nodes connected to more than 1000 entities. Maximum subgraph sizes reach thousands of triples across all datasets, requiring the graph encoder to handle substantial structural variability. The largest subgraphs may also exceed the context

Table 18: Statistics of the retrieved question-specific subgraphs by PCST used as input to `KG-Guard`. Avg./Max #Nodes and Avg./Max #Triples report node and edge statistics per subgraph, respectively.

Dataset	Split	Avg. #Nodes	Max #Nodes	Avg. #Triples	Max #Triples
WebQSP	train	15.5	307	17.4	848
	val	16.2	302	20.7	1792
	test	16.4	282	19.9	1699
CWQ	train	16.5	530	18.6	1783
	val	17.2	851	19.9	2691
	test	16.3	441	17.6	497
PUGG	train	97.6	13614	120.5	16765
	val	72.2	13574	83.2	16702
	test	87.8	13575	108.0	16703

budget used by a text-based baseline, motivating graph-based verification. The full subgraph size distributions are shown in Figure 6. All three datasets are heavy-tailed.

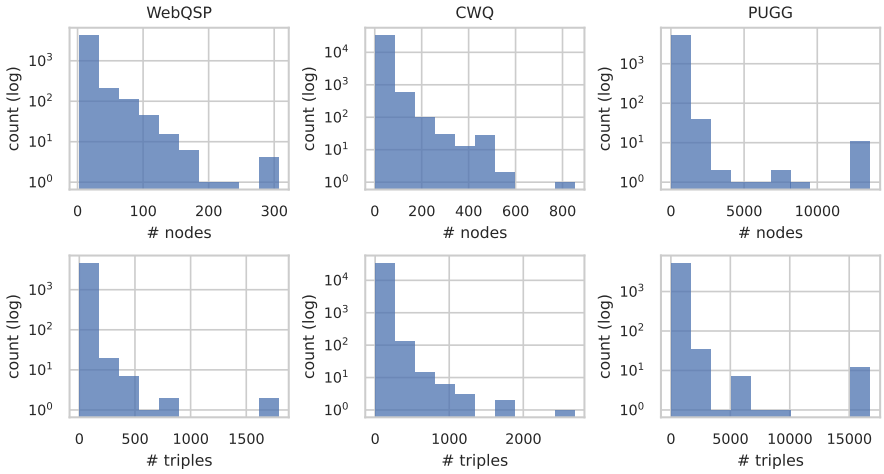

Figure 6: Subgraph size distributions across the three datasets. Top row: number of nodes; bottom row: number of triples. Counts are on a log scale.

E.2 Answer Statistics

Table 19: Average answer count per question. #GT Ans./Q: gold (ground truth) answer nodes; #LLM Ans./Q: nodes returned by the LLM-based KBQA method.

Dataset	Split	#GT Ans./Q	#LLM Ans./Q
WebQSP	train	11.5	3.3
	val	10.4	3.1
	test	10.2	3.0
CWQ	train	2.2	1.6
	val	2.3	1.7
	test	1.9	1.7
PUGG	train	1.8	1.8
	val	1.8	1.5
	test	1.8	1.7

Table 19 reports the average number of gold and LLM-predicted answer nodes per question. WebQSP contains substantially more gold answers per question (avg. 10.2–11.5) than CWQ (1.9–2.3) or PUGG

(≈ 1.8), reflecting that WebQSP questions admit multiple valid answer entities. The LLM consistently under-predicts on WebQSP, returning 3.0–3.3 nodes on average against $10+$ gold answers. On CWQ and PUGG, the LLM returns a similar number of nodes to the gold count.

E.3 LLM Abstention and Hallucination Patterns

Table 20: LLM abstention rate (%Unknown) and gold answer coverage (%Answer in Graph: fraction of examples where at least one gold answer node appears in the retrieved subgraph).

Dataset	Split	%Unknown	%Answer in Graph
WebQSP	train	17.6	62.9
	val	16.2	60.6
	test	16.8	63.0
CWQ	train	37.7	41.4
	val	34.8	42.4
	test	37.8	39.9
PUGG	train	21.5	46.6
	val	19.1	48.9
	test	21.2	47.7

Table 21: LLM-based KBQA abstention rate (*unknown* answers) conditioned on gold answer presence in the retrieved subgraph. %Answer absent/%Answer present: abstention rate when the gold answer is absent from or present in the subgraph, respectively.

Dataset	Split	%Answer absent	%Answer present
WebQSP	train	40.0	4.4
	val	37.6	2.3
	test	39.1	3.6
CWQ	train	56.6	10.9
	val	52.5	10.8
	test	55.4	11.2
PUGG	train	37.8	2.9
	val	35.8	1.8
	test	39.6	1.0

Table 22: Hallucination rate among non-abstained LLM-based KBQA answers, conditioned on gold answer presence in the retrieved subgraph. %Answer absent/%Answer present: hallucination rate when the gold answer is absent from or present in the subgraph, respectively. %Answer absent is always 100% by definition: if the gold answer is not in the subgraph, any non-abstained node returned by the system is hallucinated.

Dataset	Split	%Answer absent	%Answer present
WebQSP	train	100.0	39.8
	val	100.0	39.2
	test	100.0	38.7
CWQ	train	100.0	42.5
	val	100.0	46.9
	test	100.0	42.4
PUGG	train	100.0	41.8
	val	100.0	45.2
	test	100.0	42.5

Table 20 shows that a substantial fraction of LLM responses are abstentions (*unknown*): 16–18$\%$ on WebQSP, 19–22$\%$ on PUGG, and 35–38$\%$ on CWQ. In all cases, the gold answer is present in the retrieved subgraph for fewer than two-thirds of examples, reflecting the inherent difficulty

of subgraph retrieval. Larger subgraphs increase answer coverage but add noise and exceed LLM context limits, while smaller subgraphs are more tractable but risk omitting the correct evidence. Tables 21 and 22 show how abstention and hallucination rates vary with answer presence. When the gold answer is absent from the subgraph, the LLM abstains at rates of 35–57% depending on the dataset. When the gold answer is present, abstention rates drop to 1–11% and hallucination among non-abstaining responses falls to 38–47%.

These patterns confirm that hallucination in this setting is tightly coupled to whether the answer evidence is accessible in the retrieved subgraph. When no gold answer is retrieved, every returned node is labeled hallucinated by definition. This motivates both answer verification and improvements to retrieval.

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 state the same F1 scores (82.0, 87.4, 84.3) and improvement ranges (13.0–14.5 pp F1, 16.9–17.6 pp EM) that appear in the experimental results (Sections 5.1 and 5.2). The five contribution claims in the introduction are each substantiated in the corresponding sections and appendices.

   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: A dedicated "Limitations and Future Work" section discusses three primary limitations: the requirement for hallucination-labelled training data, inability to handle LLM abstentions, and inability to recover answers whose gold node is absent from the retrieved subgraph.

   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: [N/A]

   Justification: The paper presents no novel theoretical results. The method is built from components (Graph Transformer, GAT, and Adam optimizer) whose theoretical properties are established in their respective original papers, which we cite.

   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: The method section describes our approach in full detail. Section A reports all architecture details, optimizer settings, text encoder choices, LLM temperatures, subgraph retrieval procedure, and evaluation protocol; baseline configurations are documented there as well. We release the full code, all experimental data, and the newly constructed KBQA hallucination detection dataset.

   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: [Yes]

   Justification: Code and data including the hallucination detection dataset are released under CC BY-SA 4.0 (GitHub and HuggingFace links in a footnote in Section 1).

   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: Section A covers data splits, all hyperparameters, and evaluation details.

   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: All `KG-Guard` results are reported as mean $\pm$ standard deviation over 3 random seeds (Section A). For LLM-based baselines, single-run results are reported with an explicit explanation that the high inference cost makes repeated runs impractical.

   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: Section A states that Llama-4-Scout inference was run with vLLM on 4 $\times$ H100 GPUs and that `KG-Guard` training used a single NVIDIA A40 GPU. Appendix C provides a model-size comparison across all methods.

   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 uses publicly available KBQA datasets, involves no human subjects, collects no sensitive data, and poses no identified ethical risks.

   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: [Yes]

Justification: The "Limitations and Future Work" section (Section 6) includes a "Broader impacts" paragraph discussing positive impacts and negative risks.

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 released assets (hallucination detection dataset derived from public KBQA datasets) pose no significant misuse risk.

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 are cited by their original papers with license information provided in footnotes: WebQSP (CC-BY 4.0), PUGG (CC BY-SA 4.0), CWQ (GPL v2+). Our derived hallucination dataset is released under CC BY-SA 4.0 for the WebQSP and PUGG subsets, and GPL v3 for the CWQ subset, consistent with the upstream licenses.

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: [Yes]

Justification: The hallucination detection dataset is documented in Section 4.1.2 with construction methodology, statistics (Table 1), subgraph sizes (Appendix E.1), answer distributions (Appendix E.2), and abstention patterns (Appendix E.3).

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: No crowdsourcing or human subject research was conducted. All hallucination labels are derived automatically by comparing LLM-predicted answer nodes against gold answer sets.

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: No human subjects research was conducted; IRB approval is not required.

    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: [N/A]

    Justification: LLMs are not an original or non-standard component of the core method. LLMs appear only as standard tools: Llama-4-Scout, Gemma 4 E4B, and GPT OSS 20B as KAPING answering LLMs; Llama-4-Scout and GPT-5-mini as LLM-judge baselines; GPT-5.4-nano for parsing G-Retriever's free-text output. LLM usage also includes writing assistance (editing, grammar, synonyms, etc.), coding assistance (generating code based on instructions), and results visualization (generating code for plots, tables, etc.). These roles do not impact the originality or scientific rigor of the proposed approach. All generated content and code was manually reviewed and tested.

    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.