Identifying the Achilles' Heel: An Iterative Method for Dynamically Uncovering Factual Errors in Large Language Models

Wenxuan Wang[1] **Yuk-Kit Chan**[2*] **Zixuan Ling**[2*] **Juluan Shi**[2*] **Youliang Yuan**[3]
Jen-tse Huang[4] **Yifei Zhang**[5] **Wenxiang Jiao**[6] **Zhaopeng Tu**[7] **Michael R. Lyu**[2]
[1]Renmin University of China [2]The Chinese University of Hong Kong
[3]The Chinese University of Hong Kong, Shenzhen
[4]Johns Hopkins University [5]Nanyang Technological University
[6]Xiaohongshu Inc. [7]Tencent Inc.
[1]wangwenxuan@ruc.edu.cn [4]jhuan236@jh.edu [5]wenxiangjiaonju@gmail.com

Abstract

Large Language Models (LLMs) like ChatGPT are foundational in various applications due to their extensive knowledge from pre-training and fine-tuning. Despite this, they are prone to generating factual and commonsense errors, raising concerns in critical areas like healthcare, journalism, and education to mislead users. Current methods for evaluating LLMs' veracity are limited by the need for extensive human labor, test data contamination, or limited scope, hindering efficient and effective exposure of errors. To address these challenges, we propose HalluHunter, a novel, fully automated framework for systematically uncovering factual inaccuracies in LLMs. HalluHunter employs a knowledge-graph-based approach, extracting fact triplets to generate diverse question types for single- and multi-hop reasoning using rule-based Natural Language Processing (NLP) techniques. Its iterative process starts with random triplet selection for question generation, followed by adaptive selection in subsequent iterations, targeting triplets where LLMs frequently err based on their performance analysis. Our extensive tests on nine prominent LLMs reveal that HalluHunter can trigger factual errors in up to 55% of tested questions. Moreover, we demonstrate that HalluHunter 's test cases, particularly in adaptive selection, could further expose the weaknesses in benchmarking the factuality in LLMs meanwhile maintaining the coverage of questions. All code, data, and results are available at this link[1].

1 Introduction

Recent advancements in LLMs have propelled artificial intelligence to a notable milestone. For example, ChatGPT has become one of the most prominent LLMs, demonstrating rapid adoption with 100 million monthly active users within two months of its launch, making it the fastest-growing software in history (Gordon, 2023). Moreover, LLMs have significantly impacted various applications, including machine translation (Jiao et al., 2023), grammatical error correction (Wu et al., 2023) and program synthesis (Gao et al., 2023).

A significant barrier to the development of LLM-based intelligent applications, such as intelligence tutoring system, is their intrinsic proneness to errors, particularly in factual accuracy. Prior studies, for instance, have shown that models like ChatGPT often produce plausible yet factually incorrect or nonsensical outputs, a phenomenon known as "hallucinations" (Bang et al., 2023). As these models advance and user trust in their outputs increases, such inaccuracies could lead to more serious consequences. This is especially problematic in sectors like journalism, academia, and healthcare where accuracy and reliability are paramount. Therefore, identifying, analyzing, and mitigating these factual inaccuracies is essential to improve the safety and dependability of LLM-based intelligent software.

While the progress on benchmarking the factuality in LLMs made by recent works is noteworthy, the current methods in evaluating factual accuracy have several shortcomings that require attention, as shown in Table 1. These limitations hinder the ability to explore and address factual errors in LLMs:

1. High Cost: Existing benchmarks (Lin et al., 2021; Talmor et al., 2019; Laskar et al., 2023) rely heavily on question formulation and human annotation, demanding significant effort. **2. Data Contamination**: LLM evaluation often suffers from data contamination due to the static nature of evaluation datasets, making the results unreliable. Unlike earlier models, LLMs use extensive internet-sourced corpora, potentially including publicly available evaluation data (Aiyappa et al., 2023; OpenAI, 2023). **3. Limited Coverage**: Prior research methods exhibit limitations in topic and question type, such as often focusing

[*] Denote Equal Contribution.
[1]https://github.com/Mysterchan/HalluHunter

Findings of the Association for Computational Linguistics: ACL 2026, pages 34288–34309
July 2-7, 2026

Table 1: A comparison of HalluHunter to other factual evaluation works on the issues of high cost, data contamination, limited coverage and lack of effective expose of error

Dataset	Auto Gen?	Dynamic Gen?	Effective Gen?	Question Types	Multi -Hop?	Cover Any Topics?	LLMs Tested?
LAMA Probe (Petroni et al., 2019)	✗	✗	✗	1	✗	✗	✗
MLAMA (Kassner et al., 2021a)	✗	✗	✗	1	✗	✗	✗
GrailQA (Gu et al., 2021)	✗	✗	✗	1	✓	✗	✗
ParaRel (Elazar et al., 2021)	✗	✗	✗	1	✗	✗	✗
SQuAD2.0 (Cunxiang Wang, 2021)	✗	✗	✗	1	✓	✗	✗
SimpleQuestions (Cunxiang Wang, 2021)	✗	✗	✗	1	✗	✗	✗
KQA Pro (Cao et al., 2022)	✓	✓	✗	1	✓	✗	✗
PopQA (Mallen et al., 2023)	✗	✗	✗	1	✗	✗	✓
PAQ (Lewis et al., 2021)	✓	✗	✗	1	✗	✗	✗
TruthfulQA (Lin et al., 2021)	✗	✗	✗	1	✗	✗	✓
SimpleQA (Wei et al., 2024)	✗	✗	✗	1	✗	✗	✓
Omar et al. (2023)	✗	✗	✗	2	✗	✗	✓
Head-to-Tail (Sun et al., 2023)	✓	✓	✗	1	✗	✗	✓
DyKnow (Mousavi et al., 2024)	✓	✗	✗	1	✗	✗	✓
Ours	✓	✓	✓	3	✓	✓	✓

narrowly on specific relations like individuals and their birthplaces (Petroni et al., 2019; Kassner et al., 2021b), or only using multiple choice questions, which have been shown to be a biased evaluation method (Li et al., 2024). **4. Ineffective Error Exposure Mechanisms**: Current methods rely on single-round evaluations with static datasets, lacking effective mechanisms to generate questions that dynamically target areas where LLMs are prone to factual inaccuracies, limiting their ability to expose model weaknesses.

To address these limitations, we propose HalluHunter, a novel, fully automated framework designed to comprehensively evaluate and uncover factual inaccuracies in LLMs. HalluHunter leverages a knowledge-graph-based and rule-based NLP approach, by selecting facts triplets to generate diverse question types, including Yes/No, Multiple-Choice (MC), and WH questions and supporting both single-hop and multiple hops reasoning. By eliminating reliance on manual annotation and static datasets, HalluHunter reduces costs and mitigates data contamination risks with questions generated dynamically at test-time. Its flexible design allows customization across various domains and knowledge bases, ensuring broad topic coverage.

A core innovation of HalluHunter is its iterative and adaptive test case generation algorithm (Algorithm 1), which analyzes LLM performance from prior iterations to dynamically select fact triplets entities structurally similar to those in incorrect responses and relations with low accuracy. This adaptive approach enhances the framework's ability to generate targeted questions that expose LLM inaccuracies across diverse domains, iteratively increasing test difficulty to probe deeper vulnerabilities. The architecture is illustrated in Figure 1.

To comprehensively assess the effectiveness of HalluHunter, we evaluate it on nine widely deployed LLMs: GPT-3.5-Turbo, GPT-4-Turbo, GPT-4o, DeepSeek-v3, Claude-Sonnet-4, Claude-3.5-Haiku, Gemini-2.0-Flash, Qwen-3, and Qwen-3-Reasoning. Our evaluation shows that HalluHunter successfully uncovers factual inaccuracies in these models. The key contributions of this work are:

- We design and implement HalluHunter, a novel, fully automated framework that exposes factual inaccuracies in LLMs by generating diverse question types using knowledge graphs, eliminating reliance on human annotation and mitigating data contamination risks.
- We develop an iterative and adaptive algorithm within HalluHunter that dynamically generates questions by selecting fact triplets based on prior LLM performance, targeting entity- and relation-specific weaknesses to show deeper vulnerabilities.
- We conduct an comprehensive evaluation of HalluHunter using random and iterative question generation across nine state-of-the-art LLMs, demonstrating their ability to identify and expose significant factual errors.

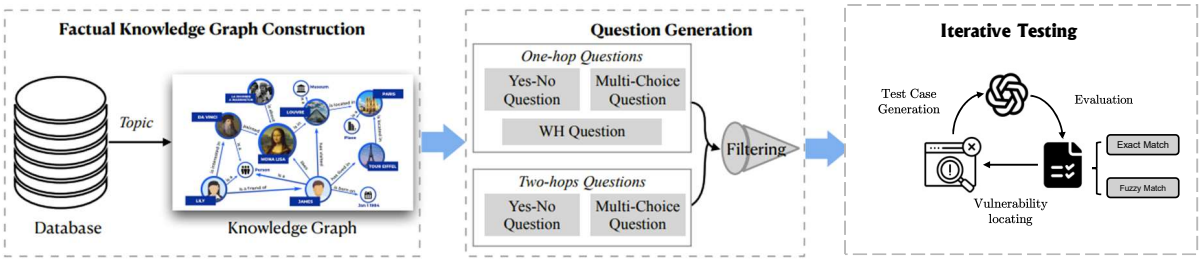

Figure 1: An illustration of the framework of HalluHunter.

2 Approach and Implementation

In this section, we present HalluHunter, a novel framework designed to identify factual errors in LLMs. The architecture comprises four stages:

1. *Knowledge Graph Construction*: Building a structured knowledge graph (KG) from fact triplets sourced from an external database.

2. *Random Question Generation*: Producing diverse one-hop and multi-hop questions by randomly selecting triplets from the KG.

3. *Answer Assessment*: Querying target LLMs and identifying potential factual errors through matching algorithms.

4. *Iterative and Adaptive Question Generation*: Employing an iterative and adaptive algorithm that dynamically updates the question generation strategy based on evaluation results from prior rounds, effectively selecting triplets to target LLM vulnerabilities and enhance the efficiency of exposing factual inaccuracies in subsequent iterations.

2.1 Knowledge Graph Construction

The first step in HalluHunter is the construction of a structured factual knowledge graph (KG), which serves as the foundation for generating evaluation questions. The primary contribution of HalluHunter lies in its flexible and robust framework design, which can seamlessly integrate with various knowledge bases to extract fact triplets. For this study, we demonstrate the framework's capability using Wikidata, a widely accessible knowledge base with over 100 million items [2].

HalluHunter enables users to focus on specific topics by defining criteria, such as "occupation: emperor." These criteria are translated into SPARQL queries [3] to retrieve relevant triplets from the chosen knowledge base. The selected fact triplets are represented as (SUBJECT, relation, OBJECT), for example, (USA, capital, Washington D.C.), indicating that Washington D.C. is the capital of the USA.

After retrieving the triplets, a directed graph is constructed by HalluHunter, denoted as $G = (V, E)$, where the vertex set V comprises SUBJECT and OBJECT entities, and the edges in E represent relations pointing from the SUBJECT vertex to the OBJECT vertex.

2.2 Question Generation

HalluHunter employs a systematic rule-based approach to generate questions from the KG, supporting Yes-No, MC, and WH question types, including single-hop and multi-hop complexities. Implementation details and templates are in Appendix C.

2.2.1 One-Hop Questions Generation

For each randomly selected triplet, HalluHunter creates targeted questions, supporting all major English question types, as shown in Table 2 and Figure 6.

Yes-No Questions: Using a triplet (Subject, Relation, Object), HalluHunter conducts Part-of-Speech (PoS) analysis on the relation to select an appropriate auxiliary verb (e.g., "is" for nouns, "does" for verbs), yielding questions like "Is Washington D.C. the capital of the USA?" from (USA, capital, Washington D.C.). For passive verbs, the structure adjusts for clarity.

To ensure balanced testing, HalluHunter generates an equal number of "No"-answer questions by selecting an incorrect but relation-consistent Object, such as "Is London the capital of the USA?"

Multiple-Choice Questions: HalluHunter uses Named Entity Recognition (NER) to select inter-

[2] https://www.wikidata.org/wiki/Wikidata:Main_Page

[3] https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service

Table 2: Examples of generated questions. The first column shows single-hop questions while the second column shows multi-hop ones.

Tuple	Type	Question	Answer
(Napoleon, native language, Corsican)	Yes-No	Is Corsican the native language of Napoleon?	Yes
	MC	What is the native language of Napoleon? A. Latin B. Chinese C. Corsican D. Marathi	C
	WH	What is the native language of Napoleon?	Corsican
(Michelle Obama, spouse,	Yes-No	Was Michelle Obama's spouse educated at Harvard University?	Yes
educated at, Harvard University)	MC	Where was Michelle Obama's spouse educated at? A. Harvard University B. UCLA C. Stanford University D. MIT	A

rogative pronouns (e.g., "What," "Who," "Where") for SUBJECT or OBJECT queries. For SUBJECT queries, PoS analysis determines the auxiliary verb, forming questions like "Which country's capital is Washington D.C.?" For OBJECT queries, a similar approach applies.

For each triplet, HalluHunter generates four options: one correct answer and three distractors sourced from other knowledge graph edges with the same relation. For example, for (Donald Trump, child, Ivanka Trump), HalluHunter formulates "Who is a child of Donald Trump?" and selects distractors like Malia Obama (child of Barack Obama), Chelsea Clinton (child of Bill Clinton), and Jennifer Gates (child of Bill Gates), randomly assigned as options A, B, C, and D, to test the LLM's ability to identify correct relationships.

WH Questions: HalluHunter has stricter requirements for fact triplets, to ensure that the questions have a unique answer in Wh questions. For instance, instead of generating the question "What is the city of China?" for the fact triplet (China, city, Shanghai), it is more appropriate to generate the question "What is the capital of China?" based on the fact triplet (China, capital, Beijing).

To achieve the above requirement, HalluHunter only selects triplets with a single outgoing edge for the relation. For example, for (China, city, Shanghai), when considering the source entity "China," there are multiple out-edges labeled as "city" pointing to different cities. While for (China, capital, Beijing), there will only be one out-edge of "China" labeled as "capital" pointing to "Beijing." By guaranteeing the uniqueness of the answer for the generated question, HalluHunter limits the variation in correct answers, making the final verification process much more straightforward.

2.2.2 Multi-Hop Questions Generation

HalluHunter generates multi-hop questions that require chained reasoning, such as "Where was Michelle Obama's spouse educated at?" from Table 2. These questions demand multiple steps, like identifying Michelle Obama's spouse as Barack Obama and then determining his education at Harvard University.

For multi-hop questions, HalluHunter uses extended triplets (SUBJECT, relation-list, OBJECT), e.g., (Michelle Obama, {spouse, educated at}, Harvard University). It concatenates the SUBJECT with the first relation (e.g., "Michelle Obama's spouse") and applies PoS analysis to the final relation to select the auxiliary verb, producing questions like "Where was Michelle Obama's spouse educated at?" using the same generation process as one-hop questions.'

2.3 Answer Assessment

2.3.1 LLM Responses Collection

Once HalluHunter has generated a significant number of questions, we can utilize them as test cases to query LLMs. Detail Prompts are provided in Appendix H.

2.3.2 LLM Errors Identification

After collecting LLM responses, HalluHunter evaluates performance to identify factual errors. **For Yes-No and MC questions**: accuracy is assessed via exact matching against ground-truth answers. **For WH questions**: to account for entity variations (e.g., "Great Britain" vs. "United Kingdom"), HalluHunter employs five similarity metrics to determine response correctness:

- **Levenshtein distance**: It is a string metric that quantifies the minimum number of single-character edits required to transform one word into another (Po, 2020).

- **N-grams similarity**: It measures the similarity of two sequences by comparing the overlapping ratio of sub-sequences they contain (Papineni et al., 2002). We used N=1.

- **Word embedding similarity**: It measures the semantic similarity between words represented as dense vector embeddings in a high-dimensional space, adopted in (Chen et al., 2021).

- **Sentence transformer similarity**: It utilizes the sentence transformer model[4] to represent the whole sentences in a vector form.

- **ChatGPT**: HalluHunter directly asks ChatGPT (gpt-3.5-turbo-0613) whether the LLM response is equivalent to the question answer.

The questions that can not be answered correctly by the LLMs will be collected as suspicious errors for further human analysis.

2.4 Iterative and Adaptive Question Generation

The iterative and adaptive question generation stage is a cornerstone of the HalluHunter framework, designed to expose factual inaccuracies in LLMs by dynamically generating targeted questions based on prior iteration results. This stage integrates the question generation and answer assessment stages (Sections 2.2 and 2.3) to iteratively target the LLMs weaknesses. Formally, given a knowledge graph $G = (V, E)$, where V represents entities and T contains triplets $t = (s, r, o)$ with $s, o \in V$ as subject and object respectively and r as a relation. This algorithm processes the question list $Q^{(l)}$, LLM response list $A^{(l)}$, triplet set $T^{(l)}$, and relation-accuracy map $R^{(l)}$ at iteration l to produce a new question list $Q^{(l+1)}$, updated triplet set $T^{(l+1)}$, and updated relation-accuracy map $R^{(l+1)}$.

The algorithm's design is motivated by the hypothesis that LLM factual inaccuracies arise from unfamiliarity with specific entities or relations, treated as knowledge points in the knowledge graph. For example, if an LLM fails to answer a question based on the triplet (hydrogen, atomic mass, 1.008), it may also struggle with similar questions in chemistry, such as "What is the atomic mass for oxygen?" If it fails on a question from the triplet (13, prime factor, 91), it may lack understanding of the relational concept of prime factors.

[4]https://github.com/UKPLab/sentence-transformers

Algorithm 1: Iterative and Adaptive Test Case Generation

Input : Knowledge graph $G = (V, E)$, question list $Q^{(l)}$, LLM response list $A^{(l)}$, triplet set $T^{(l)}$, Relation-accuracy $R^{(l)}$, embedding model $\mathcal{M}$, top-k parameter k, explore constant e, low-accuracy constant a

Output : New question list $Q^{(l+1)}$, Triplet set $T^{(l+1)}$, Relation-accuracy $R^{(l+1)}$

Initialize new question list;
$Q^{(l+1)} \leftarrow \emptyset$;
Remove used triplets to prevent duplication;
$T^{(l+1)} \leftarrow T^{(l)} \setminus \{t_i \mid q_i \in Q^{(l)}\}$;
Shuffle($T^{(l+1)}$);
$R^{(l+1)} = R^{(l)}$
for *each* $(q_i, a_i) \in (Q^{(l)}, A^{(l)})$ *with triplet* $t_i = (s_i, r_i, o_i)$ **do**
  $c_i \leftarrow$ Evaluator(q_i, a_i, t_i) ▷ c_i: True if response is correct, False otherwise ;
  Update $R^{(l+1)}(r_i)$;
for *each* $(q_i, a_i) \in (Q^{(l)}, A^{(l)})$ *with triplet* $t_i = (s_i, r_i, o_i)$ **do**
  $e_i \leftarrow$ random() ▷ value between [0,1] ;
  if $e_i < e$ **then**
    Explore the triplets with low accuracy relation ;
    $T' \leftarrow \{t' \in T^{(l+1)} \mid R^{(l+1)}(t') < a\}$;
  else
    if *not* c_i **then**
      $C \leftarrow$ TopKSimilar$(s_i, k, \mathcal{M})$;
      $T' \leftarrow \{t' \in T^{(l+1)} \mid t'[0] \in C\}$;
    else
      $T' \leftarrow T^{(l+1)}$;
  for $(s', r', o') \in T'$ **do**
    Generate question q' for (s', r', o') with q_i.type;
    if $q' \neq$ *Null* **then**
      $Q^{(l+1)} \leftarrow Q^{(l+1)} \cup \{q'\}$;
      break;
return $Q^{(l+1)}, T^{(l+1)}$;

To address this, Algorithm 1 evaluates question-response pairs $(q_i, a_i) \in (Q^{(l)}, A^{(l)})$ with corresponding triplets $t_i = (s_i, r_i, o_i)$, computing a boolean correctness indicator c_i (True for correct responses and False otherwise) as described in Section 2.3. It updates the relation-accuracy map $R^{(l+1)}$ to track the running average performance of the tested model on different relations. The triplet set $T^{(l+1)}$ is shuffled and depleted of used triplets to ensure diversity and avoid repetition. During each generation, with a fixed probability e, the exploration constant and a, defined as low accuracy, the algorithm selects triplets with low-accuracy relations ($R^{(l+1)}(r_i) < a$) with higher priority to exploit known weaknesses.

If the previous response is incorrect (c_i = False),

a knowledge graph embedding model $\mathcal{M}$, trained using QuatE (Zhang et al., 2019) within the PyKEEN framework (Ali et al., 2021), will be used to identify entities similar to s_i in the embedding space to generate challenging questions that probe related knowledge points. New questions q' for triplets (s', r', o') preserve the type of q_i. If the previous response is correct (c_i = True), a random valid triplet will be picked to generate a new question. By balancing exploitation of weak relations and subject entity, the algorithm effectively exposes the LLM weaknesses in various domains.

3 Evaluation

3.1 Experimental Setup

Software and Models Under Test To assess the effectiveness of the HalluHunter, we employ it to evaluate nine widely-utilized LLMs models: gpt-3.5-turbo-0125, gpt-4-turbo-2024-04-09, gpt-4o-2024-11-20, deepseek-v3-0324, claude-sonnet-4-20250514, claude-3-5-haiku-20241022, gemini-2.0-flash, qwen3-32b and qwen3-32b-reasoning. All are with the default temperature.

We focus on a closed-book evaluation setting, where models answer without retrieval augmentation or external grounding. This setting is important in practice because many real-world interactions with LLMs remain weakly grounded, and even in grounded systems, parametric knowledge can still influence answer selection and reasoning behavior (Mallen et al., 2023; Tan et al., 2025). Therefore, evaluating parametric knowledge remains a meaningful and complementary way to diagnose factual weaknesses in LLMs.

Test Cases Generation To comprehensively evaluate LLMs' performance, we conduct experiments by generating questions from three big domains: Humanity, Social Science and Science, technology, engineering, and mathematics (STEM). We use HalluHunter to generate 1000 questions for each question type within each domain. The iterative algorithm will generate 1000 new questions for each loop to support a same scale comparison.

Evaluation Metrics To rigorously assess performance of LLMs and our novel algorithm, we employ two primary metrics tailored to the objectives of HalluHunter:

- **Accuracy**: We measure the factual accuracy of each LLM by the percentage of correct answers. For Yes-No and MC questions, accuracy is determined through exact matching, For WH questions, accuracy is evaluated using different metrics as detailed in Section 2.3.

- **Weighted Coverage**: Our goal is to ensure that our method maintains a balanced coverage of the entire knowledge graph, avoiding a focus on rare, unconventional facts that are difficult to reach. We aim to give more weight to entities that are more easily reachable—those that have a larger number of neighbors. Inspired from Group Degree Centrality (Karlovčec et al., 2022), a metric commonly used to evaluate the importance or centrality of groups of nodes in networks, we calculate the Group Degree Centrality of the entity set that is selected to form the final question set as its weighted coverage. Let $G = (V, E)$ be a graph and $S \subseteq V$ the selected (visited) set; define the open neighborhood $N(S) = \{\, v \in V \setminus S : \exists\, u \in S \text{ with } (u, v) \in E \,\}$. The (unnormalized) group degree centrality is $C_{\deg}(S) = |N(S)|$, and we report its normalized form $\widehat{C}_{\deg}(S) = \dfrac{|N(S)|}{|V| - |S|} \in [0, 1]$.

3.2 Baseline Performance Analysis

This section evaluates the performance of LLMs under the initial random question generation phase of HalluHunter, referred to as Trial 0, to establish a baseline for factual accuracy across diverse question types and domains. Detailed numerical results are provided in Appendix J.

HalluHunter reveals diverse factual errors in LLMs. By generating and evaluating responses to random question sets, HalluHunter effectively identifies factual inaccuracies across different LLMs. As shown in Trial 0 of Figure 2, GPT-4o achieves the highest accuracy, followed by GPT-4. Notably, WH questions pose the greatest challenge, with an average accuracy of 37.4% across all LLMs, indicating their difficulty in precise factual recall.

Multi-hop questions amplify LLM challenges. HalluHunter generates 1,000 multi-hop questions per domain and question type to assess complex reasoning. As depicted in Figure 3, LLMs exhibit higher error rates on questions with more hops, with accuracy declining sharply from 1-hop to 2-hop questions and more gradually from 2-hop to 4-hop. This trend highlights the increased complexity of multi-hop reasoning, exacerbating factual

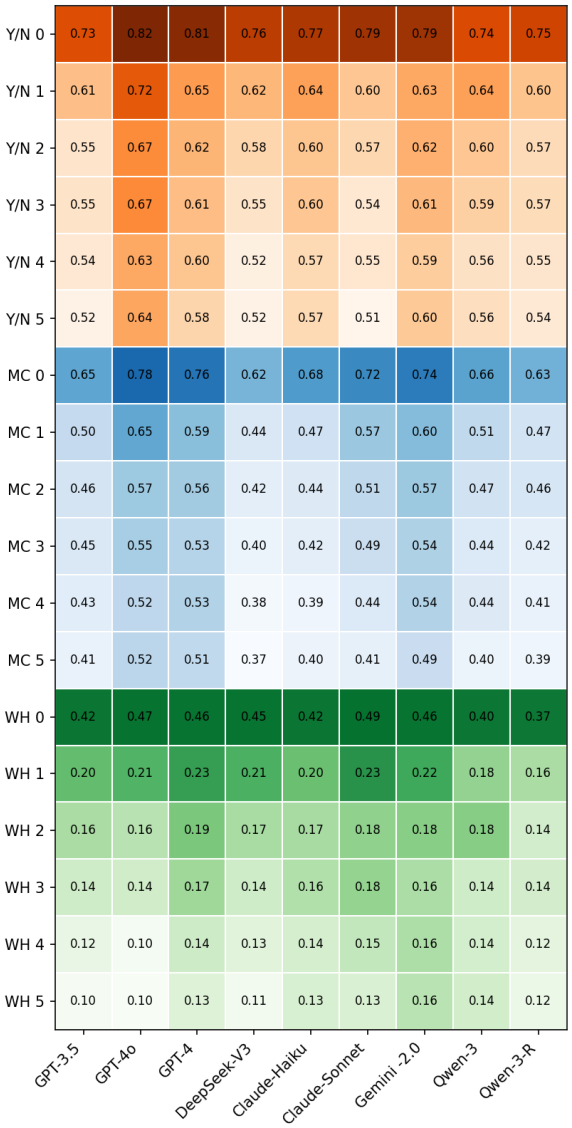

Figure 2: Heat map of LLMs' average accuracy across sequential trials in 1-Hop questions (darker shades indicate higher accuracy)

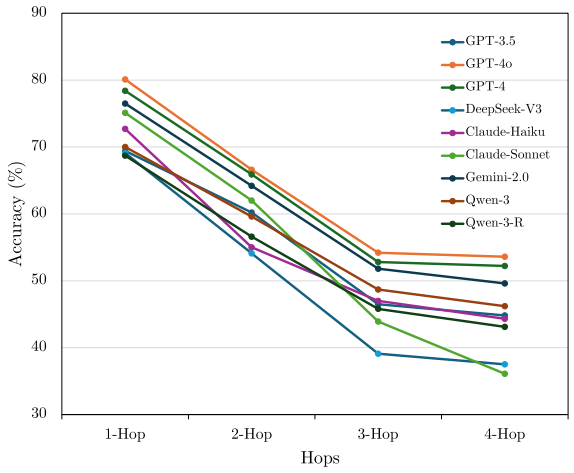

Figure 3: Average LLM accuracy across 1-hop to 4-hop tasks in all domains and question types.

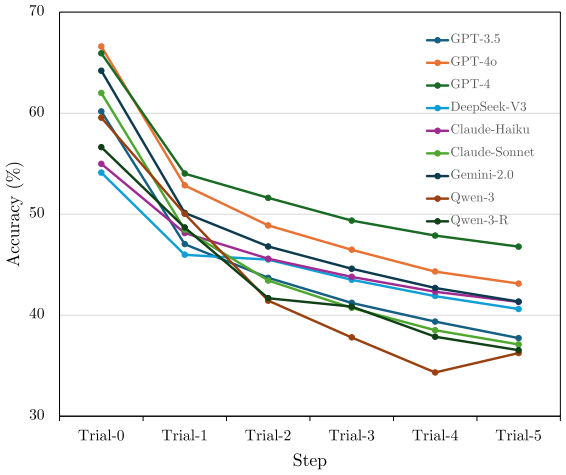

Figure 4: Average LLM accuracy trends across sequential trials for 2-hop questions.

inaccuracies, though the rate of accuracy decline slows with additional hops.

3.3 Effectiveness of Iterative and Adaptive Question Generation

This section assesses the efficacy of the iterative and adaptive question generation algorithm in enhancing exposure of factual error across Trials 0-5 in three question formats with fading color intensity, as shown in Figure 2. Detailed results are provided in Appendix J.

Targeted exposure of factual inaccuracies across domains. HalluHunter 's iterative algorithm (Algorithm 1) enhances the exposure of factual inaccuracies by generating targeted question sets that leverage LLM weaknesses identified from prior iterations, as demonstrated by consistent performance declines across the Humanity, Social Science, and STEM domains. As shown in Figure 2, subsequent trials exhibit a significant reduction in LLM accuracy, driven by strategic triplet selection that probes deeper vulnerabilities. Table 4 details the median accuracies by Trial 5, with Social Science and STEM showing pronounced declines at 0.384 (-40.2%) and 0.373 (-41.8%), respectively, compared to Humanity's milder drop at 0.462 (-32.7%). These results suggest that LLMs struggle with the nuanced, complex questions in Social Science and the precision-demanding queries in STEM, while maintaining relatively stable performance in Humanity. This underscores HalluHunter's ability to expose both factual and rea-

Table 3: Coverage comparison between random questions and Trial 5 (cumulatively).

Domain	Question Type	Trial 5 Coverage	Random Coverage
Hum.	Yes/No	0.442	0.408
	MC	0.480	0.366
	WH	0.473	0.369
Soc. Sci.	Yes/No	0.530	0.530
	MC	0.523	0.551
	WH	0.486	0.563
STEM	Yes/No	0.445	0.333
	MC	0.442	0.250
	WH	0.437	0.280
Average		0.473	0.406

Table 4: Median accuracy (absolute value) and median performance change (percentage vs. seed), aggregated across LLMs and question types, in different domains.

Trial	Hum.	Soc. Sci.	STEM
Seed	0.712 (0.0%)	0.699 (-0.0%)	0.649 (-0.0%)
Trial 1	0.542 (-19.5%)	0.524 (-28.1%)	0.478 (-24.8%)
Trial 2	0.516 (-24.1%)	0.462 (-31.5%)	0.428 (-31.7%)
Trial 3	0.492 (-29.2%)	0.439 (-37.5%)	0.406 (-36.1%)
Trial 4	0.495 (-28.5%)	0.405 (-38.8%)	0.415 (-38.5%)
Trial 5	0.462 (-32.7%)	0.384 (-40.2%)	0.373 (-41.8%)

soning limitations across diverse domains through adaptive question generation.

Robust error exposure in multi-hop scenarios. The algorithm's effectiveness extends to multi-hop questions requiring complex reasoning across multiple relations. For 2-hop questions, Figure 4 illustrates a consistent accuracy decline across trials, highlighting Algorithm 1's capability to probe deeper reasoning vulnerabilities. This underscores HalluHunter 's strength in revealing both factual and reasoning limitations in LLMs.

Enhanced knowledge graph exploration. To evaluate whether the iterative algorithm compromises exploration of the knowledge graph, we compare coverage between random question generation and Trial 5, both in total 5000 questions, as shown in Table 3. The results indicate that the algorithm either mostly maintains or increases coverage, ensuring that question generation remains diverse and covers a broad range of entities and relations in the knowledge graph.

4 Related Work

The prevalent approach of factual evaluation involves reference-based techniques, which rely on benchmarks created through manual question design or test input labeling (Clark et al., 2019; McCoy et al., 2019; Lin et al., 2021; Talmor et al., 2019; Laskar et al., 2023). Despite their utility, these methods demand substantial human effort and are dependent on the development of comprehensive benchmarks. Lewis et al. (2021) is a pioneering work that automatically generates massive question-answer pairs to evaluate open-domain question-answering models. Recently, Sun et al. (2023) and Omar et al. (2023) also adopted knowledge graphs to evaluate the factual correctness of LLMs. However, their scope of question types and topics, testbed models, and evaluation methods are limited, compared with HalluHunter.

Additionally, static benchmarks are prone to data contamination, posing challenges in accurately evaluating LLMs and efficiently identifying errors. Recent advancements have seen the emergence of automatic test case generation, independent of manually pre-annotated labels (Chen et al., 2021; Liu et al., 2022; Shen et al., 2022). However, these techniques still depend on existing benchmarks for question formulation, limiting the breadth of test case generation.

Although automated question generation from knowledge graphs (KGs) or LLMs reduces human annotation costs, the resulting questions are often generic and fail to target specific LLM weaknesses effectively. Recent approaches using LLMs to autonomously expose errors (Chen et al., 2024; Cheng et al., 2024) rely heavily on LLMs for the evaluation pipeline, risking the propagation of inherent model biases and limiting test case diversity and grounding. In contrast, HalluHunter employs an iterative and adaptive algorithm that leverages a structured KG and embedding models to generate targeted, objective, and diverse test cases, ensuring effective uncovering of LLM weaknesses.

To sum up, our approach differs significantly in several key aspects: (1) **Scope**: HalluHunter offers a more extensive scope, capable of generating three types of questions on any topic, as opposed to their method which is restricted to probing specific relationships, like "place of birth" or "date of birth", in a cloze-style format. (2) **Testbed**: HalluHunter evaluates nine leading LLMs, while most of the previous works focused solely on the traditional models. (3) **Testing Logic**: HalluHunter is an automated testing framework designed to dynamically generate a diverse array of test cases each time it is run, aiming to avoid data contamination. (4) **Effective Generation**: HalluHunter 's algorithm uses performance-driven triplet selection to generate tar-

geted, high-quality test cases, ensuring robust and effective uncovering of LLM weaknesses.

5 Conclusion

In this paper, we design and implement HalluHunter, an automated framework dedicated to uncovering factual errors in LLMs. Distinct from previous approaches that depend on extensive human annotation or are prone to data contamination, HalluHunter leverages a structured KG to autonomously and iteratively generate a wide array of questions spanning various topics. We conducted comprehensive evaluations using nine models. Our empirical findings reveal that HalluHunter can successfully identify factual errors in LLMs.

Limitations

This paper has two primary limitations that offer avenues for future research:

- The effectiveness of HalluHunter is affected by the quality of the knowledge graph (*e.g.*Wikidata), which is hard to guarantee. Please refer to our discussion in Appendix A.
- This paper does not provide any novel method to improve the factual correctness of LLMs. More effective and efficient methods are needed to further enhance the factual correctness of LLMs.

References

Rachith Aiyappa, Jisun An, Haewoon Kwak, and Yong-Yeol Ahn. 2023. Can we trust the evaluation on chatgpt? *ArXiv*, abs/2303.12767.

Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. 2021. PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *Journal of Machine Learning Research*, 22(82):1–6.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *ArXiv*, abs/2302.04023.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6101–6119, Dublin, Ireland. Association for Computational Linguistics.

Songqiang Chen, Shuo Jin, and Xiaoyuan Xie. 2021. Testing your question answering software via asking recursively. *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 104–116.

Yulong Chen, Yang Liu, Jianhao Yan, Xuefeng Bai, Ming Zhong, Yinghao Yang, Ziyi Yang, Chenguang Zhu, and Yue Zhang. 2024. See what llms cannot answer: A self-challenge framework for uncovering llm weaknesses. *Preprint*, arXiv:2408.08978.

Jiale Cheng, Yida Lu, Xiaotao Gu, Pei Ke, Xiao Liu, Yuxiao Dong, Hongning Wang, Jie Tang, and Minlie Huang. 2024. Autodetect: Towards a unified framework for automated weakness detection in large language models. *Preprint*, arXiv:2406.16714.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *North American Chapter of the Association for Computational Linguistics*.

Yue Zhang Cunxiang Wang, Pai. 2021. Can generative pre-trained language models serve as knowledge bases for closed-book QA? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Preprint*, arXiv:2102.01017.

Shuzheng Gao, Xinjie Wen, Cuiyun Gao, Wenxuan Wang, and Michael R. Lyu. 2023. Constructing effective in-context demonstration for code intelligence tasks: An empirical study. *ArXiv*, abs/2304.07575.

Cindy Gordon. 2023. Chatgpt is the fastest growing app in the history of web applications. `https://www.forbes.com/sites/cindygordon/2023/02/02`. Accessed: 2023-07-01.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488. ACM.

Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Xing Wang, Shuming Shi, and Zhaopeng Tu. 2023. Is chatgpt a good translator? yes with gpt-4 as the engine. *arXiv preprint arXiv:2301.08745*.

Mario Karlovčec, Matjaž Krnc, and Riste Škrekovski. 2022. Evaluating group degree centrality and centralization in networks. *Informatica*, 46.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021a. Multilingual LAMA: Investigating knowledge in multilingual pretrained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3250–3258, Online. Association for Computational Linguistics.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021b. Multilingual lama: Investigating knowledge in multilingual pretrained language models. *EACL*.

Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq R. Joty, and J. Huang. 2023. A systematic study and comprehensive evaluation of chatgpt on benchmark datasets. In *Annual Meeting of the Association for Computational Linguistics*.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Kuttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. Paq: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

Wangyue Li, Liangzhi Li, Tong Xiang, Xiao Liu, Wei Deng, and Noa Garcia. 2024. Can multiple-choice questions really be useful in detecting the abilities of llms? *ArXiv*, abs/2403.17752.

Stephanie C. Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. In *Annual Meeting of the Association for Computational Linguistics*.

Zixi Liu, Yang Feng, Yining Yin, J. Sun, Zhenyu Chen, and Baowen Xu. 2022. Qatest: A uniform fuzzing framework for question answering systems. *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *Preprint*, arXiv:2212.10511.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *ACL*.

Belinda Mo, Kyssen Yu, Joshua Kazdan, Joan Cabezas, Proud Mpala, Lisa Yu, Chris Cundy, Charilaos Kanatsoulis, and Sanmi Koyejo. 2025. Kggen: Extracting knowledge graphs from plain text with language models. *arXiv preprint arXiv:2502.09956*.

Seyed Mahed Mousavi, Simone Alghisi, and Giuseppe Riccardi. 2024. Dyknow: Dynamically verifying time-sensitive factual knowledge in llms. *Preprint*, arXiv:2404.08700.

Reham Omar, Omij Mangukiya, Panos Kalnis, and Essam Mansour. 2023. Chatgpt versus traditional question answering for knowledge graphs: Current status and future directions towards knowledge graph chatbots. *ArXiv*, abs/2302.06466.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *ArXiv*, abs/1909.01066.

Daw Khin Po. 2020. Similarity based information retrieval using levenshtein distance algorithm. *International Journal of Advances in Scientific Research and Engineering*.

Qingchao Shen, Junjie Chen, J Zhang, Haoyu Wang, Shuang Liu, and Menghan Tian. 2022. Natural test generation for precise testing of question answering software. *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*.

Kai Sun, Y. Xu, Hanwen Zha, Yue Liu, and Xinhsuai Dong. 2023. Head-to-tail: How knowledgeable are large language models (llms)? a.k.a. will llms replace knowledge graphs? *ArXiv*, abs/2308.10168.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *NAACL*.

Yuqiao Tan, Shizhu He, Huanxuan Liao, Jun Zhao, and Kang Liu. 2025. Dynamic parametric retrieval augmented generation for test-time knowledge enhancement. *Preprint*, arXiv:2503.23895.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *Preprint*, arXiv:2411.04368.

Hao Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael R. Lyu. 2023. Chatgpt or grammarly? evaluating chatgpt on grammatical error correction benchmark. *ArXiv*, abs/2303.13648.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. *Preprint*, arXiv:1904.10281.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

A Threats to Validity

The validity of this work may be subject to several potential threats.

The first concern is the reliance on NLP techniques employed by HalluHunter for error detection. Given the inherent limitations of NLP methods, HalluHunter might generate false positives or overlook errors, resulting in false negatives. This is particularly evident in scenarios where varying interpretations of correct responses to WH questions challenge accurate validation. To mitigate this issue, we evaluated the efficacy of several prominent similarity methods, selecting the most effective one based on performance metrics. Additionally, we conducted human annotation to demonstrate that HalluHunter achieves high accuracy in error detection, as evidenced by the results.

The second threat is from the implementation of HalluHunter, which covers only one knowledge base, Wikidata. Like any knowledge base, Wikidata is prone to factual inaccuracies or suffers from incomplete data, leading to sub-optimal question generation. Additionally, it is vulnerable to issues like data leakage. To address these two concerns, we adopt strategies respectively: (1) HalluHunter is designed for flexibility, allowing easy substitution of Wikidata with alternative knowledge bases. Incorporating multiple knowledge bases can enhance the robustness and quality of the generated questions. (2) One advantage of Wikidata is the graph format for information storage, a method not extensively employed in training most LLMs despite its public availability. Our primary contribution lies in the development of an automated testing framework. This framework aims to minimize the human effort needed to identify factual inaccuracies within LLMs. Essentially, HalluHunter flags potential errors, which are then subjected to further human analysis to assess their validity.

The third limitation of our study is the limited exploration of various LLMs during evaluation. Our current analysis does not encompass a broad assessment of HalluHunter's performance across numerous systems. To address this limitation, we focus on testing the most prevalent conversational LLMs and SOTA academic models developed by major corporations. Future work, utilizing HalluHunter, could expand this scope to include additional commercial and research models, thereby enhancing the robustness of our findings.

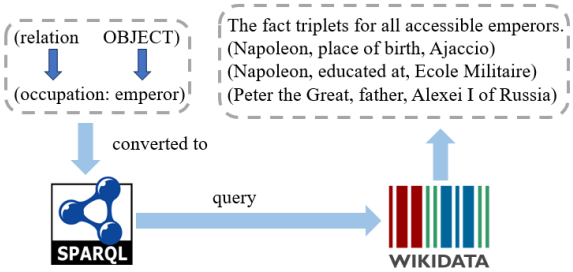

Figure 5: The retrieval process for fact triplets.

B Illustration of The Retrieval Process of Fact Triplets.

A fact triplet is represented in the form of (SUBJECT, relation, OBJECT). For instance, the triplet (USA, capital, Washington D.C.) denotes the fact that the capital of the USA is Washington D.C. HalluHunter enables users to obtain fact triplets pertaining to specific topics. As illustrated in Figure 5, when a user expresses interest in the topic of emperors, HalluHunter proceeds to convert the "occupation: emperor" specification into a SPARQL query language[5], which is utilized for querying related triplets in Wikidata. The resulting SPARQL query will retrieve all accessible fact triplets about emperors, including examples such as "Napoleon, place of birth, Ajaccio" and "Peter the Great, father, Alexei I of Russia".

C Illustration of the Rule-Based Method for Question Generation

HalluHunter utilizes a rule-based approach to generate questions from the constructed KG. It can generate various types of questions, including different question types, *i.e.*, Yes-No questions, MC questions and WH questions. For each triplet in the constructed knowledge graph, HalluHunter converts it to the question form according to their POS and NER features. Figure 6 shows some examples of HalluHunter's question generation method.

D Multi-Hop Question Construction

To address concerns about potential brittleness and ambiguity in multi-hop question construction, we provide a detailed specification of HalluHunter 's procedure, incorporating safeguards to prevent spurious chains and ensure clarity. The multi-hop generation is constrained to simple, two-hop chains

[5] https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service

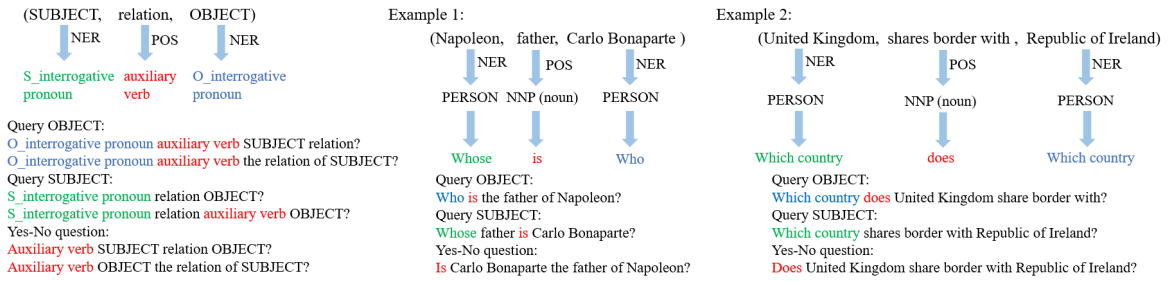

Figure 6: The proposed rule-based method for Question Generation.

of the form $(Entity_A, relation_X, Entity_B) \wedge (Entity_B, relation_Y, Entity_C)$, we turn them to $(Entity_A, \{relation_X, relation_Y\}, Entity_C)$. For example, to generate a question like "Where was Michelle Obama's spouse educated at?" (Table 2), HalluHunter retrieves the triplet (Michelle Obama, spouse, Barack Obama) and chains it with (Barack Obama, educated at, Harvard University). The intermediate entity ("spouse") is concatenated into a fluent subject ("Michelle Obama's spouse") using Part-of-Speech (PoS) analysis to determine the auxiliary verb ("was") and WH interrogative ("Where").

Generating WH questions for multi-hop scenarios poses a challenge, primarily concerning the assurance of answer uniqueness. As an illustration, consider the triplet (Michelle Obama, child, educated at, Harvard Law School). To assess the feasibility of employing this case for WH question generation, one HalluHunter must initially explore all of Michelle Obama's children. In cases where Michelle Obama has multiple children, HalluHunter must then verify the educational background of each child. Only when there exists a sole viable answer, the fact triplet becomes suitable for formulating the WH question.

E Costs of using HalluHunter

The cost-efficiency of HalluHunter is a key advantage of its design. HalluHunter takes a few hours to generate tens of thousands of high-quality, multi-hop, multi-format questions. This automation eliminates the need for expensive crowd-sourced or expert annotations, which are commonly used in other benchmarks, and significantly reduces operational costs. Additionally, the framework avoids the reliance on LLM-driven question generation, which would incur additional API costs and potential biases. By iteratively re-mining the same KG to generate new, targeted questions that expose model weaknesses, HalluHunter maximizes the utility of the KG, further enhancing cost-efficiency. While the construction of a high-quality knowledge graph may require initial time and human effort, recent research has demonstrated the feasibility of automatic KG construction. For example, (Mo et al., 2025) leverages LLMs to generate high-quality questions directly from automatically constructed KGs.

Table 5: Selected topics for evaluation.

Domain	Topic	Example
Hum.	Art	Monna Lisa
	History	World War II
	Philosophy	Plato
Soc. Sci	Psychology	Sigmund Freud
	Sociology	Elitism
	Geography	Earthquake
STEM	Mathematics	Pythagorean theorem
	Physics	Planck constant
	Computer Science	Dijkstra's algorithm

F Details of Selected Topics

To ensure comprehensive evaluation, HalluHunter generate 1,000 questions per domain (Humanities, Social Sciences, STEM), with each domain comprising five topics (200 questions each). This distribution is maintained across trials for consistency. Table 5 lists the topics and example entities.

G Preliminary Experiments

In this section, we conducted an initial experiment to validate our choice of employing a rule-based method for question generation as opposed to directly instructing ChatGPT(gpt-3.5-turbo-0613) to craft questions from fact triplets. Additionally, we conducted experiments to assess the effectiveness of our grammar-checking and polishing modules.

We also meticulously investigated the comparison of five evaluation metrics.

G.1 Prompts for ChatGPT

To clarify, no prompt tuning was performed to optimize performance for ChatGPT experiments. Below, we provide three types of prompts for two question types (Yes-No and WH/MC) and two entity cases (asking about Entity A or B in relation (A, relation, B)) in MC and WH questions, presented in Table 6 and Table 7.

Table 6: Yes/No Question template drafting.

Your are now given a triplet in format of <A, relation, B>, and your task is to create a yes/no question template for asking whether the relation holds between A and B. In crafting the template, you should replace entity A and entity B with <Subject> and <Object>, in order to build template for certain relation. #Example 1 Triplet: <Barack Obama, born in, Hawaii> Question: Was <Subject> born in <Object>? #Example 2 Triplet: <Albert Einstein, profession, physicist> Question: Was <Subject> a <Object>? Now, create a question for the following triplet: Triplet: {SAMPLES} Question:

Table 7: Wh Question template drafting.

You are now given a triplet in format of <A, relation, B>, and your task is to create a wh- question template for asking Object B of the triplet. In crafting the template, you should replace entity A with <Subject> A, in order to build template for certain relation. #Example 1 Triplet: <Barack Obama, born in, Hawaii> Question: Where was <Subject> born? #Example 2 Triplet: <Albert Einstein, profession, physicist> Question: What is the profession of <Subject>? Now, create a question for the following triplet: Triplet: {SAMPLES} Question:

G.2 Can ChatGPT outperform the proposed rule-based methods on question generation?

Given the capabilities of ChatGPT, an alternative approach for generating questions from fact triplets involves instructing ChatGPT to generate the desired questions based on the extracted fact triplets. To verify the viability of this approach, we prompted ChatGPT to generate 200 questions from fact triplets and compared the results with the rule-based method we proposed. Subsequently, we enlisted the assistance of three annotators, each holding a Bachelor's degree or higher and proficient in English, to independently evaluate the quality of the generated questions. Any discrepancies in their assessments were resolved through discussion. The results indicate that while ChatGPT is capable of producing some high-quality questions from fact triplets, it may occasionally deviate from our instructions, introducing unreliability. Among the 200 questions generated, the annotators found that 26 did not align with our expectations. On the other hand, despite introducing some grammatical errors, the rule-based method produced questions where 98.5% adhered to the intended semantic meaning.

G.3 Are the modules of grammar-checking and rephrasing effective?

In order to address potential grammatical errors introduced by rule-based approaches in question generation, we have implemented and compared two modules within our method. The first approach incorporates the use of a grammar checker API to filter out questions exhibiting grammatical errors. Following this filtering process, the remaining questions maintain a high level of quality. However, this approach has a drawback as it tends to be overly sensitive, leading to the elimination of approximately 50% of all generated questions, resulting in a notable false positive rate. The second alternative entails instructing ChatGPT to paraphrase the generated questions, thereby rectifying grammatical errors and enhancing the natural sound of the questions. Our hired annotators observed that by directly leveraging ChatGPT for paraphrasing, the question formats became more diverse, and all 48 questions initially containing grammar errors were successfully corrected. The final results demonstrate that relying solely on the rewriting approach yields better overall performance. Thus, HalluHunter adopts the rewriting powered by ChatGPT to obtain fluent test cases.

G.4 Which similarity metric performs the best?

Due to the diverse nature of the responses generated for WH questions, utilizing a straightforward exact match criterion is not sufficient for addressing these variations effectively. As a result, we compare five distinct evaluation methods described in Section 2.3.2. The objective is to identify the most effective method that yields satisfactory results. To conduct the evaluation, we randomly selected 500 questions along with their corresponding generated responses from LLMs and the ground-truth an-

swers. Subsequently, the recruited three annotators are required to annotate whether the generated response matches the ground-truth answers. Finally, we obtained 238 cases that the responses are annotated as not aligned with the ground truth. Then, we use the annotated 500 data as a benchmark to evaluate the performance of the five matching methods. The results are shown in Table 8, demonstrating that the sentence transformer method exhibits the most promising performance, with the highest F1 score of 87

It is important to note that this F1 score specifically applies to WH questions, which account for 33% of our benchmark (1,000 out of 3,000 questions per trial per domain). Other question types, such as Yes-No and Multiple-Choice, are evaluated deterministically via exact string matching, achieving 100% evaluation accuracy. Thus, while WH questions inherently pose challenges due to answer variations (e.g., "Great Britain" vs. "United Kingdom" vs. "UK"), we consider the sentence transformer's F1 score of 87% reasonably high, given these complexities. In this context, we respectfully argue that our evaluation is both valuable and reliable.

Additionally, we emphasize that the sentence transformer method minimizes false negatives, which supports the reliability of our evaluation. As shown in Table 8, the sentence transformer achieves relatively high precision—indicating few false positives—and exceptionally high recall, meaning nearly all true errors are captured. This high recall aligns with our primary focus on the reliability and quality of the identified set of factual inaccuracies. While there is an inherent trade-off between precision and recall, we prioritize recall to ensure that the identified error set is comprehensive and trustworthy for analyzing model weaknesses. Consequently, we consider a small number of missed errors acceptable, further supporting the use of sentence transformers as the matching metric for WH questions.

Thus, HalluHunter adopts the sentence transformer as the matching metric for WH questions.

H Prompt Design for LLM Evaluation

To ensure precise and consistent evaluation of Large Language Models (LLMs), HalluHunter employs carefully crafted prompts tailored to each question type, minimizing ambiguity and focusing on direct responses to assess factual accuracy. The specific prompts used are as follows:

Table 8: Performance of different evaluation methods.

Evaluation Method	Precision	Recall	F1
Levenshtein distance	72.6	99.2	83.8
1-grams	61.7	100	76.3
Word embedding	72.8	91.2	81.0
Sentence transformer	78.2	97.9	87.0
ChatGPT	100	65.5	79.2

- **Yes-No Questions**: "The following question's topic is about `TOPIC`. Only answer 'Yes' or 'No', and do not provide an explanation."
- **Multiple-Choice Questions**: "The following question's topic is about `TOPIC`. Choose the only correct option from ('A', 'B', 'C', or 'D') and do not provide an explanation."
- **WH Questions**: "The following question's topic is about `TOPIC`. Provide the answer in 'phrase' or 'word' format only. Do not provide an explanation or use a sentence."

It is important to note that this framework does not allow the model to abstain or refrain from providing an answer. Abstention benchmarks like SimpleQA (Wei et al., 2024) are designed to evaluate whether models know when they don't know, measuring calibration and epistemic uncertainty. In contrast, HalluHunter focuses on whether models correctly recall and reason over facts, measuring parametric knowledge coverage and effectively exposing factual inaccuracies in different models. Both dimensions are crucial for building trustworthy LLMs. However, while abstention benchmarks prioritize precision by focusing on reducing false positives, our study prioritizes recall by targeting the identification of factual inaccuracies.

From this perspective, our study regards abstention as incorrect, as the goal is to rigorously test the model's factual correctness. Nevertheless, we emphasize that in real downstream applications, abstention is often preferable to unsupported guessing when reliability and safety are the primary concern.

I Reproducibility Details

To enhance reproducibility, we clarify key configurations and hyperparameters for HalluHunter. The knowledge graph is constructed from Wikidata, with each domain containing over 500,000 triplets

and more than 10,000 distinct entities. Random selection experiments use a fixed seed for deterministic outcomes. The iterative algorithm (Algorithm 1) employs the following hyperparameters: an explore constant $e = 0.2$ balances exploitation of low-accuracy relations (R_{low}) with exploration of new relations; a low-accuracy constant $a = 0.4$ targets the 40th percentile of relation accuracies; and a top-k parameter $k = 10$ limits the QuatE embedding model (PyKEEN) to the 10 most similar entities for distractor generation. For answer assessment, the sentence transformer similarity threshold is set to 0.6, with performance detailed in Table 8. Experiments were conducted on an A100 GPU (40GB VRAM), with API calls to nine LLMs (e.g., GPT-4o, Claude-3.5-Haiku) costing approximately $400 USD.

J Comprehensive Evaluation Results

This section details HalluHunter 's evaluation of LLM factual accuracy across single-hop, 2-hop, and multi-hop questions, highlighting the iterative algorithm's effectiveness in exposing inaccuracies.

J.1 Multi-Hop Question Performance

Table 12 shows LLM accuracy on multi-hop questions (1-4 hops) across Humanities, Social Sciences, and STEM for Yes-No and Multiple-Choice questions. Each trial includes 1,000 unique questions per domain and question type. Accuracy decreases with increasing hops, with Multiple-Choice questions posing greater challenges, especially in STEM.

J.2 Single-Hop Question Performance

Figure 2 employs three distinct color gradients to differentiate between question types: orange represents Yes/No questions, blue signifies Multiple-Choice (MC) questions, and green corresponds to WH questions. Within each gradient, the intensity of the color reflects the models' performance across sequential trials, with darker shades indicating higher accuracy and better performance, while lighter shades, closer to white, denote lower accuracy. This gradation effectively highlights the impact of our iterative algorithm in exposing factual inaccuracies.

Table 13 presents LLM accuracy on single-hop questions across three domains and question types (Yes-No, Multiple-Choice, WH). Each trial includes 1,000 unique questions. Algorithm 1 reduces accuracy in later trials (e.g., GPT-4o's accuracy drops from 84.4% to 65.8% for Yes-No and 82.9% to 54.1% for Multiple-Choice), with WH questions showing the lowest average accuracy (37.4%).

J.3 2-Hop Question Performance

Table 14 reports LLM accuracy on 2-hop questions across domains and question types. The iterative algorithm significantly lowers accuracy in subsequent trials, highlighting LLMs' challenges with sequential reasoning.

J.4 Validation of Factual Errors

To investigate whether HalluHunter reveals novel weaknesses in LLMs beyond reformulating known deficiencies, we conducted a comparative analysis of its performance against established benchmarks. Figure 3 illustrates LLM performance on single-hop questions generated through random triplet selection, representing the baseline (Trial 0) without HalluHunter 's iterative probing mechanism. These results align closely with the baseline accuracies reported in Table 13, where models such as GPT-4o, GPT-4, and Gemini-2.0 achieve accuracies ranging from 65% to 70% across domains.

However, the application of HalluHunter 's iterative and adaptive question generation algorithm, as detailed in Algorithm 1, significantly amplifies the exposure of LLM vulnerabilities. As shown in Tables 13 and 14, accuracies across Trials 1–5 decline markedly, with most LLMs dropping to approximately 50%, 40%, and 10% for Yes-No, Multiple-Choice (MC), and WH questions, respectively. This pronounced reduction, which intensifies with successive trials, demonstrates HalluHunter 's efficacy in systematically revealing error-prone areas that challenge LLMs' factual recall and reasoning capabilities.

In contrast, benchmarks such as Head-to-Tail (Sun et al., 2023) report GPT-4o achieving approximately 40% accuracy on WH questions in open-domain settings. HalluHunter 's iterative probing further exacerbates this decline, projecting accuracies as low as 10% by Trial 5 across all domains, as evidenced in Table 13. This substantial degradation highlights HalluHunter 's unique ability to uncover novel failure modes that conventional benchmarks may overlook, thereby providing deeper insights into LLMs' factual limitations and enhancing the evaluation of their robustness.

J.5 Error Pattern Analysis

Table 9: Error Patterns of different models in specific relations

Models	Domains	Relations	Accuracy
GPT-4o	Physics	Binding energy	0.258 (74/287)
		Mass excess	0.237 (69/291)
Claude Haiku	Mathematics	Prime factor	0.313 (260/831)
	Biology	Chromosome	0.237 (69/291)
Gemini	Physics	Mass excess	0.241 (67/278)
	Computer Science	Operating system	0.308 (104/367)
DeepSeek	Economics	Central bank/issuer	0.253 (39/154)

We have conducted a detailed examination of the complete six-trial dataset, comprising 6,000 questions. This analysis reveals several key observations regarding topic-level trends and model-specific blind spots. For instance, GPT-4o, the overall strongest model in our evaluation, exhibits clear weaknesses in nuclear and particle physics concepts. Relations such as "binding energy" and "mass excess" yield accuracies of only 0.258 and 0.237, respectively. In contrast, GPT-4o performs near ceiling on biomedical associations, achieving an accuracy of 0.778 across questions related to "genetic association" These results highlight GPT-4o's blind spots in physics-related domains despite its exceptional performance in biology, suggesting that its training data or reasoning capabilities may be biased toward biomedical topics.

Similarly, Claude-3.5-Haiku demonstrates a reproducible weakness in elementary number-theory reasoning. For questions related to "prime factor," Claude-3.5-Haiku achieves an accuracy of only 0.313 across 831 questions, whereas other models such as Gemini-2.0 and GPT-4o consistently perform 0.60 on the exact same domain questions. This discrepancy underscores a specific limitation of the Claude-3.5-Haiku in mathematical reasoning, which our adaptive rounds systematically surface. These cases are similar in Cytology domains. These findings not only validate the effectiveness of our framework in exposing model-specific weaknesses but also provide actionable insights into the areas where LLMs require further improvement. Future researchers can use our framework to improve their models based on the performance analysis.

J.6 Coverage Analysis

Our algorithm 1 strengthens exploitation through targeted probing of model weaknesses, without sacrificing coverage compared to random sampling. Specifically, exploitation focuses only on queries where the model fails. Furthermore, the model does not get stuck in specific regions of the KG. We have calculated the coverage metric across multiple trials, which shows a steady increase from the initial seed trial to the final trial, instead of showing a sign of stuck. Table 15 reports the coverage scores of our algorithm for each trial, domain, and LLM. A higher coverage score indicates that the algorithm explores and covers more of the knowledge graph effectively.

J.7 Exploration-Exploitation Tradeoff Analysis

The adaptive algorithm in HalluHunter balances exploitation of identified weaknesses against exploration of new areas of the knowledge graph. As shown in Algorithm 1, this tradeoff is controlled by two hyperparameters: the exploration constant e, which determines how often the algorithm prioritizes low-accuracy relations, and the low-accuracy threshold a, which defines what counts as a weak relation.

To better characterize the exploration-exploitation (EE) dynamics, we conduct a sensitivity analysis by varying one hyperparameter at a time. Specifically, we vary $a \in \{0.3, 0.4, 0.5\}$ with $e = 0.2$ fixed, and vary $e \in \{0.1, 0.2, 0.3\}$ with $a = 0.4$ fixed. For each setting, we report the average accuracy and average coverage across question types and domains.

Sensitivity to the low-accuracy threshold a. Table 10 shows that a lower threshold ($a = 0.3$) induces more aggressive exploitation of weak relations. This yields slightly lower final accuracy, but also noticeably lower coverage, suggesting that the algorithm focuses too narrowly on a smaller subset of relations. In contrast, a larger threshold ($a = 0.5$) relaxes exploitation, resulting smaller initial accuracy drops and stabilizing at higher values, with a comparable coverage to $a = 0.4$. This highlights the tradeoff that stricter constants enhance short-term error exposure but risk coverage stagnation, while looser ones promote broader exploration at the cost of slower vulnerability detection.

Sensitivity to the exploration constant e. Table 11 shows that $e = 0.2$ produces the steepest and most consistent decline in accuracy, indicating the strongest exposure of factual weaknesses. By comparison, $e = 0.1$ and $e = 0.3$ lead to more modest drops. Although the final coverage of $e = 0.2$

Table 10: Sensitivity analysis for the low-accuracy threshold a with $e = 0.2$ fixed.

Trial	$a = 0.3$		$a = 0.4$		$a = 0.5$	
	Accuracy	Coverage	Accuracy	Coverage	Accuracy	Coverage
Seed	0.605	0.314	0.605	0.314	0.605	0.314
1	0.460	0.307	0.481	0.326	0.522	0.346
2	0.443	0.370	0.444	0.391	0.507	0.386
3	0.420	0.397	0.446	0.425	0.495	0.409
4	0.404	0.411	0.424	0.449	0.489	0.448
5	0.371	0.417	0.373	0.471	0.450	0.468

is similar to that of $e = 0.3$, the setting $e = 0.1$ exhibits less stable behavior across trials.

How many evaluation points are needed? At the scale used in our experiments, each domain-specific KG subset contains roughly 500k-600k triplets and 10k-12k entities, and each iteration generates 1,000 new questions per question type and domain. The results in Tables 10 and 11 show that the first two iterations quickly exploit initial weaknesses, producing sharp drops in accuracy but only modest coverage. Stopping too early therefore risks missing substantial vulnerabilities that emerge only after broader exploration.

In practice, we recommend a dynamic stopping criterion that they continue iterations until marginal growth in coverage diminishes or accuracy stabilizes per trial. This rule ensures efficient scaling, balances the EE tradeoff, and prevents unnecessary over-sampling.

K Statistical Analysis of Iterative Algorithm Effectiveness

To address concerns regarding the statistical rigor of HalluHunter 's evaluation and the need for a more measured narrative on its impact, we conducted additional statistical analysis to quantify the effectiveness of the iterative algorithm (Algorithm 1). Specifically, we performed linear regression on the average accuracy of the nine evaluated LLMs (GPT-3.5-Turbo, GPT-4-Turbo, GPT-4o, DeepSeek-V3, Claude-Sonnet-4, Claude-3.5-Haiku, Gemini-2.0, Qwen-3, Qwen-3-Reasoning) across six iterations (Trials 0–5, with Trial 0 as the initial random question generation). The analysis was separated for single-hop and multi-hop questions to assess the algorithm's ability to progressively uncover LLM vulnerabilities.

For single-hop questions, linear regression on the average accuracy across iterations yielded a p-value of 0.031, indicating a statistically significant negative trend (declining accuracy as iterations target model weaknesses). For multi-hop questions, the p-value was 0.01, further confirming the algorithm's effectiveness in exposing factual inaccuracies through iterative probing. These results, detailed in Sections J.2 and J.1, are supported by corresponding graphs in the revised paper, enhancing analytical transparency. Claims about HalluHunter 's impact are now confined to its demonstrated ability to iteratively uncover LLM vulnerabilities through automated question-answering probing, avoiding overgeneralization.

L Full Text of Human Annotation Instruction

L.1 Instructions for Question-Triplet Alignment (Section G.2)

Thank you for participating in this evaluation. Your task is to assess 200 questions generated from fact triplets to determine if they accurately reflect the triplet's semantic meaning. For each assigned question-triplet pair, review the triplet and question, then record a Yes (question aligns with the triplet's content and intent) or No (question misrepresents the triplet or deviates from instructions) in the provided options. Noted that all data will only be used as research purpose.

L.2 Instructions for Factual Error Validation (Section J.4)

Thank you for participating in this evaluation. Your task is to validate 100 failure cases identified by HalluHunter, where LLMs provided incorrect responses. For each question, triplet, and LLM response: (1) Answer using reliable internet sources. (2) Compare LLM response to the correct answer. (3) Discuss with annotators to resolve disagreements. (4) Annotate as valid error (incorrect LLM response) or false negative (correct LLM response). Data is used exclusively for research.

Table 11: Sensitivity analysis for the exploration constant e with $a = 0.4$ fixed.

Trial	$e = 0.1$		$e = 0.2$		$e = 0.3$	
	Accuracy	Coverage	Accuracy	Coverage	Accuracy	Coverage
Seed	0.605	0.314	0.605	0.314	0.605	0.314
1	0.450	0.348	0.481	0.326	0.498	0.315
2	0.436	0.380	0.444	0.391	0.468	0.378
3	0.462	0.408	0.446	0.425	0.435	0.413
4	0.455	0.452	0.424	0.449	0.429	0.435
5	0.430	0.460	0.373	0.471	0.412	0.472

M Ethical Statement

All participants were informed beforehand that their feedback would be used for research purposes and potentially published in this paper. No personal or private information was collected, and all data collected was anonymized to ensure privacy. Participation was voluntary, and the recruited volunteers were proficient in using LLMs to ensure informed contributions.

N Beyond Evaluation — Using HalluHunter to Improve LLM Accuracy

To complement our main study, we conducted a preliminary fine-tuning experiment using errors identified by HalluHunter. While not the focus of this work, the results indicate that these errors can be leveraged to improve factual accuracy in LLMs.

For fine-tuning the research models, we gathered 900 questions that were answered incorrectly by LLaMA-2-13B-Chat model and adopted a cross-format fine-tunning design. For example, if the model incorrectly answered an MC question about (Napoleon, native language, Corsican) during fine-tuning, we tested it with a Yes/No question on the same triplet. For the training set used for fine-tuning, we augmented it with the LIMA (Zhou et al., 2023) instruction tuning dataset so the model is able to answer questions in the expected format. We trained the LLaMA-2-13B-Chat model on 8 V100-32G GPUs using DeepSpeed Zero3, with a per-GPU batch size of 4, a learning rate of 2e-5, and a cosine learning rate schedule over 1 epoch. Post-fine-tuning, we re-evaluated the model with HalluHunter 's different format set.

The cross-format testing design provides evidence of factual learning rather than memorization of specific question-answer test cases. If the model merely memorized question-answer patterns, it would fail when tested with different question types on the same triplets. The consistent 30% improvement across different formats shows the model internalized the underlying knowledge, which is the expected and intended outcome of knowledge injection through fine-tuning on identified error cases. The result demonstrates that the factual errors identified by HalluHunter can substantially enhance the factual accuracy, resulting in a notable enhancement of 33.2% for this fine-tuning method, from 35.3% to 68.5%.

To examine whether this improvement comes at the expense of unrelated retained knowledge, we additionally evaluated the pre-SFT and post-SFT models on a held-out set of 500 randomly sampled Yes-No questions that were disjoint from the 900 questions used for fine-tuning. The two stages achieved nearly identical accuracies, 35.2% before fine-tuning and 35.5% after fine-tuning. This result suggests that, in our preliminary setting, the SFT procedure improved targeted factual errors without causing an observable degradation on unrelated knowledge of the same evaluation type.

We note, however, that this experiment is limited in scope: it only probes unrelated factual knowledge on a held-out Yes-No set and does not constitute a comprehensive evaluation of the model's broader general capabilities. A more complete study of trade-offs—including instruction following, reasoning, and other downstream abilities—is an important direction for future work.

Table 12: Factual accuracy of different LLMs on multi-hop questions.

LLM	Domain	Question Type	Hop			
			1-hop	2-hop	3-hop	4-hop
GPT-3.5	Hum.	Yes/No	72.2	68.7	54.9	51.4
		MC	68.6	64.2	46.0	42.4
	Soc. Sci.	Yes/No	74.1	66.2	53.9	50.0
		MC	67.0	50.3	39.2	43.8
	STEM	Yes/No	73.9	66.4	53.8	51.4
		MC	60.5	45.2	31.3	29.5
GPT-4o	Hum.	Yes/No	84.4	74.4	62.9	60.6
		MC	82.9	74.8	56.7	50.6
	Soc. Sci.	Yes/No	82.1	68.8	59.7	64.9
		MC	79.2	61.5	54.1	55.4
	STEM	Yes/No	79.4	70.5	57.0	56.4
		MC	72.6	49.6	34.6	33.4
GPT-4	Hum.	Yes/No	81.0	73.4	61.2	60.9
		MC	79.2	72.4	55.0	47.7
	Soc. Sci.	Yes/No	80.7	68.3	58.7	65.4
		MC	77.6	62.0	53.0	53.8
	STEM	Yes/No	81.0	69.2	56.5	53.9
		MC	70.9	50.2	32.3	31.3
DeepSeek-V3	Hum.	Yes/No	76.2	67.7	56.0	52.1
		MC	65.0	56.3	31.5	28.0
	Soc. Sci.	Yes/No	75.6	66.2	55.6	51.0
		MC	65.0	45.2	24.5	26.5
	STEM	Yes/No	77.2	62.4	53.0	51.5
		MC	55.5	26.9	14.1	16.0
Claude-3.5-Haiku	Hum.	Yes/No	76.9	70.5	60.4	55.6
		MC	71.2	55.7	39.7	33.6
	Soc. Sci.	Yes/No	78.9	55.4	60.0	60.0
		MC	69.9	43.8	36.0	34.9
	STEM	Yes/No	76.5	68.1	56.8	54.9
		MC	62.9	36.4	29.3	27.0
Claude-Sonnet-4.0	Hum.	Yes/No	78.1	70.2	60.1	52.7
		MC	75.3	68.5	33.4	19.3
	Soc. Sci.	Yes/No	79.0	70.2	59.9	55.9
		MC	74.3	56.8	31.4	22.6
	STEM	Yes/No	78.9	66.1	54.8	51.6
		MC	65.2	40.2	23.5	14.2
Gemini-2.0	Hum.	Yes/No	81.3	73.1	62.0	57.8
		MC	77.7	70.1	52.1	45.9
	Soc. Sci.	Yes/No	78.9	67.6	57.4	56.2
		MC	75.6	59.4	49.1	51.0
	STEM	Yes/No	76.0	68.5	55.1	54.4
		MC	69.6	46.5	35.2	32.1
Qwen-3	Hum.	Yes/No	74.4	69.3	59.0	55.6
		MC	67.4	63.4	45.1	42.0
	Soc. Sci.	Yes/No	72.9	65.5	59.8	56.2
		MC	66.0	51.6	43.4	43.1
	STEM	Yes/No	74.5	65.0	53.9	52.1
		MC	64.9	42.5	31.0	28.4
Qwen-3-Reasoning	Hum.	Yes/No	72.8	67.6	56.1	54.1
		MC	63.9	59.9	41.0	34.5
	Soc. Sci.	Yes/No	75.1	65.8	57.6	52.0
		MC	62.6	44.0	37.3	38.3
	STEM	Yes/No	76.6	61.2	52.9	51.6
		MC	61.2	41.3	30.1	28.3

Table 13: Factual accuracy of LLMs across multiple trials on single-hop questions.

LLM	Trial	Hum.			Soc. Sci.			STEM		
		Yes/No	MC	WH	Yes/No	MC	WH	Yes/No	MC	WH
GPT-3.5	Trial 0	72.2	68.6	45.3	74.1	67.0	44.3	73.9	60.5	35.3
	Trial 1	61.8	52.6	18.5	58.1	50.4	23.4	62.1	48.1	18.4
	Trial 2	57.5	49.6	12.5	52.8	44.2	21.1	55.1	44.4	15.8
	Trial 3	56.2	46.9	11.4	53.0	42.4	15.4	57.2	44.6	15.5
	Trial 4	55.3	45.5	9.8	51.9	39.9	14.4	55.4	42.4	11.1
	Trial 5	53.6	46.2	7.5	48.1	38.2	12.1	53.2	37.3	11.1
GPT-4o	Trial 0	84.4	82.9	50.8	82.1	79.2	51.4	79.4	72.6	38.4
	Trial 1	71.0	68.3	20.4	74.5	65.2	21.4	71.0	61.3	21.1
	Trial 2	66.2	58.9	15.0	68.1	54.1	16.5	65.7	57.4	15.2
	Trial 3	67.3	56.4	12.0	67.4	54.7	14.9	65.7	54.9	15.1
	Trial 4	65.7	51.7	8.8	63.2	50.8	11.4	61.0	52.0	10.8
	Trial 5	65.8	54.1	9.1	63.6	51.5	10.6	61.2	50.5	10.3
GPT-4	Trial 0	81.0	79.2	49.1	80.7	77.6	48.4	81.0	70.9	39.3
	Trial 1	69.0	67.1	25.4	63.4	57.2	24.2	61.9	54.1	18.4
	Trial 2	59.4	61.6	18.4	65.1	56.5	21.8	60.8	50.1	16.8
	Trial 3	61.3	61.1	16.0	59.2	49.6	20.8	61.9	48.2	14.6
	Trial 4	62.3	60.8	14.3	60.2	53.1	15.2	57.4	46.2	13.0
	Trial 5	61.0	57.5	14.4	57.4	49.1	14.5	55.0	45.8	9.8
DeepSeek-V3	Trial 0	76.2	65.0	51.7	75.6	65.0	44.9	77.2	55.5	39.0
	Trial 1	63.1	52.3	23.6	61.8	43.4	20.1	59.7	37.5	20.0
	Trial 2	57.8	49.9	16.7	56.6	43.8	18.8	58.8	32.1	14.2
	Trial 3	52.8	49.2	11.3	58.3	41.5	17.1	55.0	28.4	13.9
	Trial 4	55.6	45.9	11.5	50.5	38.0	14.4	50.8	30.0	12.2
	Trial 5	52.3	43.2	10.1	51.6	37.9	10.9	51.1	29.6	10.8
Claude-3.5-Haiku	Trial 0	76.9	71.2	45.5	78.9	69.9	45.4	76.5	62.9	34.1
	Trial 1	66.4	54.2	17.9	65.2	47.0	22.8	60.7	41.3	18.7
	Trial 2	62.1	49.1	15.3	60.6	42.7	18.6	57.4	40.0	15.9
	Trial 3	61.1	48.1	13.7	60.1	40.5	19.0	58.3	38.1	15.3
	Trial 4	57.4	45.4	11.8	59.4	37.0	16.1	54.4	34.7	13.0
	Trial 5	57.6	44.9	11.5	57.8	38.2	16.3	56.1	36.4	12.3
Claude-Sonnet-4.0	Trial 0	78.1	75.3	52.8	79.0	74.3	50.2	78.9	65.2	44.3
	Trial 1	64.8	66.4	23.1	58.3	59.3	25.8	56.7	45.6	21.6
	Trial 2	63.0	62.8	16.2	52.7	56.7	21.6	56.6	34.4	15.9
	Trial 3	60.7	59.5	14.6	49.8	55.4	21.6	52.5	30.8	16.7
	Trial 4	61.0	58.6	14.0	48.6	47.6	17.4	54.4	26.0	13.5
	Trial 5	55.0	57.1	10.4	44.9	42.5	16.0	53.1	24.7	13.4
Gemini-2.0	Trial 0	81.3	77.7	48.8	78.9	75.6	50.3	76.0	69.6	37.7
	Trial 1	65.7	63.4	21.2	62.8	64.4	24.3	61.3	52.0	20.1
	Trial 2	63.5	61.1	15.3	63.4	62.1	23.3	59.9	47.7	16.1
	Trial 3	62.7	60.5	12.0	62.4	57.8	22.2	59.0	43.5	14.7
	Trial 4	60.9	60.2	12.4	60.5	58.2	20.3	55.6	42.6	15.9
	Trial 5	62.4	55.2	11.6	59.7	50.0	23.7	56.5	41.6	11.3
Qwen-3	Trial 0	74.4	67.4	42.0	72.9	66.0	41.2	74.5	64.9	35.7
	Trial 1	66.8	52.3	18.9	62.3	52.4	18.1	63.2	47.8	16.2
	Trial 2	61.9	51.6	18.0	57.3	46.2	18.9	60.3	42.8	18.1
	Trial 3	60.1	47.7	14.9	57.9	43.9	14.1	58.1	40.6	13.2
	Trial 4	58.6	49.5	12.4	53.3	40.5	15.2	55.2	41.5	13.5
	Trial 5	59.0	42.6	13.8	55.3	38.4	14.1	54.4	39.1	13.8
Qwen-3-Reasoning	Trial 0	72.8	63.9	41.8	75.1	62.6	37.2	76.6	61.2	33.1
	Trial 1	58.8	51.1	18.5	60.9	47.2	15.3	60.6	42.1	15.6
	Trial 2	57.3	49.1	15.1	58.9	45.5	14.9	54.0	42.2	11.5
	Trial 3	56.3	45.7	14.7	54.6	42.1	12.9	58.9	37.4	13.8
	Trial 4	58.7	45.7	12.5	52.1	40.5	12.0	54.0	35.7	12.6
	Trial 5	55.7	44.5	12.2	53.9	37.1	11.6	52.5	35.6	10.7

Table 14: The factual accuracy of LLMs across multiple trials on 2-hop questions.

LLM	Trial	Hum.		Soc. Sci.		STEM	
		Yes/No	MC	Yes/No	MC	Yes/No	MC
GPT-3.5	Trial 0	68.7	64.2	66.2	50.3	66.4	45.2
	Trial 1	54.5	47.5	54.8	37.7	53.7	34.0
	Trial 2	52.4	36.5	54.0	35.1	51.2	33.1
	Trial 3	52.3	30.1	53.0	32.5	50.4	29.0
	Trial 4	51.4	27.8	51.9	30.6	48.3	26.2
	Trial 5	51.1	26.4	51.0	29.4	44.7	23.8
GPT-4o	Trial 0	74.4	74.8	68.8	61.5	70.5	49.6
	Trial 1	62.6	52.1	56.9	44.5	58.7	42.3
	Trial 2	60.4	45.9	55.2	40.4	56.0	35.5
	Trial 3	58.5	41.5	54.5	37.5	54.4	32.5
	Trial 4	57.5	38.0	53.3	34.8	52.3	30.1
	Trial 5	57.1	35.6	53.0	33.1	51.7	28.3
GPT-4	Trial 0	73.4	72.4	68.3	62.0	69.2	50.2
	Trial 1	62.2	55.7	57.9	49.8	54.8	43.7
	Trial 2	58.4	47.2	59.0	54.2	53.7	37.3
	Trial 3	56.4	41.5	56.8	53.8	53.0	34.7
	Trial 4	56.4	38.2	55.3	53.4	51.9	32.1
	Trial 5	55.8	35.9	54.7	52.9	51.1	30.3
DeepSeek-V3	Seed	67.7	56.3	66.2	45.2	62.4	26.9
	Trial 1	54.7	48.6	53.9	38.4	54.2	26.1
	Trial 2	53.2	44.4	51.9	35.3	51.6	36.8
	Trial 3	52.1	38.8	51.1	32.4	51.3	35.4
	Trial 4	51.7	35.5	50.5	30.8	48.7	34.2
	Trial 5	51.5	33.2	50.0	28.7	47.2	33.1
Claude-3.5-Haiku	Trial 0	70.5	55.7	55.4	43.8	68.1	36.4
	Trial 1	59.5	38.8	54.9	40.6	57.7	37.3
	Trial 2	56.7	38.3	54.3	37.9	54.1	32.3
	Trial 3	56.0	34.5	53.5	35.1	53.7	30.0
	Trial 4	54.7	32.8	53.1	32.5	52.0	28.9
	Trial 5	53.6	31.8	52.6	31.6	50.2	28.0
Claude-Sonnet-4.0	Trial 0	70.2	68.5	70.2	56.8	66.1	40.2
	Trial 1	55.7	48.2	54.5	40.5	52.0	39.9
	Trial 2	54.4	36.6	53.8	30.8	50.2	34.8
	Trial 3	54.4	31.0	53.6	25.2	49.1	31.0
	Trial 4	53.5	26.8	52.9	21.9	47.9	28.2
	Trial 5	52.6	24.0	52.5	19.7	47.7	26.2
Gemini-2.0	Trial 0	73.1	70.1	67.6	59.4	68.5	46.5
	Trial 1	60.3	50.6	52.8	42.9	52.4	41.8
	Trial 2	58.4	45.3	51.6	37.3	50.9	37.4
	Trial 3	56.5	40.4	51.7	34.7	50.5	33.8
	Trial 4	55.9	37.4	50.7	32.3	49.7	30.3
	Trial 5	55.3	35.1	50.5	30.5	48.9	27.8
Qwen-3	Trial 0	69.3	63.4	65.5	51.6	65.0	42.5
	Trial 1	59.3	49.3	54.5	43.8	53.5	39.8
	Trial 2	55.6	36.8	52.7	36.0	43.9	23.6
	Trial 3	54.8	31.0	53.2	31.1	33.2	23.5
	Trial 4	54.0	30.6	50.0	28.7	24.2	18.5
	Trial 5	53.7	30.5	47.3	26.3	42.8	16.9
Qwen-3-Reasoning	Trial 0	67.6	59.9	65.8	44.0	61.2	41.3
	Trial 1	56.8	44.5	53.3	39.5	55.5	42.5
	Trial 2	57.1	35.7	52.2	29.5	49.4	26.2
	Trial 3	54.5	31.3	51.8	29.4	52.2	25.9
	Trial 4	52.9	31.5	51.2	25.9	45.5	20.2
	Trial 5	54.9	25.6	51.0	25.5	43.9	18.3

Table 15: Coverage scores of LLMs across multiple trials on single-hop questions.

LLM	Trial	Hum.			Soc. Sci.			STEM		
		Yes/No	MC	WH	Yes/No	MC	WH	Yes/No	MC	WH
GPT-3.5	Trial 0	0.357	0.360	0.359	0.395	0.391	0.400	0.312	0.306	0.254
	Trial 1	0.358	0.374	0.351	0.416	0.404	0.386	0.306	0.319	0.304
	Trial 2	0.406	0.431	0.392	0.461	0.434	0.416	0.357	0.379	0.345
	Trial 3	0.438	0.467	0.426	0.487	0.461	0.441	0.389	0.408	0.375
	Trial 4	0.464	0.489	0.449	0.518	0.493	0.466	0.417	0.429	0.397
	Trial 5	0.495	0.512	0.473	0.541	0.524	0.486	0.439	0.447	0.414
GPT-4o	Trial 0	0.357	0.360	0.359	0.395	0.391	0.412	0.312	0.306	0.254
	Trial 1	0.379	0.385	0.368	0.431	0.407	0.421	0.325	0.307	0.327
	Trial 2	0.432	0.435	0.418	0.497	0.463	0.473	0.371	0.354	0.383
	Trial 3	0.471	0.474	0.446	0.525	0.506	0.500	0.412	0.401	0.413
	Trial 4	0.500	0.498	0.466	0.546	0.529	0.531	0.453	0.430	0.438
	Trial 5	0.521	0.521	0.488	0.566	0.545	0.550	0.472	0.457	0.458
GPT-4	Trial 0	0.356	0.360	0.359	0.395	0.391	0.412	0.312	0.306	0.254
	Trial 1	0.369	0.378	0.365	0.450	0.436	0.397	0.327	0.354	0.338
	Trial 2	0.422	0.414	0.413	0.489	0.473	0.424	0.382	0.385	0.393
	Trial 3	0.448	0.439	0.441	0.513	0.506	0.445	0.413	0.404	0.421
	Trial 4	0.474	0.453	0.458	0.531	0.521	0.462	0.431	0.424	0.444
	Trial 5	0.495	0.470	0.479	0.542	0.535	0.480	0.449	0.439	0.459
DeepSeek-V3	Trial 0	0.356	0.360	0.359	0.395	0.391	0.412	0.312	0.306	0.254
	Trial 1	0.374	0.370	0.360	0.426	0.415	0.379	0.277	0.363	0.332
	Trial 2	0.417	0.409	0.401	0.458	0.455	0.405	0.343	0.396	0.383
	Trial 3	0.454	0.433	0.436	0.483	0.487	0.423	0.400	0.415	0.413
	Trial 4	0.480	0.453	0.460	0.502	0.505	0.442	0.428	0.434	0.433
	Trial 5	0.504	0.472	0.482	0.520	0.520	0.457	0.453	0.449	0.448
Claude-3.5-Haiku	Trial 0	0.356	0.360	0.359	0.395	0.391	0.412	0.312	0.306	0.254
	Trial 1	0.366	0.358	0.355	0.413	0.441	0.387	0.327	0.334	0.339
	Trial 2	0.403	0.394	0.395	0.449	0.479	0.423	0.377	0.367	0.379
	Trial 3	0.435	0.415	0.420	0.470	0.504	0.454	0.412	0.393	0.405
	Trial 4	0.447	0.435	0.436	0.487	0.521	0.472	0.429	0.415	0.427
	Trial 5	0.458	0.451	0.456	0.503	0.533	0.488	0.451	0.430	0.444
Claude-Sonnet-4.0	Trial 0	0.356	0.360	0.359	0.395	0.391	0.412	0.312	0.306	0.254
	Trial 1	0.366	0.355	0.352	0.400	0.418	0.400	0.317	0.323	0.335
	Trial 2	0.423	0.412	0.396	0.443	0.463	0.430	0.357	0.367	0.370
	Trial 3	0.451	0.445	0.430	0.469	0.490	0.452	0.384	0.389	0.396
	Trial 4	0.474	0.469	0.454	0.485	0.511	0.476	0.404	0.406	0.416
	Trial 5	0.492	0.487	0.474	0.498	0.528	0.490	0.422	0.417	0.434
Gemini-2.0	Trial 0	0.356	0.360	0.359	0.395	0.391	0.412	0.312	0.306	0.254
	Trial 1	0.382	0.369	0.355	0.443	0.414	0.405	0.345	0.338	0.302
	Trial 2	0.422	0.413	0.393	0.486	0.461	0.435	0.381	0.380	0.349
	Trial 3	0.449	0.434	0.423	0.514	0.486	0.463	0.405	0.409	0.378
	Trial 4	0.467	0.452	0.446	0.531	0.512	0.483	0.425	0.435	0.397
	Trial 5	0.482	0.471	0.465	0.541	0.534	0.501	0.441	0.449	0.414
Qwen-3	Trial 0	0.356	0.360	0.359	0.395	0.391	0.412	0.312	0.306	0.254
	Trial 1	0.362	0.367	0.360	0.445	0.402	0.381	0.312	0.346	0.290
	Trial 2	0.397	0.404	0.400	0.482	0.436	0.410	0.364	0.376	0.330
	Trial 3	0.428	0.440	0.427	0.503	0.460	0.429	0.394	0.400	0.362
	Trial 4	0.454	0.459	0.453	0.521	0.470	0.445	0.418	0.420	0.436
	Trial 5	0.473	0.475	0.474	0.537	0.478	0.455	0.437	0.441	0.457
Qwen-3-Reasoning	Trial 0	0.356	0.360	0.359	0.395	0.391	0.412	0.312	0.306	0.254
	Trial 1	0.372	0.355	0.353	0.409	0.411	0.383	0.327	0.342	0.292
	Trial 2	0.412	0.395	0.387	0.457	0.446	0.411	0.367	0.388	0.333
	Trial 3	0.446	0.423	0.417	0.477	0.483	0.432	0.393	0.412	0.360
	Trial 4	0.478	0.445	0.442	0.500	0.502	0.449	0.415	0.434	0.381
	Trial 5	0.502	0.464	0.464	0.521	0.512	0.465	0.437	0.449	0.403