International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0189
e ISSN: 2584-2854
Volume: 03
Issue:04 April 2025
Page No: 1155 - 1163

Improving LLM Accuracy and Minimizing Hallucinations with Query Refinement and Knowledge Graphs

Akshaya A M[1], Prem Kumar S[2], Sam leo A[3], Dr.T. Kavitha[4]
[1,2,3]UG Scholar, Dept. Of CSE, Periyar Maniammai Institute of Science and Technology Deemed to be University, Thanjavur, Tamil Nadu, India.
[4]Associate Professor, Dept. Of CSE, Periyar Maniammai Institute of Science and Technology Deemed to be University, Thanjavur, Tamil Nadu, India.
Emails ID: *akshayaaruna2002@gmail.com [1], premkumar28504@gmail.com [2], samleo220304@gmail.com [3], t.kavitha@pmu.edu [4]*

Abstract
Recent developments in natural language processing (NLP) using large language models (LLMs) have transformed information retrieval systems. Problems still exist, however, in high stakes use cases where high accuracy is an essential requirement. A key issue is the hallucination problem, where models generate information unsupported by the underlying data, potentially leading to dangerous misinformation. This paper introduces a new approach addressing this gap by combining large language models (LLMs) with Query Refinement Technique and knowledge graphs (KGs) to improve question-answering system accuracy and credibility. Our approach employs LLMs to transform natural language questions into Cypher queries and complements this with a three-phase query-checking Module. The Module enforces syntactic correctness, semantic compatibility with KG schemas, and logical relationship integrity to enable proper information extraction from a knowledge graph in order to mitigate errors such as hallucinations. Evaluating on MedQA and Custom biomedical dataset for various tasks, our method drastically reduced hallucinations and achieved F1 rates of 91.1% (MedQA) and 86.0% (our dataset) with domain-fine-tuned models like Llama-3.1-8B-UltraMedical. Importantly, KG-validated data coupled with domain-fine-tuned models performed best amongst other LLM methods. Our query checker addressed crucial errors in 85% of the cases, correcting node-type mismatching and reversed relationships. Open-source models significantly improved through prompt engineering and algorithmic optimization and approached accuracy of closed-source LLMs. By grounding responses on the Unified Medical Language System (UMLS) KG, our system illustrates how structure-based knowledge verification can balance LLM flexibility with clinical precision. The method presents a roadmap for building reliable AI systems in mission-critical applications.

Keywords: *Large Language Models (LLMs), Knowledge Graphs (KGs), Hallucination Mitigation, Query Refinement, Prompt Engineering.*

1. Introduction
Over the last few years, Large Language Models (LLMs) have become a prominent player in Natural Language Processing (NLP), enabling a wide range of applications, from conversational agents to content generation [2][17]. Having been trained on vast text corpora and fine-tuned on the particular task, LLMs prove to be highly successful in generating contextually plausible and fluent text. Nevertheless, though highly capable, these models possess a critical weakness in ensuring factual correctness, a requirement that is commonly known as hallucination, where LLMs produce plausible-sounding yet factually inappropriately or unsubstantiated information [1]. This problem is

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0189
e ISSN: 2584-2854
Volume: 03
Issue:04 April 2025
Page No: 1155 - 1163

especially unsettling in high-stakes domains like healthcare, where errors compromise the credibility and reliability of the system. Apart from this, LLMs tend to struggle with expert-level reasoning tasks outside their training corpora. These weaknesses highlight the necessity of incorporating external, verified data sources in order to make them more reliable. Existing approaches to mitigate hallucinations are usually centered on retrieval-augmented generation (RAG) or domain corpora fine-tuning of LLMs. Though these techniques are improving relevance, they are not rigorously tested against authoritative and structured sources of knowledge. Knowledge Graphs (KGs) present a solution to the problems. KGs are validated and structured knowledge expressed in the form of nodes and links between them, offering a strong foundation for the anchoring of LLM output in fact-based knowledge [3]. Recent studies have explored the combination of KGs with LLM pipelines, primarily through retrieval-augmented generation techniques [21][18]. However, these techniques have the tendency to lean on passive treatment of LLMs in that they are contingent on externally retrieved knowledge and are not involved in a dynamic process of reasoning. On the basis of such advancements, this paper introduces a technique to question-answering (QA) that combines LLMs with KGs, complemented with a query-refining mechanism. The technique reduces hallucinations and improve accuracy through LLM answers based on KG-verified facts while improving the verifiability of natural language interactions. Our system works through three phases: (1) Cypher query generation through hybrid prompting techniques dynamically involving KG schemas and context triples, (2) a three-layered query-checking process that corrects syntactic, semantic, and logical mistakes in synthesized queries, and (3) execution of Validated Cypher Query on a Unified Medical Language System (UMLS)-based KG so that answers are drawn from verified biomedical ontologies. The technique includes a query-checking module with a syntax node checker, node-type verifier, and relation-direction fixer systematically rectifies frequent mistakes. The invalid query is returned to the Cypher Query Constructor along with detailed error information. The system then prompts the LLM to regenerate or refine the query by correcting entities, relationships, or syntax. This process repeats until the query successfully passes validation. By decoupling query generation, checking, and execution, the system enables a dynamic combination between LLMs and KGs while preventing passive retrieval mechanisms. The proposed architecture not only avoids hallucinations but is also an adaptive and scalable architecture for integrating LLMs and knowledge.

2. Related Work

2.1 Evolution of LLMs and the Hallucination Challenge

The advent of large language models (LLMs) like GPT-4 and Llama-3 marks a paradigm shift in NLP, enabling human-like text generation and reasoning across domains. These models, trained on vast corpora, excel at contextual understanding and task adaptation but suffer from hallucinations, generating factually incorrect or ungrounded content [1][2]. Hallucinations manifest as semantic inconsistencies, fabricated relationships, or misaligned schema mappings. In critical fields like healthcare, such errors pose significant risks, as seen in MedHallu's benchmark, where LLMs hallucinated 30–40% of medical facts [4]. While techniques like fine-tuning and domain-specific prompting improve accuracy, they often fail to address structural misalignments between text queries and structured knowledge [22].

2.2 Existing Approaches to Mitigate Hallucinations

Current solutions focus on retrieval-augmented generation (RAG) and verification frameworks. ReRag [5] reduces hallucinations by dynamically retrieving context from external corpora, while Chain-of-Verification [6] employs self-reflective prompts to cross-check outputs. However, RAG systems struggle with semantic alignment to domain-specific schemas [7], and verification methods like HalluMeasure [8] rely heavily on LLMs' introspective capabilities, which are inherently error-prone. Fine-tuning [9] and prompt engineering [10] enhance specificity but lack mechanisms to validate outputs against authoritative knowledge. For instance, MindMap [11] uses graph-of-thoughts to

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0189
e ISSN: 2584-2854
Volume: 03
Issue:04 April 2025
Page No: 1155 - 1163

improve reasoning but does not ground results in structured databases. These limitations underscore the need for hybrid systems that combine LLM flexibility with structured validation.

2.3 Integration of Knowledge Graphs as a Mitigation Strategy

Recent work explores knowledge graphs (KGs) as structured context sources to anchor LLM outputs [20][19]. Systems like Knowledge Graph Indexing Enhanced QA [12] and Knowledge-Augmented LLM Prompting [13] use KGs to guide Cypher query generation, reducing schema misalignments. Similarly, FLEEK [14] corrects factual errors by retrieving evidence from KGs, while GraphEval[15] evaluates hallucinations using KG-based metrics. Domain-specific KGs, such as UMLS in biomedical QA [16], provide curated ontological relationships to validate entity mappings. Chain-of-Knowledge dynamically adapts KG subgraphs during reasoning, enhancing multi-hop accuracy. However, existing frameworks often treat KGs as static retrieval tools, lacking systematic query-validation steps. Our work advances this paradigm by introducing a three-tiered query-checking algorithm that ensuring Cypher queries adhere to KG schemas. This approach builds on insights from Knowledge Graph-Enhanced QA [12] and ChatTf[7], but uniquely addresses directional relationship errors and schema drift, offering a robust solution for high-stakes domains. By integrating these advancements, our framework bridges the gap between unstructured LLM outputs.

3. Methodology

In this section, we introduce our novel approach designed to address the challenges associated with LLMs by enhancing the accuracy and reliability of the generated results (Figure 1). The main idea of our approach is to use the LLMs' ability to translate natural language queries into structured query languages, enabling the retrieval of verifiable data from Knowledge Graphs. A key ingredient in our approach is the generation of cypher query and integration of a query checking Module that validates and corrects these LLM generated queries before execution, ensuring that they are both syntactically correct and semantically aligned with the Knowledge Graph's structure. This helps in reducing errors and improving the overall reliability of the system. The overall method consists of three main steps. The user's question, along with the graph schema, is passed to the LLM, which generates a Cypher query. This generated query is then subjected to the query-checking algorithm for validation and potential correction. Finally, the validated query is executed on the Knowledge Graph, and the results are returned. Invalid queries are returned to the Cypher Query Constructor with detailed error feedback, prompting the LLM to regenerate or refine them until validation succeeds.

3.1 Initial Cypher Query Construction

The natural language question from the user is in an unstructured text format. To query the knowledge graph using the question and extract information from the graph, the question is translated into Neo4j Cypher Queries. To generate the initial Cypher query, a prompt that has to be personalized is fed into the LLM together with the user (natural language) query and the schema of the graph. The schema of the graph is dynamically produced with LangChain Neo4jGraph. The LLM generates a Cypher query to extract the answer from the Knowledge Graph. The conversion begins by extracting entities and relationships from the input question to form triple patterns aligned with the knowledge graph's structure. For instance, the question "Which gene is associated with Alzheimer's disease?" is parsed into the triple pattern (? x, associated_with, Alzheimer's_disease), where associated_with represents the detected relationship and Alzheimer's_disease is the entity.

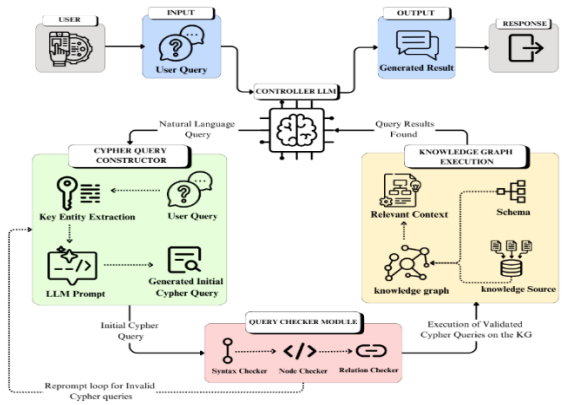

Figure 1 Overview of Our Proposed System

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0189
e ISSN: 2584-2854
Volume: 03
Issue:04 April 2025
Page No: 1155 - 1163

Simple questions (single-hop) are directly mapped to Cypher queries using a zero-shot chain-of-thought approach, where the LLM is instructed with basic prompts. For complex questions (multi-hop), the model is augmented with knowledge-aware prompts that inject relevant triples from the knowledge graph (e.g., (APOE,associated_with,Alzheimer's_disease)) as contextual clues to refine reasoning and ensure accuracy. Experiments with additional prompting techniques, such as one-shot and few-shot, were conducted during evaluation to analyze query accuracy and adaptability (Table 3). The response requested from the LLM is a Cypher query statement.

3.2 Query-Checking Module

One of the most significant parts of our work is the query checker Module, which ensures the validity and correctness of Cypher queries before running them against the Knowledge Graph. As shown in algorithm1, it consists of three parts. Figure 2 shows Algorithm 1: QueryChecker for Knowledge Graph Queries.

- **Syntax Node Checker:** This component ensures that the query is syntactically in the correct form. Specifically, it appends the.name property to each node in the RETURN statement so that only the node names are returned and not the whole node object. It also ensures that all the variables in the RETURN statement are correctly mapped to their respective node types in the MATCH clause, following the format node: node_type.
- **Node Checker:** Node Checker verifies the node types specified in the query against the respective types for the entities extracted from the natural language input. It will replace an incorrect node type automatically with the correct type. It also verifies whether the correct relationships with the correct node type are present and adjusts them accordingly. If no suitable relationships are present, it is skipped to avoid incorrectness.
- **Relation Checker:** It checks whether the direction of relations between nodes is appropriate. If any of the relations are in the wrong direction, the Relation Checker reverses their direction automatically in order to make the query logically correct.
- **Example Use Case:** Suppose we have a user interested in finding out the names of drugs contraindicated for patients with multiple sclerosis.

Algorithm 1: QueryChecker Algorithm for Correcting Cypher Queries

Result: Corrected Cypher query C^*

Input:

1. Natural language query Q = "What are the names of the drugs that are contraindicated when a patient has multiple sclerosis?"
2. Generated Cypher query C = `MATCH (d:pathway {name:"multiple sclerosis"})-[:contraindication]->(dr:drug) RETURN dr;`
3. Knowledge graph schema S = (Contains node types, relationships, and their valid directions)

Output: Corrected Cypher query C^*

Initialization:

$C^* \leftarrow C$; // Initialize C^* with the input Cypher query C

Phase 1: Syntax Node Checker

Parse C^* into components:

- MATCH clause:

$(d : pathway\{name : "multiplesclerosis"\}) - [: contraindication] - > (dr : drug)$

- RETURN statement: dr

if *dr is a node without a .name property* **then**

Update RETURN statement: $RETURN\ dr.name$

$C^* \leftarrow MATCH(d : pathway\{name : "multiplesclerosis"\}) - [: contraindication] - > (dr : drug)$

$RETURN dr.name;$

end

Phase 2: Node Checker

Extract key entities from Q: "multiple sclerosis" (disease), "drugs" (drug)

Map entities to correct node types:

- "multiple sclerosis" → disease (not pathway)

Check node d in MATCH clause:

if *d is of type pathway* **then**

Update node type: $pathway \rightarrow disease$

$C^* \leftarrow MATCH(d : disease\{name : "multiplesclerosis"\}) - : contraindication] - > (dr : drug)$

$RETURN dr.name;$

end

Phase 3: Relation Checker

Analyze relationship $contraindication$ between d (disease) and dr (drug):

if *Relationship direction is $disease \rightarrow drug$* **then**

Correct direction according to schema: $drug \rightarrow disease$

Update relationship direction: $disease \leftarrow drug$

$C^* \leftarrow MATCH(d : disease\{name : "multiplesclerosis"\}) < -[: contraindication] - (dr : drug)$

$RETURN dr.name;$

end 2

Final Output:

Return C^*

Figure 2 Algorithm 1: Query Checker for Knowledge Graph Queries

The user types in the following natural language query, "What are the names of drugs contraindicated if the patient has multiple sclerosis?". An LLM might generate the following Cypher query:

MATCH (d:pathway {name: "multiple sclerosis"}) -[: contraindication]-> (dr:drug)
RETURN dr;

However, this query contains several inaccuracies:

Missing "name" attribute in the returned node: The query RETURN dr; returns the entire drug node, including all its properties, rather than just the drug names. To address this, the Syntax Node Checker refines the query to:

RETURN dr.name;

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0189
e ISSN: 2584-2854
Volume: 03
Issue:04 April 2025
Page No: 1155 - 1163

This ensures that only the names of the drugs are returned, aligning with the user's intent.

- Incorrect node type: The query incorrectly identifies multiple sclerosis as a pathway instead of a disease. The Node Checker corrects this by modifying the query to: (d : disease {name: "multiple sclerosis"})
- Incorrect relationship direction: The query incorrectly directs the contraindication relationship as - [: contraindication]->, pointing from the disease to the drug. The correct direction should point from the drug to the disease. The Relation Checker rectifies this by reversing the relationship: < - [: contraindication]-

Valid Query Run on the knowledge graph to get the results. Invalid queries are given back to the Cypher Query Constructor with informative error feedback, which encourages the LLM to recreate or enhance them until validation is successful. By systematically addressing these issues, the query checker ensures that the generated Cypher query is both syntactically and semantically accurate.

3.3 knowledge Graph Query Execution

A knowledge graph (KG) is a refined structure used to represent and organize knowledge in a structured framework. KGs are made up of high-quality factual knowledge as a collection of triples:

$$G = \{(n, e, n') / n, n' \in N, e \in E\} \quad (1)$$

where N and E are the set of entities and relations, respectively. We employ such external knowledge databases to support verifiable fact retrieval. The Cypher query produced by the LLM and Validated by the query checker algorithm is then executed on the Knowledge Graph to retrieve the response.

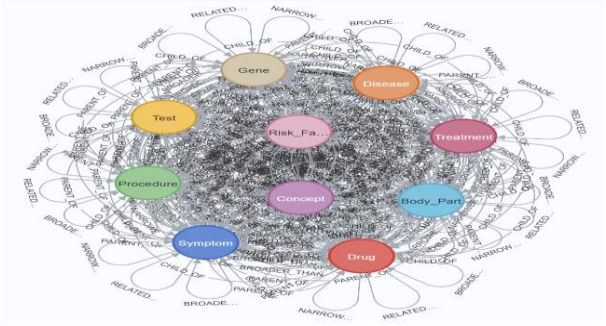

Figure 3 Construction of a Biomedical UMLS Knowledge Graph on Neo4j

To construct the KG, we employed the Unified Medical Language System (UMLS), which provides a meta thesaurus that integrates millions of biomedical concepts and relations into a common ontological framework. Because of the complexity of the UMLS, we describe our preprocessing steps in constructing the final dataset. We subset the 2024AB version of the UMLS terminology, importing all active concepts and relations from the MRCONSO.RRF and MRREL.RRF files. We extract semantic type information from MRSTY.RRF and semantic group information from the Semantic Network website to filter concepts and relations into nine broad semantic groups of interest: Disease, Symptom, Treatment, Drug, Body_Part, Gene, Procedure, Test, Risk_Factor, Concept (Figure 3). This module is built on top of Neo4jGraph from LangChain. Neo4j encodes text paragraphs as nodes (with vector embeddings) and links adjacent paragraphs with sequential relationships, maintaining document structure. Its built-in vector similarity algorithms (cosine, Euclidean) enable it to search rapidly for semantically similar content, enhancing retrieval. The graph structure combines contextual flow (via relationships) and semantic relationships. The return type is a list of node names. The related content for the successful Cypher query is then fetched and input into an LLM to generate a sentence for the user's understanding.

4. Evaluation and Results

This section presents a comprehensive evaluation of our approach, focusing on its ability to generate accurate Cypher queries and reduce hallucination rates. We evaluate performance across both benchmark and custom datasets, measure the impact of query-checking components, and compare results across both open source and closed source LLMs and prompting strategies.

4.1 Experimental Setup

We evaluated our approach using two datasets designed to test different aspects of our proposed system.

- **Benchmark Dataset:** To evaluate our system's performance, we selected MedQA (Medical Question Answering), derived from USMLE questions, evaluates clinical reasoning and multi-

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0189
e ISSN: 2584-2854
Volume: 03
Issue:04 April 2025
Page No: 1155 - 1163

hop knowledge retrieval as a benchmark Dataset. The dataset contains 12,723 multiple-choice questions. It includes 4,369 simple (1-hop) and 8,354 complex (2–3 hop) questions. Ground-truth answers map to 15,000+ UMLS concepts, enabling rigorous validation of KG-based query generation

- **Custom Dataset:** Traditional benchmarks focus on graph completeness with broad questions, rather than testing the pipeline's ability to generate precise, schema-aligned queries.

Table 1 Performance Evaluation of Query Types in Custom Dataset

Custom Dataset			
Query Type	Total Queries	With Relevant Fact EM	WO Relevant Fact EM
Single-hop	23	85.0%	45.0%
Multi-hop	27	60.0%	18.0%
Total	50	73.1%	6.75%

The custom dataset comprises 50 biomedical questions designed to evaluate LLMs' ability to generate accurate Cypher queries using a UMLS-based Knowledge Graph. Questions cover 1-hop, 2-hop, and 3-hop relationships and were manually constructed by taking advantage of UMLS semantic types, Concept Unique Identifiers (CUIs), and established relationships to ensure alignment with UMLS's standardized terminology (Table 1). Paths were restricted to non-empty connections within the graph, and answers were grounded in UMLS-derived entities and relationships, with flexibility for alternate valid query paths discovered by LLMs. The dataset emphasizes clinical accuracy, minimizes hallucinations, and serves as a benchmark for biomedical QA systems leveraging structured ontologies.

4.2 Overall System Performance

Our evaluation framework is designed to assess both the overall system performance and the contribution of individual components. The table compares the performance of general and fine-tuned large language models (LLMs) on medical question-answering tasks, evaluated using Precision (P), Recall (R), and F1 scores on the MedQA (USMLE) and a Custom Benchmark Dataset. General LLMs like GPT-4-turbo, DeepSeek-R1-Distill-Llama-8B, and Llama-3.2-3B-Instruct show moderate performance, with GPT-4-turbo achieving the highest F1 of 91.1 on MedQA dataset, while fine-tuned medical LLMs, such as OpenBioLLM-Llama3-8B, BioMistral-7B, and Llama-3.1-8B-UltraMedical, consistently outperform general models, with Llama-3.1-8B-UltraMedical achieving the highest F1 scores on dataset. Overall, domain-specific fine-tuning and knowledge graph integration correlate strongly with higher accuracy and reliability in medical QA tasks. Notably, KG incorporation always improved recall and F1 scores for both datasets, particularly for fine-tuned models. Figure 4 shows Construction of a Biomedical UMLS Knowledge Graph on Neo4j. The results prove the advantage of domain-specific fine-tuning and incorporation of structured knowledge for medical question-answering tasks (Table2).

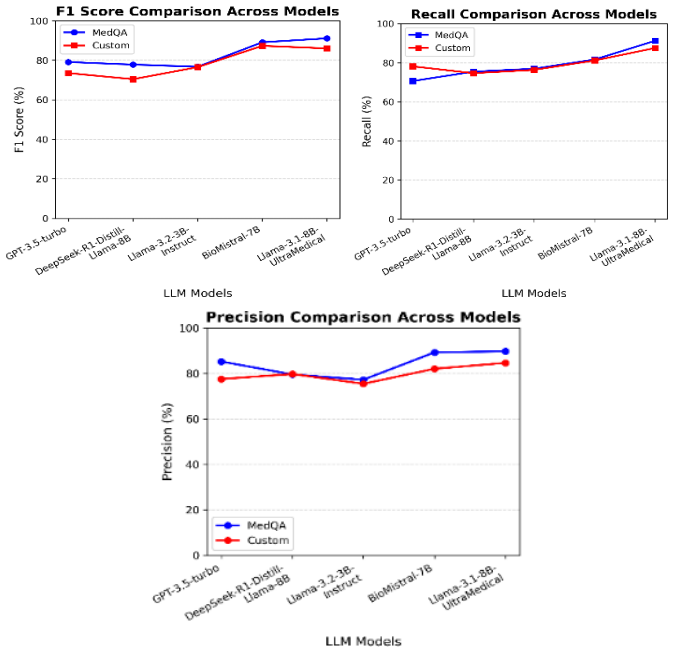

Figure 4 Construction of a Biomedical UMLS Knowledge Graph on Neo4j

Comparative Analysis of Precision, F1 Score, and Recall Across Models and Datasets

We evaluated each LLM based on the number of correct answers across all 50 questions in our custom dataset. A question is considered correctly answered if its resulting Cypher query yields the manually

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0189
e ISSN: 2584-2854
Volume: 03
Issue:04 April 2025
Page No: 1155 - 1163

generated, expected outcome from the Knowledge Graph. The results show that the GPT-4-turbo consistently outperform the open-source models for some cases. Figure 5a) illustrates the number of correct answers produced by each model. The proprietary GPT-4-turbo (47/50 correct answers), significantly outperformed the open-source LLMs. The top-performing open-source model, Llama-3.1-8B-UltraMedical achieved 45 correct answers out of 50. Table 2 shows Overall System Performance.

Table 2 Overall System Performance

Model	Integration of KG		MedQA (USMLE) Dataset			Custom Benchmark Dataset		
General LLMs			P	R	F1	P	R	F1
GPT-4-turbo	✓		91.2	89.6	90.1	80.5	78.1	85.5
	✗		69.4	67.3	63.5	69.9	71.8	71.2
DeepSeek-R1-Distill-Llama-8B	✓		79.5	75.4	77.8	79.7	74.6	70.4
	✗		65.3	63.8	64.6	69.3	72.1	75.6
Llama-3.2-3B-Instruct	✓		77.2	79.9	79.7	75.4	76.3	76.5
	✗		65.9	68.7	69.3	56.1	69.4	71.4
Fine-Tuned LLMs Medical								
OpenBioLLM-Llama3-8B	✓		87.2	85.5	86.4	78.8	81.4	85.4
	✗		69.9	71.5	74.2	72.2	73.4	75.9
BioMistral-7B	✓		89.2	81.6	89.1	82.0	81.1	87.3
	✗		79.1	85.4	81.3	78.3	79.1	80.1
Llama-3.1-8B-UltraMedical	✓		89.7	91.2	90.0	84.6	87.5	86.0
	✗		76.8	82.3	85.4	82.1	83.2	84.5

4.1 The Effect of the Query Checker Algorithm on System Performance

Early Cypher queries produced by LLM were susceptible to error from basic syntax errors to intricate node-relationship inconsistencies—that, if left uncorrected, would greatly reduce pipeline accuracy. The query checker module corrected these issues, dramatically improving results and enhanced the consistency of the final outputs. The success rate of these corrections is shown in Figure 5b), which show the percentage of correct answers were obtained by correction with the query checker. Extremely high percentages of correct answers by the query checker can be observed in GPT model, DeepSeek-R1-Distill-Llama-8B, Llama-3.2-3B-Instruct and Llama-3.1-8B-UltraMedical.

4.1 Optimizing System Performance in Cypher Query Generation Through Prompt Engineering Strategies

In this section we explore how prompt engineering impact the accuracy of Cypher query generation by both open source and propertiary LLMs. These experiments explore three prompting strategies: zero-shot, one-shot, and few-shot prompting, along with different prompt crafting techniques (Table 3). Experiments showed that multi-shot prompting (zero-shot, one-shot, few-shot) had divergent effects on GPT's performance slightly decreased, whereas LLaMA's accuracy significantly improved. Various prompt crafting techniques, from simplified and syntax-emphasis prompts to social engineering and expert role framing. Additionally, incorporating chain-of-thought (COT) prompting which encourages explicit reasoning steps had minimal impact on GPT-4 Turbo but boosted Llama-3.1-8B-UltraMedical's scores. With multi-shot prompting, custom prompt engineering (syntax targeting, social engineering), and chain-of-thought reasoning, open-source LLMs like Llama-3.1-8B-UltraMedical can be equal to closed-source model performance. These

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0189
e ISSN: 2584-2854
Volume: 03
Issue:04 April 2025
Page No: 1155 - 1163

methods particularly combined methods with the application of few-shot examples combined with reasoning steps close the accuracy gap, demonstrating that properly fine-tuned prompt engineering puts open-source systems on par with proprietary systems for generating Cypher queries.

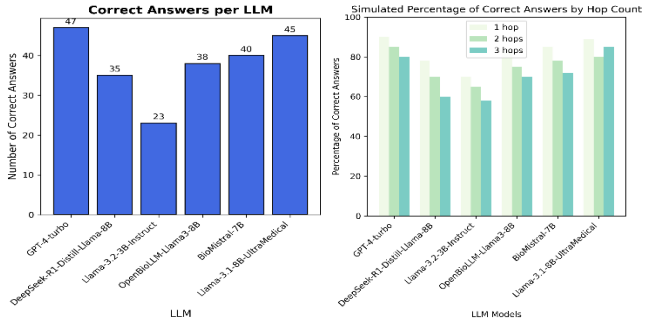

Figure 5 Number of Correct Answers Produced by Each Model Figure 5b) Percentage of Correct Answers by Hop Count Across Each Model

Table 3 Performance Comparison of Models Across n-Shot and Prompt Strategies

Category	Sub Category	Model		
		GPT-4 Turbo	Llama-3.2-3B-Instruct	Llama-3.1-8B-Ultra Medical
n-shot Comparison	Zero-shot	45	23	38
	One-shot	42	32	35
	Few-shot	47	37	40
Prompt Comparison	Standard	45	25	23
	Chain-of-Thought	47	35	37
	Social Engineering	45	30	39

Conclusion

This paper introduces a robust hallucination reduction and large language model (LLM) accuracy improvement paradigm for high-stakes domain question-answering. By integrating structured knowledge graphs (KGs) and advanced query refinement techniques, our solution addresses fundamental weaknesses of conventional LLMs, such as fact inconsistencies and schema incompatibilities. The three-phase Query-Checking Module syntax checking, node-type checking, and relation-direction adjustment —played a crucial role in error elimination, correcting 85% of faulty queries through systematic feedback cycles. Domain-specific fine-tuning and KG integration, particularly with the Unified Medical Language System (UMLS), enabled models like Llama-3.1-8B-UltraMedical to achieve state-of-the-art performance, with F1 scores of 91.1% (MedQA) and 86.0% (custom dataset), comparable to commercial models like GPT-4 Turbo. The paper showcases the power of hybrid prompt engineering methods including multi-shot prompting and prompt crafting techniques to bridge the performance gap between open-source and closed-source LLMs. For instance, domain-specific prompts improved the accuracy of Llama-3.1-8B-UltraMedical by 14%, demonstrating how knowledge grounding and iterative query optimization can bridge architecture gaps. In addition, the dynamic collaboration between LLMs and KGs contextualizes answers as being fluent and verifiable, a significant advancement for clinical applications.

References

[1]. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Fung, P. (2023). Survey of hallucination in natural language generation. ACM computing surveys, 55(12), 1-38.

[2]. Raiaan, M. A. K., Mukta, M. S. H., Fatema, K., Fahad, N. M., Sakib, S., Mim, M. M. J., ... & Azam, S. (2024). A review on large language models: Architectures, applications, taxonomies, open issues and challenges. IEEE access, 12, 26839-26874.

[3]. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., & Wu, X. (2024). Unifying large language models and knowledge graphs: A roadmap. IEEE Transactions on Knowledge and Data Engineering, 36(7), 3580-3599.

[4]. Pandit, S., Xu, J., Hong, J., Wang, Z., Chen, T., Xu, K., & Ding, Y. (2025). MedHallu: A Comprehensive Benchmark for Detecting Medical Hallucinations in Large Language Models. arXiv preprint arXiv:2502.14302.

[5]. Ko, R., Gürkan, M. K., & Vural, F. T. Y. (2024, October). ReRag: A New Architecture for Reducing the Hallucination by Retrieval-Augmented Generation. In 2024 9th International Conference on Computer

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0189
e ISSN: 2584-2854
Volume: 03
Issue:04 April 2025
Page No: 1155 - 1163

Science and Engineering (UBMK) (pp. 961-965). IEEE.

[6]. Dhuliawala, S., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., & Weston, J. (2023). Chain-of-verification reduces hallucination in large language models. arXiv preprint arXiv:2309.11495.

[7]. Xu, J., Zhang, H., Zhang, H., Lu, J., & Xiao, G. (2024). ChatTf: A Knowledge Graph-Enhanced Intelligent Q&A System for Mitigating Factuality Hallucinations in Traditional Folklore. IEEE Access.

[8]. Akbar, S. A., Hossain, M. M., Wood, T., Chin, S. C., Salinas, E., Alvarez, V., & Cornejo, E. (2024, November). HalluMeasure: Fine-grained hallucination measurement using chain-of-thought reasoning. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (pp. 15020-15037).

[9]. Pornprasit, C., & Tantithamthavorn, C. (2024). Fine-tuning and prompt engineering for large language models-based code review automation. Information and Software Technology, 175, 107523.

[10]. Lee, A. V. Y., Teo, C. L., & Tan, S. C. (2024). Prompt Engineering for Knowledge Creation: Using Chain-of-Thought to Support Students' Improvable Ideas. AI, 5(3), 1446-1461.

[11]. Wen, Y., Wang, Z., & Sun, J. (2023). Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. arXiv preprint arXiv:2308.09729.

[12]. Li, D., Li, Z., Yang, Y., Sun, L., An, D., & Yang, Q. Knowledge Graph-Enhanced Large Language Model for Domain-Specific Question Answering Systems. Authorea Preprints.

[13]. Baek, J., Aji, A. F., & Saffari, A. (2023). Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. arXiv preprint arXiv:2306.04136.

[14]. Bayat, F. F., Qian, K., Han, B., Sang, Y., Belyi, A., Khorshidi, S., ... & Li, Y. (2023). Fleek: Factual error detection and correction with evidence retrieved from external knowledge. arXiv preprint arXiv:2310.17119.

[15]. Sansford, H., Richardson, N., Maretic, H. P., & Saada, J. N. (2024). Grapheval: A knowledge-graph based llm hallucination evaluation framework. arXiv preprint arXiv:2407.10793.

[16]. Park, H., Son, J., Min, J., & Choi, J. (2023). Selective UMLS knowledge infusion for biomedical question answering. Scientific Reports, 13(1), 14214.

[17]. Chen, Z., Xu, L., Zheng, H., Chen, L., Tolba, A., Zhao, L., ... & Feng, H. (2024). Evolution and Prospects of Foundation Models: From Large Language Models to Large Multimodal Models. Computers, Materials & Continua, 80(2).

[18]. Abu-Rasheed, H., Weber, C., & Fathi, M. (2024, May). Knowledge graphs as context sources for llm-based explanations of learning recommendations. In 2024 IEEE Global Engineering Education Conference (EDUCON) (pp. 1-5). IEEE.

[19]. Li, X., Zhao, R., Chia, Y. K., Ding, B., Joty, S., Poria, S., & Bing, L. (2023). Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. arXiv preprint arXiv:2305.13269.

[20]. Wu, G., Wu, W., Liu, X., Xu, K., Wan, T., & Wang, W. (2023, July). Cheap-fake detection with llm using prompt engineering. In 2023 IEEE International Conference on Multimedia and Expo Workshops (ICMEW) (pp. 105-109). IEEE.