Vol. 06, No. 02, pp. 342 –357 (2025)
ISSN: 2708-0757

JOURNAL OF APPLIED SCIENCE AND TECHNOLOGY TRENDS

www.jastt.org

A Hybrid LLM–Knowledge Graph Framework for Accurate Biomedical Question Answering

Havraz Y. Omar[1,2]*, Abdulhakeem O. Mohammed[3]

[1]Department of Information Technology, Technical College of Duhok, Duhok Polytechnic University, Duhok, Kurdistan Region, Iraq. havraz.omar@dpu.edu.krd
[2] Department of Information Technology, Technical College of Informatics – Akre, Akre University for Applied Sciences, Akre, Kurdistan Region, Iraq.
[3] Department of Computer Science, College of Science, University of Zakho, Zakho, Kurdistan Region, Iraq.
a.mohammed@uoz.edu.krd

*Correspondence: havraz.omar@dpu.edu.krd

Abstract

Biomedical question answering requires accurate and interpretable systems; however, existing approaches often face challenges such as language model hallucinations and limited reasoning when relying solely on standalone knowledge graphs. To address these limitations, this study proposes a hybrid framework that integrates the LLaMA-3B language model with a Neo4j-based drug–disease–symptom knowledge graph. The system translates natural language questions into executable Cypher queries, operates on an iBKH-derived graph comprising over 65,000 entities and 3 million relationships, and returns answers with supporting evidence through a transparent interface. Experiments conducted on 60 biomedical questions across three levels of difficulty demonstrate the robustness of the approach: 96% exact match for simple queries, 95% for medium queries, and 86.7% for complex queries. Overall, the system achieves Precision@5 of 96.1%, Recall@5 of 89.0%, F1@5 of 91.0%, Hits@k of 96.1%, and an MRR of 94.4%, while maintaining an average response time of only 6.07 seconds. These results indicate that the system retrieves nearly all relevant answers, ranks them correctly, and delivers them with latency low enough for interactive use. Moreover, unlike cloud-based APIs such as ChatGPT, which require internet connectivity and external data transmission, the proposed framework operates fully offline, ensuring privacy, reproducibility, and compliance with biomedical data governance. Overall, this pipeline provides an accurate, efficient, and privacy-preserving solution for biomedical question answering, making it a practical alternative to cloud-dependent approaches in sensitive healthcare contexts.

Keywords: *Knowledge Graph, LLM, Question Answering, Neo4j, Biomedical Informatics, Healthcare AI, LLaMA 3.*

Received: August 14th, 2025 / Revised: October 10th, 2025 / Accepted: October 16th, 2025 / Online: October 20th, 2025

I. INTRODUCTION

Answering questions in the biomedical field is a difficult task due to the complexity of medical knowledge and the need for precision. In recent years, large language models (LLMs) like LLaMA, GPT-4 have made progress in understanding and generating human-like responses to medical questions [1, 2]. These models can process large amounts of information and respond in natural language, which makes them helpful in healthcare settings [3]. However, they often struggle to provide accurate answers when dealing with specialized biomedical content [4, 5].

One major issue with LLMs is a problem called hallucination, where the model generates information that sounds right but is actually incorrect or unsupported [6]. In medical applications, this can be dangerous, as healthcare professionals rely on precise and trustworthy information [7]. Therefore, researchers are exploring ways to combine LLMs with structured sources of knowledge to improve their reliability [8].

LLM-only systems in biomedicine still hallucinate and are hard to verify, limiting safe use [9, 10]. Biomedical knowledge graphs (BKGs) such as iBKH and SPOKE curate multi-source facts and enable multi-hop reasoning, yet they neither interpret free text nor generate answers [11, 12]. Recent hybrids (KG-aware RAG) improve grounding but often lack explicit path-level justifications and robust end-to-end answer evaluation [13, 14].

Recent studies have increasingly integrated Knowledge Graphs (KGs) with Large Language Models (LLMs) to improve factual accuracy, reasoning, and reduce hallucinations. Notable examples include DR.KNOWS, which combines UMLS-based KGs with LLMs for better diagnostic reasoning [15], KnowNet

doi: 10.38094/jastt62404

for visualizing and validating LLM outputs [16], and MedKA for KG-enhanced question answering [17].

To address these challenges, several recent works have explored the integration of large language models with biomedical knowledge graphs (KGs). A biomedical KG is a structured network that connects entities such as diseases, drugs, and symptoms using defined relationships [18, 19]. These graphs store verified medical knowledge from trusted databases, allowing for more accurate and explainable responses [12]. KGs are especially useful in multi-step reasoning tasks, where finding an answer requires connecting different pieces of information [20]. These entities and relationships can be visually represented in a biomedical knowledge graph, as shown in Fig. 1, where nodes represent medical concepts such as drugs, diseases, symptoms, and pathways, and edges denote their semantic relationships.

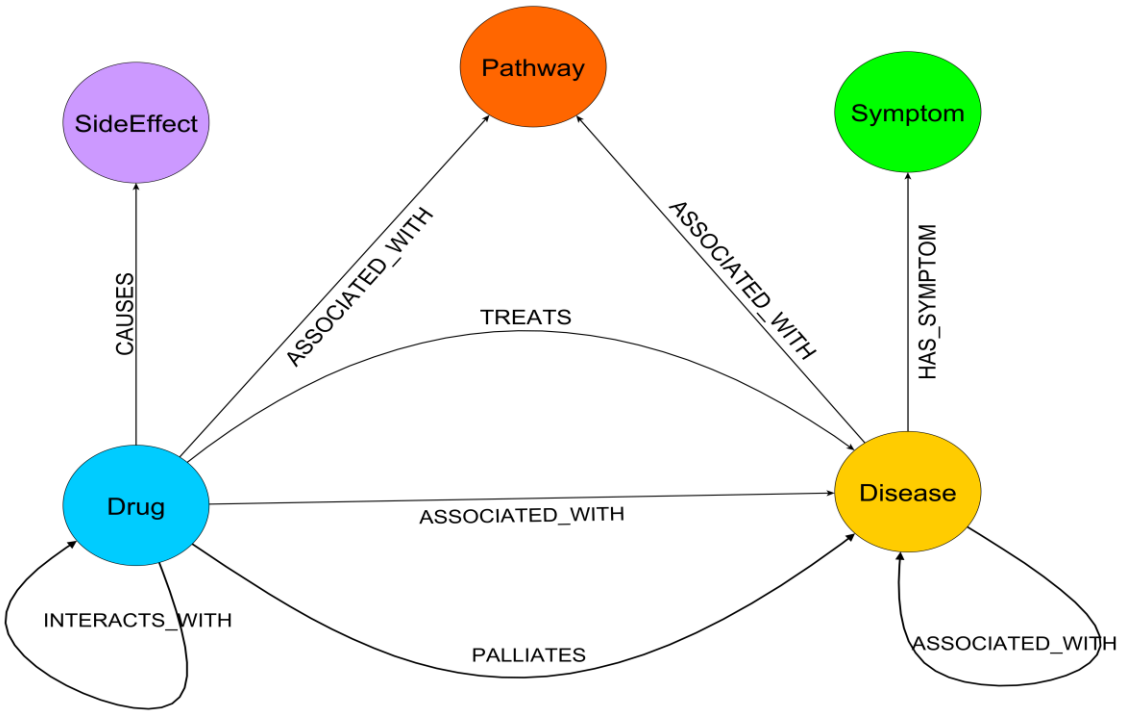

Fig. 1. Overview of Biomedical Knowledge Graph Entities and Relationships

One example of a widely used biomedical KG is SPOKE, which includes millions of nodes and relationships from over 40 biomedical databases [12]. Integrating an LLM with a KG allows the strengths of both technologies to work together: the LLM provides language understanding, and the KG provides structured, factual knowledge [21, 22]. A common method is retrieval-augmented generation (RAG), where the LLM retrieves information from the KG and uses it to generate more accurate responses [13, 23]. In more advanced setups, the LLM can even generate queries like Cypher to fetch specific data from the graph [24, 25]. Neo4j is a popular graph database that supports fast and flexible storage and querying of knowledge graphs using Cypher [21]. It is well-suited for biomedical applications because it allows easy exploration of complex medical relationships. Recent work has shown that combining Neo4j with LLMs can lead to better accuracy, fewer hallucinations, and more explainable results [24, 26].

Despite improvements, building a reliable hybrid system that combines an LLM with a biomedical KG remains a technical challenge. Some approaches require complex pipelines or large training datasets, while others rely on fine-tuning specific to a narrow set of questions [27, 28]. There is still a need for systems that are both accurate and easy to scale, especially in domains like medicine where the cost of errors is high [22].

Recent advances in KG-augmented LLMs have improved performance, yet biomedical QA continues to face three practical gaps:

1. **Traceability:** LLM-only or text-retrieval-only pipelines rarely provide *graph-grounded* justifications; users lack the ability to inspect the exact nodes and edges that support an answer.
2. **Evaluation:** Prior work often judges quality via surface-form checks (e.g., matching a Cypher template), which fails to capture *end-to-end answer correctness* or ranking quality across different difficulty levels.
3. **Deployment:** Many solutions assume cloud resources or domain-specific fine-tuning, yet biomedical contexts typically demand a *local, privacy-preserving* system with low latency and reproducible behavior.

Timestamp-aware execution and periodic KG refresh help avoid deprecated or contraindicated links, making the tool better suited for safety-critical clinical contexts (e.g., drug–drug interactions).

To address these limitations, Our work introduces a locally deployable pipeline that translates biomedical questions into executable Cypher queries over a Neo4j knowledge graph. The system returns answers with supporting nodes and edges, and is

evaluated using Exact Match, Precision, Recall, F1, Hits@k, MRR, and latency across simple, medium, and complex question sets. Unlike prior template-based methods, our approach enables traceable, outcome-level validation. In summary, the main contributions of this work are as follows:

- **Hybrid LLM to Cypher QA:** A system that translates natural language questions into accurate, executable Cypher over a Neo4j drug, disease, and symptom KG.
- **Prompt-driven query generation:** Schema, entity aware prompting that reliably maps diverse biomedical questions to graph queries.
- **Evidence transparency:** Along with each answer, we surface the generated Cypher and the supporting *justification subgraph* (nodes, edges) plus a brief natural language rationale.
- **Answer-level evaluation:** End-to-end assessment using *Exact Match, F1, Precision/Recall, Hits@k, MRR* and *latency* across simple, medium and complex tiers.
- **Local, reproducible deployment:** On-prem LLaMA 3 inference (no cloud dependency) suitable for biomedical settings requiring low latency and strong data control.

The remainder of this paper is organized as follows: Section 2 reviews related work on biomedical knowledge graphs and LLM-based QA systems. Section 3 provides background on knowledge graphs, large language models, and question answering frameworks. Section 4 details the proposed methodology, including system architecture, dataset construction, and query translation. Section 5 presents the experimental results through both quantitative metrics and qualitative case studies. Section 6 discusses the findings, analyzes limitations, and compares performance against baseline models. Finally, Section 7 concludes this paper and outlines directions for future work.

II. Related Work

Recently, studies have concentrated on the integration of clinical and medical knowledge graphs (LLM) to improve the answer to medical questions. Researchers have derived several biomedical KGs using Neo4j and incorporated the application of LLMs like LLaMA and GPT to convert natural language questions into graph queries. Improvements in answer correctness, reduction of hallucination errors, one-to-many relationships, and support for complex reasoning were the objectives of these efforts. Some frameworks also adopted retrieval methods to ground responses in secure data.

Su et al.[11] developed an integrative Biomedical Knowledge Hub (iBKH), a huge biomedical knowledge graph that comprises 18 of the very best data sources. The deployment of the iBKH in Neo4j allows for a user-friendly web portal to allow fast and interactive knowledge retrieval. The system implemented advanced graph learning techniques to enable the discovery of biomedical knowledge, illustrated by an example of repurposing in silico drugs for Alzheimer's disease. iBKH achieved promising predictive performance for known drugs and proposed possible new drug candidates.

Rajabi and Kafaie[19] proposed a disease knowledge graph using a cross-referential disease database comprising diseases, symptoms, and drugs interconnected with relationships. They transferred the data into Neo4j to create a graph of 9,400 nodes and 45,000 relationships representing the semantic links between medical concepts. Applying Cypher queries enabled answering complex medical questions regarding identifying drugs that may cause certain diseases; it was demonstrated that the graph inferred new relationships not explicitly existing in the original data. The conclusion was that disease knowledge graphs sped up clinical discovery and contributed to understanding complex medical relationships.

Hou et al.[3] assessed and contrasted ChatGPT (both GPT-3.5 and GPT-4) and the biomedical knowledge graphs (BKGs) concerning their ability to answer biomedical questions, generate new knowledge, and reason. Their datasets were focused on dietary supplements and drugs, while evaluation criteria entailed accuracy, novelty, and reasoning ability. The results indicate that while GPT-4 surpassed GPT-3.5 and BKGs in knowledge provision, it proved inconsistent with regard to citations and reasoning. Compared to them, BKGs scored higher in accuracy and reliability, especially in discovering novel links as well as within structured reasoning.

Soman et al.[13] presented a novel framework called KG-RAG that integrates a large biomedical knowledge graph (SPOKE) with LLaMA 2, GPT-3.5, and GPT-4 (LLMs) to produce accurate biomedical text. They optimized the retrieval of relevant graph context to cut over 50% tokens without losing accuracy. It aided LLMs in performing better on biomedical question answering with very high accuracy boosts, especially in the case of LLaMA 2. They compared KG-RAG to other retrieval methods and indicated its comparatively more robust and efficient results. The framework produced reliable evidence-based answers grounded in biomedical knowledge.

Luo et al.[23] created ChatKBQA, a new framework with a question-and-answer approach over knowledge bases that first generates logical forms with the help of fine-tuned LLMs and then retrieves the relevant entities and relations. This generate-then-retrieve approach is supposed to handle a couple of issues with the earlier methods concerning tedious retrieval and error propagation. They fine-tuned open-source LLMs like LLaMA 2 to change natural-language questions into logical forms with high accuracy. The retrieval phase uses unsupervised phrase-level semantic matching in a way that enhances the alignment of entities and relations. Experiments on benchmark datasets indicate ChatKBQA to be superior to its predecessors, with the highest accuracy to date.

Pusch and Conrad[6] conducted work under a hybrid approach conflating LLMs and biomedical Knowledge Graphs (KGs) to suppress hallucination errors in question-answering. They proposed query-checking algorithms for validating, correcting, and executing the KG Cypher queries that LLMs generated, thereby attaining accurate and understandable answers. The system used retrieval-augmented generation (RAG) to ground answers within KG data. The methodology was validated on a biomedical KG called PrimeKG using 50 benchmark questions, assessing models like GPT-4 Turbo and LLaMA 3. Commercially available GPT-4 Turbo obtained

record-high accuracy, while open-source models achieved impressive gains through prompt optimization.

Feng et al.[22] developed the Knowledge Graph-based Thought (KGT) framework that integrated LLMs with a pan-cancer knowledge graph for biomedical question answering. KGT was designed to reason on the knowledge graph schema and identify optimal subgraphs to use for directing accurate answer generation, all without fine-tuning the LLMs. The framework is benchmarked against a new dataset (PcQA) designed specifically for pan-cancer KGQA tasks and has outperformed all existing state-of-the-art approaches by a rather large margin. KGT's practicality in biomedical issues was highlighted through case studies for drug repositioning, drug resistance, and biomarker discovery. Their approach exhibited robust adaptability among various LLMs.

Rezaei et al.[26] developed AMG-RAG, a dynamic framework that utilizes autonomous LLM agents with medical search tools in the continuous construction and real-time updating of Medical Knowledge Graphs (MKGs). Their system incorporated confidence scoring and multi-hop reasoning to improve accuracy and interpretability in medical question answering. AMG-RAG outperformed size models on both very hard MEDQA benchmarks and more accessible MedMCQA ones, proving that it could conduct efficient reasoning based on current structured medical knowledge. They also used Neo4j to manage the knowledge graphs while adding external searches to ensure the latest data.

Tiwari et al.[24] presented Auto-Cypher, a recent automated pipeline for producing high-quality synthetic data for training LLMs by mapping natural language to Cypher queries for graph databases like Neo4j. The pipeline deployed the novelty of LLM-as-database-filler to synthesize Neo4j databases for the execution of generated queries to ensure their correctness. A sizable dataset called SynthCypher was created, spanning multiple domains and complex queries, leading to a 40% improvement in LLM performance on Cypher generation. The datasets were used to fine-tune open-source models such as LLaMA, Mistral, and Qwen, and the SPIDER benchmark was adapted for evaluation purposes.

Mohammed et al.[29] proposed a hybridized GraphRAG framework combining Neo4j-based UMLS knowledge graphs with a vector store for medical textbooks to create an improved U.S.M.L.E.-style clinical question-answering approach. The project integrated symbolic reasoning from knowledge graphs with semantic retrieval performed on text embeddings to enhance relevance and accuracy via adaptive re-ranking and query expansion. The system had the answers produced by GPT-4o-Mini, with different prompting strategies encouraging evidence-based and traceable responses grounded in verified medical knowledge. Experiments showed that the hybrid approach improved factual accuracy and citation fidelity as compared to the L.L.M.-only approach, enhancing transparency and reliability. It is shown that binding both structured and unstructured medical knowledge sources could aid in ameliorating hallucinations and hence improve clinical trustworthiness in AI-driven medical QA.

Yang et al.[30] articulated sepsis knowledge graph was crafted by combining multicenter clinical data from over 10,000 patients with the help of GPT-4 for entity recognition and relationship extraction. Real-world data were collected from three hospitals and integrated with clinical guidelines and databases from the public domain. The knowledge graph contained 1,894 nodes and 2,021 relationships pertaining to diseases, symptoms, biomarkers, treatments, and complications. GPT outperformed other models in every resolution on sepsis-specific datasets to obtain high F1-score results. The constructed graph highlighted complex interactions in sepsis for assisting clinical decision-making and was implemented on Neo4j.

Guan et al.[20] proposed a novel method for constructing a local knowledge graph from retrieved biomedical documents by extracting propositional claims. They carried out layer wise summarization on this graph to capture multi-document relationships and provide comprehensive contextual information to a language model for question-answering purposes. The method resolved issues in multi-document biomedical QA, such as noise cancellation and efficient context usage. They then tested their method on several benchmarks for biomedical question answering, achieving performance at least comparable to, if not better than, existing retrieval-augmented generation (RAG) baselines. The study established enhanced reasoning and answer accuracy of the model achieved through structured graph summarization.

Previous studies have improved biomedical QA using KGs and LLMs, but important gaps remain. Most systems lack transparent, graph-based justifications, rely on limited evaluation methods, or depend on cloud resources that reduce privacy and reproducibility. Our framework addresses these gaps by providing visible Cypher queries with evidence subgraphs, applying comprehensive performance metrics across difficulty levels, and ensuring fully local, privacy-preserving deployment.

Table I summarizes key previous studies on biomedical knowledge graphs and question answering, outlining their methods, datasets, and main limitations.

III. Preliminaries

This section outlines the fundamental concepts required to understand the proposed system. It introduces biomedical knowledge graphs, explains how Neo4j stores data in graph form, and describes the use of Cypher for querying. It also provides a brief overview of large language models (LLMs) and their role in interpreting natural language.

A. Biomedical Knowledge Graphs

Biomedical Knowledge Graphs (BKGs) provide a structured representation of complex biomedical information by modeling diverse medical entities, such as diseases, drugs, symptoms, and biological pathways, as interconnected nodes within a graph structure. The edges in these graphs represent the semantic relationships between these entities, including 'treats', 'causes', 'interacts with' and many others, as illustrated in Fig 1. This form of representation enables the integration of heterogeneous biomedical data from a wide range of sources, including

scientific literature, clinical records, genomic databases, and experimental findings [19, 31].

Such integration creates a comprehensive biomedical knowledge base that supports advanced analytics and discovery. For example, biomedical knowledge graphs can reveal previously unknown relationships (e.g., between drugs and diseases) and help prioritize potential biomarkers for complex conditions. The Integrative Biomedical Knowledge Hub (iBKH) is one such large-scale graph that consolidates diverse biomedical resources into a unified hub, enabling discovery at scale [11]. Beyond iBKH, large biomedical knowledge graphs such as SPOKE further illustrate how graph integration accelerates research and supports precision-medicine use cases [12]. Overall, these graphs serve as foundational resources for data-driven and personalized medicine. These knowledge graphs serve as foundational resources for precision medicine, where treatment can be tailored to the individual's biological profile, improving outcomes and minimizing side effects [19, 31].

TABLE I. SUMMARY OF RELATED RESEARCH ON BIOMEDICAL KGS AND QUESTION ANSWERING

Ref.	Year	Data/Graph	Method	Baselines	Key Metric	Limitation
[11]	2023	iBKH (18 biomedical sources, Neo4j)	Integrative KG + Graph learning; drug repurposing case	Known drugs, Alzheimer's study	Predictive performance (drug repurposing)	Limited to Alzheimer's case study; scalability and updates not detailed
[19]	2023	Disease KG (9,400 nodes, 45,000 relations in Neo4j)	Cypher queries for disease–drug–symptom reasoning	Cross-referential disease DB	New relation inference; complex query answering	Limited to single domain; lacks large-scale evaluation
[3]	2023	BKGs vs. GPT-3.5/4	Comparative QA study: LLMs vs. KGs	GPT-3.5, GPT-4, KG reasoning	Accuracy, Novelty, Reasoning	GPT-4 inconsistent in reasoning/citations; KG less fluent but more reliable
[13]	2024	SPOKE KG + LLaMA2, GPT-3.5, GPT-4	KG-optimized retrieval for LLMs (RAG)	Other retrieval methods	Accuracy, token reduction >50%	Focus on retrieval optimization, not KG construction
[23]	2024	Benchmark KB datasets	Generate-then-retrieve (LLM → logical form → KB retrieval)	Prior KBQA methods	Accuracy (highest to date)	Risk of error in logical form generation
[6]	2024	PrimeKG	LLM + KG hybrid, Cypher query validation, RAG	GPT-4 Turbo, LLaMA 3	Accuracy, Explainability	Dependent on KG coverage; computationally intensive
[22]	2025	Pan-cancer KG (PcQA dataset)	KG-enhanced reasoning (subgraph selection)	SOTA KGQA methods	Outperformed SOTA on PcQA	Limited to pan-cancer focus; no fine-tuning explored
[26]	2025	Dynamic Medical KG + Neo4j	LLM agents + multi-hop reasoning	MEDQA, MedMCQA baselines	Accuracy, Interpretability	High system complexity; requires continuous updating
[24]	2025	SynthCypher dataset (Neo4j + synthetic queries)	LLM-supervised Cypher generation and verification	SPIDER benchmark	Cypher accuracy 40%	Synthetic dataset may not capture all real-world cases
[29]	2025	UMLS KG + Neo4j	Hybrid GraphRAG	LLM-only QA	Accuracy, Citation fidelity	More complex pipeline; relies on external vector store
[30]	2025	Clinical data (10k patients, 1,894 nodes, Neo4j)	KG construction using GPT-4 for entity/relation extraction	Other KG construction methods	High F1-scores	Focus only on sepsis; limited generalization
[20]	2025	Local KG from biomedical documents	Multi-level summarization over KG for QA	RAG baselines	QA accuracy, reasoning	Tested mainly on document QA; scalability not proven

B. Neo4j Graph Database

To manage the complexity and large size of biomedical knowledge graphs, specialized graph databases are needed. Neo4j is one of the most popular graph databases designed to store and query data structured as nodes (entities) and relationships (edges), both of which can have descriptive properties [32, 33]. It uses the property graph model, which makes it easy to represent complex, connected biomedical data such as drug-gene interactions or disease pathways. Neo4j's Cypher query language is especially advantageous because it allows users to write expressive and efficient queries to explore multi-step connections in the data [34].

Neo4j works well for biomedical data because it can quickly run complicated queries over highly interconnected datasets. This is important in biology and medicine, where relationships between entities like proteins, diseases, and drugs are often complex and layered. Studies have shown that Neo4j handles large biomedical graphs efficiently, making it a favorite among researchers and industry users alike [33, 35, 36]. Its indexing and caching mechanisms also help speed up query processing and data retrieval [37].

Moreover, Neo4j integrates smoothly with many programming languages and analytics tools, which makes it easier to build interactive biomedical applications and clinical decision support systems that can turn complex graph data into useful insights [38, 39].

C. Large Language Models (LLMs) in Biomedical Question Answering

Large Language Models (LLMs) are powerful AI systems trained on vast amounts of text data. They learn the structure and patterns of language, enabling them to understand questions, generate responses, summarize information, and perform other complex language tasks. Well-known models such as LLaMA and GPT-3 have greatly advanced the field of natural language processing by showing strong performance across many tasks [40, 41].

In biomedical research and clinical settings, LLMs help translate natural language questions from doctors, researchers, or patients into precise, structured queries that can be executed on biomedical knowledge graphs and databases. This makes it easier to retrieve detailed biomedical information like drug interactions, gene-disease associations, and symptom descriptions [42, 43].

Despite their power, LLMs can sometimes generate incorrect or fabricated responses, a phenomenon known as hallucination, which poses risks in sensitive biomedical contexts. These hallucinations occur because the models generate plausible answers based on patterns learned from data rather than verified knowledge. To mitigate this, researchers integrate LLMs with biomedical knowledge graphs to ground answers in factual data, significantly improving accuracy and reducing misinformation [4]. Further improvements come from fine-tuning LLMs on biomedical corpora and carefully engineering prompts, which enhance their reliability and relevance in medical question answering.

Additionally, combining LLMs with knowledge graphs and reasoning techniques is an active area of research that promises to increase the interpretability and trustworthiness of AI systems in biomedicine. These advances are critical for developing tools that assist clinical decision-making and accelerate biomedical discoveries [43, 44].

IV. Methods and Materials

This section describes the methodology used to build a biomedical question-answer system. The proposed method consists of three main stages; First, a biomedical knowledge graph is constructed in the data ingestion phase, using structured data sources (e.g., diseases, drugs, symptoms). Second, a language model (LLaMA 3) interprets the user's question written in English in the user interaction phase and converts it into a Cypher query. Third, a graphical user interface allows users to type questions and view the results interactively.

A. System Architecture

The proposed framework is organized as a step-by-step pipeline that integrates a local large language model (LLM) with a biomedical knowledge graph stored in Neo4j. The overall workflow is illustrated in Fig. 2. Each module performs a specific function, and together they ensure that the system delivers accurate, reliable, and explainable answers.

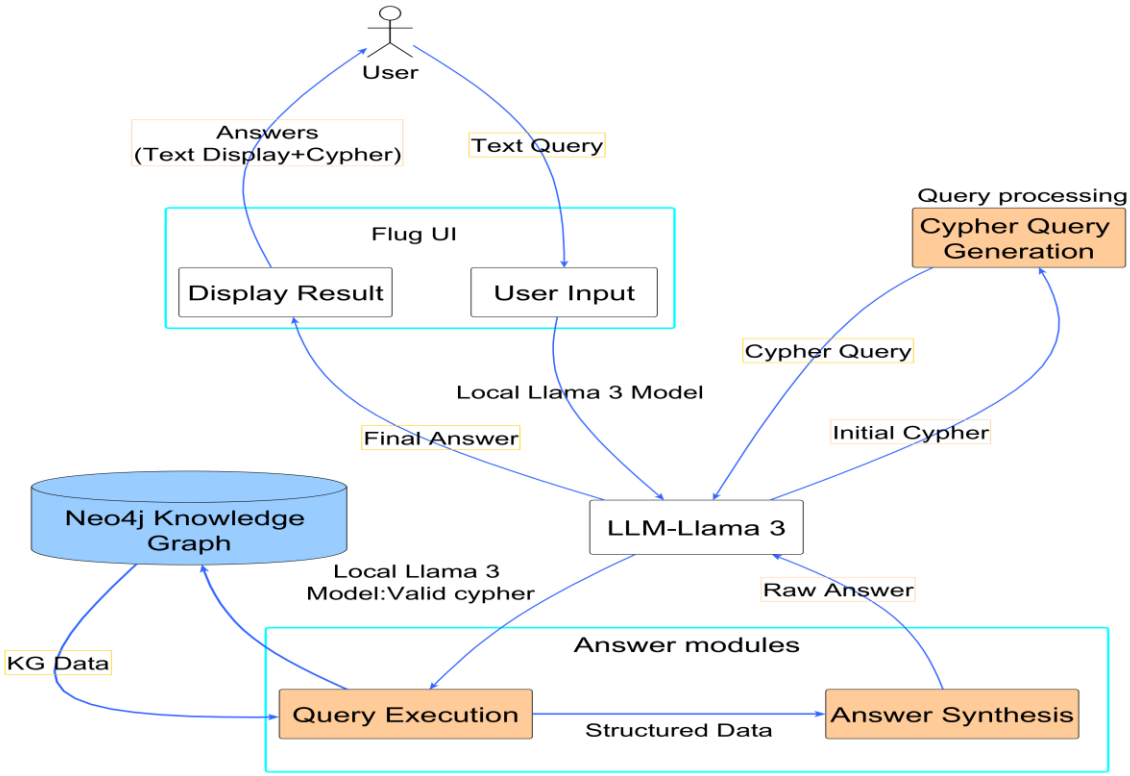

Fig. 2. Workflow of the LLaMA 3 and Neo4j-Based QA System

Step 1. User Input (Flask Web Interface): A user submits a biomedical question in natural language through a Flask-based web interface. The same interface will later display the answer, the executed query, and a compact preview of the retrieved rows.

Step 2. LLM Processing and Initial Cypher: The text query is forwarded to a local **LLaMA 3** module, which interprets the intent and drafts an *initial Cypher* pattern suitable for querying the graph.

Step 3. Cypher Query Generation: The Cypher Query Gen block receives the initial pattern from LLaMA 3, canonicalizes and completes it (projection, DISTINCT, filters), and returns a finalized *Cypher query* to the model.

Step 4. Query execution on Neo4j: LLaMA 3 passes the finalized query to the Query execution component (inside the 'Answer modules' box), which runs it against the Neo4j Knowledge Graph. KG supplies *knowledge graph data* (e.g.,

drugs, diseases, symptoms) and execution returns *structured data* (tuples) relevant to the question.

Step 5. Answer Synthesis: The structured tuples flow to Answer Synthesis, which aggregates and formats them into a concise *raw answer*. This raw answer is sent back to LLaMA 3 to optionally refine the phrasing while preserving the retrieved facts.

Step 6. Result Presentation: LLaMA 3 produces the *final answer*, which the interface displays together with the executed *Cypher query* and an optional preview of the returned rows, improving transparency and trust.

The pipeline couples LLM-based language understanding (LLaMA 3) with a schema-grounded Neo4j knowledge graph. The *Cypher Query Gen* refines the query formulation, *Query Execution* retrieves evidence and *Answer Synthesis* converts structured results into readable outputs that produce answers that are accurate, interpretable, and easy to audit directly from the displayed query and evidence.

B. Dataset and Knowledge Graph Construction

1) Dataset

The integrated Biomedical Knowledge Hub (iBKH), a large biomedical knowledge base, forms the first level of the system and integrates information from various curated high-quality biomedical databases. This implies that the data set includes various types of entities, such as diseases, symptoms, drugs, biological pathways, etc. This study used the representative subset of the iBKH dataset, which contained 65828 biomedical entities. These entities are semantically interconnected through a total of 3004166 relationships, thus creating a rich knowledge graph. The iBKH dataset was originally introduced in [11], and it is freely available at (https://github.com/wcm-wanglab/iBKH). This dataset is the core semantic foundation upon which this study is built. The knowledge graph is populated from multiple tabular sources (CSV files), each listing entities or relationships. The main input files and their contents are as follows:

- **Disease vocabulary**(disease_vocab.csv): Contains columns such as primary (a unique disease ID), name, do_id (Disease Ontology ID), kegg_id, and umls_cui (UMLS Concept ID). Each row represents a disease node with external identifiers.
- **Drug vocabulary** (drug_vocab.csv): Includes primary (unique drug ID), name, drugbank_id, kegg_id, pharmgkb_id, umls_cui, mesh_id, iDISK_id and CID (PubChem ID). Each row defines a drug node with standard database identifiers.
- **Symptom vocabulary (symptom_vocab.csv)**: Contains primary (unique symptom ID), name, mesh_id, umls_cui and iDISK_id. Each row defines a symptom node.
- **Side effect vocabulary (side_effect_vocab.csv)**: Includes primary (unique side-effect ID) and name. Each row represents a side-effect node (with UMLS ID when available).
- **Pathway vocabulary (pathway_vocab.csv)**: Contains primary (unique pathway ID), name, reactome_id, go_id, and kegg_id. Each row defines a biological pathway node.

Relationship files (each row typically contains two entity IDs and one or more boolean flags or codes) include:

- **Disease–Symptom links** (Di_Sy_res.csv): Rows include Disease and Symptom IDs, a presence flag (1 or 0) and a data source. If Present = 1, a HAS_SYMPTOM edge is created from the disease to the symptom, with properties for presence and source.
- **Disease–Disease links** (di_di_res.csv): Rows include Disease_1 and Disease_2 IDs with binary flags for *is_a* and *Resemble*. If is_a = 1, an (IS_A) edge is created (Disease_1 → Disease_2); if Resemble = 1, a RESEMBLES edge is created. The source field is used for provenance.
- **Drug–Disease links** (D_Di_res.csv): Includes Drug and Disease IDs with several binary flags. If a flag equals 1, a corresponding edge is created:
  - TREATS (Treats = 1)
  - PALLIATES (Palliates = 1)
  - ASSOCIATED_WITH (Associate = 1)
  - ALLEVIATES_REDUCES (alleviates = 1)
  - TREATMENT_THERAPY (treatment/therapy = 1)
  - INHIBITS_CELL_GROWTH (inhibits cell growth = 1)
  - HAS_BIOMARKER (biomarkers = 1)
  - PREVENTS_SUPPRESSES (prevents/suppresses = 1)
  - ROLE_IN_PATHOGENESIS (role in disease pathogenesis = 1)
- **Drug–SideEffect links** (D_SE_res.csv): Contains Drug and SideEffect IDs with a Source column. Each row creates a CAUSES edge from the drug to the side effect, with source as an edge property.
- **Drug–Drug interactions (D_D_res.csv)**: Rows include Drug_1 and Drug_2 IDs with flags for Interaction and Resemble. If Interaction = 1, an INTERACTS_WITH edge is created (bidirectional). If Resemble = 1, a RESEMBLES edge is added.
- **Drug–Pathway links** (D_Pwy_res.csv): Includes Drug ID and Pathway ID. Each row generates an ASSOCIATED_WITH edge from the drug to the pathway.
- **Disease–Pathway links (Di_Pwy_res.csv)**: Contains Disease ID and Pathway ID. Each row creates an ASSOCIATED_WITH edge from the disease to the pathway.

2) Data Upload Performance

The time required to upload different types of entities and relationships into the Neo4j biomedical knowledge graph, measured in seconds. These measurements reflect both the size and complexity of the data being processed.

As shown in Table II, the longest upload time is for *Drug-Drug Relationships*, which takes approximately 190 seconds due to the large number of edges (over 3 million). Following this, *Disease-Disease* and *Drug-Disease Relationships* also require considerable time for loading. On the other hand, individual

entities such as *Diseases* and *Drugs* are uploaded much faster, generally under 2 seconds.

TABLE II. DATA UPLOAD TIMES FOR DIFFERENT ENTITY AND RELATIONSHIP TYPES IN NEO4J

Entity / Relationship Type	Upload Time (seconds)
Disease	0.81
Drugs	1.08
Symptoms	0.06
Side Effects	0.14
Pathways	0.08
Disease-Disease Relationships	30.97
Drug-Disease Relationships	30.28
Drug-SideEffect Relationships	5.24
Drug-Drug Relationships	190.09
Drug-Pathway Relationships	0.14
Disease-Pathway Relationships	0.06
Disease-Symptom Relationships	0.12

Fig. 3, presents a vertical bar chart that visually compares these upload times across the different entity and relationship types. The chart clearly shows the significant difference in upload duration between nodes and edges, emphasizing the higher cost of ingesting complex relationships in the graph.

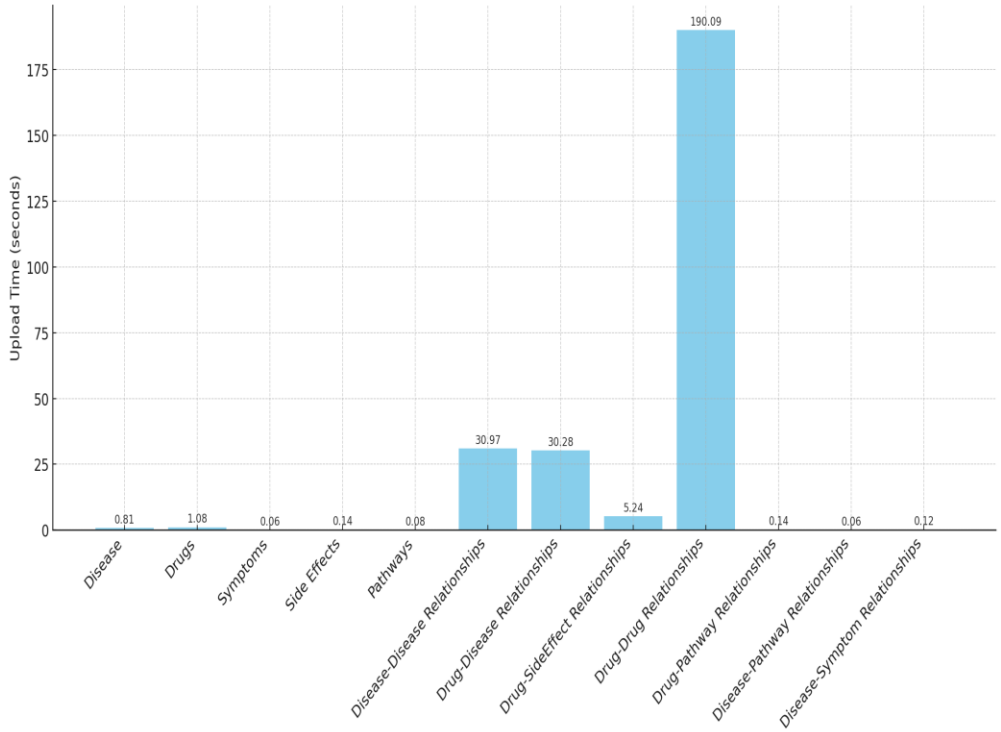

Fig. 3. Upload times for various biomedical entities and relationships in Neo4j.

We run LLaMA 3 locally (not via cloud APIs) to satisfy biomedical privacy/governance (no data leaves the host) and to maximize reproducibility (fixed GGUF checkpoint, pinned llama.cpp commit, controlled seeds settings, constant hardware). Local execution yields predictable cost availability and stable latency (no network jitter) and lets us enforce executable Cypher grounding with per edge provenance and timestamp aware execution.

3) Experimental Environment

In this study, the proposed biomedical question answering system was evaluated using a locally hosted environment. All experiments were conducted on a Windows 11 Pro (64-bit) system equipped with an Intel Core i5-10500H processor running at 2.50 GHz (12 logical CPUs), 24 GB of RAM, and an NVIDIA GeForce GTX 1650 GPU with Max-Q Design. The Neo4j graph database (v4.4.5) was managed through Neo4j Desktop (v1.6.2), and the LLaMA 3B language model was executed locally using optimized configurations suitable for this hardware setup.

Each Cypher query generated by the system was executed multiple times to calculate an average response time, ensuring consistency across varying levels of question difficulty. The knowledge graph was constructed using the iBKH dataset, and data loading and system performance were carefully monitored to maintain stability during testing. This experimental setup provides a reliable and reproducible environment for benchmarking the effectiveness and responsiveness of the hybrid QA system.

4) Knowledge Graph(KG) Construction

The Neo4j graph database was used as the backend to store and query the KG, and it is a graph database designed for highly connected data. Before loading data, uniqueness constraints were created on the primary property for each node label (Disease, Drug, Symptom, SideE ffect, Pathway). This enforces that each primary ID appears only once, preventing duplicate entities. For efficient lookups in queries, a search index was created based on the name property of each node label. As noted in the Neo4j

documentation, indexes "enable quicker and more efficient pattern matching" by allowing the query planner to rapidly locate nodes by label and property.

With the schema in place, data was imported using Cypher's LOAD CSV commands. For each vocabulary file, rows with nonnull primary IDs were processed: the code used MERGE to create (or match existing) nodes with the given label and primary property, and then SET the remaining attributes from the CSV columns. For example, in importing disease_vocab.csv, each row produced a node (:Disease primary: <id>) with properties name, do_id, kegg_id, and umls_cui set from the row (empty strings were turned into null). Similarly, drug_vocab.csv rows produced drug nodes with properties for DrugBank, KEGG, PharmGKB IDs, etc. This approach follows best practice: using MERGE on the unique ID ensures that nodes are not duplicated during multiple passes.

After all nodes were created, the relationship CSVs were loaded. Each row in those files was matched to the corresponding source and target nodes by their primary IDs, and conditional logic was used to create edges. For example, the disease-symptom file (Di_Sy_res.csv) was processed by matching a disease node and a symptom node for each row, then executing MERGE (d)-[r:HAS_SYMPTOM]-(s) if the present column is nonzero; the edge was given a present property and a source property from the CSV. The disease-disease file (di_di_res.csv) was processed by matching disease nodes d1 and d2:If is_a = 1, a (:Disease)-[:IS_A]->(:Disease) edge was merged; if Resemble = 1, a (:Disease)-[:RESEMBLES]->(:Disease) edge was merged. Drug-disease relationships were handled similarly: the script matched a Drug node and a Disease node for each row of D_Di_res.csv, then for each flag that was 1, it merged the corresponding edge label (such as TREATS, PALLIATES, ASSOCIATED_WITH, etc.) from the drug to the disease. Drug–side-effect rows produced (:Drug)-[:CAUSES]->(:SideEffect) edges with the source noted, and drug–drug rows produced either INTERACTS_WITH or RESEMBLES edges between matched drug pairs. Finally, the drug-pathway and disease-pathway files each produced:ASSOCIATED_WITH edges linking drugs or diseases to pathway nodes.

In this graph model, most relationships are directional (for example, a drug TREATS a disease; a disease does not TREAT a drug). This follows common practice in biomedical KGs. The same relations (such as RESEMBLES or INTERACTS_WITH) are inherently symmetric, but were stored as two directed edges or one undirected edge depending on implementation. All relationship creation steps used Cypher's MERGE so that repeated loads or out-of-order imports did not create duplicate edges.

This study used a static KG snapshot for reproducibility, but the system also supports incremental updates through the Neo4j MERGE and batch import functions. A summary of the different types of nodes and relationships is provided, together with their counts, descriptions, and examples in Table III.

TABLE III. DETAILED SUMMARY OF KNOWLEDGE GRAPH COMPONENTS IN IBKH SUBSET

Component Type	Entity/Relationship	Count	Description	Examples
Nodes	Disease	19,236	Medical conditions and disorders	Diabetes, Hypertension, Cancer
	Drug	37,997	Pharmaceutical compounds	Aspirin, Insulin, Amoxicillin
	Symptom	1,361	Clinical signs and symptoms	Headache, Fever, Nausea
	SideEffect	4,251	Negative outcomes of drugs	Nausea, Drowsiness
	Pathway	2,983	Biological processes and pathways	Apoptosis, Glycolysis
Relationships	ASSOCIATED_WITH	101,534	General associations (disease–pathway, drug–pathway, etc.)	Diabetes ASSOCIATED_WITH Pathway
	CAUSES	145,321	Drug-SideEffect relationships	Aspirin CAUSES Gastric_Bleeding
	HAS_SYMPTOM	3,357	Disease-symptom links	COVID-19 HAS_SYMPTOM Fever
	HAS_BIOMARKER	412	Biomarkers linked to disease	PSA HAS_BIOMARKER Prostate_Cancer
	INHIBITS_CELL_GROWTH	1,508	Drugs inhibiting cell growth	Chemo INHIBITS_CELL_GROWTH Tumor
	INTERACTS_WITH	2,682,142	Drug-drug interactions	Aspirin INTERACTS_WITH Warfarin
	IS_A	10,529	Subtype hierarchy	Flu IS_A Viral_Infection
	PALLIATES	388	Drug palliates disease	Morphine PALLIATES Cancer
	PREVENTS_SUPPRESSES	859	Preventive links	Vaccine PREVENTS_SUPPRESSES Measles
	RESEMBLES	7,000	Similarity relationships	DrugA RESEMBLES DrugB
	TREATMENT_THERAPY	44,852	Therapy relationships	Radiotherapy TREATMENT_THERAPY Tumor
	TREATS	5,491	Drug-disease links	Insulin TREATS Diabetes
	ALLEVIATES_REDUCES	~180,000	Symptom relief	Paracetamol ALLEVIATES_REDUCES Fever
Total	Nodes	65,828	Total biomedical entities	—
	Relationships	3,004,166	Total knowledge graph links	—

C. Natural Language to Cypher Query Translation

A key feature of the system is its ability to accept questions written in plain English and automatically generate the corresponding Cypher queries. This is accomplished using Meta's LLaMA 3 large language model, which runs entirely on a local machine through the open-source llama.cpp framework. Running the model locally ensures low-latency execution and keeps sensitive queries within the user's environment.

To generate a Cypher query, LLaMA 3 is prompted with examples of natural language questions along with their correct Cypher translations. The prompt also includes instructions on how to navigate the structure of the graph schema. When a user enters a question (e.g., 'What are the symptoms of Alzheimer's disease?'), the system inserts it into the prompt and asks LLaMA 3 to produce a corresponding query. For example, the model may return:

```
MATCH (d:Disease)-[:HAS_SYMPTOM]->(s:Symptom)
WHERE toLower(d.name) CONTAINS "alzheimer"
RETURN s.name
```

This query searches for a disease node whose name contains 'alzheimer' and follows HAS_SYMPTOM edges to list related symptom names. The system then executes this cypher to retrieve answers. The prompts (such as few-shot examples and schema hints) were carefully designed to help LLaMA 3 generate correct Cypher queries. The model learns how to use the graph's labels and relationships properly. For example, if a user asks, 'Which drugs treat diabetes?', LLaMA might respond with a suitable Cypher query:

```
MATCH (d:Drug)-[:TREATS]->(di:Disease)
WHERE toLower(di.name) CONTAINS "diabetes"
RETURN d.name
```

This queries for drug nodes that have a TREATS edge to a diabetes disease node. By leveraging LLaMA 3 in this way, our system can flexibly handle many phrasing variations without manual mapping rules.

D. Model Configuration & Decoding

We run a local **LLaMA 3.2-3B** model in GGUF format (llama-3.2-3b-instruct-q4_k_m.gguf) via llama.cpp, as shown in Table IV.

TABLE IV. MODEL RUNTIME AND DECODING SETTINGS

Runtime settings	Decoding settings
n_ctx = 1024	temperature = 0.2
n_threads = 12	top_p = 0.95
n_gpu_layers = 33	top_k = 40
n_batch = 512	repeat_penalty = 1.1
	max_tokens = 80
	seed = 42

E. Graph Subset and Versioning

We use an iBKH derived subgraph (≈ 65.8k nodes; ≈ 3.0M edges) spanning DRUG, DISEASE, SYMPTOM, PATHWAY. IDs are normalized to CURIEs and duplicates collapsed across UMLS/DrugBank/DisGeNET/SIDER/KEGG. Each edge stores provenance/licensing metadata (source, source_version, license, retrieved_at, evidence_pmids/urls) and, when available, temporal fields (valid_from, valid_to). We report *coverage* as the percentage of evaluated questions whose gold entities/relations are present.

F. Query Execution and Reliability

After a Cypher query is generated, it is executed on the Neo4j database through the official Neo4j Python driver, which manages the secure connection and returns the results. Instead of restricting the output with a fixed LIMIT (e.g., LIMIT 5), the system retrieves candidate results and evaluates them using standardized retrieval metrics such as Hits@1, Hits@5, and Hits@10. This approach ensures that the system remains responsive while providing a fair assessment of ranking quality across different cutoff levels, rather than depending solely on a fixed number of returned items. Neo4j's indexing on key node properties, such as name and primary identifiers, also helps speed up lookups as the knowledge graph grows. In cases where the language model generates an incomplete or incorrect query, such as referencing nodes or relationships that do not exist, the system catches the error and either retries with a simpler prompt or informs the user. Together, these steps make sure that queries run quickly, return valid results, and keep the overall experience smooth and reliable for biomedical question-answering.

G. User Interface for Query Transparency

The system includes a lightweight, cross-platform graphical user interface (GUI) implemented as a web application using the Flask framework in Python, with HTML and Bootstrap for interactive visualization. The interface is designed to make the question-answering process transparent and accessible to users without technical expertise. It consists of three main panels:

1. Input Panel: Where the user can enter a biomedical question in natural language.
2. Query Panel: Which displays the Cypher query generated by the language model, allowing users to verify how their question was interpreted.
3. Results Panel: Which presents the retrieved answers in a clear, readable format, accompanied by a brief natural language explanation generated by the system.

By showing both the query and the answer, the GUI promotes user trust and enables validation of the system's reasoning process. The interface is lightweight enough to run smoothly on standard desktop machines without additional dependencies, making it practical for local deployments in clinical or research settings. Fig. 4 illustrates the overall layout.

V. EXPERIMENTAL RESULTS

In this section, the proposed system is evaluated based on its ability to translate natural-language biomedical questions into executable Cypher queries over the iBKH knowledge graph. The assessment is conducted end-to-end and focuses on three main aspects: (i) the accuracy of query generation, measured by the proportion of correctly produced Cypher queries; (ii) system efficiency, quantified through total response time from question submission to final answer delivery, covering both query generation and execution; and (iii) the quality of retrieved answers, evaluated using standardized information retrieval metrics including Precision@k, Recall@k, F1@k, Hits@k, and Mean Reciprocal Rank (MRR). For clarity, all these metrics are formally defined in the following subsections, with their corresponding equations, and will be used consistently throughout the results section. Together, these dimensions provide a comprehensive view of both correctness and efficiency across simple, medium, and complex biomedical queries.

A. Description of the Experimental Data

To evaluate the proposed system, this work used a carefully selected set of biomedical questions designed to test how well the system understands natural language and converts it into Cypher queries for a biomedical knowledge graph.

To evaluate the system, a benchmark of 60 biomedical questions was constructed by the authors, guided by the schema and relations in iBKH. The questions were grouped into simple, medium, and complex levels to assess performance across different reasoning requirements. Gold-standard answers were manually prepared for each question to enable quantitative evaluation using Exact Match, Precision@k, Recall@k, F1@k, Hits@k, and MRR. The complete set of 60 questions is available at this link https://drive.google.com/drive/my-drive.

The dataset contains 60 questions divided into three difficulty levels based on how complex the language is and how deep the biomedical reasoning needs to be:

- Level 1: 25 simple questions focusing mostly on easy-to-understand facts, such as symptoms of certain diseases or drugs used to treat specific conditions.
- Level 2: 20 medium-level questions that involve more detailed relationships, such as drug interactions and SideEffect.
- Level 3: 15 hard questions requiring multi-step reasoning across multiple biomedical concepts or biological pathways, similar to the complexity found in real clinical cases.

The set of evaluation questions was designed to span a wide range of common biomedical topics and to reflect clinically relevant query types reported in prior literature. Each question is paired with its corresponding gold standard cypher query and categorized by difficulty level, as summarized in Table V, where three illustrative examples are shown. The questions were derived from publicly available biomedical QA benchmarks and adapted from established knowledge bases (e.g., iBKH schema relations), ensuring both coverage and diversity across diseases, drugs, symptoms, and pathways.

TABLE V. QUESTION DIFFICULTY LEVELS AND SAMPLE CYPHER QUERIES

Level	Definition	Example	Cypher Query
1	Single-hop question using one relationship	What are the symptoms of Alzheimer?	MATCH (d:Disease)-[:HAS_SYMPTOM]->(s:Symptom) WHERE toLower(d.name) CONTAINS 'alzheimer' RETURN s.name AS symptom
2	Questions requiring one relationship	What are the side effects of drugs used to treat asthma?	WITH toLower('asthma') AS disease_name MATCH (d:Disease)<-[:TREATS]-(dr:Drug) WHERE toLower(d.name) CONTAINS disease_name MATCH (dr:Drug)-[:CAUSES]->(se:SideEffect) RETURN DISTINCT se.name AS side_effect, dr.name AS drug
3	Questions requiring two or more relationships	What cholesterol medications cause side effects, and what are some of those effects?	WITH toLower('cholesterol') AS disease_name MATCH (d:Disease)<-[:TREATS]-(dr:Drug) WHERE toLower(d.name) CONTAINS disease_name MATCH (dr:Drug)-[:CAUSES]->(se:SideEffect) RETURN DISTINCT dr.name AS drug, se.name AS side_effect

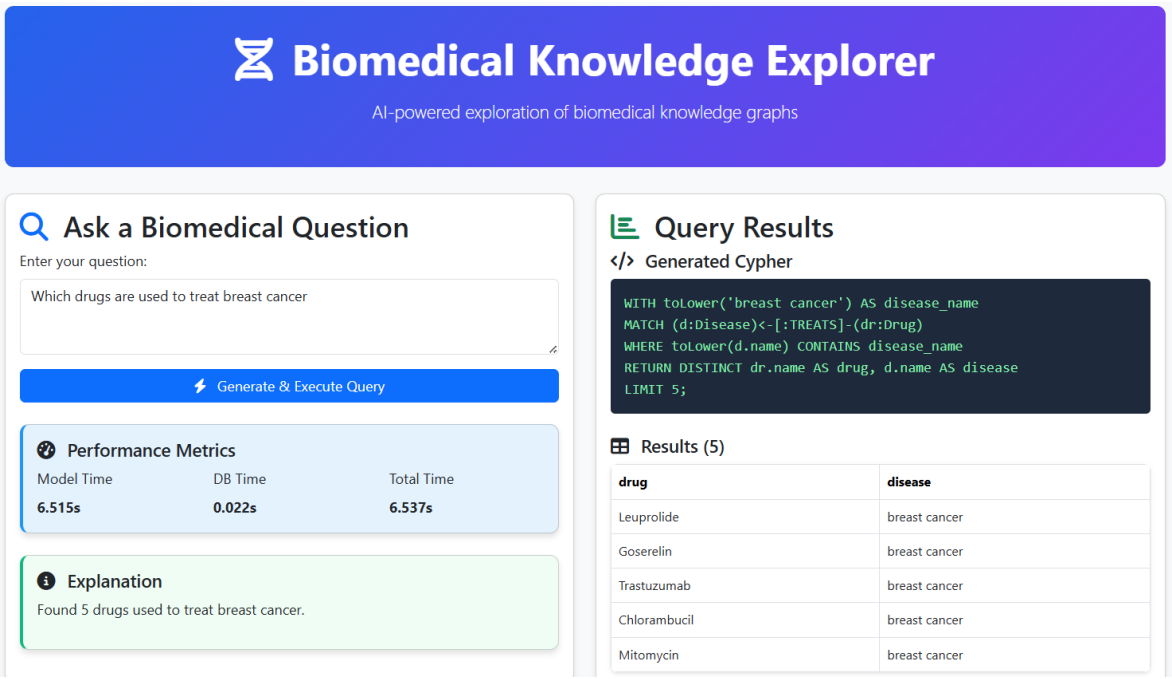

Fig. 4. Graphical User Interface of the Biomedical Knowledge Explorer System

B. *Quantitative Evaluation*

The performance of the proposed system is evaluated on the iBKH knowledge graph using Exact Match, Precision@k, Recall@k, F1@k, Hits@k, MRR, and total response time across simple, medium, and complex biomedical queries. These metrics were measured for each difficulty level (simple, medium, and complex) to see how performance changes as questions become more challenging.

TABLE VI. QUERY GENERATION EXACT MATCH BY DIFFICULTY LEVEL

Difficulty Level	Total Questions	Correct Queries	Cypher Exact Match (EM) (%)
Simple	25	24	96%
Medium	20	19	95%
Complex	15	13	86.7%
avg			92.6%

To better understand the quality of the responses returned by the system, this work examined real examples. One of the test questions was "What drugs are used to treat breast cancer?" As shown in Fig. 4, the system was able to understand the question, generate the correct Cypher query using the TREATS relationship, and retrieve the right information from the biomedical knowledge graph. It returned a list of five drugs, including Leuprolide, Goserelin, and Trastuzumab, which are known treatments for breast cancer. This result shows that the system is able to connect the question to the right part of the graph and provide accurate and helpful answers, even for medical questions that involve specific treatments.

1) Exact Match of Query Generation

As shown in Table VI, how often the system generated the correct query for each difficulty level. Here, accuracy is defined as the percentage of questions for which the system's generated Cypher query matched the expected query. It is calculated using the Eq. (1)

$$\text{Exact Match (\%)} = \frac{\text{Correct Queries}}{\text{Total Questions}} \times 100 \quad (1)$$

These findings highlight the purpose of this experiment to test whether the framework can reliably map natural language to Cypher across varying levels of complexity. The graceful drop from 96% on simple to 86.7% on complex queries indicates that the system is robust for straightforward questions but still challenged by multi-hop reasoning. This points to clear opportunities for improvement, such as synonym expansion, constrained decoding, or enhanced error handling for multi-step queries.

2) Latency

Table VII reports the average latency per difficulty level and decomposes it into query generation and Neo4j execution. The total response time is computed as in Eq. (2). Execution time is effectively constant across all levels (≈ 0.04–0.05 s), so variation in total latency is dominated by query generation. As difficulty increases, the mean total time rises from 5.12 s (simple) to 5.75 s (medium) and 7.35 s (complex). Dispersion (Std. Dev.) grows with task complexity 0.72 s (simple), 0.32 s (medium), 2.09 s (complex) reflecting more variable planning and reasoning needed to assemble correct Cypher for harder questions. Pooled over all questions, the overall mean is 6.07 s with an overall SD of 1.38 s, keeping latencies within single-digit seconds and practical for interactive, real-world use.

$$T_{\text{total}} = T_{\text{gen}} + T_{\text{exec}} \quad (2)$$

Standard Deviation (SD). Unless otherwise stated, SD is the unbiased sample standard deviation computed over per-question total times within each difficulty group G with n_G questions:

$$\bar{T}_G = \frac{1}{n_G}\sum_{i=1}^{n_G} T_{\text{total}}^{(i)}, \quad (3)$$

$$s_G = \sqrt{\frac{1}{n_G - 1}\sum_{i=1}^{n_G}\left(T_{\text{total}}^{(i)} - \bar{T}_G\right)^2}. \quad (4)$$

Overall values are computed by pooling all questions across levels, with $N = \sum_G n_G$:

$$\bar{T}_{\text{overall}} = \frac{1}{N}\sum_{i=1}^{N} T_{\text{total}}^{(i)}, \quad (5)$$

$$s_{\text{overall}} = \sqrt{\frac{1}{N - 1}\sum_{i=1}^{N}\left(T_{\text{total}}^{(i)} - \bar{T}_{\text{overall}}\right)^2}. \quad (6)$$

TABLE VII. AVERAGE LATENCY (S) BY DIFFICULTY LEVEL (SD = STANDARD DEVIATION)

Difficulty Level	Query Generation (s)	Query Execution (s)	Total Time (s)	Std. Dev. (s)
Simple	5.09	0.03	5.12	0.72
Medium	5.69	0.01	5.75	0.32
Complex	6.94	0.4	7.35	2.09
Overall	5.9	0.15	6.07	1.38

3) Answer-Level Evaluation Metrics

As shown in Table VIII, the quality of the returned items is evaluated using five standardized metrics: Precision@k, Recall@k, F1@k, Hits@k, and MRR. Precision@k measures the proportion of correct answers among the top-k retrieved items Eq. (7), while Recall@k quantifies the fraction of gold-standard answers covered within the top-k results Eq. (8). F1@k combines both aspects through the harmonic mean Eq. (9). Hits@k reports whether at least one correct answer appears in the top-k Eq. (10, 11), and MRR captures how early the first correct answer is retrieved in the ranking Eq. (12). Together, these metrics provide a comprehensive view of both the correctness and completeness of retrieval, as well as the ranking quality across simple, medium and complex queries.

$$\text{Precision@}k = \frac{|\{\text{relevant} \cap \text{retrieved@}k\}|}{k} \quad (7)$$

Precision@k measures the fraction of the top-k retrieved items that are correct (i.e., appear in the gold set). For example,

P@1 refers to the accuracy of the very first retrieved item, P@5 evaluates correctness within the top three results, and P@10 considers the top ten. Higher values indicate that relevant items tend to appear early in the ranked list.

TABLE VIII. EVALUATION METRICS ACROSS LEVELS

	Simple	Medium	Complex	Overall
P@1	100	95	86.67	93.8
P@5	100	95	93.33	96.11
P@10	100	95	88	94.33
R@1	88	90	86.67	88.22
R@5	90.65	89.67	86.67	88.9
R@10	89.60	89	83.33	87.31
F1@1	88	90	86.67	88.22
F1@5	93.20	91.25	88.67	91.03
F1@10	93.28	91.39	85.24	89.97
Hits@k	100	95	93	96.1
MRR	100	95	88	94.4

$$\text{Recall@}k = \frac{|\{\text{relevant} \cap \text{retrieved@}k\}|}{|\{\text{relevant}\}|} \quad (8)$$

Recall@k measures the proportion of all relevant items (in the gold set) that are successfully retrieved within the top- k positions. For instance, R@5 indicates how many of the expected answers are covered by the top three results. This metric is particularly important when the gold answer set is larger than k.

$$\text{F1@}k = \frac{2 \cdot \text{Precision@}k \cdot \text{Recall@}k}{\text{Precision@}k + \text{Recall@}k} \quad (9)$$

F1@k is the harmonic mean of Precision@k and Recall@k. It balances the trade-off between retrieving only relevant items (precision) and covering as many relevant items as possible (recall). For example, F1@10 reflects the combined quality of the system when retrieving the top ten results.

$$\text{Hits@}k = \begin{cases} 1 & \text{if } \{\text{relevant} \cap \text{retrieved@}k\} \neq \varnothing \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

Hits@k is a binary measure at the query level: it equals 1 if at least one correct answer is present among the top-k retrieved items, and 0 otherwise. For example, Hits@5 reports the percentage of queries where the system was able to "hit" at least one correct answer in the top five results.

$$\text{Hits@}k = \frac{1}{|Q|}\sum_{q \in Q} \mathbf{1}\left(\{\text{relevant}_q \cap \text{retrieved}_q@k\} \neq \varnothing\right) \quad (11)$$

When averaged across a set of queries Q, Hits@k gives the proportion of queries for which at least one relevant item is returned in the top-k results. This measure is less sensitive to ranking quality but emphasizes coverage across queries.

$$\text{MRR} = \frac{1}{|Q|}\sum_{q \in Q} \frac{1}{\text{rank}_q} \quad (12)$$

Mean Reciprocal Rank (MRR) averages the reciprocal of the rank of the first correct answer for each query $q \in Q$. For example, if the first correct answer appears in the top position, the reciprocal rank is $1/1 = 1.0$; if it appears at rank 5, the contribution is $1/5 = 0.2$. MRR therefore rewards systems that retrieve correct answers as early as possible.

According to the results in Table VIII, this work reports standardized metrics, including Precision, Recall, F1, Hits@k, and MRR at different cutoffs ($k = 1,5,10$). Precision@k captures the proportion of retrieved items among the top-k that are correct, while Recall@k measures the coverage of gold-standard answers within the same cutoff. F1@k balances both aspects. Hits@k reflects whether at least one correct answer appears in the top-k, and MRR evaluates how early the first correct answer is retrieved.

For simple questions ($N = 25$), the performance is consistently near perfect, with P @ 10 = 100%, R @ 10 (89. 6%) and F1 @ 10(93. 3%), along with Hits @ 10 = 100% and MRR = 100%. For medium questions ($N = 20$), P @ 10 = 95%, R @ 10 (89%) and F1 @ 10 (91. 4%), with a strong ranking quality reflected in Hits@10 = 95% and MRR = 95%. For complex queries ($N = 15$), the performance remains robust but slightly lower, with P @ 10 (93. 3%), R @ 10 (88%) and F1 @ 10 (90. 2%), alongside Hits @ 10 (93. 3%) and MRR 93. 3%.

In general, at all levels, the system achieves P @ 10 (96. 1%), R @ 10 (88. 9%) and F1 @ 10 (91. 6%), with Hits @ 10 (96. 1%) and MRR 96. 1%. These results indicate that the system not only retrieves nearly all expected answers but also ranks them highly, ensuring both completeness and correctness. This level of reliability is particularly valuable in biomedical applications where precision and trustworthiness are critical.

The main purpose of this experiment was to assess not only whether queries execute, but whether the returned results are both correct and complete. The precision consistently above 95% confirms that almost all the items retrieved are clinically valid, while Hits@5 near 95% shows that the system usually returns close to the expected five answers per question. Together, these metrics demonstrate that knowledge graph grounding effectively minimizes hallucinations and ensures trustworthy biomedical output.

We evaluated performance differences across difficulty levels using a two-sample t-test, which revealed statistically significant differences. Error analysis indicates that failures in complex queries are mainly due to missing relation hops, whereas errors in medium-level queries are mostly caused by syntax mismatches. These findings highlight the challenges associated with query complexity and provide insights for targeted improvements.

C. Qualitative Evaluation

In addition to quantitative metrics, the system's outputs were evaluated for contextual accuracy and alignment with the structure of the knowledge graph. Two plain English questions were selected and for each, the generated Cypher query, the Neo4j output, and the visual graph representation were reviewed to verify that the answers reflected valid relationships in the graph.

For the question 'What are the symptoms of brain cancer?' The system generated a Cypher query that correctly followed the HAS_SYMPTOM relationship from disease nodes to symptoms nodes, filtering by the specified disease name. The results retrieved included terms such as aphasia (Broca, Wernicke), anomia, agraphia, agnosia, amnesia (including retrograde amnesia), olfaction disorders, and apnea symptoms consistent with established neurological manifestations of brain tumors. In Neo4j, the data formed a clear center-and-spoke pattern, with brain cancer at the center and its associated nodes of symptoms radiating outward, as shown in Fig. 5.

Cypher:
```
WITH toLower("brain cancer") AS disease_name
MATCH (d:Disease)-[:HAS_SYMPTOM]->(s:Symptom)
WHERE toLower(d.name) CONTAINS disease_name
RETURN DISTINCT s,d
LIMIT 10;
```

Fig. 5. Graphical user interface displaying Cypher query and results for breast cancer treatment

A second query, What are the side effects of insulin?, produced a Cypher statement starting from the drug node for insulin and traversing the CAUSES relationship to the connected SideEffect nodes. The resulting list included dizziness, diarrhea, cough, back pain, weakness, rash/dermatitis, and hypersensitivity side effects well documented in insulin safety profiles. In the visual representation in Fig. 6, insulin appeared centrally connected to these side effect nodes, further strengthening the correctness of the relationship mapping of the system.

Cypher:
```
WITH toLower("insulin") AS drug_name
MATCH (d:Drug)-[:CAUSES]->(se:SideEffect)
WHERE toLower(d.name) CONTAINS drug_name
RETURN se,d
LIMIT 20;
```

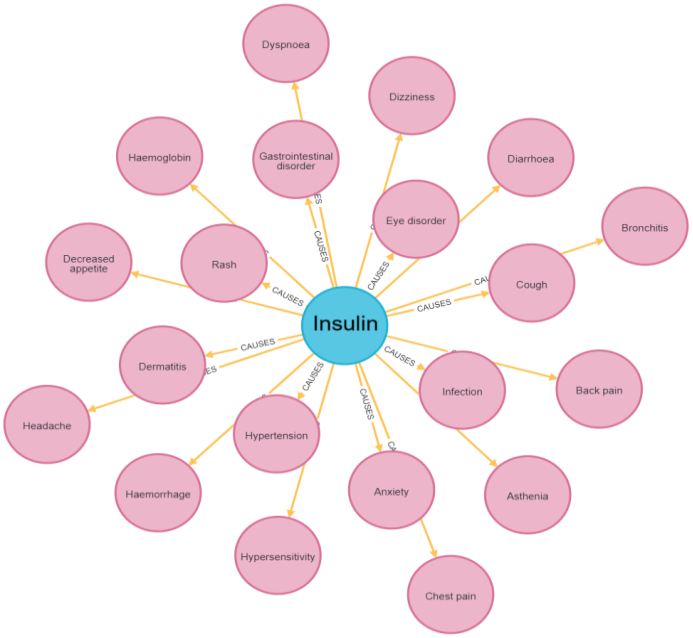

Fig. 6. Knowledge Graph Visualization of Insulin and Its Related Side Effects

These examples demonstrate the system's ability to interpret plain English biomedical questions, generate correct Cypher queries, and return results that are clinically plausible, easy to interpret, and directly traceable to specific graph relationships. This supports both the accuracy and the transparency of the proposed approach.

To illustrate how biomedical questions can be translated into graph queries, consider the natural language question: 'What are the side effects of drugs that treat epilepsy?'. The following Cypher query demonstrates how this question is mapped in the knowledge graph, where diseases are connected to drugs through the TREATS relation and drugs are connected to side effects through the CAUSES relation.

Cypher:
```
MATCH (d:Disease)<-[:TREATS]-(dr:Drug)
WHERE toLower(d.name) CONTAINS "epilepsy"
MATCH (dr:Drug)-[:CAUSES]->(se:SideEffect)
RETURN DISTINCT se,d,dr
limit 10
```

Fig. 7. Side effects of drugs that treat epilepsy

Executing this query returns drugs associated with epilepsy and their corresponding side effects. For example, the query identifies Pregabalin as a treatment for epilepsy and retrieves multiple side effects such as anxiety, arrhythmia, gastrointestinal pain, and others. The visualization of the graph in Fig. 7, highlights this pattern, where the drug node is linked to epilepsy via TREATS and to several side effect nodes via CAUSES, providing an interpretable biomedical knowledge structure.

VI. Discussion

Under a unified evaluation protocol reporting **P@k/R@k/F1@k**, **Hits@k**, and **MRR** at $k \in \{1,5,10\}$, the pipeline exhibits consistent end-to-end behavior across the three difficulty tiers. The k-ablation shows strong early precision (high P@1 and elevated MRR), while recall increases with larger k, indicating that correct answers typically surface near the top yet persist deeper in the ranked slate. At the query-generation level, exact-match (EM) is high for simple questions (e.g., ∼96%) and lower for complex, multi-hop questions (e.g., ∼86.7%), which aligns with increased compositionality. End-to-end latency (mean total ≈6.07 s from question to final answer) remains within interactive bounds on our local setup. Practically, a fully offline deployment avoids internet connectivity, API keys, and external data transfer, strengthening privacy, compliance, and reproducibility in biomedical contexts.

Several challenges qualify these findings. *First, the comparability challenge:* to our knowledge, no prior study evaluates **iBKH** using the same metric suite, making cross-paper numeric comparisons not "apples to apples." We therefore interpret the results as controlled evidence about this pipeline under a single, consistent protocol rather than as a cross-study leaderboard. Beyond comparability, performance may vary with other biomedical knowledge graphs; the current iBKH snapshot is static, limiting real-time updates; and scaling to larger or dynamically refreshed graphs can introduce latency and consistency trade-offs.

Error analysis shows that residual failures concentrate in complex, multi-hop queries where missing relation hops or brittle name-based matching (synonyms, abbreviations, homonyms) lead to partial answers. These observations motivate concept-level normalization via biomedical identifiers (e.g., UMLS, SNOMED, RxNorm) with synonym expansion, as well as schema-constrained query generation and path-guided decoding to better satisfy multi-hop constraints. Finally, correctness is assessed primarily at the *answer level* (Hits@k, MRR, precision/recall) and does not yet include full *semantic-equivalence* checks across alternative Cypher queries, which may overlook cases where different queries yield the same correct results. Complementing template EM with *result-set equivalence* checks, expanding the metric suite (e.g., nDCG/MAP) with per-question 95% bootstrap confidence intervals, and supporting incremental graph updates with distributed storage are promising steps to enhance robustness, scalability, and external validity.

VII. Conclusion

This study introduced a hybrid biomedical question answering framework that couples the LLaMA-3B language model with a Neo4j-based iBKH knowledge graph to enable the automatic generation of executable Cypher queries and to deliver transparent, evidence-grounded answers through justification subgraphs. Evaluation in simple, medium, and complex queries demonstrated consistently high performance, with strong precision, recall, F1, Hits@k, and MRR values, while maintaining low latency suitable for interactive biomedical applications. Beyond quantitative performance, the proposed system provides a reproducible and privacy-preserving solution by operating fully offline, a property of particular importance in sensitive clinical and research contexts. However, certain limitations remain. The reliance on a static iBKH snapshot constrains coverage and adaptability, recall is lower for complex multi-hop reasoning, and the absence of canonical entity normalization (e.g., UMLS, SNOMED, RxNorm) may reduce semantic robustness. Future research will therefore focus on integrating standardized biomedical entity normalization, enabling dynamic and incremental knowledge graph updates, and leveraging domain-adapted or fine-tuned biomedical LLMs. These directions are expected to further strengthen the robustness, scalability, and applicability of the framework in real-world biomedical and clinical environments.

References

[1] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, *et al.*, "Sparks of artificial general intelligence: Early experiments with gpt-4," *arXiv preprint arXiv:2303.12712*, 2023.

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[3] Y. Hou, J. Yeung, H. Xu, C. Su, F. Wang, and R. Zhang, "From answers to insights: unveiling the strengths and limitations of chatgpt and biomedical knowledge graphs," *Research Square*, pp. rs–3, 2023.

[4] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[5] C. Malaviya, S. Lee, S. Chen, E. Sieber, M. Yatskar, and D. Roth, "Expertqa: expert-curated questions and attributed answers," *arXiv preprint arXiv:2309.07852*, 2023.

[6] L. Pusch and T. O. Conrad, "Combining llms and knowledge graphs to reduce hallucinations in question answering," *arXiv preprint arXiv:2409.04181*, 2024.

[7] H. Nori, N. King, S. M. McKinney, D. Carignan, and E. Horvitz, "Capabilities of gpt-4 on medical challenge problems," *arXiv preprint arXiv:2303.13375*, 2023.

[8] T. Sekar, Kushal, S. Shankar, S. Mohammed, and J. Fiaidhi, "Investigations on using evidence-based graphrag pipeline using llm tailored for usmle style questions," *medRxiv*, pp. 2025–05, 2025.

[9] S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, "Detecting hallucinations in large language models using semantic entropy," *Nature*, vol. 630, no. 8017, pp. 625–630, 2024.

[10] E. Asgari, N. Montaña-Brown, M. Dubois, S. Khalil, J. Balloch, J. A. Yeung, and D. Pimenta, "A framework to assess clinical safety and hallucination rates of llms for medical text summarisation," *npj Digital Medicine*, vol. 8, no. 1, p. 274, 2025.

[11] C. Su, Y. Hou, M. Zhou, S. Rajendran, J. R. Maasch, Z. Abedi, H. Zhang, Z. Bai, A. Cuturrufo, W. Guo, *et al.*, "Biomedical discovery through the integrative biomedical knowledge hub (ibkh)," *Iscience*, vol. 26, no. 4, 2023.

[12] J. H. Morris, K. Soman, R. E. Akbas, X. Zhou, B. Smith, E. C. Meng, C. C. Huang, G. Cerono, G. Schenk, A. Rizk-Jackson, *et al.*, "The scalable precision medicine open knowledge engine (spoke): a massive knowledge graph of biomedical information," *Bioinformatics*, vol. 39, no. 2, p. btad080, 2023.

[13] K. Soman, P. W. Rose, J. H. Morris, R. E. Akbas, B. Smith, B. Peetoom, C. Villouta-Reyes, G. Cerono, Y. Shi, A. Rizk-Jackson, *et al.*, "Biomedical knowledge graph-optimized prompt generation for large language models," *Bioinformatics*, vol. 40, no. 9, p. btae560, 2024.

[14] F. Frau, P. Loustalot, M. Törnqvist, N. Temam, J. Cupe, M. Montmerle, and F. Augé, "Connecting electronic health records to a biomedical knowledge graph to link clinical phenotypes and molecular endotypes in atopic dermatitis," *Scientific Reports*, vol. 15, no. 1, p. 3082, 2025.

[15] Y. Gao, R. Li, E. Croxford, J. Caskey, B. W. Patterson, M. Churpek, T. Miller, D. Dligach, and M. Afshar, "Leveraging medical knowledge graphs into large language models for diagnosis prediction: design and application study," *Jmir AI*, vol. 4, p. e58670, 2025.

[16] Y. Yan, Y. Hou, Y. Xiao, R. Zhang, and Q. Wang, "Knownet: guided health information seeking from llms via knowledge graph integration," *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[17] Y. Deng, S. Zhao, Y. Miao, J. Zhu, and J. Li, "Medka: a knowledge graph-augmented approach to improve factuality in medical large language models," *Journal of Biomedical Informatics*, p. 104871, 2025.

[18] L. Ehrlinger and W. Wöß, "Towards a definition of knowledge graphs," *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, no. 1–4, p. 2, 2016.

[19] E. Rajabi and S. Kafaie, "Building a disease knowledge graph," in *Caring is Sharing – Exploiting the Value in Data for Health and Innovation*, pp. 701–705, IOS Press, 2023.

[20] L. Guan, Y. Huang, and J. Liu, "Biomedical question answering via multi-level summarization on a local knowledge graph," *arXiv preprint arXiv:2504.01309*, 2025.

[21] D. Steinigen, R. Teucher, T. H. Ruland, M. Rudat, N. Flores-Herr, P. Fischer, N. Milosevic, C. Schymura, and A. Ziletti, "Fact finder – enhancing domain expertise of large language models by incorporating knowledge graphs," *arXiv preprint arXiv:2408.03010*, 2024.

[22] Y. Feng, L. Zhou, C. Ma, Y. Zheng, R. He, and Y. Li, "Knowledge graph–based thought: a knowledge graph–enhanced llm framework for pan-cancer question answering," *GigaScience*, vol. 14, p. giae082, 2025.

[23] H. Luo, Z. Tang, S. Peng, Y. Guo, W. Zhang, C. Ma, G. Dong, M. Song, W. Lin, Y. Zhu, *et al.*, "Chatkbqa: a generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models," *arXiv preprint arXiv:2310.08975*, 2023.

[24] A. Tiwari, S. K. R. Malay, V. Yadav, M. Hashemi, and S. T. Madhusudhan, "Auto-cypher: improving llms on cypher generation via llm-supervised generation-verification framework," in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 623–640, 2025.

[25] R. Wang, Z. Zhang, L. Rossetto, F. Ruosch, and A. Bernstein, "Nlqxform: a language model-based question to sparql transformer," *arXiv preprint arXiv:2311.07588*, 2023.

[26] M. R. Rezaei, R. S. Fard, J. L. Parker, R. G. Krishnan, and M. Lankarany, "Agentic medical knowledge graphs enhance medical question answering: bridging the gap between llms and evolving medical knowledge," *arXiv preprint arXiv:2502.13010*, 2025.

[27] Z. Dong, B. Peng, Y. Wang, J. Fu, X. Wang, Y. Shan, and X. Zhou, "Effiqa: efficient question-answering with strategic multi-model collaboration on knowledge graphs," *arXiv preprint arXiv:2406.01238*, 2024.

[28] Y. Duan, Q. Zhou, Y. Li, C. Qin, Z. Wang, H. Kan, and J. Hu, "Research on a traditional chinese medicine case-based question-answering system integrating large language models and knowledge graphs," *Frontiers in Medicine*, vol. 11, p. 1512329, 2025.

[29] S. Mohammed, J. Fiaidhi, T. Sekar, K. Kushal, and S. Shankar, "Investigations on using evidence-based graphrag pipeline using llm tailored for answering usmle medical exam questions," *medRxiv*, pp. 2025–05, 2025.

[30] H. Yang, J. Li, C. Zhang, A. P. Sierra, and B. Shen, "Large language model–driven knowledge graph construction in sepsis care using multicenter clinical databases: development and usability study," *Journal of Medical Internet Research*, vol. 27, p. e65537, 2025.

[31] K.-L. Hsieh, G. Plascencia-Villa, K.-H. Lin, G. Perry, X. Jiang, and Y. Kim, "Synthesize heterogeneous biological knowledge via representation learning for alzheimer's disease drug repurposing," *Iscience*, vol. 26, no. 1, 2023.

[32] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, pp. 1–39, 2008.

[33] B. Chicho and A. O. Mohammed, "An empirical comparison of neo4j and tigergraph databases for network centrality," *Science Journal of University of Zakho*, vol. 11, no. 2, pp. 190–201, 2023.

[34] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases: New Opportunities for Connected Data*, O'Reilly Media, 2015.

[35] A. Lysenko, I. A. Roznovăţ, M. Saqi, A. Mazein, C. J. Rawlings, and C. Auffray, "Representing and querying disease networks using graph databases," *BioData Mining*, vol. 9, no. 1, p. 23, 2016.

[36] M. Šestak, M. Heričko, T. W. Družovec, and M. Turkanović, "Applying k-vertex cardinality constraints on a neo4j graph database," *Future Generation Computer Systems*, vol. 115, pp. 459–474, 2021.

[37] M. Desai, R. G. Mehta, and D. P. Rana, "An empirical analysis to identify the effect of indexing on influence detection using graph databases," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 9S, pp. 414–421, 2019.

[38] S. Beis, S. Papadopoulos, and Y. Kompatsiaris, "Benchmarking graph databases on the problem of community detection," in *New Trends in Database and Information Systems II*, pp. 3–14, Springer, 2015.

[39] R. Wang, Z. Yang, W. Zhang, and X. Lin, "An empirical study on recent graph database systems," in *International Conference on Knowledge Science, Engineering and Management*, pp. 328–340, Springer, 2020.

[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 4171–4186, 2019.

[41] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[42] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[43] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. McDermott, "Publicly available clinical bert embeddings," *arXiv preprint arXiv:1904.03323*, 2019.

[44] Z. He, S. Sunkara, X. Zang, Y. Xu, L. Liu, N. Wichers, G. Schubiner, R. Lee, and J. Chen, "Actionbert: leveraging user actions for semantic understanding of user interfaces," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 5931–5938, 2021.