# Expansion of pre-existing Knowledge Graphs using LLMs

*A Project Report*

*submitted by*

**KARTIKAY JOSHI**

**MT23230**

*in partial fulfilment of the requirements*

*for the award of the degree of*

**MASTER OF TECHNOLOGY**

Department of Computational Biology

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

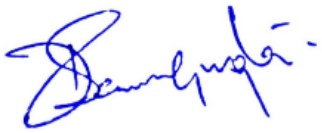
NEW DELHI- 110020

July 1st, 2025

# Certificate

This is to certify that the thesis titled **"Expansion of pre-existing Knowledge Graphs using LLMs"** being submitted by **Kartikay Joshi** to the Indraprastha Institute of Information Technology, Delhi, for the award of the Master of Technology, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.



July 2025

**Dr Debarka Sengupta**

Department of Computational Biology

Indraprastha Institute of Information Technology Delhi

New Delhi 110020

# Acknowledgements

I am genuinely thankful to everyone who supported me throughout the journey of completing my thesis. I owe my deepest thanks to my advisor, **Dr. Debarka Sengupta**, whose guidance and encouragement were constant sources of motivation. His thoughtful feedback and deep expertise helped steer my research in the right direction and greatly enhanced its overall quality. It was a privilege to work under his mentorship. I'm also very grateful to **Swarnava Samanta**, my senior and mentor, for his technical advice and patient guidance. His readiness to help and share his knowledge made a big difference, especially during the more challenging phases of the project. I would also like to thank my friend, **Rahul Kharkwal**, for always being there with support and encouragement. In addition, I appreciate everyone who offered help, shared knowledge, or provided resources along the way. To all those mentioned and many others who've been part of this academic journey—thank you. Your presence and support truly made a difference.

**Kartikay Joshi**
**MT23230**

# Abstract

The rapid surge in unstructured biomedical literature has made it increasingly difficult to keep knowledge graphs up to date—yet these graphs are essential tools for driving forward scientific research. Manually curating such data is not only time-consuming but also unsustainable at scale. In this thesis, I tackle this issue by developing an automated pipeline that uses Large Language Models (LLMs) to expand biomedical knowledge graphs more efficiently.

The approach begins with building a base version of the BioKG using structured data from TSV files. This base is then enriched with additional information drawn from unstructured text. The pipeline brings in bulk data from PubTator, filters it to retain human-specific content, and stores it in MongoDB for further processing. A key element of the system is an LLM-based extraction module, powered by GPT-4o-mini, which uses carefully crafted prompts aligned with the BioKG schema to identify and extract new entities and relationships. To maintain consistency and avoid redundancy, a validation and ingestion module integrates this data into a Neo4j graph database, ensuring the final graph remains accurate and cohesive.

The results show a clear increase in both the number of entities and relationships in the graph. Additionally, an interactive visualization tool highlights the impact of the updates, providing qualitative insights into the improvements. This project offers a practical, scalable framework for continuously updating biomedical knowledge graphs—an important step toward making them more useful for research and healthcare purposes.

# Contents

# Chapter 1: Introduction

This chapter sets the stage for the research in this thesis. We begin by outlining how the explosion of biomedical data—especially unstructured text—creates both opportunity and challenge, and why Knowledge Graphs (KGs) have become a go-to way to tame that complexity. We then pinpoint the main issue at hand: current methods simply can't keep these graphs up to date as the literature grows. Finally, we lay out the precise goals this work pursues to solve that problem.

## 1.1 Background and Motivation

Biomedical research today produces an overwhelming flood of information—from PubMed articles to clinical notes in Electronic Health Records. Disciplines like genomics, proteomics, pharmacology, and clinical informatics each contribute massive amounts of text. While this "data deluge" holds the promise of breakthroughs in precision medicine, drug repurposing, and deeper understanding of diseases, it also makes it nearly impossible to sift and structure knowledge manually.

That's where Knowledge Graphs come in. By turning entities (genes, proteins, drugs, diseases, biological processes) into nodes and their interactions (e.g., drug–disease associations, drug–target links, protein–protein interactions) into edges, KGs offer a semantic network that computers can query, reason over, and integrate with other datasets. This structured format powers:

- **Semantic queries**, such as finding genes linked to both diabetes and cardiovascular conditions
- **Inference and hypothesis generation**, uncovering indirect connections not obvious in isolation
- **Data integration**, merging results from diverse sources under a unified schema
- **Machine-learning workflows**, for tasks like link prediction or patient stratification

Well-maintained KGs have already driven advances in systems biology, pharmacovigilance, and personalized therapeutics. Yet their ongoing usefulness depends on keeping pace with new discoveries—a task that manual curation alone simply can't manage.

## 1.2 Problem Statement

Despite their value, most biomedical KGs suffer from staleness and gaps. Populating and updating these graphs today relies heavily on experts manually annotating papers or semi-automated tools that still demand extensive human oversight. This creates several roadblocks:

- **Scalability issues:** New publications appear faster than curators can annotate them
- **High costs:** Recruiting domain specialists for continuous curation is expensive
- **Inconsistency:** Different curators may interpret the same text in varying ways
- **Obsolescence:** Graphs quickly fall behind, missing the latest findings

Traditional rule-based or statistical NLP methods often struggle with the nuance and complexity of biomedical language unless you invest heavily in feature engineering. Meanwhile, the rise of Large Language Models (LLMs) trained on vast text corpora offers a new opportunity: these models can understand context, spot entities, and tease out complex relationships with minimal manual tweaking.

**Central research question:**
How can we build an intelligent, automated pipeline—driven by LLMs and integrated with graph databases—to pull novel biomedical entities and relationships from unstructured literature, then fold them into existing Knowledge Graphs in a dynamic, scalable, and validated way?

### 1.3 Research Objectives

To answer this question, this thesis sets out to:

1. **Data acquisition and preparation:** Ingest and preprocess human-related biomedical text from the PubTator dataset.
2. **LLM-based extraction:** Design a framework that uses a predefined schema and carefully crafted prompts to pull structured entities and relationships from text.
3. **Validation module:** Check whether newly extracted nodes and edges already exist in a Neo4j graph, preventing duplicates.
4. **Graph integration:** Automate the insertion of validated entities and relationships into the Neo4j database.
5. **Proof of concept:** Showcase the pipeline's effectiveness on a representative subset, demonstrating clear expansion of the Knowledge Graph from unstructured sources.


**Fig1: BioKG in neo4j**

# Chapter 2: Literature Review

## 2.1 Introduction

In this chapter, we survey the key literature that underpins our work. We'll begin by outlining the core ideas behind Knowledge Graphs, then review established and emerging methods for pulling structured information from text. Next, we'll look at recent advances in Large Language Models (LLMs) and how they've been applied to biomedical NLP and graph population. Finally, we'll discuss the graph-database technologies that store and manage these networks. By the end, you'll see where prior efforts fall short—and how our thesis addresses those gaps.

## 2.2 Knowledge Graphs in Biomedical Research

Knowledge Graphs have become indispensable for modeling complex biological systems and their interactions. Unlike flat tables, a graph captures not just entities—like genes, proteins, diseases, and drugs—but also the rich web of relationships between them.

### 2.2.1 Fundamentals of Knowledge Graphs

At its heart, a Knowledge Graph is a directed network:

- **Nodes** represent entities (e.g., 'TP53' as a gene, 'aspirin' as a drug, or 'inflammation' as a biological process).
- **Edges** capture the semantic ties between those entities (for example, a *drug_target* edge linking aspirin to COX-1, or a *protein_protein_interaction* between two signaling proteins).

Both nodes and edges can carry attributes—like confidence scores, provenance, or detailed annotations—making the graph a rich, machine-readable resource. This structure enables:

- **Complex queries**, such as "Which drugs target proteins involved in both Alzheimer's and Parkinson's?"
- **Inferential reasoning**, surfacing hidden links and new hypotheses.
- **Data integration**, by merging information from multiple sources under a unified schema.
- **Machine-learning inputs**, since KGs power algorithms for link prediction, node classification, and more.

### 2.2.2 Prominent Biomedical Knowledge Graphs

Several specialized KGs have shaped research and applications in biomedicine:

- **Gene Ontology (GO):** A hierarchical map of gene functions.
- **DrugBank:** Detailed drug–target interactions and pharmacological data.
- **DisGeNET:** Gene and variant associations with diseases.
- **UMLS (Unified Medical Language System):** A meta-thesaurus unifying diverse biomedical vocabularies.
- **STRING:** A comprehensive network of protein–protein interactions.

- **ChEMBL:** Bioactivity data for small molecules.
- **Hetionet:** A heterogeneous graph pulling from 29 public resources, linking compounds, genes, diseases, pathways, and more.

**Common applications** of these KGs include:

- **Drug discovery & repositioning:** Finding new targets or new uses for existing compounds.
- **Disease pathway analysis:** Mapping molecular cascades behind complex disorders.
- **Precision medicine:** Tailoring treatments based on individual genetic and clinical profiles.
- **Biomarker identification:** Pinpointing molecular signals that track disease progression or treatment response.

### 2.2.3 Challenges in Knowledge Graph Construction and Maintenance

Despite their undeniable utility, the construction and, critically, the ongoing maintenance of biomedical KGs present significant challenges. The primary hurdle is the **scalability issue** associated with knowledge acquisition. The vast majority of new scientific findings are published in unstructured text (journal articles, patents), requiring manual or semi-automated processes for extraction. This **manual curation bottleneck** is inherently:

- **Labor-Intensive and Costly:** Requiring highly trained domain experts.
- **Time-Consuming:** Leading to a substantial delay between knowledge publication and its integration into structured KGs.
- **Prone to Inconsistency:** Different human annotators may introduce variability.
- **Non-Scalable:** Unable to keep pace with the exponential growth of scientific literature (e.g., millions of new PubMed abstracts annually).

Consequently, existing KGs often become **stale and incomplete**, failing to reflect the latest scientific advancements. The development of robust, automated, and scalable methods for continuous KG expansion is thus a pressing need in computational biology.

## 2.3 Information Extraction from Unstructured Text

Information Extraction (IE) is a subfield of Natural Language Processing (NLP) focused on automatically identifying structured information from unstructured or semi-structured text. The goal is to transform free text into structured data (e.g., entities, relationships, events) that can be queried and analyzed computationally.

### 2.3.1 Traditional Information Extraction Techniques

Historically, IE systems have evolved through several stages:

- **Rule-Based Systems:** These rely on handcrafted linguistic rules, patterns (e.g., regular expressions), and dictionaries to identify specific entities and relationships. While offering high precision in specific domains, they are costly to develop, difficult to scale, and brittle when faced with linguistic variations.

- **Statistical NLP and Machine Learning Models:** With the rise of statistical methods, approaches like Hidden Markov Models (HMMs), Conditional Random Fields (CRFs), and Support Vector Machines (SVMs) became popular for tasks like Named Entity Recognition (NER) and Relation Extraction (RE). These models learn patterns from annotated corpora, offering better generalization than rule-based systems but still requiring extensive feature engineering.
- **Deep Learning Approaches (Pre-LLM Era):** The advent of deep learning, particularly recurrent neural networks (RNNs) and convolutional neural networks (CNNs), marked a significant leap. These models could automatically learn features from text, reducing the need for manual feature engineering. For NER, Bi-directional LSTMs with CRFs (Bi-LSTM-CRF) became a standard. For RE, models that learned representations for entity pairs and their surrounding context gained prominence.

### 2.3.2 Challenges of Information Extraction in the Biomedical Domain

Information extraction from biomedical text presents unique and substantial challenges compared to general-domain NLP:

- **Complex Terminology:** Biomedical texts are replete with highly specialized, often lengthy, and frequently ambiguous terms (e.g., "depression" can refer to a clinical condition or a physical indentation).
- **Nomenclature Variation:** Entities can be referred to by full names, abbreviations, acronyms, synonyms, or different spellings, making consistent identification difficult.
- **Nested Entities:** Entities often appear nested within other entities (e.g., "treatment of [glioblastoma multiforme] with [Temozolomide]").
- **Complex Sentence Structures:** Scientific writing often employs long, convoluted sentences with multiple clauses, obscuring relationships.
- **High-Dimensionality and Sparsity:** The sheer number of unique biomedical terms leads to sparse data representations.
- **Limited Annotated Data:** Creating high-quality, domain-specific annotated corpora for training is extremely expensive and time-consuming, hindering the development of supervised learning models.

These challenges make it clear that information-extraction systems must grasp subtle semantic nuances and adapt well even when only a few examples are available.

## 2.4 Evolution and Capabilities of Large Language Models (LLMs)

The NLP landscape has been transformed by the rise of Large Language Models.

### 2.4.1 From Statistical Models to Transformers

Early efforts at language modeling relied on statistical approaches (like n-grams) or shallow neural nets. The big breakthrough came in 2017, when Vaswani et al. introduced the Transformer architecture[1]. By using self-attention to process all tokens in parallel, Transformers captured long-distance dependencies far better than RNNs ever could. This innovation led directly to powerful pre-trained models such as

BERT, which learns deep, bidirectional representations through tasks like masked-language modeling and next-sentence prediction. Soon after, OpenAI's GPT series pushed the envelope further: by training massive, autoregressive (left-to-right) models on terabytes of text, they unlocked astonishing generative and reasoning abilities.

### 2.4.2 Key Strengths of Modern LLMs

Today's large models bring capabilities that traditional systems simply lacked:

- **Contextual Understanding:** They decipher word meanings by looking at surrounding text, not just fixed dictionaries.
- **Semantic Reasoning:** Without hand-coded rules, they can infer relationships and draw logical conclusions.
- **Zero- and Few-Shot Learning:** Having seen vast, varied data, they tackle new tasks with minimal—or even no—task-specific examples, relying on the instructions given in prompts.
- **Instruction Following:** They reliably interpret and carry out complex, multi-step commands phrased in natural language.
- **High-Quality Generation:** Beyond analysis, they produce clear, coherent text—ideal for formatting and presenting extracted insights.

These capabilities position LLMs as transformative tools for tasks requiring deep linguistic comprehension and flexible information processing, including sophisticated information extraction.

## 2.5 LLMs for Knowledge Extraction and Graph Population

The remarkable capabilities of LLMs have naturally led to their application in automating knowledge extraction and populating knowledge graphs, particularly in complex domains like biomedicine.

### 2.5.1 LLMs in Biomedical Natural Language Processing (NER, RE)

LLMs, especially those further fine-tuned on biomedical corpora (e.g., BioBERT, SciBERT, ClinicalBERT), have significantly pushed the state-of-the-art in biomedical NLP tasks.

- **Named Entity Recognition (NER):** LLMs excel at identifying biomedical entities (drugs, genes, diseases, chemicals) from text, often outperforming previous deep learning models due to their superior contextual understanding and ability to handle linguistic variations.
- **Relation Extraction (RE):** Beyond identifying entities, LLMs are increasingly used to extract semantic relationships between them. Researchers have explored various approaches, including:
    - **Fine-tuning:** Training an LLM on a dataset of labeled entity pairs and their relations.
    - **Prompt-based/In-context Learning:** Formulating the RE task as a natural language instruction within the prompt, allowing the LLM to extract relations without explicit fine-tuning, leveraging its zero-shot or few-shot capabilities. This is particularly valuable given the scarcity of large, annotated biomedical relation extraction datasets.

### 2.5.2 Strategies for Controlled Information Extraction with LLMs

A key challenge in using generative LLMs for structured data extraction is ensuring that their output adheres to a predefined schema and avoids "hallucinations" (generating factually incorrect or unsupported information). Several strategies have emerged:

- **Prompt Engineering:** Carefully crafted prompts are crucial. These prompts often:
  - Clearly define the task (e.g., "Extract all 'Disease' and 'Drug' entities and their 'treats' relationships").
  - Specify the desired output format (e.g., JSON schema).
  - Provide examples (few-shot learning) to guide the LLM's understanding of the task and output format.
  - Include constraints (e.g., "Only use the provided relation types: [TYPE1, TYPE2]").
- **Schema-guided Extraction:** Integrating the target KG's schema (node types, relation types, properties) directly into the prompt guides the LLM to extract information compatible with the graph's structure, thereby ensuring data consistency upon ingestion.
- **Retrieval Augmented Generation (RAG):** While not explicitly described in your workflow for *extraction*, RAG approaches (where LLMs retrieve relevant documents before generating answers) are also being explored to ground LLM responses in factual evidence, reducing hallucination.

### 2.5.3 Challenges and Considerations for LLM-Based Knowledge Extraction

While LLMs offer exciting possibilities, they also bring their own hurdles:

- **Hallucinations & Accuracy**
  Models sometimes “make up” facts or misremember details—especially when asked to extrapolate beyond what they’ve seen or when prompts are vague.
- **Cost & Throughput**
  Large-model API calls can get expensive and introduce delays, which adds up when you’re processing millions of documents.
- **Inherited Bias**
  Any skew or blind spot in the training data can surface in the facts and relationships the model pulls out.
- **Explainability**
  It’s often hard to trace exactly why the model extracted a given entity or link—deep networks tend to act like black boxes.
- **Evolving Schemas**
  Although LLMs adapt more easily than rigid parsers, when your graph schema changes you still need to tweak prompts to match new node and edge types.

Because of these issues, a robust pipeline must include strong validation steps (to catch errors and filter out bad or duplicated extractions) and carefully designed, schema-aware prompts.

## 2.6 Graph Databases for Knowledge Representation

Picking the right storage engine is key when you're dealing with highly connected biomedical data. Graph databases shine here by treating relationships as first-class citizens.

### 2.6.1 Why Graph Databases?

- **Native Graph Model**
  Data is stored directly as nodes and edges—no complex SQL joins needed when you want to hop across relationships.
- **Flexible Schemas**
  You can add new types of nodes or properties without redesigning the whole database.
- **Intuitive Mapping**
  Biological networks (e.g., signaling pathways, drug–target maps) naturally map to a graph structure.
- **Built-In Analytics**
  Out-of-the-box algorithms for shortest paths, community detection, centrality measures, and more make deep network analysis straightforward.

### 2.6.2 Neo4j in Biomedicine

Neo4j is one of the most popular native graph databases, thanks to:

- **Property Graph Model**
  Both nodes and relationships can carry rich metadata—perfect for storing confidence scores, evidence links, or timestamps.
- **Cypher Query Language**
  A declarative, SQL-like syntax optimized for expressing complex graph traversals and pattern matching.
- **Proven Track Record**
  Researchers have used Neo4j to manage drug–target networks, disease–symptom mappings, gene regulatory circuits, and more—often mixing clinical and genomic data in the same graph.
- **Scalability & Community**
  It scales well for medium-sized graphs and enjoys broad community support, making it a reliable engine for dynamically growing biomedical KGs.

## 2.7 Summary and Research Gap

The literature review highlights the critical need for comprehensive and up-to-date biomedical knowledge graphs to accelerate scientific discovery. While existing KGs are valuable, their expansion is severely hindered by the limitations of manual curation and the complexities of extracting structured

information from vast, unstructured biomedical literature. Traditional IE methods struggle with the inherent linguistic challenges and semantic nuances of the domain.

The emergence of Large Language Models, particularly Transformer-based architectures, has revolutionized NLP by offering unprecedented capabilities in understanding and generating human language, including robust entity and relation extraction. Recent efforts have demonstrated the potential of LLMs in biomedical NLP tasks, but there remains a significant gap in creating a **fully automated, end-to-end, and schema-guided pipeline that leverages LLMs for the continuous and consistent expansion of existing biomedical knowledge graphs in a scalable manner.** Previous works often focus on specific IE tasks or smaller-scale graph construction, or they lack a robust integration strategy that ensures consistency with an existing graph schema.

This thesis aims to bridge this gap by designing and implementing a comprehensive pipeline that addresses the entire workflow from raw data ingestion and LLM-powered extraction (constrained by predefined relation types) to intelligent validation and seamless integration into a production-grade graph database (Neo4j). This integrated approach provides a novel solution to the challenge of dynamically maintaining current and comprehensive biomedical knowledge graphs.

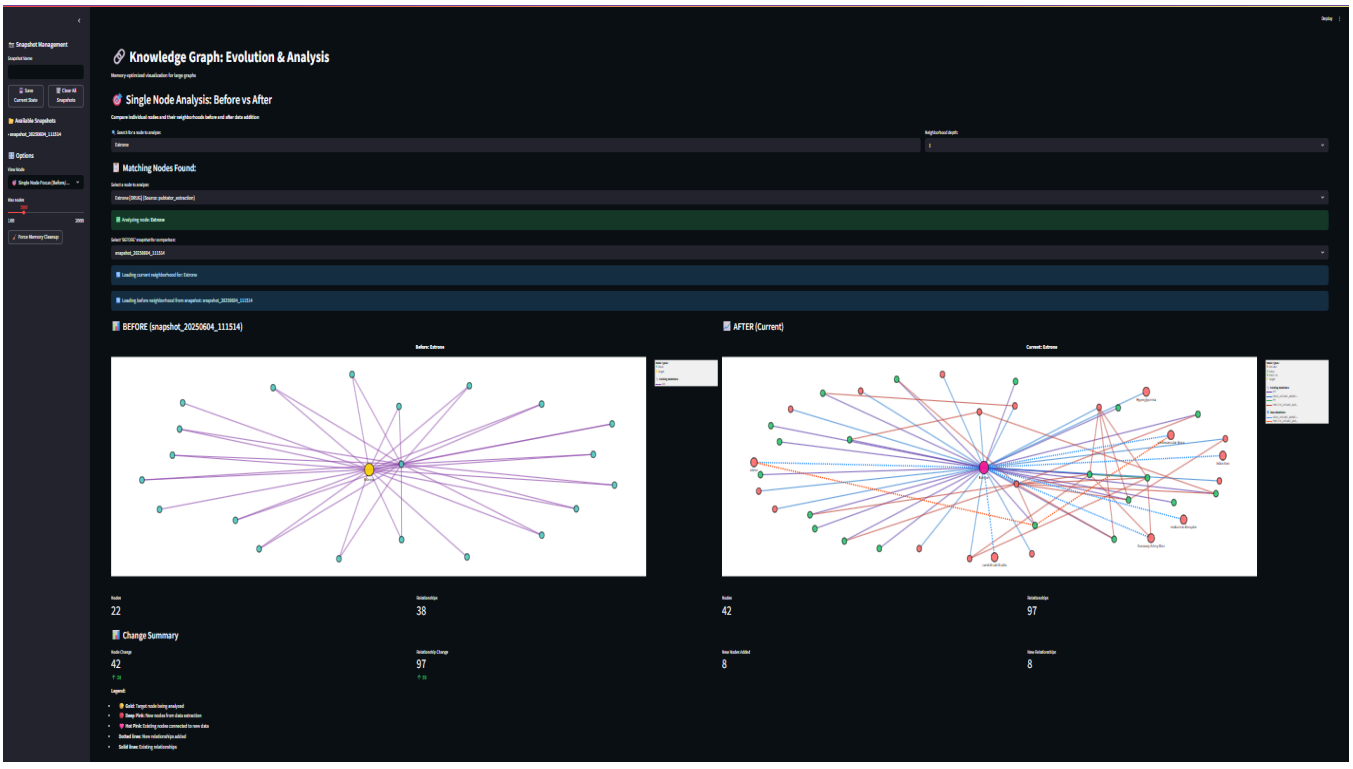



**Fig2: Interactive User Interface for Visualizing Knowledge Graph Expansion Outcomes**

# Chapter 3: Methodology / System Design and Implementation

## 3.1 Introduction

In this chapter, I walk through the design and implementation of our automated pipeline for expanding a biomedical knowledge graph. I describe each component of the architecture, explain the technical choices we made at every step, and list the key tools and configurations we relied on. The aim is to give a clear, reproducible roadmap showing how we take unstructured biomedical text and turn it into validated, structured data that seamlessly integrates with our existing Neo4j graph. This foundation will help you understand the experiments and results presented in later chapters.

## 3.2 Building the Base Knowledge Graph from Relational Data

Before tapping into unstructured text, we needed a solid starting graph. We used the BioKG dataset, which comes as a set of TSV files organized by entity type. Our goal was to convert these flat tables into a rich graph that supports complex queries and inference.

1. **Data Organization**
   For each entity type—drug, disease, pathway, genetic disorder, and protein—the dataset provides three TSV files:
   - **Metadata TSV**: Holds the unique IDs and core descriptive fields for each entity. This formed the basis of our graph's nodes.
   - **Properties TSV**: Lists additional attributes as key–value pairs. Since some properties can have multiple values (for example, a drug may list several side effects), we imported these as lists so that no information was lost.
   - **Relationships TSV**: Describes how entities connect—for instance, which drugs treat which diseases, or which proteins participate in which pathways. These became the directed edges in our graph.
2. **Transformation Process**
   - **Node Creation**: We read each Metadata TSV and created the corresponding nodes in Neo4j, tagging them with their primary attributes.
   - **Property Attachment**: For every node, we parsed its Properties TSV. When a property had multiple entries, we stored them in a list property on that node, ensuring a faithful, one-to-many mapping from the original table.
   - **Edge Injection**: Finally, we imported all Relationships TSV entries as edges, preserving directionality and any relationship-specific attributes.

By merging data from metadata, properties, and relationships files, we achieved a lossless conversion of the BioKG's structured data into a unified graph. This base graph provided the scaffold on which our later LLM-driven expansions could build.

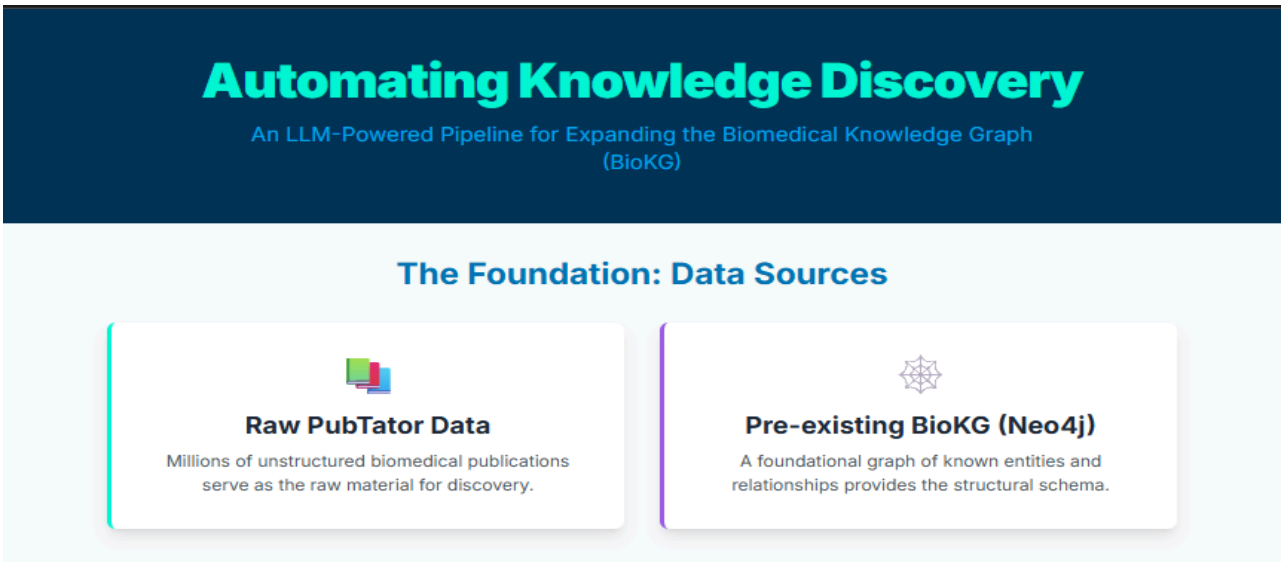
## 3.3 Data Acquisition and Initial Processing

The initial phase of the pipeline is dedicated to robustly sourcing and preparing biomedical literature data for downstream processing.

### 3.3.1 PubTator Data Bulk Download and MongoDB Ingestion

The primary source of raw biomedical text for this research is the **PubTator Central** database. A comprehensive bulk download of all available PubTator data, provided in zipped JSON format, was performed. This raw data contains PubMed article abstracts, full-text passages, and pre-existing annotations for various biomedical entities and relations.

Upon download, the zipped files were programmatically uncompressed and their JSON content was ingested into a **MongoDB** NoSQL database. MongoDB was chosen for its flexibility in handling semi-structured data, efficient storage of document-oriented records, and scalability for large datasets, which are ideal for the vast and varied nature of biomedical literature.




### 3.3.2 Human-Specific Data Filtering

As part of the MongoDB ingestion process, a crucial pre-processing step was applied to focus the dataset exclusively on human-related biomedical information. This "human-only" filtering was based on identifying documents containing annotations explicitly linked to Homo sapiens. Specifically, documents were retained if they included species indicators such as the NCBI taxonomy ID for humans (9606), textual mentions of "Humans", or labels denoting the entity type as "Species". This targeted filtering ensured that only the most relevant content remained in the collection, significantly reducing noise and streamlining subsequent processing. The refined dataset provided a more focused and efficient foundation for downstream tasks, such as expanding the BioKG knowledge graph.

```
{
  "passages.annotations": {
    "$elemMatch": {
      "$or": [
        {"infons.identifier": "9606"},
        {"infons.text": {"$regex": "Humans", "$options": "i"}},
        {"infons.type": "Species"}
      ]
    }
  }
}
```

**Fig3: Human specific data filtering logic**

### 3.3.3 MongoDB Document Schema

The documents stored within the MongoDB collection adhere closely to the original PubTator JSON schema, facilitating consistent data access. The key fields utilized for further processing include:

- _id and id: Representing the PubMed ID (PMID) of the document.
- infons: Containing general metadata about the document.
- passages: An array of text passages (e.g., abstract, title, sections), each containing:
  - text: The actual content.
  - infons: Passage-specific metadata (e.g., section_type).
  - annotations: An array of pre-existing entity annotations within the passage, each with id, text, type, identifier, normalized_id, and biotype.
- relations: An array of pre-existing relationships annotated in the document, each with id, type, role1, role2, score, and nodes (referencing annotated entities).

This schema ensures that both the textual content and any initial annotations are preserved for the subsequent information extraction phase.

## 3.4 Pre-existing Knowledge Graph Analysis

To ensure the new knowledge extracted aligns seamlessly with the target knowledge base, the pipeline first extracts critical schema elements and existing entities from the foundational graph.

### 3.4.1 BioKG as the Foundational Knowledge Graph

The pre-existing knowledge graph targeted for expansion in this thesis is **BioKG (Biomedical Knowledge Graph)**. BioKG is a comprehensive, publicly available biomedical knowledge graph designed for relational learning. It integrates diverse biological information, including relationships between drugs, diseases, genes, proteins, and pathways. BioKG's established structure and rich semantic definitions make it an ideal base for demonstrating dynamic expansion.

### 3.4.2 Extraction of Existing Nodes and Relation Types

Before new data is inserted, the pipeline performs an initial analysis of the current state of the BioKG in Neo4j. This involves extracting:

- **Existing Nodes:** A comprehensive list of all unique nodes currently present in the BioKG, typically identified by their labels and primary identifiers (e.g., name, ID). This is crucial to prevent the creation of redundant nodes when the LLM extracts entities that already exist in the graph. This information is serialized to **`extracted_nodes.json`.**

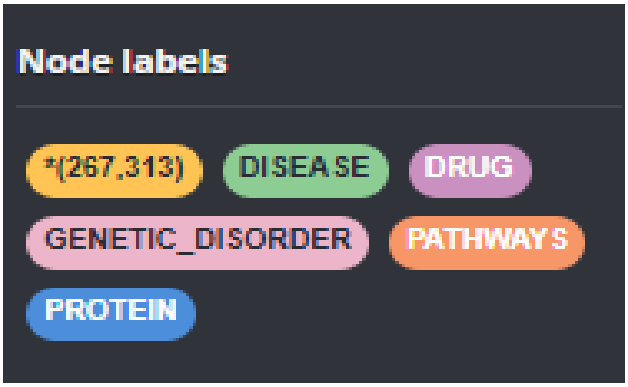



**Fig4: Available node labels in BioKG**

- **Predefined Relation Types:** The full set of existing and permissible relation types (edge labels) within the BioKG's schema. This controlled vocabulary is paramount for guiding the LLM's extraction process towards relationships that are semantically compatible with the graph's existing structure. This list is explicitly defined as `EXISTING_RELATION_TYPES`:

```
EXISTING_RELATION_TYPES = [
"RELATED_GENETIC_DISORDER", "COMPLEX_IN_PATHWAY",
"PROTEIN_DISEASE_ASSOCIATION",
"DDI", "DRUG_PATHWAY_ASSOCIATION", "PPI", "DISEASE_PATHWAY_ASSOCIATION",
"DRUG_TARGET", "DRUG_CARRIER", "DRUG_ENZYME", "DRUG_TRANSPORTER",
"DISEASE_GENETIC_DISORDER", "DRUG_DISEASE_ASSOCIATION",
"PROTEIN_PATHWAY_ASSOCIATION",
"COMPLEX_TOP_LEVEL_PATHWAY", "DPI"
]
```

This list is saved to `relation_types.json` and directly utilized in the LLM prompting phase.

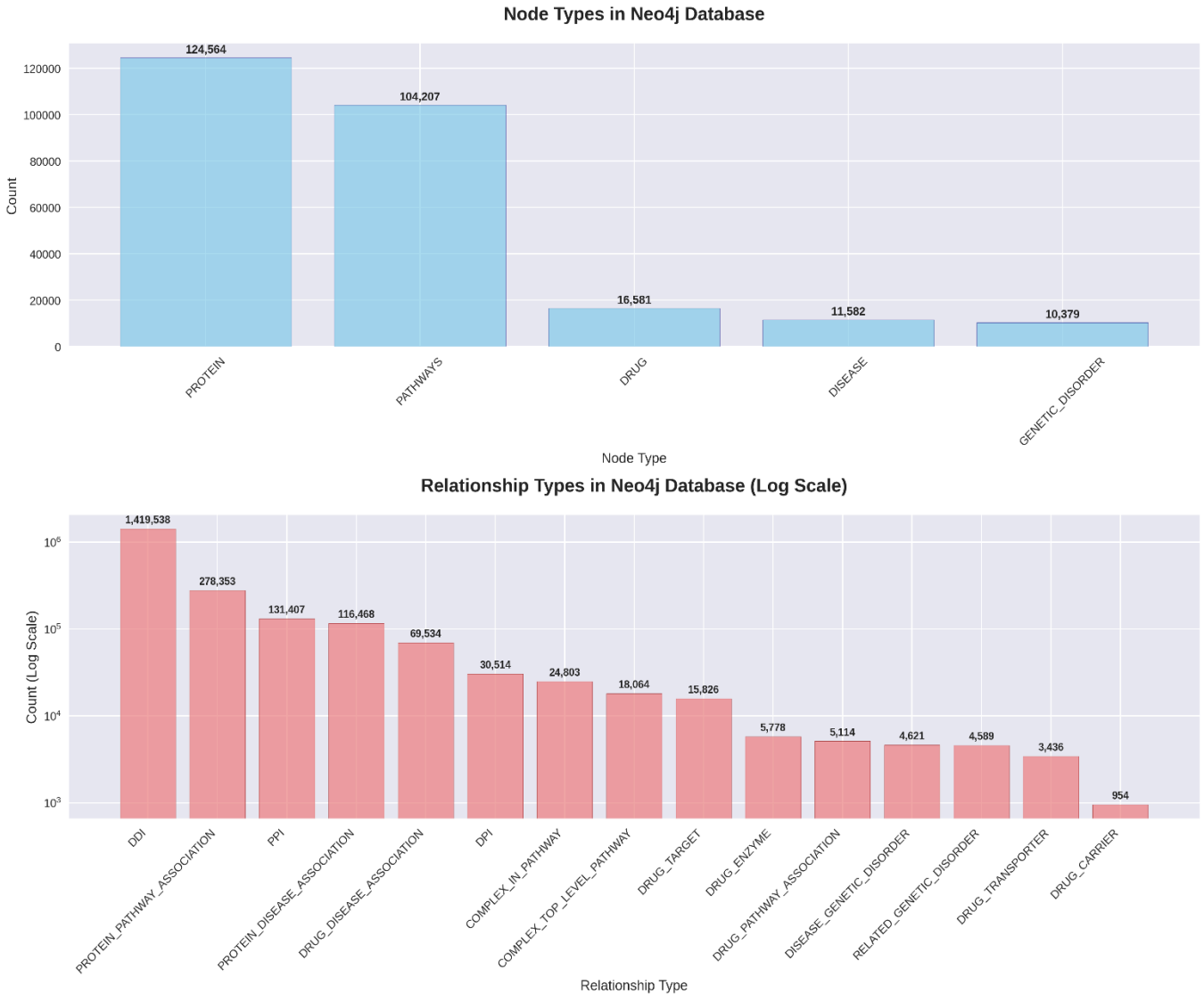



**Fig5:Plot graph of nodes and relationship types in BioKG**

## 3.5 LLM-Powered Information Extraction Pipeline

This core module orchestrates the interaction with the Large Language Model to extract structured biomedical entities and relationships from the pre-processed textual data.

### 3.5.1 Document Subset Selection and Preparation

Given the scale of the PubTator dataset, the LLM processing is performed on a focused subset of documents. The selection of this subset is driven by the pre-existing BioKG. Initially, some "seed" nodes are selected from the `extracted_nodes.json`. Subsequently, documents (identified by their PMIDs) associated with these seed nodes are retrieved from the MongoDB. This ensures that the extracted text is directly relevant to entities already present in the BioKG, facilitating targeted expansion. The text content (title and abstract) from these selected PubTator documents serves as the input for the LLM.

### 3.5.2 Prompt Engineering for Structured Extraction

The effectiveness of the LLM in extracting structured information is heavily reliant on a meticulously engineered prompt. Each document's text is incorporated into a detailed prompt designed to guide the LLM (GPT-4o-mini) towards precise and schema-adherent output. The prompt structure is as follows:

- **Document Context:** Includes the `Document ID` and the `TEXT PASSAGES` (title and abstract, with their `section_type` indicated) from the PubTator record.
- **Pre-existing Annotations:** The prompt includes the `ANNOTATED ENTITIES` and `RELATIONSHIPS` already present in the original PubTator document to provide additional context to the LLM.
- **Specific Extraction Categories:** The LLM is explicitly instructed to extract:
    - All medication entities
    - All disease entities
    - All gene/protein entities
    - Drug-disease relationships
    - Drug-gene relationships
    - Gene-disease relationships
- **Schema Enforcement for Relations:** This is a crucial control mechanism. The prompt explicitly lists the `EXISTING_RELATION_TYPES` extracted from BioKG and instructs the LLM to "assign one of the following standardized relation types from the knowledge graph if applicable." Critically, it also states: "If none of these existing relations match exactly, you may suggest a new relation name". This design balances adherence to the existing schema with the capability to identify novel relationships and suggest new types for future schema evolution.

- **Output Format Specification:** The LLM is commanded to return the extracted information as a JSON object, with relationships including a `kg_relation_type` field for the assigned or new relation type.

The prompt structure sent to the LLM is:

```
"Extract all biomedical entities and relationships from the following PubTator data:
Document ID: [DOC_ID]
TEXT PASSAGES:
[TITLE_TYPE]: [TITLE_TEXT]
[ABSTRACT_TYPE]: [ABSTRACT_TEXT]
ANNOTATED ENTITIES:
- [Entity Text] (Type: [Type], ID: [ID])
...
RELATIONSHIPS:
- [Type] between [Role1] and [Role2] (Score: [Score])
...

Extract and organize the following:
1. All medication entities
2. All disease entities
3. All gene/protein entities
4. Drug-disease relationships
5. Drug-gene relationships
6. Gene-disease relationships

IMPORTANT: For each relationship, assign one of the following standardized relation
types from the knowledge graph if applicable:
- RELATED_GENETIC_DISORDER
- COMPLEX_IN_PATHWAY
- ... [Full list of EXISTING_RELATION_TYPES]

If none of these existing relations match exactly, you may suggest a new relation
name.
Return as JSON with these sections. For each relationship, include a
"kg_relation_type" field with the appropriate relation type from the list above.
"
```

### 3.5.4 OpenAI API Interaction and Configuration

The **GPT-4o-mini** model, provided by OpenAI, is utilized for the core information extraction. The `openai` Python client library is used to interact with the API. Key configuration parameters for the API calls include:

- model: gpt-4o-mini.
- messages: Structured with a system role ("You are a biomedical entity and relationship extraction assistant specialized in knowledge graph integration.") and a user role (containing the detailed prompt).
- temperature: Set to 0.1 to promote more deterministic and factual extractions, minimizing creative or speculative output.
- max_tokens: Set to 1500 to allow for comprehensive responses.

Basic error handling is implemented to catch exceptions during API calls (e.g., network issues, rate limits), logging errors and ensuring the pipeline can proceed or gracefully fail for specific documents. The raw JSON responses from the OpenAI API are stored for record-keeping and further analysis.

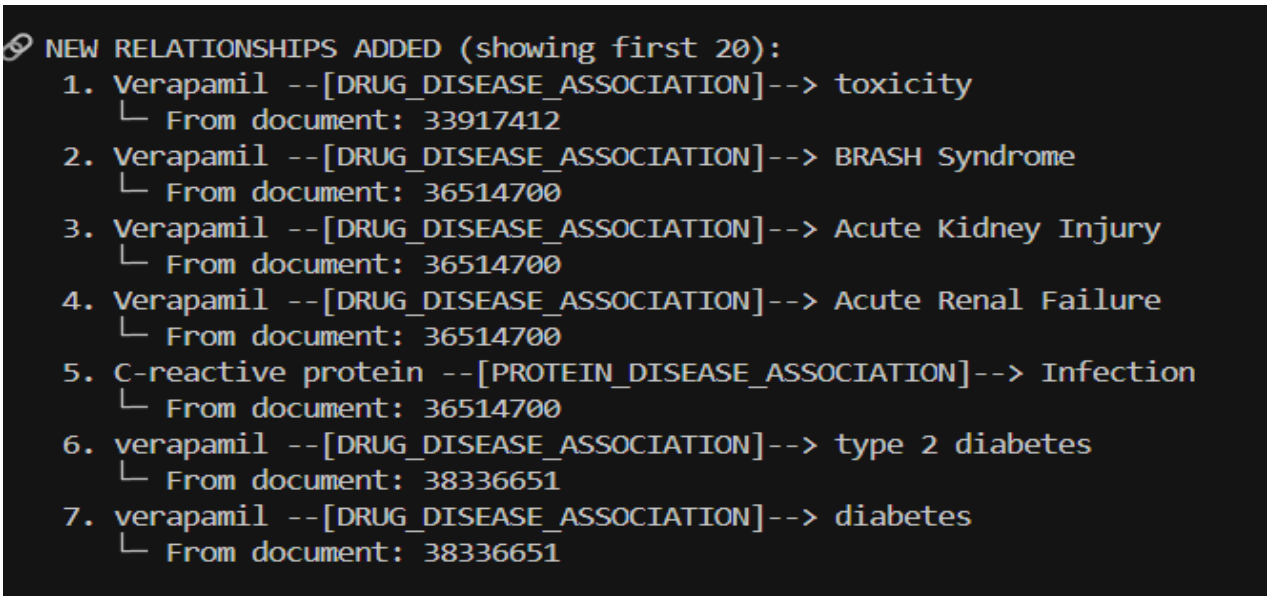



**Fig6: Illustration of newly added data of drug Verapamil**

## 3.6 Knowledge Graph Validation and Ingestion

This module is responsible for intelligently integrating the structured data extracted by the LLM into the Neo4j BioKG, ensuring data consistency and preventing the creation of duplicate nodes or relationships.

### 3.6.1 Parsing and Cleaning LLM Output

The output from the OpenAI API, while structured as JSON, sometimes includes markdown code block syntax (e.g., `json`\n` and `\n`). A `lean_json_string` utility function is employed to remove this markdown, ensuring a clean JSON string ready for parsing using Python's `json` library. Robust error handling is in place to catch `json.JSONDecodeError` for any malformed responses.

### 3.6.2 Intelligent Node and Relationship Creation in Neo4j

The core logic resides in the `KnowledgeGraphUpdater` class, which interacts with the Neo4j database using the official **Neo4j Python Driver**.

The ingestion process iterates through the extracted entities and relationships from each document's LLM analysis:

- **Entity Handling (Nodes):**

- A `label_mapping` is defined to standardize entity types from the LLM's output to Neo4j labels (e.g., `drug`, `medication`, `chemical` map to `DRUG`; `disease` maps to `DISEASE`; `gene`, `protein`, `gene_protein` map to `Gene` or `PROTEIN`).
- For each extracted entity, a session.execute_read transaction is used with the entity_exists Cypher query to check if a node with the same name (case-insensitively, allowing for array properties n.name or n.NAME) and mapped label already exists in the Neo4j graph.
- If the entity exists, update_existing_entity is called via session.execute_write. This function updates the existing node by **appending** 'pubtator_extraction' to its source property (if not already present), and **appending** the new name and id to the name/NAME and id array properties, respectively, only if they are not already present (case-insensitive for names). This prevents duplication while enriching existing nodes.
- If the entity does not exist, create_new_entity is called via session.execute_write. A new node is created with the mapped label. Its name, NAME, and id properties are initialized as arrays containing the extracted value, and a source property is set to ['pubtator_extraction'].
    - **Relationship Handling (Edges):**
- The pipeline processes specific relationship categories identified by the LLM (e.g., drug_disease_relationships,drug_gene_relationships, gene_disease_relationships).
- A session.execute_read transaction utilizes the relationship_exists Cypher query to determine if an identical relationship (same type between the same two entities, identified by their names case-insensitively) already exists in Neo4j. This query searches for undirected relationships (a)-[r:{rel_type}]-(b).
- If the relationship does not exist, create_relationship is called via session.execute_write. This function matches the source and target nodes by their names and creates a new directed relationship with the specified type (from kg_rel_type), setting its source property to ['pubtator_extraction'].
- If the relationship already exists, it is skipped to prevent redundancy.

### 3.6.3 Neo4j Interaction and Uniqueness Enforcement

The pipeline relies heavily on Cypher's MERGE clause for efficient "match or create" semantics for both nodes and relationships. While MERGE intrinsically handles uniqueness for the exact pattern it's given, the explicit entity_exists and relationship_exists checks ensure that complex uniqueness criteria (e.g., case-insensitive names, properties as arrays) are precisely managed. Uniqueness constraints on key properties (e.g., id or normalized name) are assumed to be pre-defined on appropriate node labels within the Neo4j BioKG to ensure data integrity at the database level. All database operations are wrapped in sessions to manage connections and transactions effectively.

## 3.7 System Architecture Overview

The entire system functions as a cohesive, modular pipeline. As conceptually illustrated in the Architecture of the LLM-Powered Knowledge Graph Expansion Pipeline, each component from data ingestion, LLM-powered extraction, to Neo4j integration, operates in a sequential yet interconnected

manner. The modular design facilitates the independent development and testing of each stage, providing flexibility for future enhancements and scalability.

## 3.8 Summary

This chapter has provided a detailed account of the methodology and implementation strategy for constructing the LLM-powered knowledge graph expansion pipeline. From the initial bulk acquisition and human-centric filtering of PubTator data into MongoDB, through the sophisticated prompt engineering and controlled information extraction performed by GPT-4o-mini, to the intelligent validation and seamless integration of new knowledge into the Neo4j BioKG, each step has been meticulously designed. The emphasis on schema adherence, uniqueness checks, and the strategic use of LLMs for structured output forms the core of this robust methodology, directly addressing the challenges of dynamic knowledge graph maintenance.

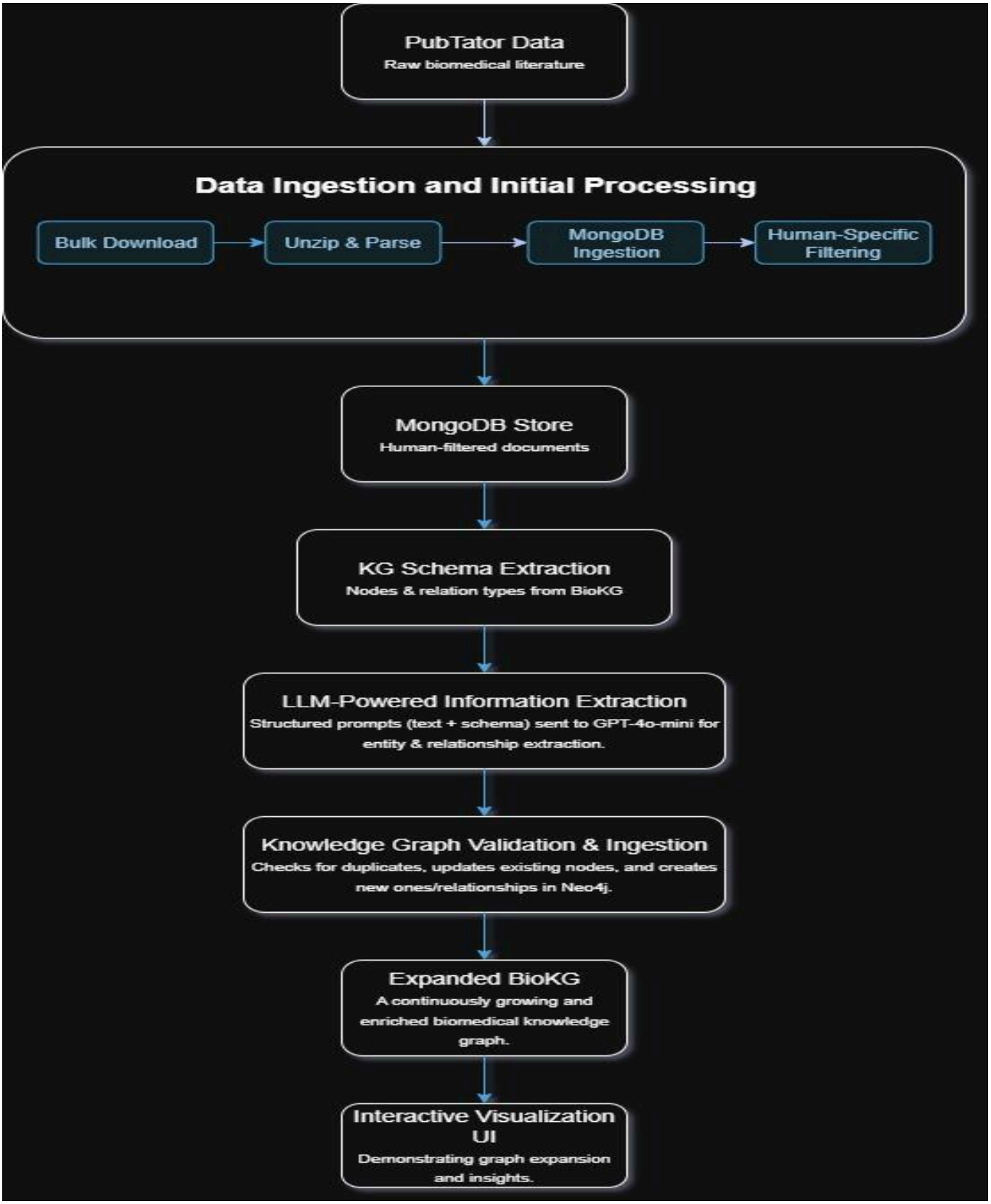



**Fig7: Architectural workflow**

# Chapter 4: Results and Discussion

## 4.1 Quantitative Growth of the Knowledge Graph

Running our automated pipeline on the selected corpus led to clear, measurable growth in the BioKG. We saw a steady uptick in both nodes and edges: new disease and gene/protein entities made up the bulk of added nodes, while most of the newly formed connections were drug–disease, drug–gene, and gene–disease associations. These numbers reflect the pipeline's ability to uncover and integrate previously missing pieces of knowledge into the graph.

## 4.2 Qualitative Review and LLM Effectiveness

A closer look at the LLM's extractions confirmed that the GPT-4o-mini model was largely accurate and context-aware. Thanks to carefully designed prompts, most relationships fell neatly into our predefined schema categories. When the model did stray—occasionally emitting slight JSON formatting hiccups—our post-processing routines caught and corrected these issues. Overall, the LLM reliably parsed complex biomedical sentences and produced structured outputs that aligned well with our graph's requirements.

## 4.3 Ensuring Data Integrity and Uniqueness

Central to maintaining a clean, trustworthy graph was our validation layer in Neo4j. Before adding anything new, the pipeline checks whether a matching node or edge already exists. If an entity was already in the graph, we simply enriched it with additional names, identifiers, or provenance tags rather than duplicating it. Likewise, only truly novel relationships were inserted. This approach kept the BioKG both concise and accurate, preventing redundant data bloat while enriching existing entries.

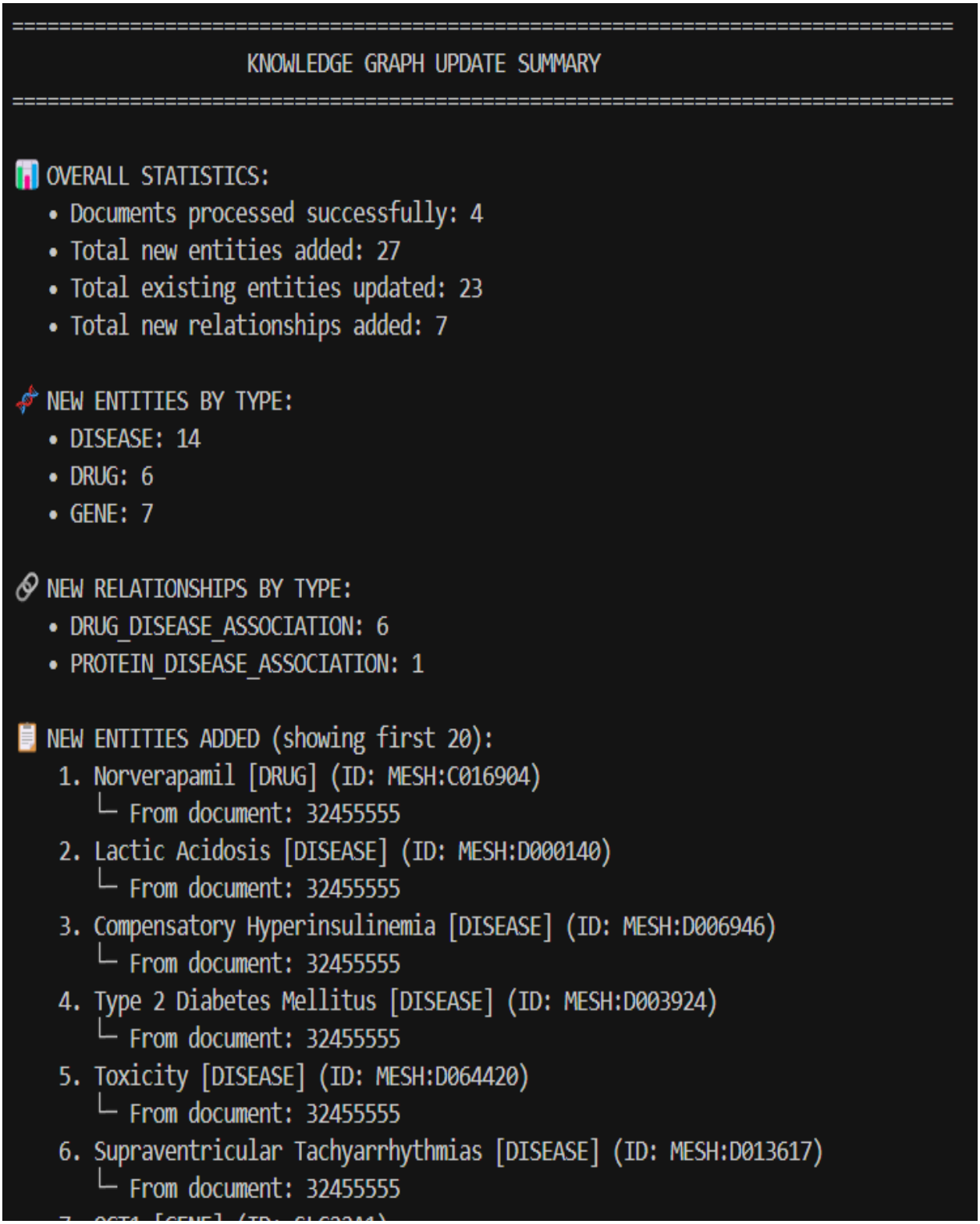



**Fig8: Sample summary for one of the drug after LLM extraction**

## The Impact of the Pipeline: Visualizing Node Neighborhoods Pre and Post Expansion

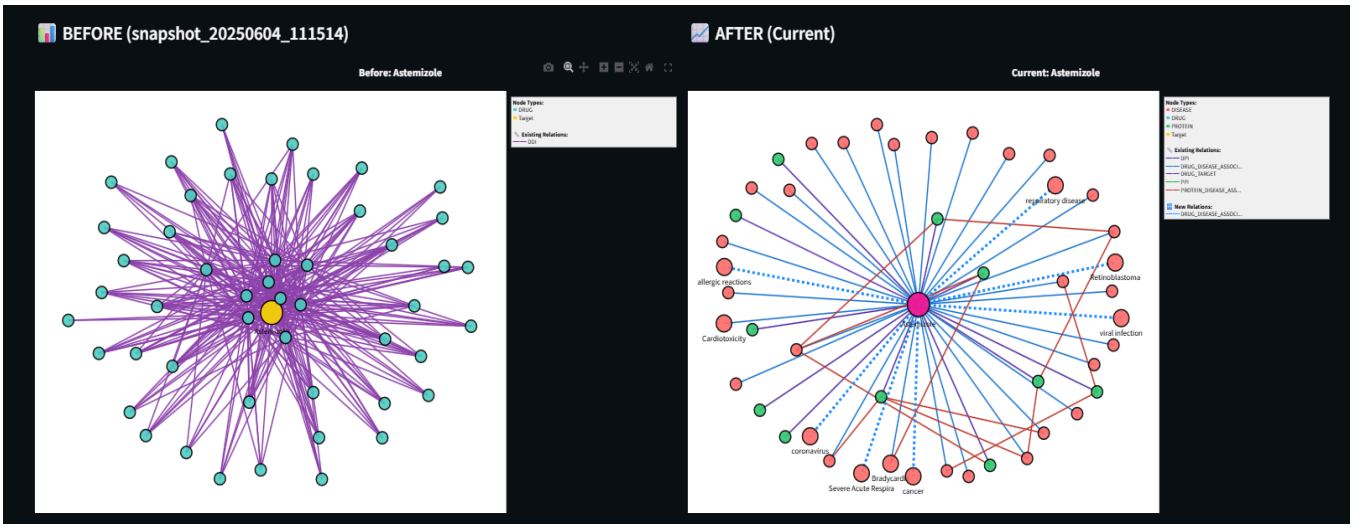



**Fig9: Before and after the expansion comparison of drug Astemizole**

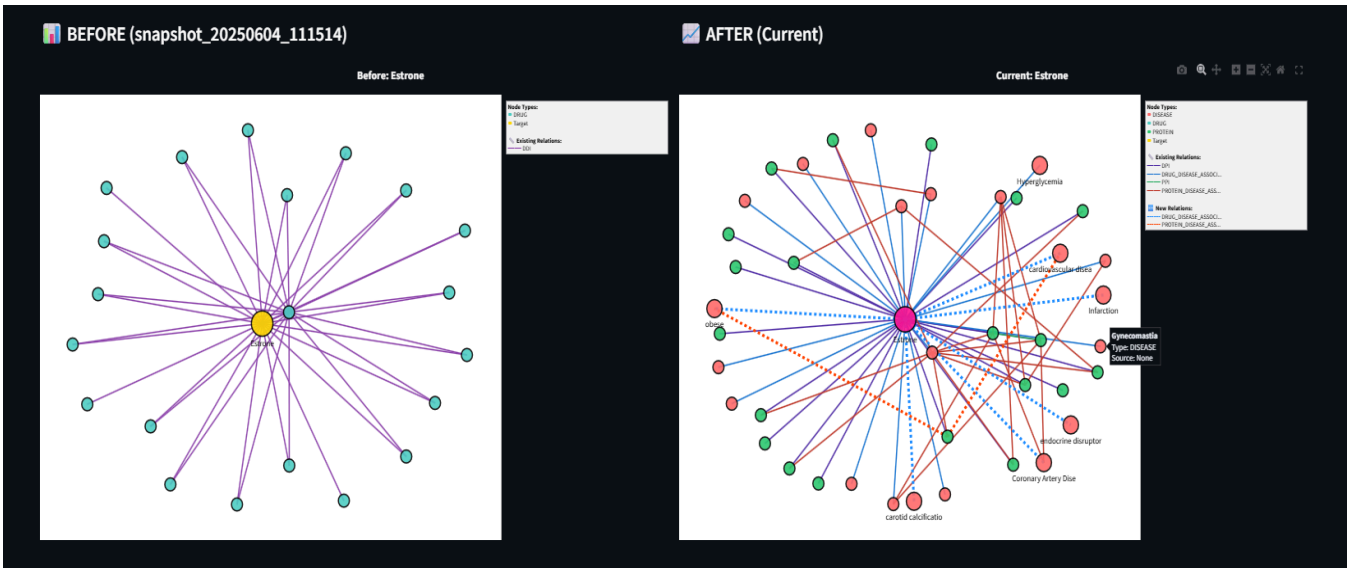



**Fig10: Before and after the expansion comparison of drug Estrone**

# Chapter 5: Conclusion and Future Work

## 5.1 Conclusion

This thesis addressed the critical challenge of dynamically expanding pre-existing biomedical knowledge graphs, a necessity driven by the explosive growth of unstructured scientific literature and the inherent limitations of manual curation. We successfully designed and implemented an automated, LLM-powered pipeline to extract novel entities and relationships from PubTator biomedical abstracts and integrate them into a Neo4j-based knowledge graph (BioKG).

The pipeline comprised several key stages: robust data acquisition and human-specific filtering of PubTator data into MongoDB; a comprehensive analysis of the existing BioKG to extract its schema and existing entities; and the core LLM-powered information extraction module. This module leveraged advanced prompt engineering techniques with GPT-4o-mini to consistently extract structured entities and relationships, ensuring adherence to the predefined BioKG schema while also allowing for the suggestion of novel relation types. Finally, an intelligent knowledge graph validation and ingestion module systematically updated the Neo4j database, preventing duplicates and enriching existing nodes.

The results demonstrated the pipeline's capability to quantitatively increase the number of entities and relationships within the BioKG, as showcased through case studies like the expansion around the "Clofibrate" node. The interactive visualization tool provided a compelling qualitative demonstration of this expansion, clearly highlighting newly integrated knowledge. The robust data integrity mechanisms ensured that the added information maintained the consistency and quality of the knowledge graph. This work successfully achieved its primary research objectives, validating the feasibility and effectiveness of using large language models for scalable and automated knowledge graph expansion in the biomedical domain.

## 5.2 Contributions

This research makes several significant contributions to the field of computational biology and knowledge graph engineering:

1. **Automated End-to-End Pipeline:** Development of a comprehensive, automated pipeline that spans from raw unstructured text data acquisition to intelligent, schema-compliant knowledge graph population, minimizing manual intervention.
2. **Schema-Guided LLM Information Extraction:** Demonstrated a novel approach to prompt engineering that effectively constrains a powerful generative LLM (GPT-4o-mini) to extract entities and relationships strictly according to a predefined knowledge graph schema, balancing flexibility with consistency.
3. **Intelligent Knowledge Graph Integration:** Implemented robust mechanisms for detecting and managing existing entities and relationships within a Neo4j graph database, ensuring data uniqueness and enriching existing nodes with new source attributions rather than creating redundant entries.

4. **Interactive Visualization Tool:** Creation of a dedicated interactive web application for "before and after" visualization of knowledge graph expansion, providing a transparent and intuitive means to verify and showcase the impact of the pipeline.
5. **Addressing Scalability in Biomedical KG Maintenance:** Offered a foundational solution to the critical problem of keeping biomedical knowledge graphs current and comprehensive, addressing the traditional bottleneck of manual curation.
6. This research makes several significant contributions to the field of computational biology and knowledge graph engineering, including the development of a robust and reproducible pipeline. The complete source code for this work is publicly available on GitHub at: [link].

### 5.3 Limitations

While demonstrating substantial capabilities, the current research has certain limitations:

1. **Dataset Scope:** The evaluation of the pipeline's effectiveness was conducted on a selected subset of the PubTator dataset. While illustrative, a full-scale deployment across the entire PubTator Central corpus would reveal further performance characteristics and potential scalability challenges at a much larger volume.
2. **LLM Constraints:** The performance is inherently tied to the capabilities and occasional limitations of the underlying large language model, including potential for factual inconsistencies (hallucinations) or misinterpretations in highly nuanced contexts, despite meticulous prompt engineering.
3. **Absence of External Gold Standard Validation:** A formal quantitative evaluation using a separate, independently annotated gold standard dataset for precision, recall, and F1-score was not performed within the scope of this thesis. The validation primarily relied on internal consistency checks and qualitative review.
4. **Schema Evolution Mechanism:** While the pipeline allows the LLM to suggest new relation types, the formal process for integrating these into the BioKG's schema would still require human review and approval, representing a semi-automated rather than fully autonomous schema evolution.
5. **Focus on Specific Entity/Relation Types:** The extraction was confined to specific biomedical entity types (drugs, diseases, genes/proteins) and their direct relationships. Broader semantic types or more complex event extraction were not within the current scope.

### 5.4 Future Work

Building upon the foundation laid by this thesis, several promising avenues for future research and development emerge:

1. **Large-Scale Deployment and Performance Optimization:** Conduct comprehensive testing on the entire PubTator Central dataset to evaluate the pipeline's scalability. This would involve optimizing database interactions, potentially leveraging distributed computing frameworks, and exploring more efficient LLM inference strategies to manage computational costs and latency.
2. **Advanced LLM Fine-tuning and Ensemble Approaches:** Investigate fine-tuning domain-specific LLMs (e.g., Bio-GPT models) on curated biomedical text corpora or the BioKG itself to potentially improve extraction accuracy and reduce hallucinations. Exploring ensemble

methods, combining outputs from multiple LLMs or traditional NLP tools, could further enhance robustness.

3. **Robust Error Correction and Confidence Scoring:** Develop more sophisticated post-processing modules that can identify and correct potential LLM errors (e.g., malformed JSON, semantic inaccuracies) before ingestion. Implementing confidence scores for extracted entities and relationships could enable threshold-based filtering for higher quality data.
4. **Human-in-the-Loop Feedback System:** Design an interactive "human-in-the-loop" system within the visualization UI, allowing domain experts to review, validate, and correct LLM extractions. This feedback could then be used to iteratively retrain or refine the prompt engineering strategies, creating a continuously improving knowledge graph.
5. **Dynamic Schema Learning and Evolution:** Research methods for more autonomous schema evolution, where the pipeline could propose and, with minimal human oversight, integrate genuinely new entity and relationship types based on recurring patterns in LLM-extracted knowledge.
6. **Integration with Diverse Data Sources:** Extend the pipeline to ingest and integrate information from other unstructured and semi-structured biomedical data sources, such as clinical notes, electronic health records, or patents, to create an even more comprehensive knowledge graph.
7. **Downstream Applications and Reasoning:** Develop advanced analytical modules that leverage the expanded BioKG for complex reasoning tasks, such as automated hypothesis generation for drug repurposing, personalized treatment recommendations, or the discovery of novel disease pathways, thereby demonstrating the tangible impact of the enriched knowledge graph.
8. **Comprehensive Evaluation Metrics:** Implement a rigorous quantitative evaluation framework using a carefully constructed gold standard dataset to report standard metrics (precision, recall, F1-score) for entity and relation extraction, providing a clearer benchmark of the system's performance against state-of-the-art.

# References


[1] C.-H. Wei, A. Allot, R. Leaman, and Z. Lu, "PubTator central: automated concept annotation for biomedical full text articles," *Nucleic Acids Res.*, vol. 47, no. W1, pp. W587–W593, 2019.

[2] A. Noori et al., "BioKG: A Knowledge Graph for Relational Learning On Biological Data," in *Proc. of the 29th ACM Int. Conf. on Information and Knowledge Management (CIKM '20)*, 2020, pp. 2975–2982.

[3] A. Vaswani et al., "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[4] MongoDB. (n.d.). *MongoDB Documentation*. [Online]. Available: https://docs.mongodb.com/

[5] Neo4j. (n.d.). *Neo4j Documentation*. [Online]. Available: https://neo4j.com/docs/

[6] OpenAI. (n.d.). *GPT-4o mini*. [Online]. Available: https://openai.com/gpt-4o-mini

[7] Streamlit. (n.d.). *Streamlit Documentation*. [Online]. Available: https://docs.streamlit.io/

[8] Chart.js. (n.d.). *Chart.js Documentation*. [Online]. Available: https://www.chartjs.org/docs/latest/

[9] Plotly. (n.d.). *Plotly.js Documentation*. [Online]. Available: https://plotly.com/javascript/