PhD Proposal: Automated Reasoning and Graph Learning for Drug Repurposing

Introduction

Drug repurposing—finding new therapeutic uses for existing drugs—can dramatically reduce development time and costnature.com. However, identifying promising repurposing candidates requires synthesizing vast amounts of biomedical knowledge (e.g. literature, omics data, clinical trial results)nature.com. Manual expert review is labor-intensive and struggles to keep up with the explosive growth of publications and databasesnature.com. This creates a need for **automated reasoning systems** that can generate and evaluate drug repurposing hypotheses at scale. Recent advances make this feasible: large language models (LLMs) demonstrate surprising capabilities in medical question-answering and reasoningnature.com, and knowledge graph (KG) based machine learning has shown success in predicting drug–disease relationships. In particular, KGs offer a way to integrate heterogeneous biomedical data and *dynamically update* analyses as new data become available. Graph neural networks (GNNs) can then be trained on these KGs to discover hidden connections ("missing links") representing potential therapies.

Hypothesis: By combining a **reasoning-capable LLM** with **structured knowledge integration** and **graph ML**, we can build a fully automated system that not only proposes plausible drug repurposing candidates but also validates them against existing data and continuously improves as new evidence emerges. This PhD project aims to develop such a system – a *reasoning agent* that reads biomedical information, hypothesizes new drug–gene–disease links, queries structured databases to assemble supporting evidence, uses GNNs on the resulting knowledge graphs to predict which links are likely valid, and finally tracks real-world outcomes (new publications, trials, approvals) to refine its hypotheses over time. Crucially, the approach will focus on **biomedical reasoning** (to ensure biological plausibility of suggestions) and **clinical relevance** (prioritizing hypotheses with supporting evidence in patient data or trials).

Envisioned Workflow of the Reasoning Agent

1. **Hypothesis Generation via Biomedical LLMs:** The process begins with an LLM fine-tuned on biomedical corpora generating hypotheses about novel drug–disease (and gene) interactions. For example, an LLM might be prompted with a disease and asked to suggest existing drugs that could treat it, along with a rationale. Recent studies have shown that GPT-4 can propose credible repurposing candidates – e.g. for Alzheimer's disease, ChatGPT (GPT-4) was able to list drugs later found to correlate with reduced disease risk in patient recordsnature.comnature.com. Here we will leverage an **open-source biomedical LLM** (such as *PMC-LLaMA*, a 13B-parameter model trained on 4.8M biomedical paperspubmed.ncbi.nlm.nih.govpubmed.ncbi.nlm.nih.gov) to ensure domain-specific knowledge. The LLM will generate candidate triples like *(Drug X –*

targets Gene Y – associated with Disease Z), accompanied by a chain-of-thought explanation. **Tools:** A domain-specific LLM (e.g. PMC-LLaMApubmed.ncbi.nlm.nih.govpubmed.ncbi.nlm.nih.gov or BioGPT) accessed via a framework like **LangChain** (to manage prompts and tool use) or deployed locally via **Ollama** for efficiency.

2. **Retrieval of Structured Data & Knowledge Graph Construction:** Next, the agent gathers evidence for each proposed hypothesis from structured biomedical databases. This step grounds the LLM's suggestions in known data. For instance, if the LLM hypothesizes that *Drug X could treat Disease Z by modulating Gene Y*, the system will query resources like **Open Targets** to retrieve known interactions: Does Drug X affect Gene Y? Is Gene Y implicated in Disease Z? Open Targets is an ideal resource here – it integrates genetics, omics, and pharmacological data to score target–disease associations. Additional databases (ChEMBL, DrugBank, DisGeNET, etc.) can be accessed for drug–target interactions, pathways, and phenotypic associations. The retrieved facts are then assembled into a **knowledge subgraph**: a small KG connecting the drug, disease, gene, and any intermediate nodes (e.g. shared pathways, similar diseases, other drugs targeting the same gene). This subgraph represents the context around the hypothesis. **Tools:** Structured query via **Open Targets API** (for gene-disease-drug evidencearxiv.org) and other public APIs (e.g. PubChem for chemical info, ClinicalTrials.gov for trial data). **NetworkX** or a graph database (Neo4j) can be used to construct and store the knowledge graph. Integration frameworks like **LangChain** allow the LLM to call these APIs automatically as part of its reasoning (agent tooling).

3. **Graph-Based Link Prediction with GNNs:** Once the knowledge graph is built, the system uses graph machine learning to evaluate the likelihood of the new connection (link) proposed by the LLM. Graph neural networks can effectively learn from biomedical KGs by considering network topology and node features. A GNN model (e.g. Graph Convolutional Network, GraphSAGE, or transformer-based GNN) is trained on known drug–disease links to predict new ones. For example, the KG-Bench framework (developed in our prior work) trains GNNs on a drug–disease–target knowledge graph derived from Open Targets. In benchmarking experiments, GNNs achieved high precision (e.g. TransformerConv model with APR 0.87 on predicting drug–disease links). Using a similar approach, our agent will feed the subgraph into a GNN link predictor to score the hypothesis: a high score would indicate that the drug–disease link is consistent with known network patterns (e.g. shared targets or pathways). If multiple hypotheses are generated, the GNN can help rank them by confidence. **Tools: PyTorch Geometric (PyG)** for implementing GNN models (message-passing networks, link prediction training loop). We will build upon open-source pipelines like **KG-Bench** or **OGB** (Open Graph Benchmark datasets) to train and evaluate models fairly. Feature handling (e.g. chemical descriptors, gene functions) can use PyG's data loaders or custom preprocessing in Python. For scalability to large graphs, distributed frameworks or **DGL** (Deep Graph Library) may be explored.

4. **Iterative Validation and Refinement:** A distinguishing aspect of this system is its **recursive learning** ability. After the agent proposes and evaluates hypotheses, it will monitor real-world outcomes over time to validate its predictions. This involves

periodically checking sources like PubMed, clinical trial databases, and regulatory approvals for any evidence that corresponds to the agent's past hypotheses. For example, if the agent predicted a Drug X–Disease Z link, we will monitor for new publications (e.g. a successful case report or a Phase II trial result) that confirm or refute this link. Notably, our previous KG-Bench work incorporated *retrospective validation* by leveraging historical data updates – similarly, we will design the system to take in **temporal updates** of databases (new releases of Open Targets, new drug indications added to DrugBank, etc.). If a prediction is confirmed by new data, it can be added as a new known link in the knowledge graph, and the GNN can be retrained with this expanded knowledge, thereby **closing the loop**. Likewise, if new data contradict a hypothesis, the system should adjust (e.g. lower that link's score or incorporate the counter-evidence). An LLM agent can be used here to ingest textual evidence (new papers) and update its knowledge or reasoning patterns. Over the PhD project, we will implement a continuous validation pipeline so that the **reasoning agent improves with time**, becoming more accurate and trustworthy. **Tools: n8n** (an open-source workflow automation tool) will be used to schedule periodic data fetches (for new publications or database updates) and trigger the LLM/GNN pipeline on a schedule (e.g. monthly). The LLM may use **literature mining** (via PubMed APIs or a tool like BioGPT) to find relevant trial results. The system will log outcomes for each hypothesis to build a feedback dataset for reinforcement learning or fine-tuning the models (e.g. using an RLHF approach where confirmed hypotheses are treated as positive reinforcement).

Throughout this workflow, we will emphasize **trustworthy AI** practices. The agent's suggestions will be accompanied by explanations and supporting evidence retrieved (to avoid LLM hallucinations). The GNN's decisions will be interrogated with interpretability techniques like **GNNExplainer**, as we did in KG-Bench to highlight what graph features influence a prediction. We will also monitor for biases (e.g. the system favoring well-studied genes or common diseases) and mitigate them by ensuring diverse training data and testing on rare conditions – leveraging the bias assessment modules from KG-Bench. Ultimately, the goal is a system that researchers and clinicians can trust to uncover credible repurposing opportunities, explain its reasoning, and adapt to new scientific knowledge.

Implementation Plan and Tools

Building this system will require integrating techniques from natural language processing, databases, and graph machine learning. Key components and tools include:

- **Language Model & Reasoning Engine:** Use an open-source biomedical **LLM** (such as PMC-LLaMA or other fine-tuned LLaMA models) to drive hypothesis generation. The **LangChain** framework can facilitate creating an agent that orchestrates the LLM and calls external data tools as needed. We may also utilize **Hugging Face Transformers** for model fine-tuning or inference. For local deployment of models, tools like **Ollama** or containerized LLM servers will be used to ensure data privacy (important when integrating unpublished data) and fast inference.

- **Biomedical Knowledge Sources:** Rely on high-quality open datasets to ground the knowledge graph. The **Open Targets** platform (via its API) will be central for known drug–target–disease associations. We will integrate additional databases: e.g. **ChEMBL** and **DrugBank** for chemical–protein interactions, **DisGeNET** for gene–disease links, **ClinicalTrials.gov** for trial outcomes, and literature mining from PubMed. Data will be parsed and stored in a graph format (using Python libraries for data handling). We might use a graph database (**Neo4j** or **ArangoDB**) if live querying is needed, but a lightweight approach with in-memory graphs (using **NetworkX** or PyG's data structures) may suffice for prototyping.

- **Knowledge Graph and GNN Training:** Construct the KG with nodes for drugs, diseases, genes, etc., and edges for their relationships. Node features can incorporate descriptors (e.g. chemical fingerprints, gene ontology terms). We will use **PyTorch Geometric (PyG)** to implement GNN models on this graph, as it provides convenient layers and loaders for heterogeneous graphs. Our prior open-source framework **KG-Bench** already provides scripts to build a similar KG from Open Targets and evaluate GNN performance – this can be extended and customized for the project. We will benchmark multiple GNN architectures (GCN, GraphSAGE, GAT, TransformerConv, etc.) for link prediction performance, using evaluation metrics like AUC/APR similar to KG-Bench. For each hypothesis subgraph, the GNN model will output a confidence score for the link. If needed, **DGL** can be explored for scaling to larger graphs or for distributed training on a cluster.

- **Interpretability and Trustworthiness:** To ensure the system's decisions are interpretable, we will incorporate tools such as **GNNExplainer** (for explaining graph model predictions by highlighting influential nodes/edges) and LLM explanation techniques (prompting the LLM to justify its answers with citations to the retrieved data). We will also track uncertainty; for example, using Bayesian neural network layers or dropout-based uncertainty in the GNN, and prompting the LLM to express its confidence or alternative hypotheses. This aligns with the project's emphasis on *trustworthy AI* for biomedical discovery.

- **Workflow Orchestration:** Using **n8n** or similar automation tools, we will link the components into a cohesive pipeline. For instance, an n8n workflow could trigger: (i) an LLM agent to generate hypotheses, (ii) a Python script to query databases and build the graph, (iii) a model inference step for the GNN, and (iv) a reporting step to summarize results. This makes the system modular and easy to update. It also enables scheduling periodic runs (to implement the recursive validation every few weeks or months, ingesting newly available data).

By the end of the project, the expected deliverable is a **fully automated platform** for drug repurposing hypothesis generation and validation. This platform will be evaluated retrospectively (e.g. can it "rediscover" known drug repurposing successes and predict recent discoveries that were confirmed in later updates?) and, prospectively, generate novel predictions for under-explored diseases. Success will be measured by the system's ability to consistently propose biologically **plausible**, **effective** drug candidates (as judged by domain experts or early experimental evidence), and by its adaptability in incorporating new data over time. If successful, this research will contribute a novel AI-driven approach to accelerate drug discovery, demonstrating how **reasoning LLMs and graph learning** can synergize for real-world scientific and medical impact.

Current Open-Source Tools for Each Stage

- **Large Language Models (LLMs):** *PMC-LLaMA* (open 13B biomedical LLM)pubmed.ncbi.nlm.nih.govpubmed.ncbi.nlm.nih.gov; *BioGPT* (biomedical transformer model); *Galactica* (scientific LLM). These can be fine-tuned on biomedical text and deployed via **Ollama** or HuggingFace pipelines.

- **LLM-Orchestration Frameworks: LangChain** (for building agents that let LLMs query tools, APIs, and handle multi-step reasoning), **LlamaIndex/GPT-Index** (for connecting LLMs with document or database indexes).

- **Biomedical Databases/APIs: Open Targets API** (query target-disease-drug associations); **ChEMBL API** (drug–target bioactivity data); **PubChem** and **DrugBank** (chemical and drug information); **DisGeNET** (gene–disease associations); **ClinicalTrials.gov** or **PubMed** APIs (for up-to-date trial and literature data).

- **Knowledge Graph Construction and Storage:** Python libraries like **pandas** (for data wrangling) and **NetworkX** (for basic graph manipulation); graph databases such as **Neo4j** (with its Cypher query language) or **ArangoDB** for persistent storage and complex querying of biomedical KGs.

- **Graph Machine Learning: PyTorch Geometric (PyG)** – a rich library for GNN layers and training pipelines on graphs; **Deep Graph Library (DGL)** – an alternative GNN library supporting multiple backends; **OGB** (Open Graph Benchmark) datasets and evaluators for standardized evaluation; our own **KG-Bench** framework (open-source on GitHub) for benchmarking drug repurposing link prediction with GNNs.

- **Interpretability and Trustworthiness: GNNExplainer** (for explaining GNN predictions) integrated via PyG; **Captum** (PyTorch's model interpretability library) for feature importance; and prompt-based explanation techniques for LLMs (e.g. ask the LLM to output supporting references for each hypothesis). Also, libraries like **SHAP** or **LIME** could be applied to simpler ML components if needed.

- **Workflow Automation and Integration: n8n** (node-based automation tool for scheduling tasks and connecting different services); **Apache Airflow** or **Luigi** (for more complex pipeline scheduling); containerization with **Docker/Kubernetes** for deploying the multi-component system (especially if scaling the agent to handle many hypotheses in parallel).

Using these tools, the project will be implemented in a modular, reproducible fashion. Each stage (LLM reasoning, data retrieval, graph analysis, validation) will have tested components that can be developed and improved independently, then seamlessly connected to form the full automated drug repurposing discovery engine.