Two-Stage Drug Repurposing Link Prediction Project on OGBL-BioKG

Problem definition and dataset

Knowledge graphs (KGs) encode information as triples ((h, r, t)), where (h) and (t) are head/tail entities and (r) is a relation type. Link prediction, or knowledge-graph completion, asks us to rank candidate tails (or heads) so that true triples score higher than false ones. For biomedical drug repurposing, this means predicting plausible drug–disease, drug–protein or other biomedical interactions by learning from observed interactions.

The **OGBL-BioKG** dataset from the Open Graph Benchmark is designed for KG completion in the biomedical domain. It is a heterogeneous KG created from multiple biomedical data repositories. The graph contains five types of entities—diseases, proteins, drugs, side-effects and protein functions—and 51 directed relation types. Relations connecting the same entity types (e.g., protein–protein or drug–drug) are symmetric and are represented with bidirectional edges. Evaluation follows the OGB KG-completion protocol: given a query triple ((h,r,?)), we rank candidate tail entities and measure performance with mean reciprocal rank (MRR) and Hits@(K); when corrupting a head or tail we only sample negative entities of the same type.

Stage 1 – Baseline establishment

The first stage builds a strong set of baselines for link prediction on OGBL-BioKG. These baselines provide a reference point for later research work and highlight strengths and weaknesses across simple heuristics, knowledge-graph embeddings (KGEs) and graph neural networks (GNNs).

1. Heuristic baseline

Implement a simple, type-aware heuristic as a sanity check.

Metapath counts. Count the number of length-2 or length-3 paths that connect head and tail entities via specific relation sequences (e.g., drug → protein → disease). These counts can serve as features for a simple scoring function.

The heuristic baseline should produce non-trivial rankings and help identify obvious failure modes (e.g., hub entities or extremely sparse relations).

2. Knowledge-graph embedding (KGE) baselines

Knowledge-graph embedding models map entities and relations into vector spaces and score triples using a learned function. A classic example is **DistMult**, which scores a triple ((h,r,t)) by the trilinear product of the entity and relation embeddings; the DistMult algorithm was introduced in 2015 and reports metrics such as MRR and Hits@10 on benchmark datasets.. A related model is **ComplEx**, which extends DistMult to

complex-valued embeddings to capture asymmetric relations. Training KGE models requires sampling negative triples; the quality of these negative samples strongly influences accuracy.

For this stage, train at least two KGE models (DistMult and ComplEx) on OGBL-BioKG. Use type-aware negative sampling, tune hyperparameters on the validation set, and report MRR and Hits@(K) with multiple seeds. Provide per-relation breakdowns to show how different relation types behave.

3. Heterogeneous GNN baseline

Graph neural networks generalize convolution to relational graphs by aggregating messages from neighbors. **Relational GCNs (R-GCNs)** are message-passing frameworks designed for relational graphs. They learn latent features by applying relation-specific weight matrices to neighbor embeddings and have become widely used in knowledge-graph applications.

Implement an R-GCN (or a variant such as CompGCN) with a triple-scoring decoder. Train it under the same evaluation protocol as the KGE baselines. Compare performance against KGEs, and include two quick ablations: collapsing relation types into a single type to simulate homogeneous GNNs, and altering the negative sampling scheme.

4. Baseline diagnostics

After establishing baselines, analyse where they succeed or fail:

- **Relation frequency slices.** Break down performance by relation frequency to see whether models handle rare relation types.
- **Degree slices.** Evaluate performance on low-degree versus high-degree entities to understand cold-start behaviour.
- **Error cases.** Inspect a handful of high-confidence false positives and false negatives; classify failures such as hub bias, ambiguous relations or contradictory signals.

This diagnostic analysis will motivate the research stage by identifying challenges (e.g., poor performance on low-degree drugs or rare side-effect relations) and will provide data splits for subsequent experiments.

Stage 2 – Evidence-grounded reranking (research focus)

The second stage builds a research-oriented system that uses graph evidence to rerank candidates produced by the baseline model. The aim is to improve predictive performance on hard cases and produce explanations grounded in the KG.

1. Candidate generation

Use the best baseline model from Stage 1 (e.g., the stronger of DistMult, ComplEx or R-GCN) as a **searcher**. Given a query ((h,r,?)), compute scores for all possible tails and extract the top (K) candidates. Choosing (K) large enough (e.g., 50 or 100) ensures the true answer is usually in the shortlist.

2. Evidence retrieval from the graph

For each candidate (t_i), retrieve a compact evidence set from the KG to support reasoning. Two common strategies are:

- **Multi-hop paths.** Find the shortest or most diverse paths of length two to four connecting the head and candidate tail (e.g., drug → protein → disease). These paths capture relational context.
- **Enclosing subgraphs.** Extract the (k)-hop neighbourhood around (h) and (t_i) (with an edge budget) and represent it as a set of typed edges.

The evidence is serialized into a machine-checkable format (e.g., a list of triples or path sentences) so it can be fed to downstream models. Retrieving such subgraphs addresses the limitation of classic document-based retrieval: knowledge graphs enable structured relationships and deeper, more contextual retrieval.

3. LLM-based reranker with grounding

The reranking stage employs a large language model (LLM) to evaluate the candidate list using the retrieved evidence. The model sees the query, the candidate entity names and relation labels, and the evidence bundle for each candidate. It must output a ranked list with confidence scores and cite which evidence pieces support each decision. Proper ordering or reranking of retrieved evidence improves downstream performance: prior work notes that reorderings of retrieved information at a fine-grained level are essential for optimal LLM performance.

Grounding the LLM is critical—its predictions must be explainable and based solely on the supplied evidence. To enforce this, the output should follow a structured schema (e.g., JSON) containing the ranking, confidence values and references to the evidence identifiers. Automatically checking whether cited evidence actually exists in the retrieved subgraph provides a **faithfulness** measure.

Example:

```
TASK
Rerank candidate tails for the query (h, r, ?), using only the provided KG evidence.

QUERY
(h=drug:1287, r=drug-disease, ?)

HEAD ENTITY NAME
drug:1287 = "DrugName_A"

CANDIDATES (from Stage 1 model; higher score means more likely)
1) disease:4421  name="DiseaseName_X"  stage1_score=12.31
2) disease:1190  name="DiseaseName_Y"  stage1_score=12.10
3) disease:9002  name="DiseaseName_Z"  stage1_score=11.95

EVIDENCE BUNDLES
Candidate 1: disease:4421 = "DiseaseName_X"
- P1: DrugName_A --drug-protein--> Protein_P53 ; DiseaseName_X --disease-protein--> Protein_P53
- P2: DrugName_A --drug-protein--> Protein_P21 ; DiseaseName_X --disease-protein--> Protein_P21
- P3: DrugName_A --drug-sideeffect--> SideEffect_Nausea  (no connection to DiseaseName_X in evidence)

Candidate 2: disease:1190 = "DiseaseName_Y"
- P4: DrugName_A --drug-protein--> Protein_P53 ; DiseaseName_Y --disease-protein--> Protein_P53
- P5: DrugName_A --drug-drug_struct_sim--> DrugName_B ; DrugName_B --drug-disease--> DiseaseName_Y

Candidate 3: disease:9002 = "DiseaseName_Z"
- P6: DrugName_A --drug-sideeffect--> SideEffect_Nausea ; DiseaseName_Z --disease-protein--> Protein_P77
- P7: (no short paths found within budget)

RULES
- Use only the evidence shown for each candidate.
- Prefer candidates with multiple independent, specific mechanistic connections (e.g., shared proteins).
- Penalize candidates supported only by very indirect similarity links unless evidence is strong.
- If evidence is absent or weak, lower the rank even if stage1_score is high.

OUTPUT JSON SCHEMA
{
  "reranked": [
    {"tail": "disease:____", "rank": 1, "score": 0.0-1.0, "cites": ["P1","P2"]},
    {"tail": "disease:____", "rank": 2, "score": 0.0-1.0, "cites": ["P4","P5"]},
    {"tail": "disease:____", "rank": 3, "score": 0.0-1.0, "cites": ["P6","P7"]}
  ],
  "notes": "one short sentence about why the top choice "
}
```

4. Research experiments and evaluation

To demonstrate that evidence-grounded reranking provides genuine benefits rather than superficial improvements, conduct the following experiments:

- **Comparison with baseline.** Compare overall MRR and Hits@(K) of the reranked system against the baseline generator on the test set. Focus on hard slices, such as rare relations and low-degree entities, to see where reranking yields the largest gains.
- **Evidence ablations.** Evaluate the reranker when given real evidence, random/shuffled evidence, or no evidence at all. A significant performance drop in the random/no-evidence conditions indicates that the model genuinely uses the graph evidence.
- **Retrieval variants.** Try different evidence retrieval strategies (shortest vs. diverse paths, paths vs. enclosing subgraphs) and observe their impact on performance and faithfulness.
- **Faithfulness metrics.** Report the citation validity rate (percentage of cited edges that appear in the retrieved evidence) and analyse cases where the reranker makes incorrect or hallucinated citations.

- **Subgroup analysis.** Stratify results by relation type frequency and node degree to evaluate generalization across challenging slices.

5. Deliverables for Stage 2

At the end of this stage, produce:

1. **Implementation:** Code for candidate generation, evidence retrieval and LLM reranking that can reproduce the experiments.
2. **Results:** Tables comparing baseline and reranked systems, ablation results, and stratified analyses.
3. **Faithfulness report:** Automated checks of citation validity and example predictions with concise evidence-based explanations.
4. **Research summary:** A brief write-up describing the method, key findings and insights into where the system helps or fails.

Resources

Free SOTA LLM API: Gemini 3 Flash (it offer free quota).

Also if you are Google AI Pro (Free for students), if will get 10$ credits per month.

(https://developers.google.com/profile/help/benefits)