

Acronym of the Team: AWESome

Names: Mochamad **Ardiansyah** Nurgaha; **William** Liaw; **Eddie** Groh; **Sri** Raam Appakutti Palani

---

## Multi-class link prediction with PyKEEN and Large Language Models

### Abstract

This report concerns the comparison of PyKEEN embedding models with Large Language Models for Knowledge Graph Completion. The study evaluates their performance in predicting missing links and classifying relationships, highlighting the strengths and limitations of each method.

### Problem Statement

Knowledge graphs are valuable tools for representing complex relationships between entities. However, a significant challenge arises from their inherent incompleteness, as many potential connections between entities are missing. This lack of information limits the utility and accuracy of downstream applications such as recommendation systems, biomedical research, and drug repurposing, which is an initial departing point for the present academic work.

The objective of this project is to address this issue by employing knowledge graph embedding techniques through PyKEEN [Ali et al., 2021] to predict and classify missing links, thereby enriching the graph. Additionally, the project explores how large language models (LLMs) could further enhance or complement traditional embedding-based approaches. We intend to investigate several LLM-based methods, including zero-shot, few-shot, and retrieval-augmented generation (RAG), to assess their effectiveness in knowledge graph completion tasks.

### Methodology

#### PyKEEN

PyKEEN [Ali et al., 2021] is an open-source Python library that facilitates training and evaluation of knowledge graph embedding models. It streamlines the process of embedding entities and

relations into continuous vector spaces, enabling efficient link prediction and relationship classification. PyKEEN supports a wide range of models, including TransE, RotatE, ComplEx, and DistMult, each with unique characteristics and performance profiles.

In this project, PyKEEN was used to predict missing links within a knowledge graph through a structured workflow involving data extraction, preparation, model training, and link prediction. Triples  $(h, r, t)$  representing head entities, relationships, and tail entities were first retrieved from the Neo4j database using Cypher queries (Listing 1). These triples were then converted into PyKEEN’s `TriplesFactory` format to enable seamless integration with the PyKEEN pipeline. The RotatE algorithm was employed to train the embedding model over 20 epochs with an embedding dimension of 128. It’s worth noting that 20 epochs may be insufficient for meaningful training on large or complex graphs, especially with a high embedding dimensionality. Additionally, applying early stopping in this scenario is unlikely to yield significant improvements due to the limited number of epochs.

Finally, the trained model was used to perform link prediction, identifying potential relationships within the knowledge graph.

Listing 1: Cypher query to retrieve triples.

1

MATCH (h) -[r]->(t)

2

RETURN id(h) AS head, type(r) AS relation, id(t) AS tail

## Neo4j Desktop

Neo4j Desktop [Neo4j, 2024] serves as a crucial tool for managing and querying the knowledge graph used in this project. It provides a local environment to import, visualize, and manipulate graph data, facilitating seamless interaction with the dataset. Neo4j’s support for Cypher, its declarative graph query language, enables efficient extraction of triples representing relationships between entities. This functionality is essential for preparing the knowledge graph data, which is subsequently processed using PyKEEN for link prediction and multi-class relationship classification. By leveraging Neo4j Desktop, we ensure that the data pipeline from graph ingestion to embedding model training remains streamlined and adaptable.

## Setup and Database Import

This section provides a step-by-step guide to set up Neo4j Desktop and import a database dump file, as depicted in Figure 1. The specific dump used in this midterm report, derived from Hetionet, requires DBMS version 4.3 to ensure compatibility.

Upon successful import, the schema can be visualized using the following Neo4j query: `CALL db.schema.visualization()`. The resulting graph, illustrates key entities and relationships that form the basis for multi-class link prediction and knowledge graph completion in subsequent tasks.

## Large Language Models

We integrate RotatE with Llama3.2:3b with Wikidata entries to entities (RLM-A) and RotatE Llama3.2:3b embeddings (RLM).

Future work will involve adjusting and/or integrating other large language models (LLMs), exploring zero-shot, few-shot, and retrieval-augmented generation (RAG) techniques to enhance knowledge graph completion.

# Experimentation

This section outlines the dataset used for knowledge graph completion, presents the results of the experiments conducted using PyKEEN and LLMs. It also provides a comparison of the models’ performance based on evaluation metrics such as Mean Reciprocal Rank (MRR), Hits@K, and Mean Rank (MR).

## Dataset and Statistics

The Hetionet dataset [Himmelstein et al., 2017] serves as the initial foundation for this project, providing a structured biomedical knowledge graph that integrates data from various sources to represent relationships like *treats*, *binds*, and *causes* between genes, compounds, diseases, and other biological entities. While Hetionet offers a diverse and well-documented schema, its role in this project is primarily exploratory. As we compare knowledge graph completion approaches using PyKEEN and large language models (LLMs), the dataset provides a valuable testbed for initial experiments. However, the suitability of Hetionet for future phases of the project remains under evaluation, and additional datasets may be considered as the scope of the task evolves.

Key dataset statistics are summarized in Table 1.

Table 1: Dataset Statistics (Hetionet Subset)

Statistic	Value
Nodes (Entities)	22634
Relationships (Edges)	561721
Unique Relation Types	10
Unique Triples	561721

## Experimental Results and Evaluation Metrics

To assess performance in predicting missing links using embedding models, the dataset is split into training (80%), validation (10%), and testing (10%) sets. With this setup, as shown in Figure 2, PyKEEN’s RotatE model’s training loss decreases steadily, indicating successful optimization over the course of training.

Evaluation is performed using standard link prediction metrics, including:

- **Mean Reciprocal Rank (MRR)**: Measures the average inverse rank of the correct entity.
- **Hits@K**: Calculates the proportion of correct predictions ranked in the top  $K$ .
- **Mean Rank (MR)**: Provides the average rank of the correct entity.

The results are summarized in Table 2, comparing several embedding models available on PyKEEN:

- Custom models with LLM (Llama3.2:3b): RLM-A, RLM
- Rotational models: RotatE
- Translation models: TransE, TransH, TransR, TransD
- Factorization models: RESCAL, TuckER, DistMult

Table 2: Performance Comparison of Models

Model	Hits@1	Hits@3	Hits@5	Hits@10	Mean Rank	Mean Reciprocal Rank
RLM-A	0.0260	0.0564	0.0768	0.1141	1216.26	0.0577
RLM	0.0258	0.0557	0.0763	0.1131	<b>1199.26</b>	0.0573
RotatE	<b>0.0304</b>	<b>0.0650</b>	<b>0.0861</b>	<b>0.1254</b>	1304.77	<b>0.0643</b>
TransE	0.0027	0.0131	0.0217	0.0399	1631.40	0.0181
TransH	0.0036	0.0095	0.0147	0.0254	2061.96	0.0135
TransR	0.0015	0.0048	0.0079	0.0154	2314.35	0.0089
TransD	0.0066	0.0187	0.0281	0.0469	1504.44	0.0232
RESCAL	0.0000	8.9e-6	2.7e-5	0.0001	9866.04	0.0002
TuckER	0.0043	0.0113	0.0176	0.0310	2487.35	0.0158
DistMult	0.0064	0.0174	0.0258	0.0427	2570.99	0.0209

Using RotatE model, predicted *treats* relationships for L-Asparagine were visualized (Figure 4) suggesting potential links to diseases such as melanoma, ulcerative colitis, and coronary disease. These predictions will require further validation to confirm biological plausibility. As illustrated in Figure 3, the graph showcases relationships between the compound L-Asparagine and breast cancer, mediated by various genes. These indirect paths offer insights into potential mechanisms that could explain the model’s predictions

## Analysis

In general, the results indicate poor performance across all tested models for knowledge graph completion on the Hetionet dataset (subset). RotatE performs the best among the models tested, achieving the highest MRR (0.0643) and Hits@10 (0.1254), along with a competitive Mean Rank (1304.77). Meanwhile, translational models like TransE, TransH, TransR, and TransD generally perform poorly, likely due to simple geometric assumptions (linear or relational translations) that fail to capture complex patterns in the dataset. Semantic matching models, such as DistMult, RESCAL, and TuckER, also struggled, with RESCAL performing exceptionally poorly showed severe overfitting and scalability issues.

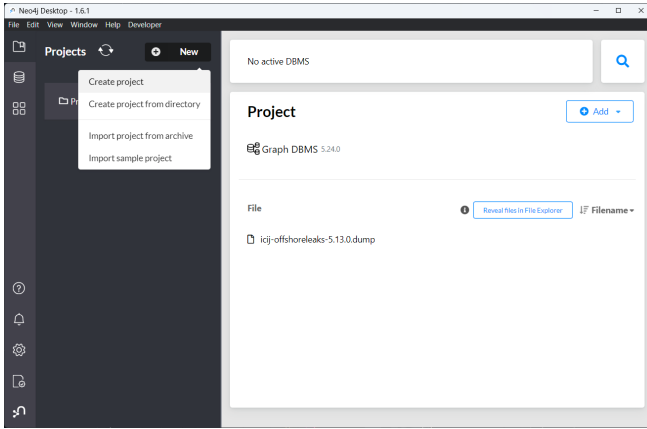
**Comparison with LLM integration (LLama3.2:3b).** This custom models did not perform as well as expected (below RotatE standalone). Both Hits@10 are around 0.11, suggesting they capture relational patterns but may lack rotational expressiveness. Training losses can be seen in Figure 5 and Figure 2.

## Discussion

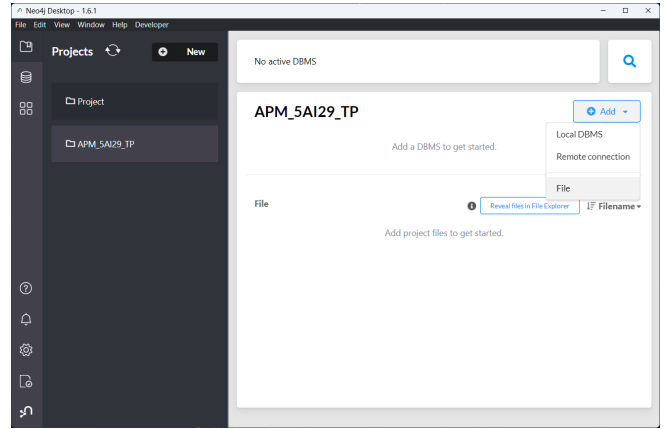
To mitigate errors encountered, future work will explore increasing the number of epochs and refining hyperparameters and incorporating negative sampling strategies to improve generalization. Moreover, Hetionet contains biomedical data with heterogeneous relationships, which may require more complex models to capture the underlying patterns. We also recommend exploring integration with other LLMs such as BioBERT (pretrained on biomedical texts), rule-based models (non-embedding methods) such as AnyBURL if embeddings fail to capture biomedical patterns, pretrained and meta-learning models (BoxE) for modeling constraints in large datasets, CNN-based models (ConvE, R-GCN) for better capture local features effectively, or advanced models (HolE, AutoSF) for complex and heterogeneous dataset like Hetionet.

# Bibliography

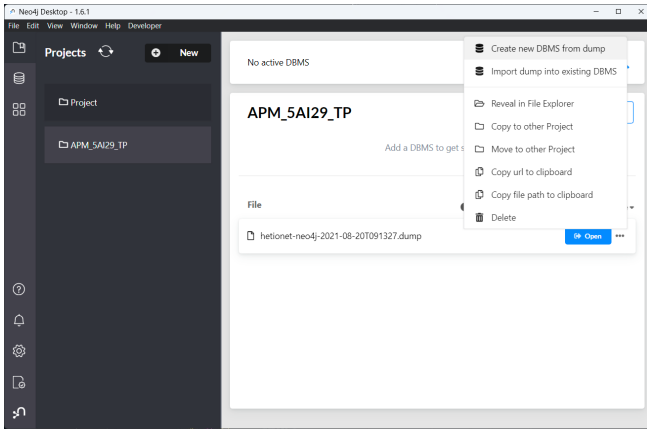
- [Ali et al., 2021] Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Sharifzadeh, S., Tresp, V., and Lehmann, J. (2021). PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *Journal of Machine Learning Research*, 22(82):1–6.
- [Himmelstein et al., 2017] Himmelstein, D. S., Lizée, A., Hessler, C., Brueggeman, L., Chen, S. L., Hadley, D., Green, A., Khankhanian, P., and Baranzini, S. E. (2017). Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife*, 6:e26726.
- [Neo4j, 2024] Neo4j (2024). Neo4j graph database & analytics | graph database management system.



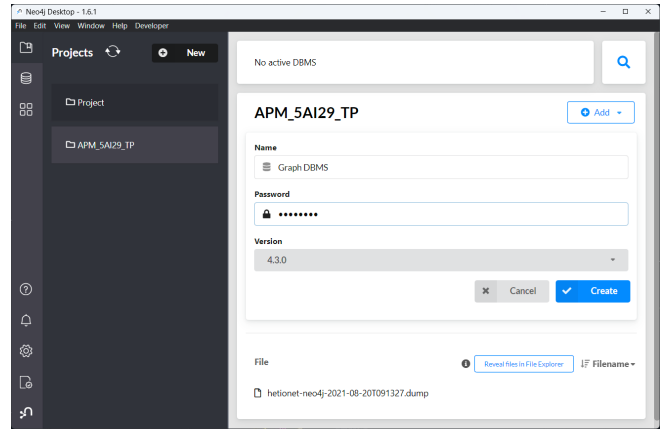
(a) Creating a project.



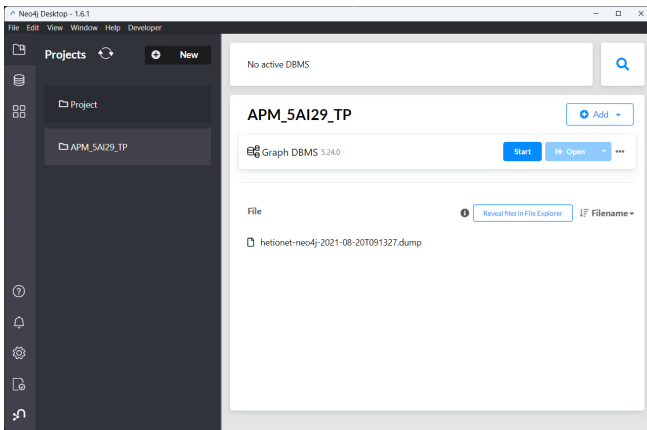
(b) Adding dump file.



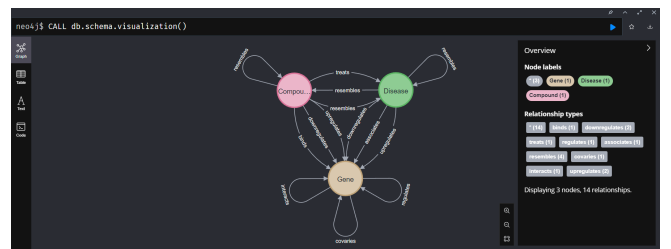
(c) Importing dump to DBMS.



(d) Configuring DBMS.



(e) Starting DBMS.



(f) Visualizing schema.

Figure 1: Steps for Neo4j Desktop Setup.

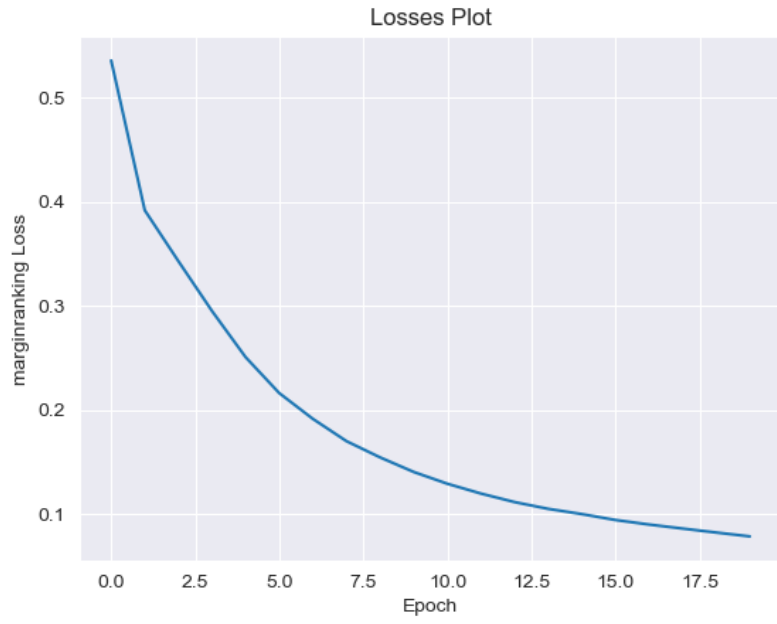


Figure 2: Training loss plot for PyKEEN RotatE model over 20 epochs.

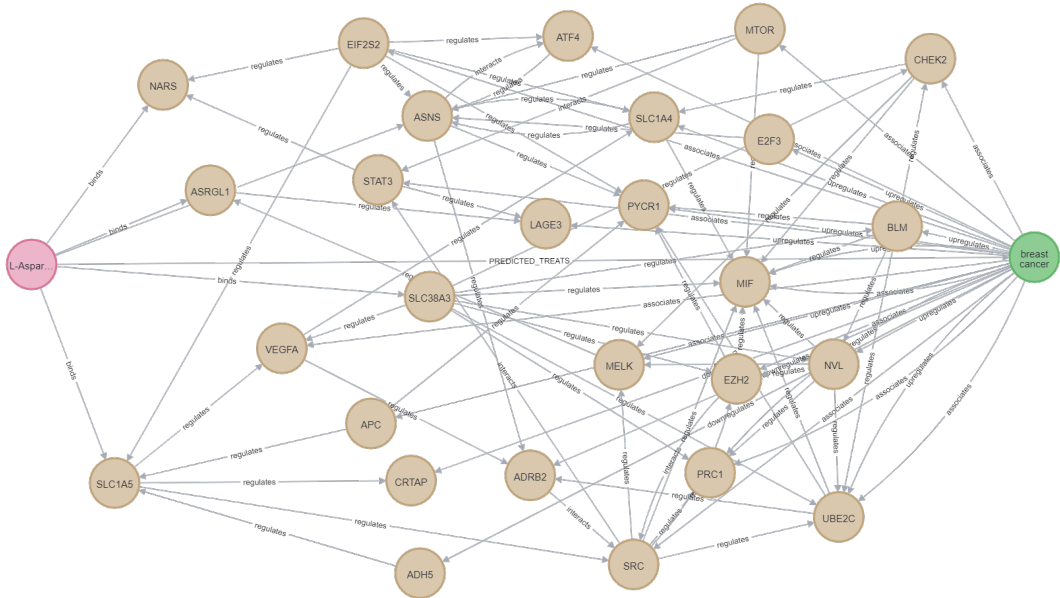


Figure 3: Visualization of connections between L-Asparagine, genes, and breast cancer, highlighting predicted relationships.



Figure 4: 5 predicted *treats* relationships for L-Asparagine.

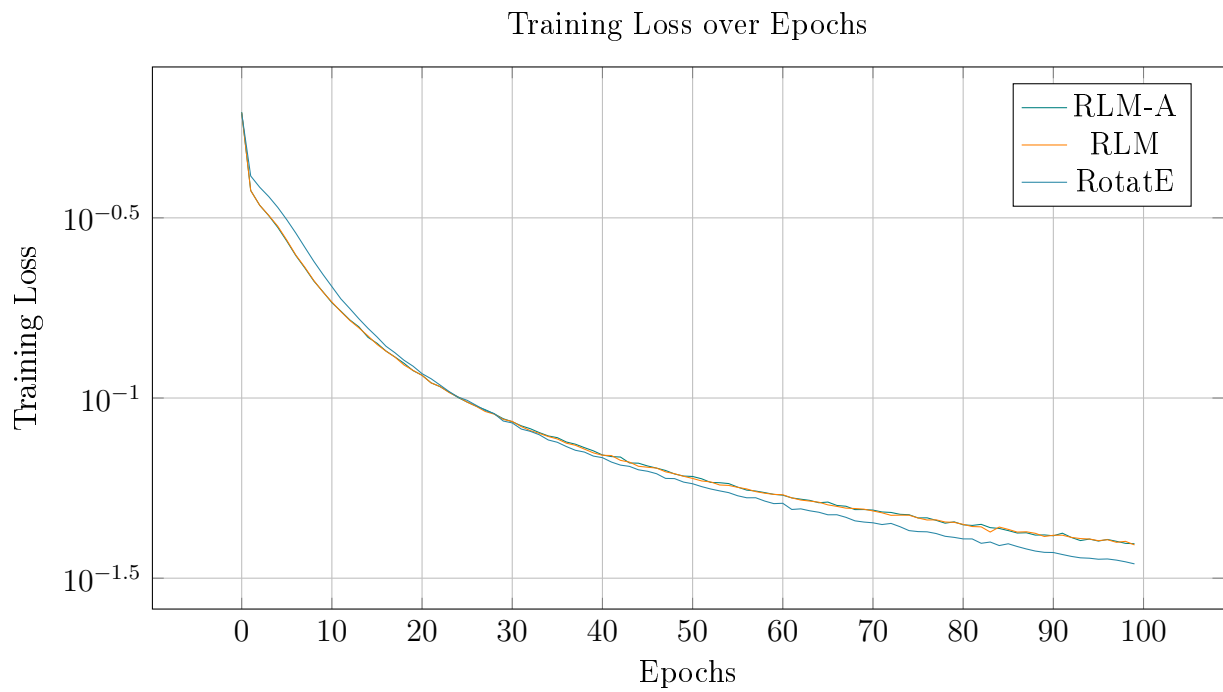


Figure 5: Logarithmic plot of training loss.