January, 2025

APM_5AI29_TP

Language Models and Structured Data

Mid-term Project Report

Acronym of the Team: AWESome
Names: Ardiansyah Nurgaha; William Liaw; Eddie Groh; Sriraam Appakutti

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.

LLM integration remains a future step. (SOMEONE STILL NEEDS TO IMPLEMENT IT AND CHANGE THIS SECTION

)

Methodology

PyKEEN

FURTHER EXPLAIN MODELS USED IN EXPERIMENTATION

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.

COULD USE HYPERPARAMETER TUNING: INCREASE EPOCHS

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.

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

SOMEONE STILL NEEDS TO IMPLEMENT IT AND CHANGE THIS SECTION

Future work will involve integrating large language models (LLMs) to complement PyKEEN's predictions, 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, and includes an analysis of errors observed during model evaluation.

Dataset and Statistics

REVISIT AND ADAPT THIS SECTION IF WE DECIDE TO CONTINUE WORKING WITH THIS DATASET

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.

Table 2: Experimental Results

Model	MRR	Hits@10	MR
RotatE	0.050	0.099	1349.6
TEST OTHER MODELS?	...	...	...
LLM (zero-shot)	...	...	...
LLM (few-shot)	...	...	...
LLM (RAG)	...	...	...

Table 3: 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	**1199.26**	0.0573
RotaE	**0.0304**	**0.0650**	**0.0861**	**0.1254**	1304.77	**0.0643**

The results are summarized in Table 2, comparing PyKEEN's models: RotatE, ...

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

Error Analysis

COMPLETE THIS SECTION

0.1 Comparison LLM integration

Fix this mess

RLM-A: Llama3.2-3B provided with Wikidata entries to entities RLM: Llama3.2-3B embeddings RDE: Pipeline with random embeddings RotatE:

Training losses can be seen in 5 and 3

To mitigate these errors, future work will explore: increasing the number of epochs and refining hyperparameters and incorporating negative sampling strategies to improve generalization.

Discussion

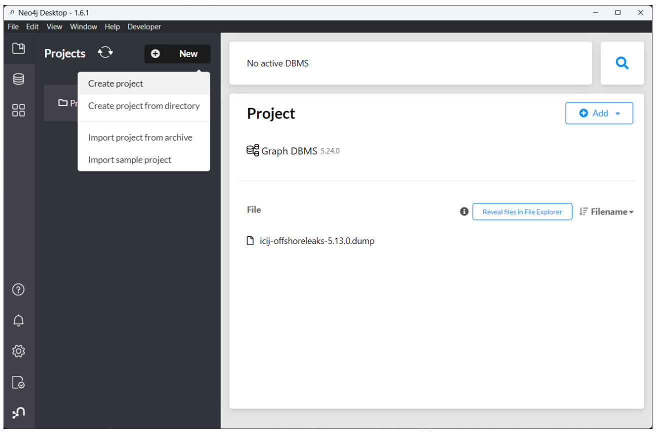
Complete this section

Bibliography

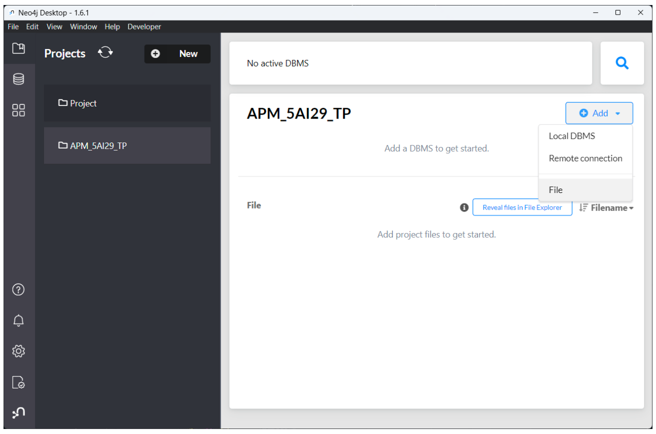
[Ali et al., 2021] Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Sharifzadeh, S., Tresp, V., and Lehmann, J. (2021). PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *Journal of Machine Learning Research*, 22(82):1–6.

[Himmelstein et al., 2017] Himmelstein, D. S., Lizee, 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.

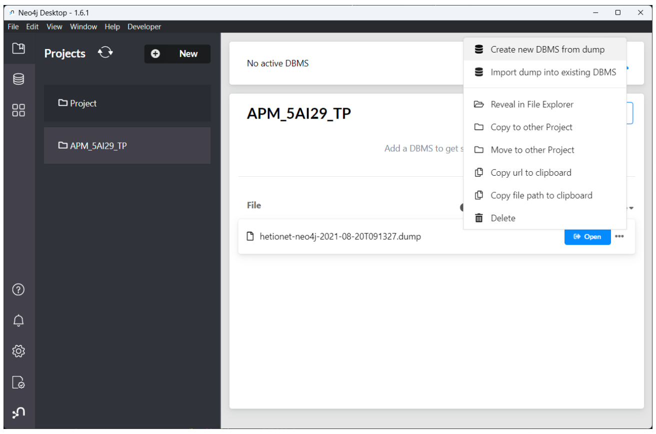
[Neo4j, 2024] Neo4j (2024). Neo4j graph database & analytics | graph database management system.


Neo4j Desktop - 1.6.1
File Edit View Window Help Developer
Projects
New
Create project
Create project from directory
Import project from archive
Import sample project
No active DBMS
Project
Add
Graph DBMS 5.24.0
File
Reveal files in File Explorer
Filename
icij-offshoreleaks-5.13.0.dump

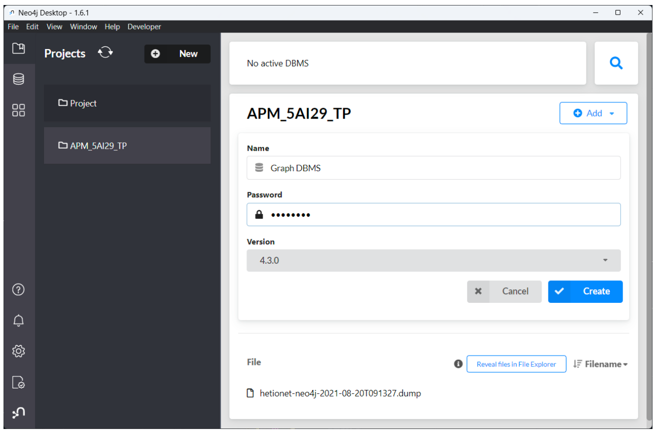
(a) Creating a project.


Neo4j Desktop - 1.6.1
File Edit View Window Help Developer
Projects
New
Project
APM_5AI29_TP
No active DBMS
APM_5AI29_TP
Add
Add a DBMS to get started.
Local DBMS
Remote connection
File
File
Reveal files in File Explorer
Filename
Add project files to get started.

(b) Adding dump file.

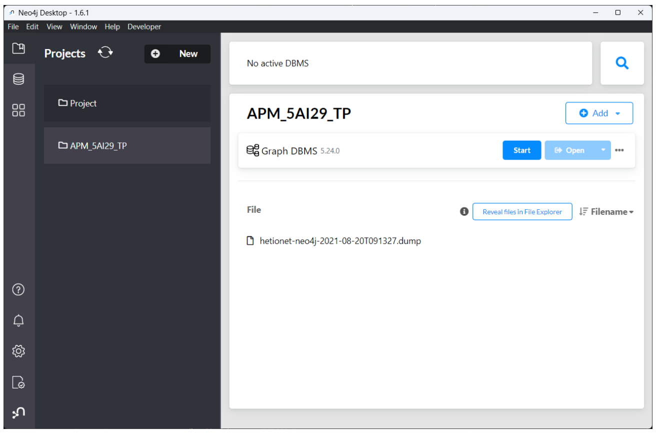


(c) Importing dump to DBMS.

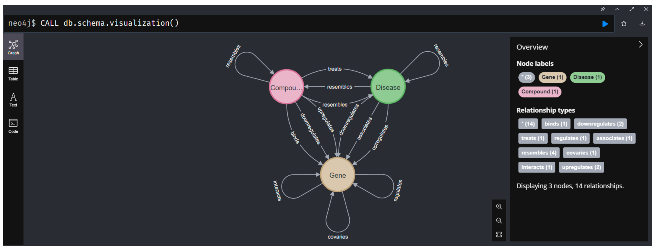


Name: Graph DBMS
Password: ••••••••
Version: 4.3.0

(d) Configuring DBMS.



(e) Starting DBMS.



```
neo4j$ CALL db.schema.visualization()
```

Displaying 3 nodes, 14 relationships.

(f) Visualizing schema.

Figure 1: Steps for Neo4j Desktop Setup.

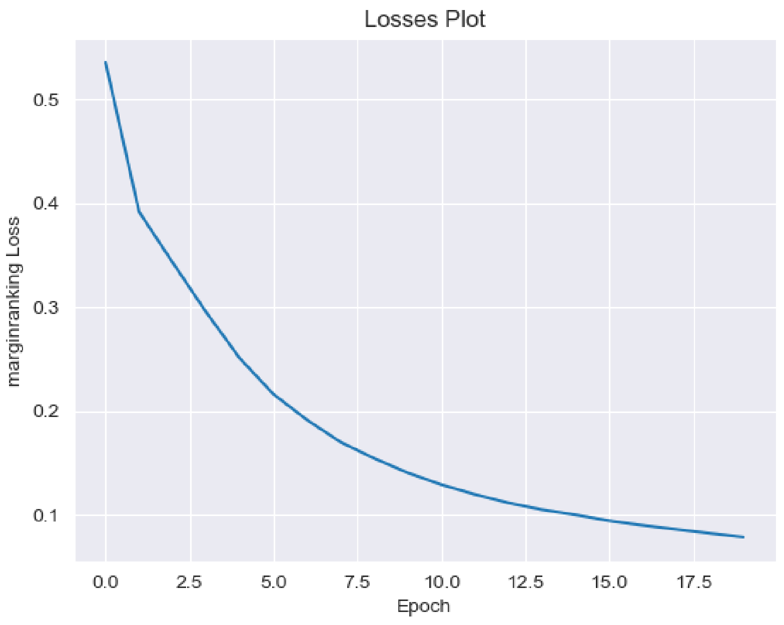


Figure 2: Training loss plot for PyKEEN RotatE model over 20 epochs.

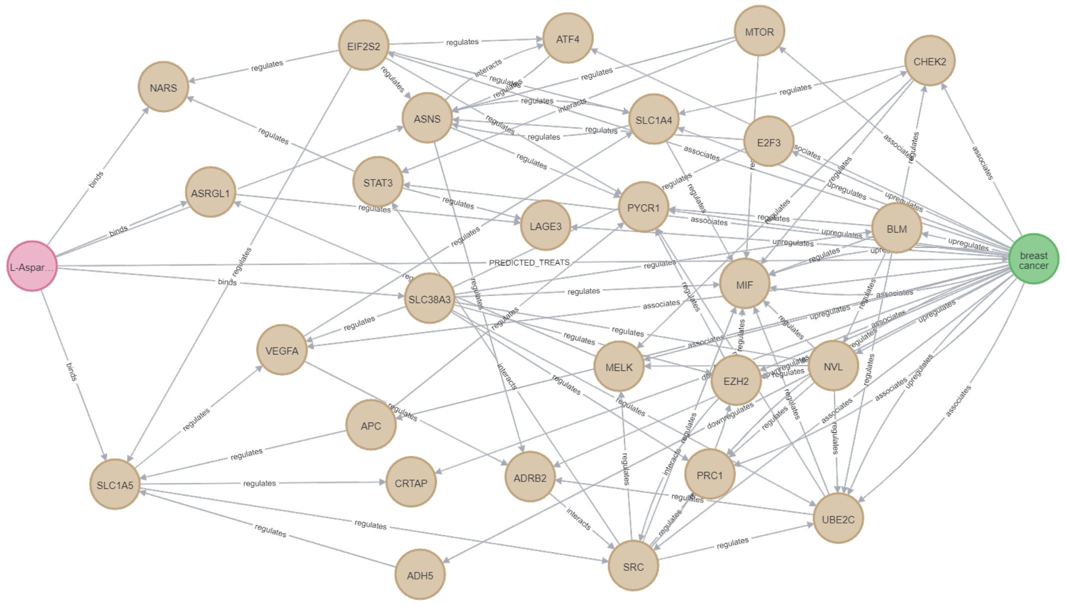


Figure 3: Visualization of connections between L-Asparagine, genes, and breast cancer, highlighting predicted relationships.

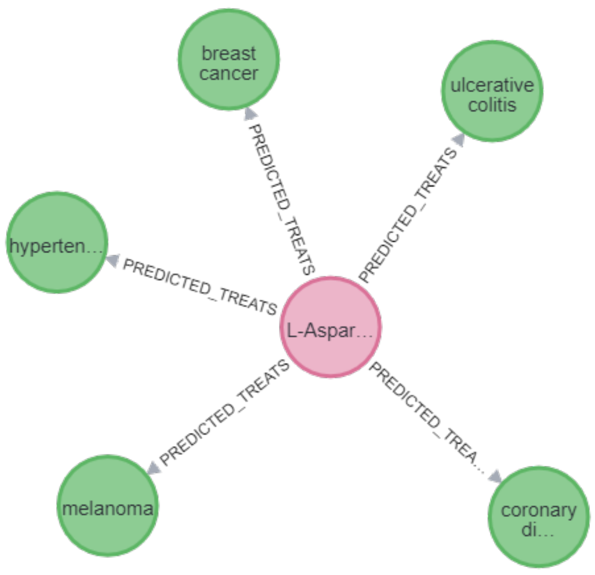


Figure 4: 5 predicted *treats* relationships for L-Asparagine.

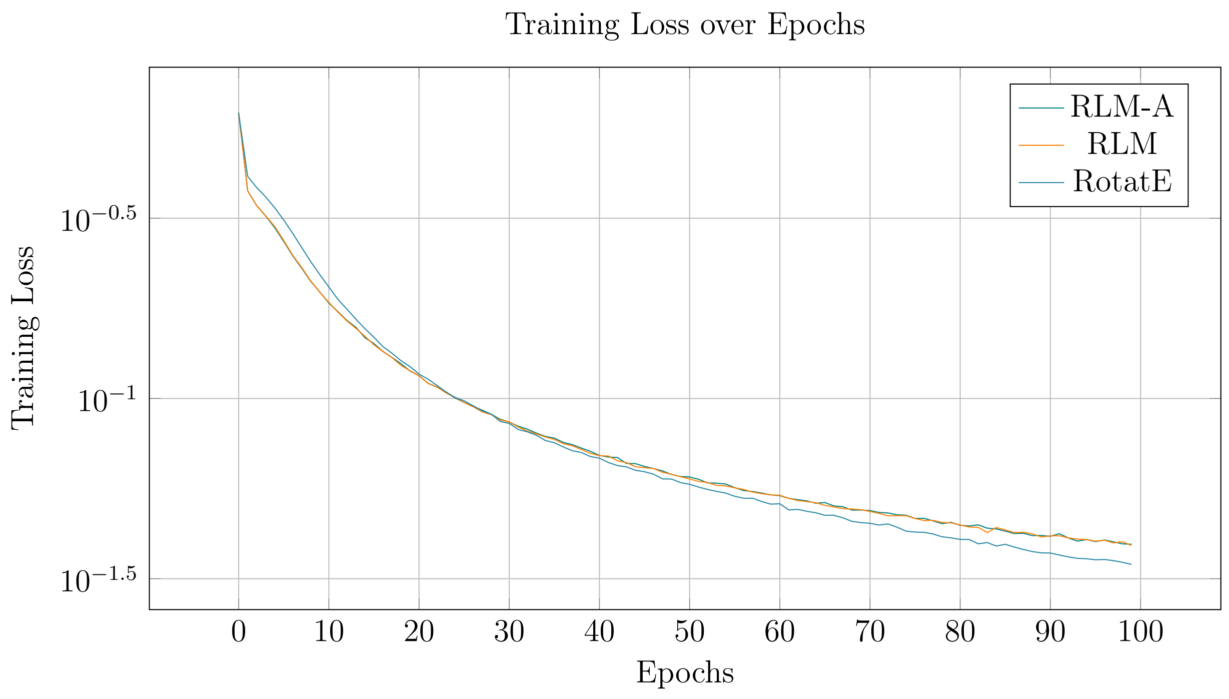


Figure 5: Logarithmic plot of training loss.