January, 2025

APM_5AI29_TP

Language Models and Structured Data

Mid-term Project Report

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

---

Multi-class link prediction with PyKEEN and Large Language Models

Abstract

This report evaluates knowledge graph embedding models for link prediction, comparing PyKEEN with large language model-based approaches. Using Hetionet graph, we assess PyKEEN and introduce a method using Llama embeddings, with and without retrieval-augmented generation.

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 therefore investigated 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.

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.

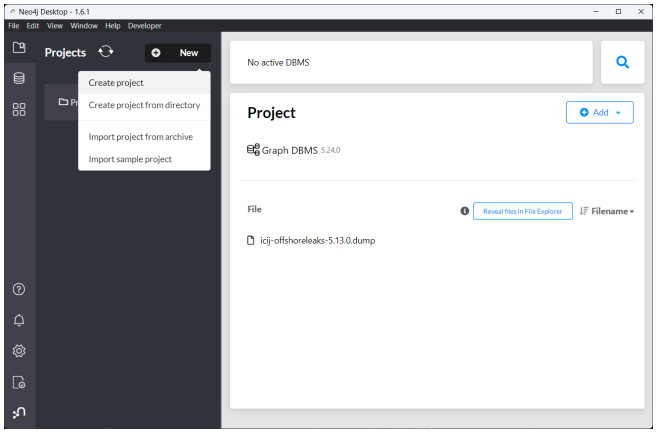
Upon successful import, the schema can be visualized using the following Neo4j query: `CALL db.schema.visualization()`. The resulting graph, illustrates key entities and relationships that form the basis for multi-class link prediction and knowledge graph completion in subsequent tasks.

Evaluated Models

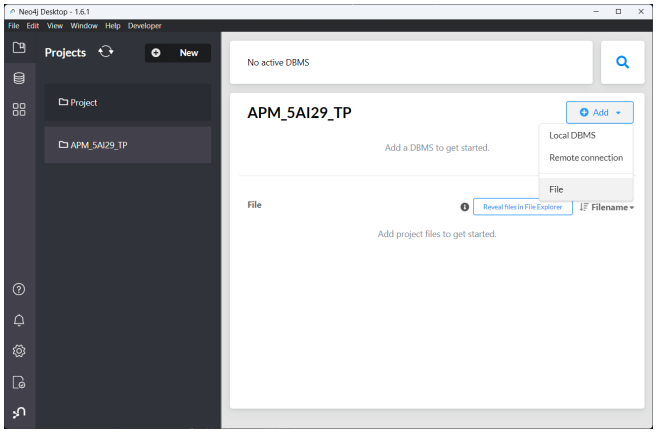
In this report, we evaluate the performance of knowledge graph completion models, with a particular focus on comparing the performance of traditional embedding models with LLM-based embeddings. As our primary baseline, we used the RotatE model, configured with 128-dimensional embeddings and trained for 100 epochs with early stopping. While we explored several additional models, their performance was found to be subpar, and as such, we will not discuss them further.

For evaluating the performance of the LLM-based approaches, we selected the Llama 3.2-3B model. This model was chosen primarily for its competitive output quality while maintaining relatively low VRAM requirements.

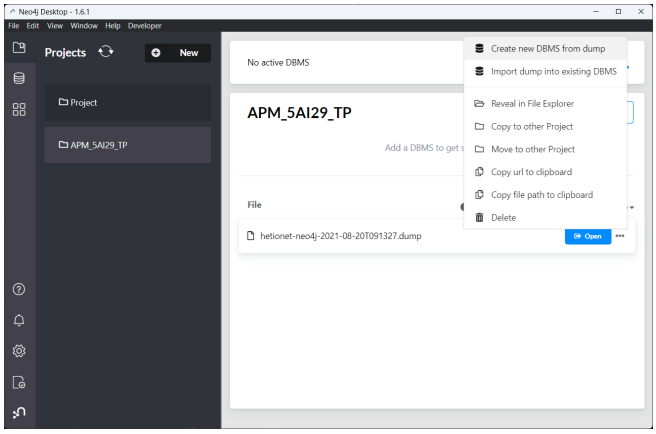
Based on the RotatE approach, we used a new architecture which combines 128 dimensions of conventional trainable embedding with 32 dimensions of projected LLM embeddings, created by a trainable linear layer which reduces the 3200 dimensional immutable LLM embedding. For creating the embeddings, we used two types of inputs. The first variant uses only the LLaMA embeddings generated from the label names, referred to as RLM. The second variant incorporates Wikidata's



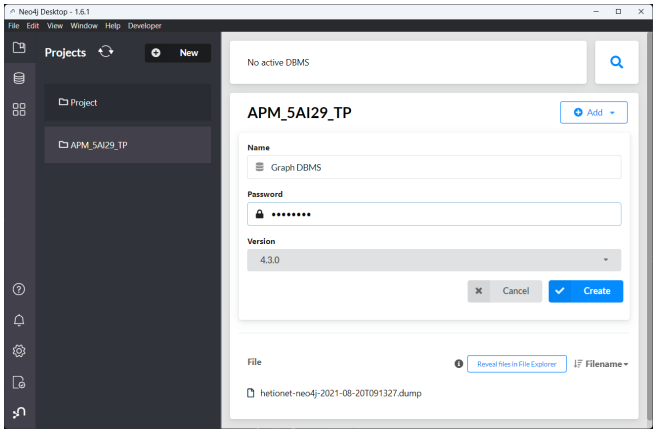
(a) Creating a project.



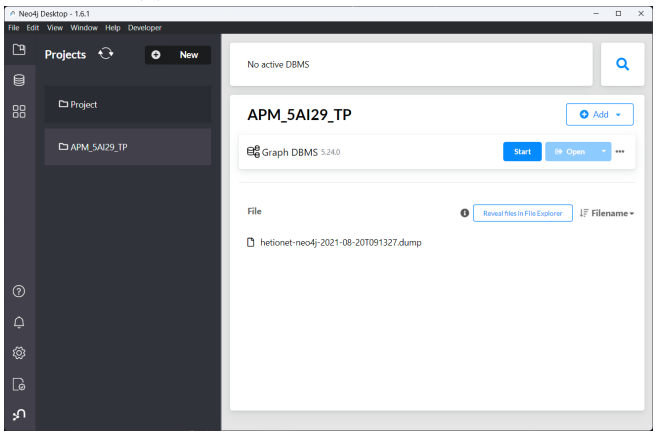
(b) Adding dump file.



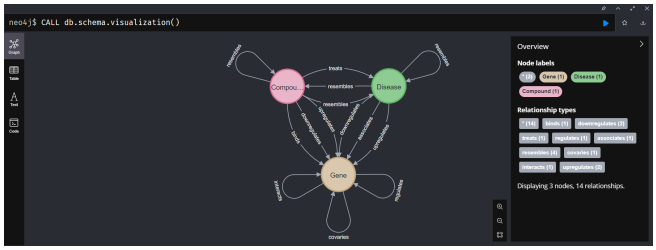
(c) Importing dump to DBMS.



(d) Configuring DBMS.



(e) Starting DBMS.



(f) Visualizing schema.

Figure 1: Steps for Neo4j Desktop Setup.

Retrieval-Augmented Generation (RAG) technique, which we term RLM-A (RLM Augmented). These inputs are then tokenized in batches of 32, with a maximum sequence length of 128 tokens. These tokenized inputs are then passed through the Llama model, and the embeddings are derived from the final hidden layer. Specifically, the first token's hidden state (similar to a [CLS] token) is selected to represent the input, resulting in a fixed-size vector for each label. These embeddings are stored in a pre-allocated tensor on the same device as the model, ensuring computational efficiency during processing.

This approach leverages both the strengths of RotatE in learning relational embeddings and the capacity of Llama embeddings for representing complex label information, enhanced by additional knowledge from Wikidata entries in the RLM-A variant.

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 serves as a valuable testbed for initial experiments. However, Hetionet may present a worst-case scenario for LLM embeddings, as many compound names are rare in natural language datasets. Additionally, the augmentation data from Wikidata is often highly repetitive and consists mostly of basic factual information.

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.

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

Using RotatE model, predicted *treats* relationships for L-Asparagine were visualized (Figure 3) 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 2, 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

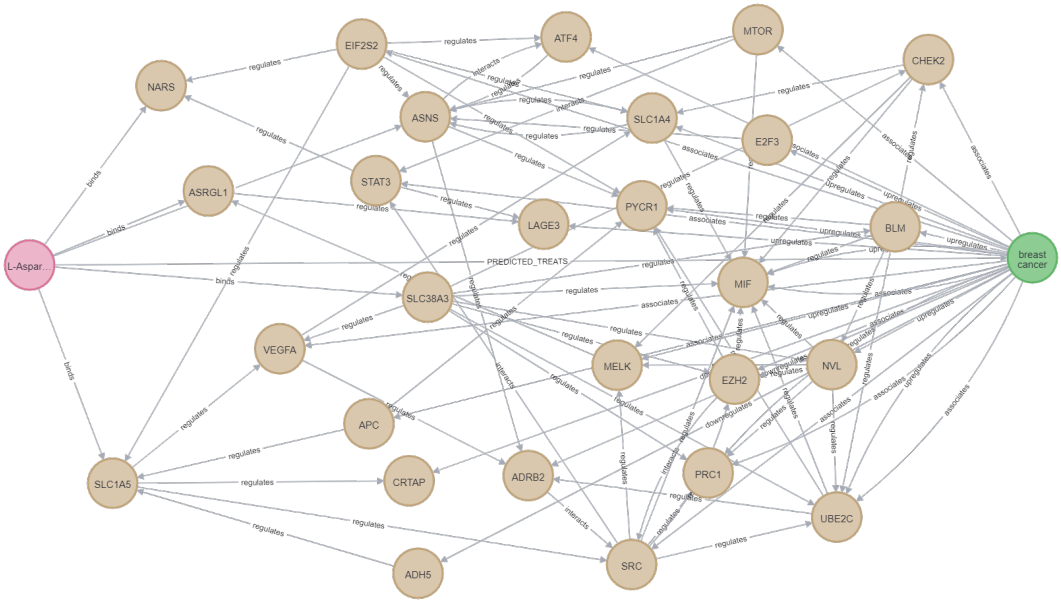


Figure 2: Visualization of connections between L-Asparagine, genes, and breast cancer, highlighting predicted relationships.

Analysis

In general, the results indicate poor performance across all tested models for knowledge graph completion on the Hetionet dataset. 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. Our models RLM and RLM-A did not perform very well (below

Figure 3: 5 predicted *treats* relationships for L-Asparagine.

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	**1199.26**	0.0573
RotatE	**0.0304**	**0.0650**	**0.0861**	**0.1254**	1304.77	**0.0643**
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

RotatE standalone). Both Hits@10 are around 0.11, suggesting they capture relational patterns but may lack rotational expressiveness. Their mean rank is lower than RotatE, suggesting the embeddings could help, but the RAG does not seem to help further in this case. Training losses can be seen in Figure 4 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

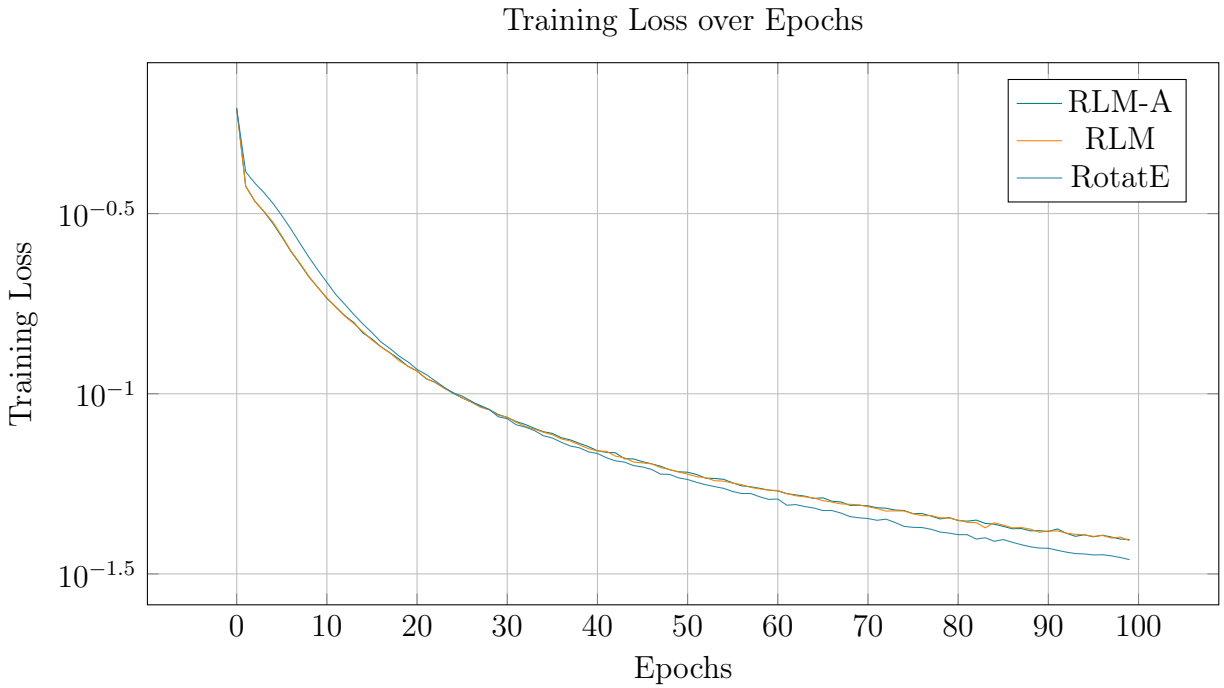


Figure 4: Logarithmic plot of training loss.

(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., 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.