	<div>January, 2025</div> <div>APM_5AI29_TP</div> <div>Language Models and Structured Data</div> <div>Mid-term Project Report</div>
---	--

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.

## Knowledge Graph Embeddings for Link Prediction

In our experiments, we focused on two primary approaches for knowledge graph completion: traditional knowledge graph embedding models and LLM-based embeddings (detail in Section Model Evaluation). Among knowledge graph embedding models available in PyKEEN, in this first iteration we used several families of models starting from the simplest approaches.

- **Translational Models (Distance-based):** TransE [Bordes et al., 2013], TransH [Wang et al., 2014], TransR [Lin et al., 2015], and TransD [Ji et al., 2015]. This family of models embeds entities and relations in the same space, with the distance between head and tail entities reflecting the relationship.
- **Rotational Models:** RotatE [Sun et al., 2019] is a rotational model that represents relations as rotations in the complex vector space. This model has shown state-of-the-art performance on several benchmark datasets.
- **Semantic Matching Models (Tensor Factorization-based):** DistMult [Yang et al., 2015], RESCAL [Nickel et al., 2011], and TuckER [Balazevic et al., 2019] are semantic matching models that capture complex interactions between entities and relations. These models are more expressive than translational models and have shown competitive performance on various datasets.

## 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 datasets. Central to this process is Neo4j’s support for Cypher, a declarative graph query language that efficiently extracts triples representing relationships between entities. These triples  $(h, r, t)$ —comprising head entities, relationships, and tail entities—form the backbone of the knowledge graph.

After retrieval through Cypher queries (Listing 1), the triples can then be converted into PyKEEN’s `TriplesFactory` format, facilitating integration with the PyKEEN pipeline for link prediction and multi-class relationship classification.

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
```

## 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.

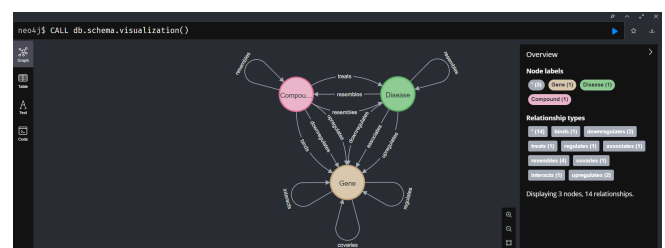
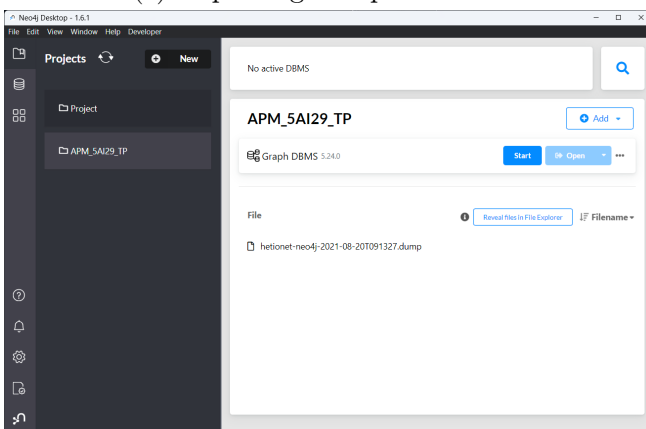
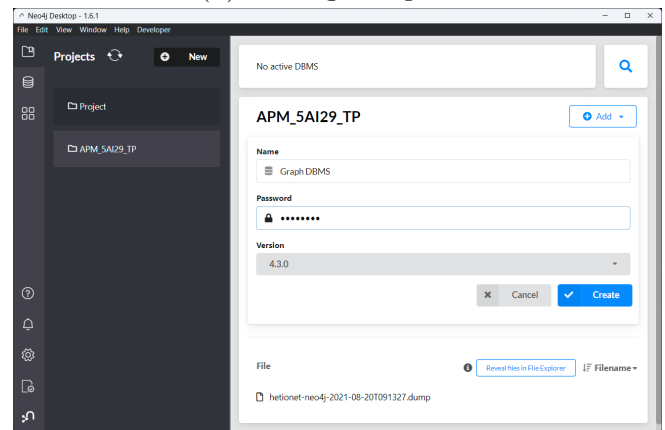
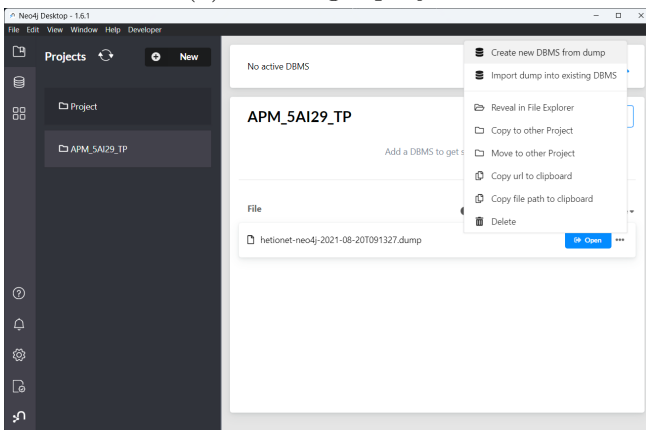
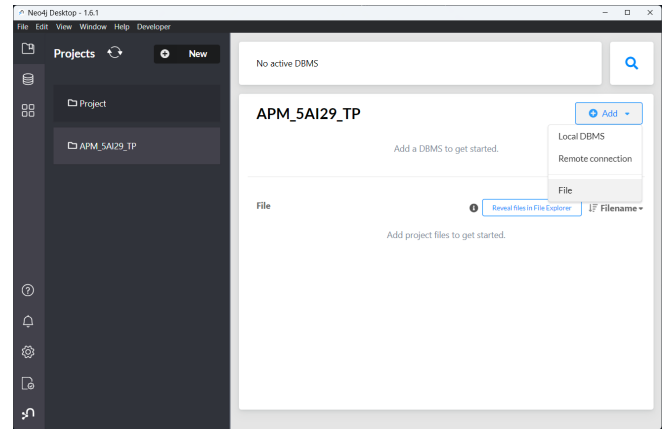
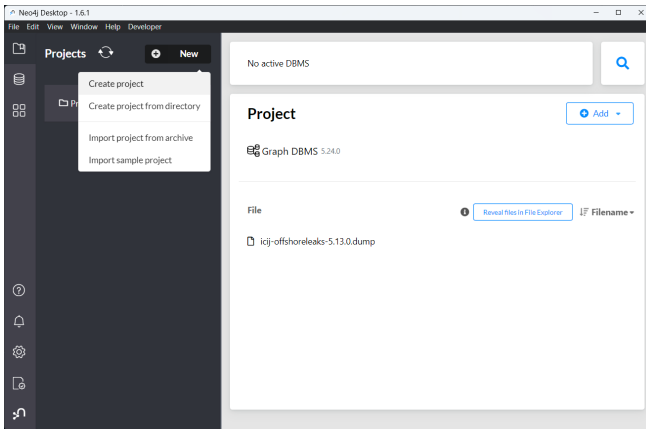


Figure 1: Steps for Neo4j Desktop Setup.

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.

## Model Evaluation

We evaluated the performance of knowledge graph completion models, with a particular focus on comparing 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, referred to as RLM, uses only the Llama embeddings generated from the label names. The second variant incorporates Wikidata’s 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. The evaluation employs standard link prediction metrics, with the results summarized in Table 2, which compares various embedding models available in PyKEEN. The metrics used are as follows:

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

Table 2: Performance Comparison of Models

Model	Hits@1	Hits@3	Hits@5	Hits@10	MR	MRR
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 2) 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



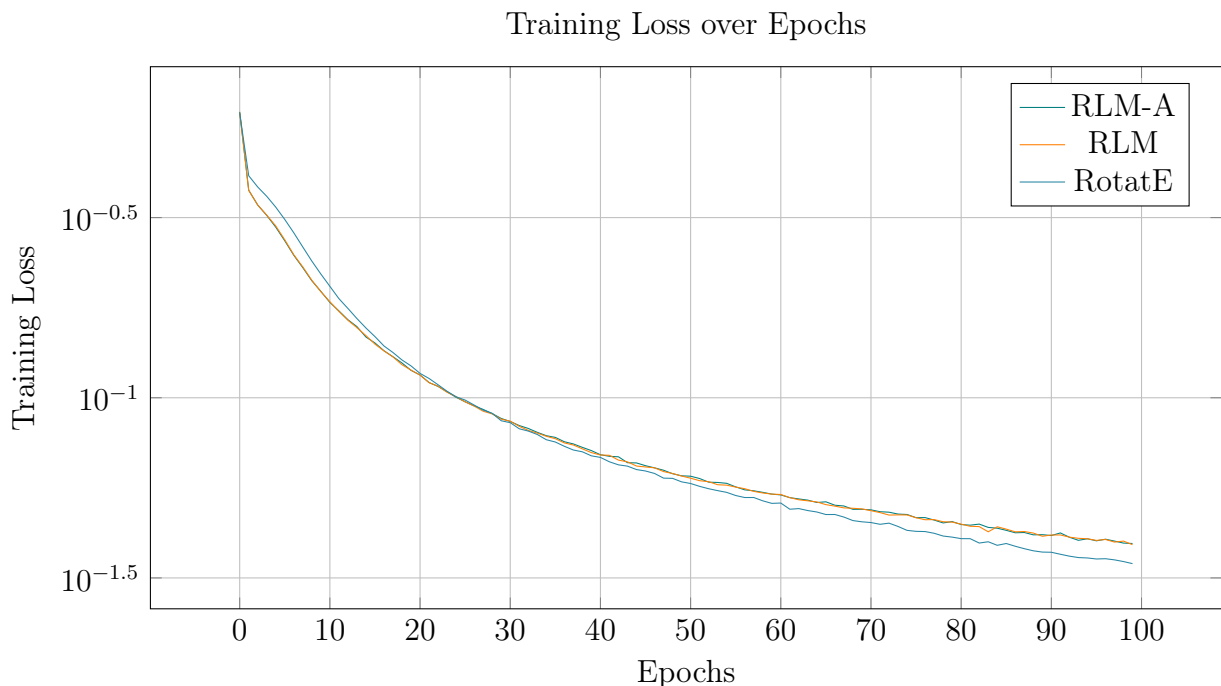


Figure 4: Logarithmic plot of training loss.

Moreover, the semantic matching models (DistMult, RESCAL, and TuckER) also yielded unsatisfactory results. These models emphasize similarity-based scoring through matrix factorization or tensor decomposition, which may fail to capture the structural dependencies present in Hetionet. Notably, RESCAL performed extremely poorly ( $\text{MRR} = 0.0002$ ), likely due to overfitting caused by its high parameter complexity and a lack of regularization. TuckER, which is tensor-based, fared slightly better but still struggled to effectively utilize the dataset’s structural information, possibly due to insufficient embedding dimensions (128).

Our models, RLM and RLM-A, performed below RotatE, with Hits@10 around 0.11, indicating they capture relational patterns but lack the rotational expressiveness of RotatE. Although their mean rank is lower than RotatE, suggesting potential improvements in embedding quality, the addition of retrieval-augmented generation (RAG) did not yield further benefits.

Training losses are shown in Figure 4, with a performance comparison in Figure 2.

## Discussion

To mitigate errors encountered, future work will explore 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 families of models such as rule-based models (non-embedding methods like 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.
- [Balazevic et al., 2019] Balazevic, I., Allen, C., and Hospedales, T. (2019). Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics.
- [Bordes et al., 2013] Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS’13*, page 2787–2795, Red Hook, NY, USA. Curran Associates Inc.
- [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.
- [Ji et al., 2015] Ji, G., He, S., Xu, L., Liu, K., and Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. In Zong, C. and Strube, M., editors, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, Beijing, China. Association for Computational Linguistics.
- [Lin et al., 2015] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI’15*, page 2181–2187. AAAI Press.
- [Neo4j, 2024] Neo4j (2024). Neo4j graph database & analytics | graph database management system.
- [Nickel et al., 2011] Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML’11*, page 809–816, Madison, WI, USA. Omnipress.
- [Sun et al., 2019] Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. (2019). Rotate: Knowledge graph embedding by relational rotation in complex space.
- [Wang et al., 2014] Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI’14*, page 1112–1119. AAAI Press.
- [Yang et al., 2015] Yang, B., tau Yih, W., He, X., Gao, J., and Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases.