Towards an Efficient Knowledge Graph Embedding Harness

Eashan Adhikarla, Chaitanya Roygaga
Department of Computer Science & Engineering
Lehigh University
{eaa418, crr221}@lehigh.edu

May 15, 2022

Abstract

"*A knowledge graph (KG) is a directed heterogeneous multigraph whose node and relation types have domain-specific semantics.*"[1] KGs are an efficient way to collect and categorize information from different entities, events or relationships, to create meaningful connections between them. Information, when communicated in any language, tends to alter based on the context of the situation. They also provide a stable way to create dynamic links between involved entities, based on additions or updations of their definitions.

We propose to do a cluster of experiments for our final project, by re-implementing the models and extending the code for a set of papers which provide efficient ways to create knowledge graph embeddings. The papers are as follows: Translating Embeddings for Modeling Multi-relational Data [3], Knowledge Graph Embedding by Translating on Hyperplanes [11], Embedding Entities and Relations for Learning and Inference in Knowledge Bases [12], Learning Entity and Relation Embeddings for Knowledge Graph Completion [6], Complex Embeddings for Simple Link Prediction[10], SimplE Embedding for Link Prediction in Knowledge Graphs [5].

Specifically, we would like to concentrate on making a harness to accomodate the task of Link Prediction on the following models (corresponding to the mentioned papers) — TransE [3], TransH [11], DistMult [12], TransR [6], ComplEx [10], and SimplE [5].

1 Problem Statement

Most regularly used knowledge graphs include billions of triples. Still, they suffer from the incompleteness problem as many valid triples are missing; finding all valid triples manually is impractical. As a result, knowledge graph completion (or link prediction in knowledge graphs) has recently garnered a lot of interest. Based on known links, link prediction automatically seeks to forecast missing relations between entities [13].

[1] https://towardsdatascience.com/introduction-to-knowledge-graph-embedding-with-dgl-ke-77ace6fb60ef

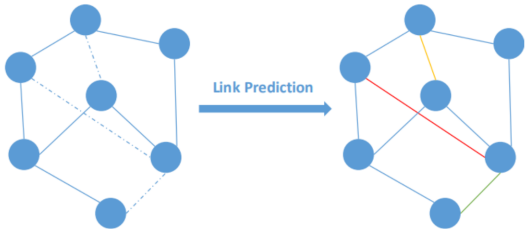


Figure 1: *Link Prediction*

The Knowledge Graph embedding (KGE) task still possess two major problems [8]. First is the problem of 'Non-exclusive relation categories': Relations that belong to more than one of the categories (reflexive, transitive, etc) cannot be embedded easily by a few of these approaches. The second is the problem of 'Zero gradients': As backpropagation is the choice of training method for such models, the incorrect usage of loss functions might generate zero gradients during training, halting updates to relation vectors. These are some of the crucial reasons for choosing to experiment with existing Knowledge Graph models, to understand and analyze the problems through experiments.

Our focus is to study the improvement of the Knowledge Graph Embedding models on the Link Prediction task [Figure 1]. It estimates the probable links between graph nodes, to identify the missing information in a network. A basic idea to solve this problem is to analyze the proximity of the network nodes, which helps in creating a set of highly probable edges between available network entities. In the context of Knowledge Graph Completion, it helps in improving the overall connectivity of the graph between entities, generating missing relations between them.

2 Goal

We aim to re-implement models from scratch, using appropriate metrics for evaluation and compare them based on some variations of hyperparameters. We then analyze the results by creating charts, tables and graphs, to show the differences in performance on the task of Link Prediction with the supporting theoretical reasoning.

3 Methods

3.1 TransE: Translating Embeddings for Modeling Multi-relational Data [3]

Bordes et al. interpret relations between entities in a knowledge graph as translations, considering multi-relational data [having a (head, relation, tail) form] in low dimensional vector spaces. This is the first work of the translation model series, and the motivation was to automatically append additional facts to knowledge graphs without new relation information between entities. The idea is to extract multiple connectivity patterns between the entities, and to generalize them with respect to a particular entity; a specific entity might represent a single type of rela-

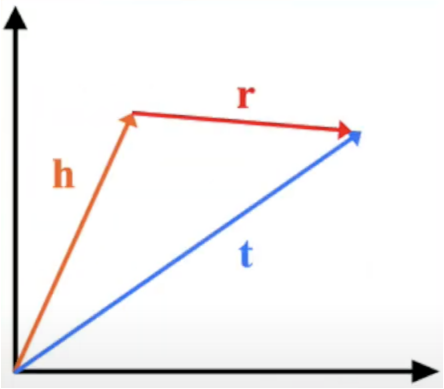


Figure 2: *TransE*

tionship with all of their relation patterns. The model aims to minimize the sum of the head vector and relation vector with respect to the tail vector, with the help of L1 or L2 norm to measure the proximity between them. A reduced set of parameters are required to train the overall model, with the focus on learning a single low-dimensional vector for every entity and relationship.

3.2 TransH: Knowledge Graph Embedding by Translating on Hyperplanes [11]

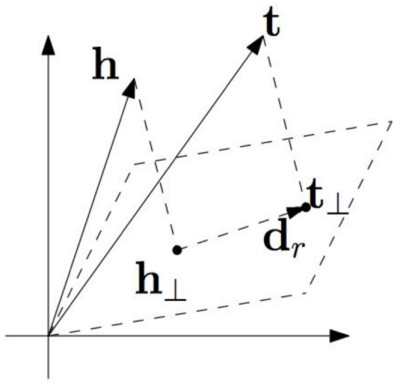


Figure 3: *TransH*

Wang et al. extend the idea of embedding the knowledge graph into a continuous vector space from Bordes et al. [3], by modeling a relation as a hyperplane instead of a single vector. TransE [3] focused on aggregating relations, but doing that is often difficult over a graph. Hence, aggregation might often result in lower performances for reflexive, one-to-many, many-to-one, and many-to-many relational properties. TransH (translation on Hyperplanes) interprets the same aggregate relation as an operation on a hyperplane, with the help of norm and translation vectors. The hyperplane's norm vector (Wr) and the translation vector (dr) on the hyperplane are both vectors in each relation. The model reduces the error for the translation vector (d_r) connecting the projections of the head and tail entities on the hyperplane, thereby eliminating the flaws of the relational properties mentioned earlier. They also reduce false negative labeling

using the proposed mapping properties of relations. The goal of TransH is to capture the one-to-many/many-to-one/many-to-many relationships while keeping a minimized modeling approach like transE.

3.3 DistMult: Embedding Entities and Relations for Learning and Inference in Knowledge Bases [12]

Yang et al. utilize neural embedding models to learn representations of entities and relations in Knowledge Bases, by comparing embedding models like NTN (Neural Tensor Network) [9] and TransE; this helps to understand the impact of model design choices on the relational learning task, providing explainability over the relational properties being captured during the embedding process. They further show the usability of their framework on mining rules for compositional reasoning.

3.4 TransR: Learning Entity and Relation Embeddings for Knowledge Graph Completion [6]

Lin et al. propose an extended idea to embedding models such as transE and transH, by creating entity and relation embeddings in different spaces. The translation task for a specific relation is done in a different space for an entity, thus preserving an entity's multiple aspects. Entities could be similar or dissimilar to one another based on their relation aspects we choose for comparison, which also creates a proximity graph between entities variable in the relation space. A relation 'location-location-contains' could reflect aspects such as country-city or country-country, so separating the entity and relation embeddings based on their variable meaning actually helps in improving the overall representation of the links between the entities. TransR represents entities and relations in the entity and relation spaces (relation-specific entity spaces), and then conducts the translation in the relevant relation space, hence the term TransR.

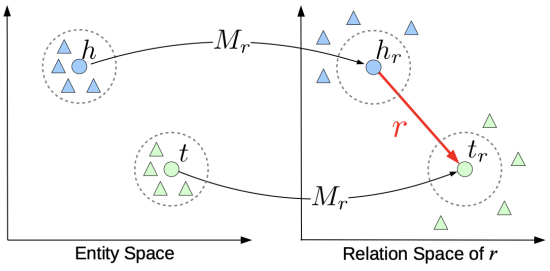


Figure 4: *TransR*

3.5 ComplEx: Complex Embeddings for Simple Link Prediction[10]

Trouillon et al. use latent factorization, with additional complex valued embeddings to handle a larger variety of binary relations in knowledge graphs, specifically the symmetric and asymmetric relations. With respect to the Neural Tensor Network [9] (who also use complex

Table 1: Hyperparameters used for different models during training.

Model	Dataset	Seed	Learning Rate	Embedding Size	No. of Batches	Epochs	Margin	Filter	Optimizer
TransH	FB15K	0	0.001	100	100	1000	1	True	Adam
TransH	WN18	0	0.001	100	100	1000	1	True	Adam

embeddings), they use a much simpler Hermitian dot product approach. A Hermitian dot product involves the conjugate-transpose of one of the two vectors, which creates asymmetry, allowing variable scores for asymmetric relational facts. They present advantages for both scalability to large datasets, and linearity in space and time.

3.6 SimplE: SimplE Embedding for Link Prediction in Knowledge Graphs [5]

Kazemi et al. also use the tensor factorization approach for link prediction. Tensor factorization approaches generally consider embeddings for every individual entity and relation to predict the probability of the relation between them. An older Canonical Polyadic decomposition approach [4] focused on learning one embedding vector for each relation, but two embedding vectors for each entity (one for head, and one for its tail implementation). SimplE (Simple Embedding) improves on this to create a dependent mode of training — the entity relations are learned based on the combined head and tail embeddings of an entity. They even show a reduction in computations when compared to a similar factorization approach in ComplEx.

4 Datasets

Evaluations of models are conducted based on the type of experiment (Link prediction, Triplet classification, Relational fact extraction), with the help of data collected from 2 knowledge graphs: WordNet [7] and Freebase [1].

1. **WordNet-based datasets**: Provide collection of words (synsets) corresponding to a singular word sense, along with lexical relations between them. Nouns, verbs, adjectives, and adverbs are split into groups of synsets, also known as cognitive synonyms, in this enormous lexical database of the English language, with each reflecting a separate contextual notion. Then, through conceptual-semantic and linguistic links, synsets are connected together.

   We are using **WN18** from the WordNet data collection [2].

   (a) No. of relations: 18

   (b) No. of triplets: 141442

   (c) No. of entities: 40943

2. **Freebase-based datasets**: Provide facts in the form of triplets. We are using **FB15k** from the Freebase data collection [2].

Table 2: Experiment results for different models [Hits@10 scores].

Model	Dataset	Test Set	One-to-One Head/Tail	One-to-Many Head/Tail	Many-to-One Head/Tail	Many-to-Many Head/Tail
TransE	FB15K	0.65	0.66/0.64	0.81/0.52	0.50/0.83	0.63/0.66
TransE	WN18	0.85	0.83/0.81	0.91/0.81	0.81/0.91	0.83/0.84
TransR	FB15K	0.54	0.51/0.51	0.69/0.37	0.41/0.73	0.52/0.55
TransR	WN18	0.63	0.57/0.62	0.72/0.54	0.55/0.73	0.64/0.62
TransH	FB15K	0.65	0.67/0.64	0.81/0.50	0.50/0.82	0.63/0.65
TransH	WN18	0.85	0.86/0.79	0.91/0.80	0.80/0.91	0.82/0.84
TransH (1000)	FB15K	0.68	0.67/0.68	0.84/0.56	0.55/0.84	0.67/0.69
TransH (1000)	WN18	0.86	0.90/0.83	0.91/0.82	0.82/0.93	0.85/0.85

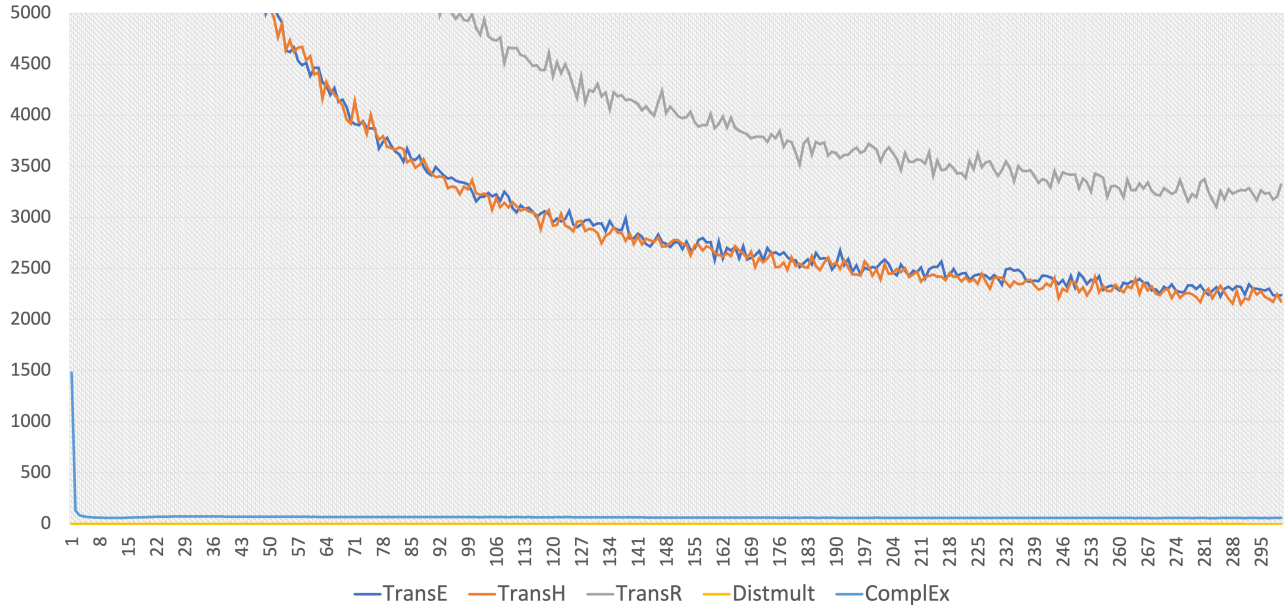


Figure 5: *The graph above represents the* ***training loss*** *for TransE, TransR, TransH, Distmult, ComplEx, SimplE model on FB15k dataset. The x-axis is the total epochs for the link prediction task.*

(a) No. of relations: 1345

(b) No. of entities: 14951

5 Experiments

5.1 Setup

Table 1. provides the entire configuration for the hyper-parameters that were used in training the model. We re-implement the entire code for the 6 models (transE, transH, DistMult, transR, ComplEx, SimplE) using PyTorch.

For training the model, we used a server backed with four GPUs - NVIDIA RTX A5000, CPU with 250GB RAM and 48 CPU cores.

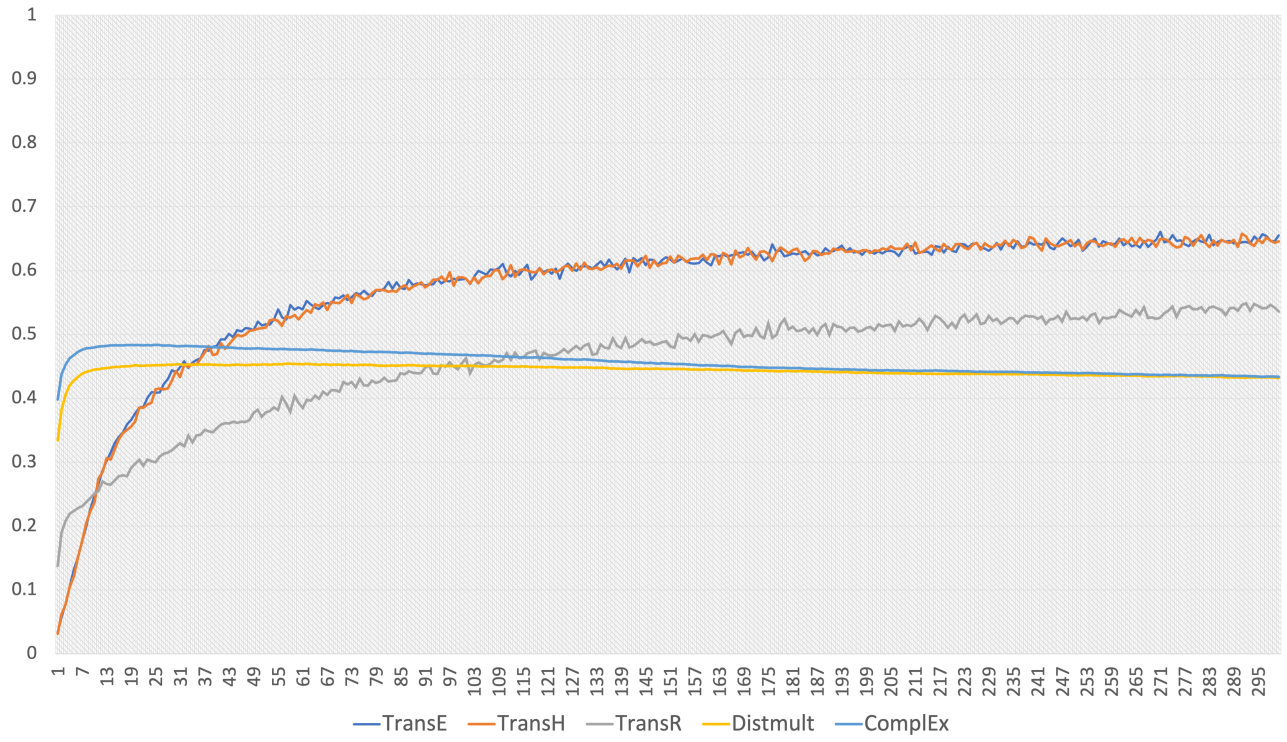


Figure 6: *The graph above represents the* ***Hits@10*** *for TransE, TransR, TransH, Distmult, ComplEx, SimplE model on FB15k dataset. The x-axis is the total epochs for the link prediction task.*

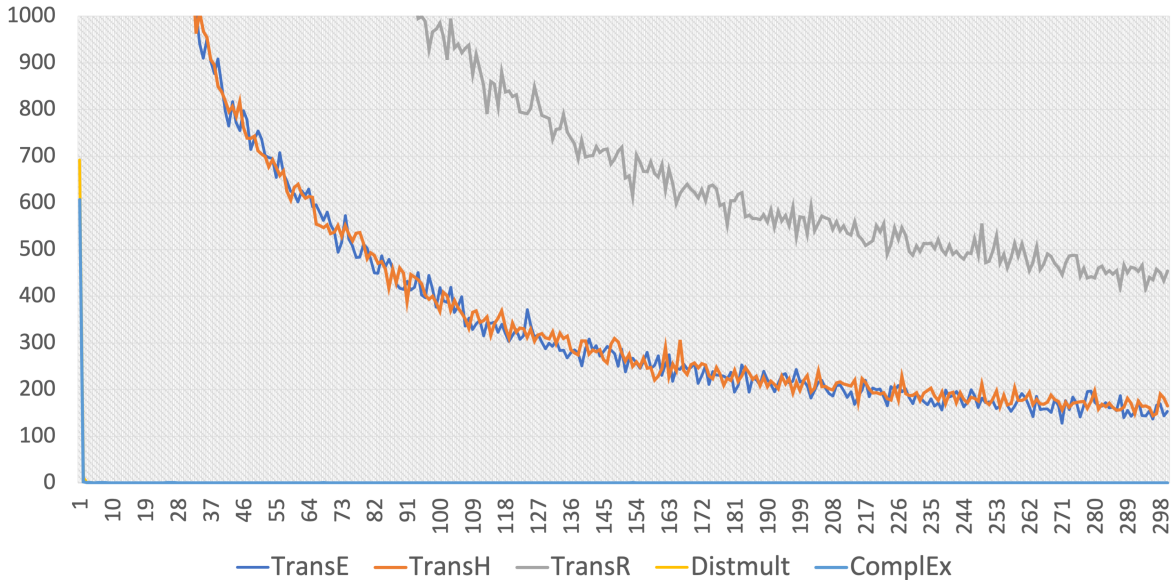


Figure 7: *The graph above represents the* ***training loss*** *for TransE, TransR, TransH, Distmult, ComplEx, SimplE model on WN18 dataset. The x-axis is the total epochs for the link prediction task.*

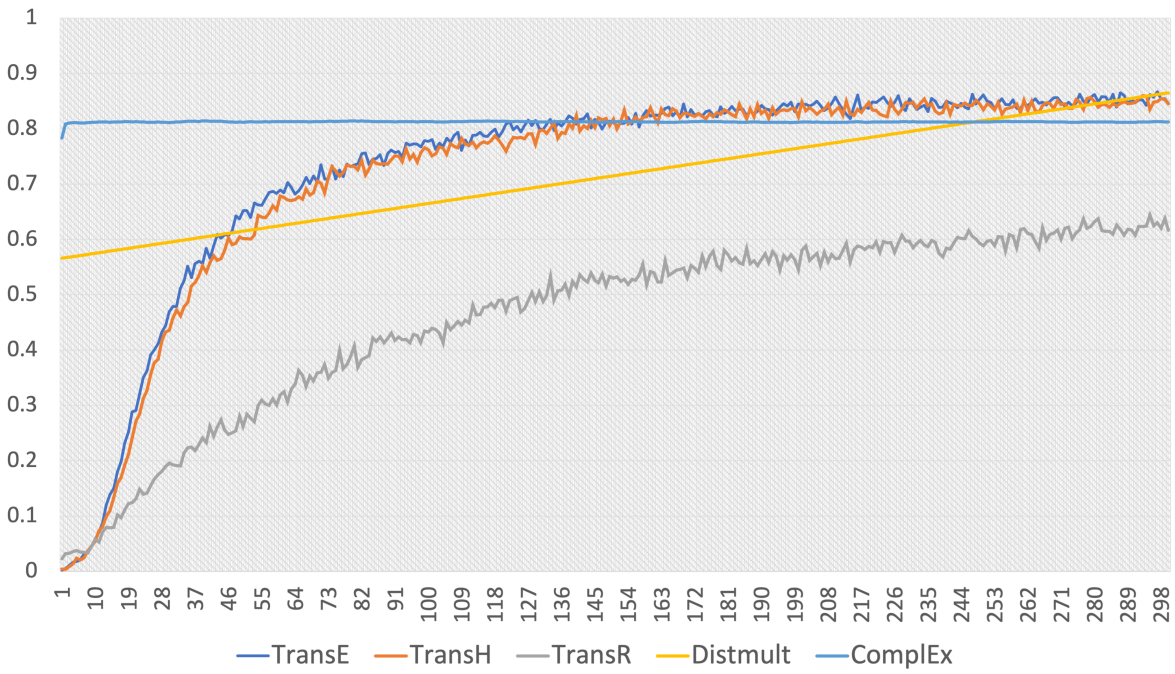


Figure 8: *The graph above represents the* ***Hits@10*** *for TransE, TransR, TransH, Distmult, ComplEx, SimplE model on WN18 dataset. The x-axis is the total epochs for the link prediction task.*

5.2 Comparison of models and Hyperparameter Optimization

We did an initial run on TransE with a set of hyperparameters (resuts not shown in table), and later performed hyperparameter optimization to get an improved perfomance for the transE model presented in Table 1. We also did a comparison for shorter vs longer training on the transH model, to see if performance improved. We see a comparatively similar performance when comparing TransH [300 epochs] with TransH (1000) [1000 epochs]. So, we ended up using a specific set of hyperparameter range from the transE experiments for 300 epochs (to keep results comparable) across all models.

5.2.1 TransE Tuning

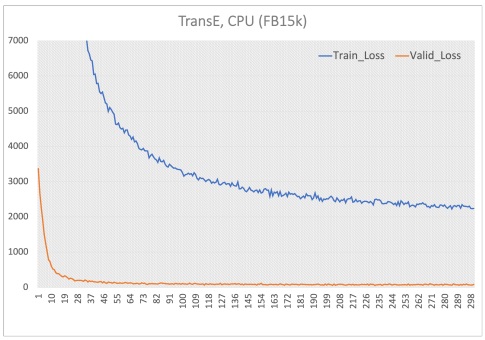
The results are as shown in Figures 9, 10, 11, and 12.

5.2.2 TransH Epoch Comparison

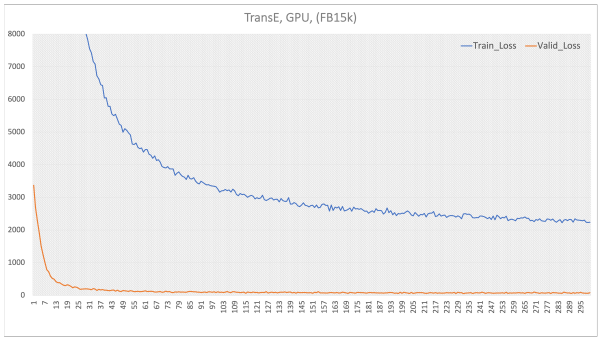
The results are as shown in Figures 13, 14, 15, and 16.

5.3 Results and Analysis

Based on the results shown in Table 3, we see that the SimplE model performs the best for the WN18 and FB15k datasets. The performance of all other models are lower but comparable to each other for the WN18 dataset, while performances for all other models are generally low for the FB15k dataset. The worst performing model on the WN18 dataset was the TransR, which

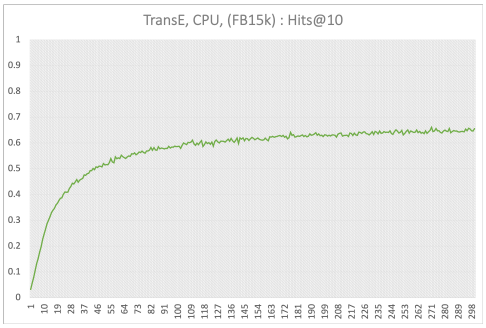


(a) *TransE training loss on FB15k on CPU with different hyperparameters, not performing well.*

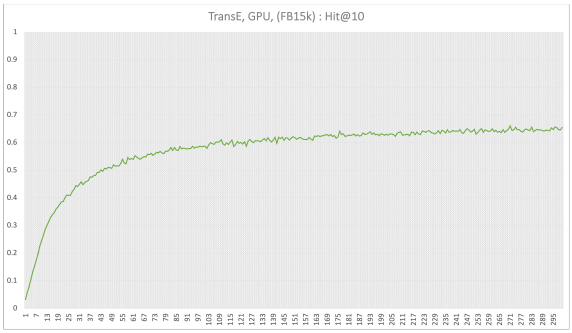


(b) *TransE training loss on FB15k on GPU with different hyperparameters, performing better.*

Figure 9

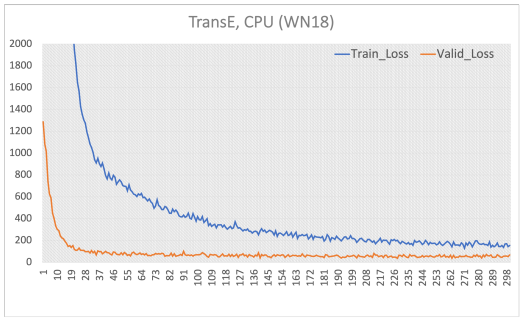


(a) *TransE training Hits@10 on FB15k on CPU with different hyperparameters, not performing well.*

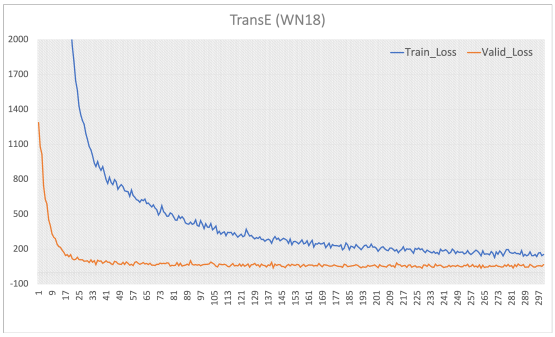


(b) *TransE training Hits@10 on FB15k on GPU with different hyperparameters, performing better.*

Figure 10



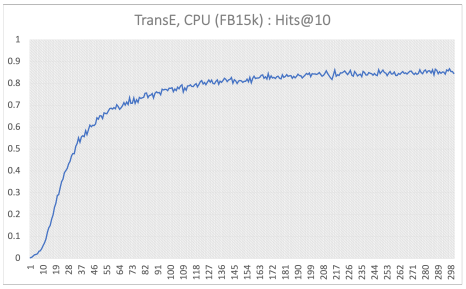
(a) *TransE training loss on WN18 on CPU with different hyperparameters, not performing well.*



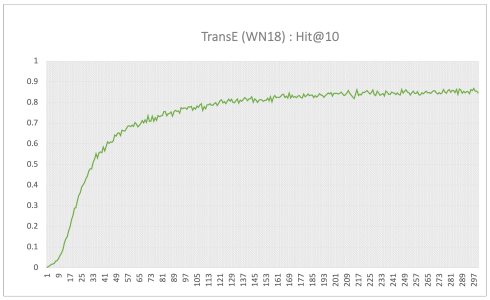
(b) *TransE training loss on WN18 on GPU with different hyperparameters, performing better.*

Figure 11

could be due to us not using pretrained embeddings from the TransE model as a starting point (which was used in [6]). Overall, we see that the tensor factorization based approaches (ComplEx and SimplE) perform better than the translational based approaches (TransE, TransH, and TransR).

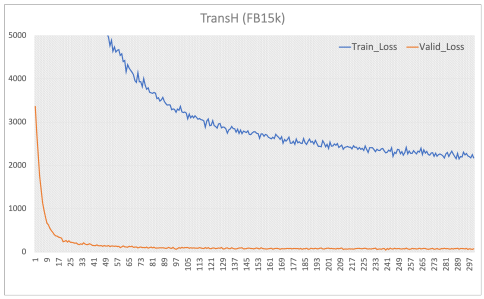


(a) *TransE training Hits@10 on WN18 on CPU with different hyperparameters, not performing well.*

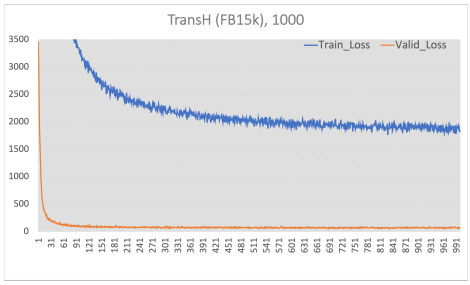


(b) *TransE training Hits@10 on WN18 on GPU with different hyperparameters, performing better.*

Figure 12

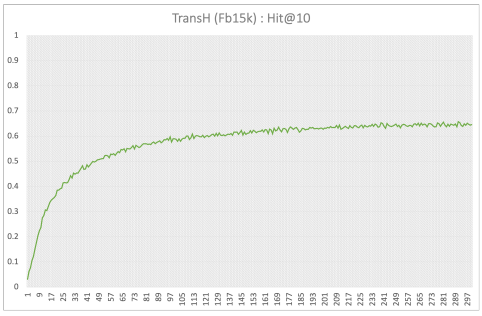


(a) *TransH training loss on FB15k with 300 epochs.*

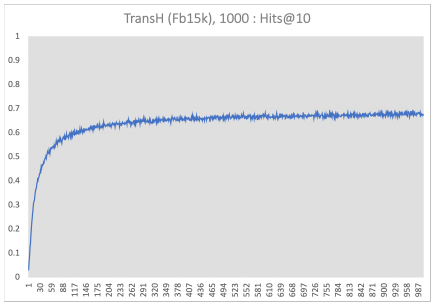


(b) *TransH training loss on FB15k with 1000 epochs.*

Figure 13



(a) *TransH training Hits@10 on FB15k with 300 epochs.*

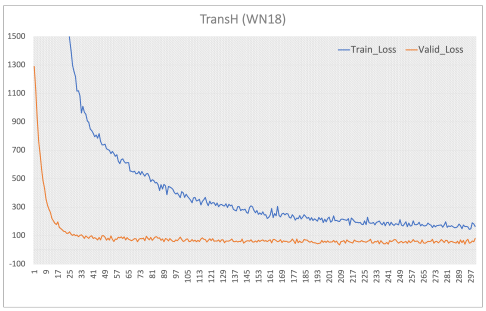


(b) *TransH training Hits@10 on FB15k with 1000 epochs.*

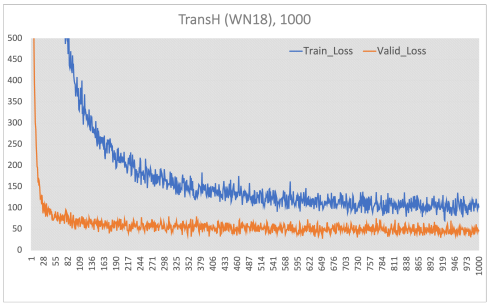
Figure 14

6 Conclusions

We started our initial assessment of Knowledge Graph Embedding models on the simplest but largely cited model of TransE, and then performed hyperparameter optimization on it to see if we could improve the model performance on our selected task of Link Prediction. We figured out a set of hyperparameters for use in other selected models. We also checked if longer training cycles improved performance (TransH), but that was not the case. So, we trained all other

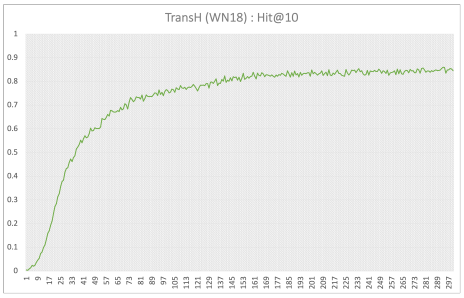


(a) *TransH training loss on WN18 with 300 epochs.*

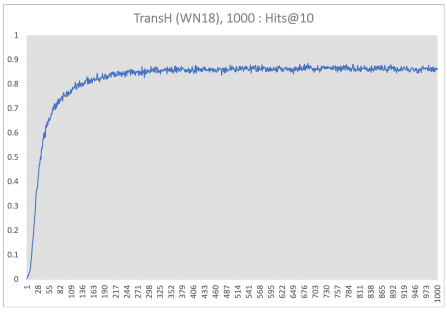


(b) *TransH training loss on WN18 with 1000 epochs.*

Figure 15



(a) *TransH training Hits@10 on WN18 with 300 epochs.*



(b) *TransH training Hits@10 on WN18 with 300 epochs.*

Figure 16

Table 3: Best Metrics on Loss, MRR, Hits@10 for all the different models.

KGEs	Dataset	Loss	MRR	Hits@10
TransE	FB15k	81.39	0.65	0.65
	WN18	67.13	6.44	0.85
TransR	FB15k	132.53	1.21	0.54
	WN18	67.27	7.26	0.63
TransH	FB15k	77.78	0.66	0.65
	WN18	79.78	6.32	0.85
ComplEx	FB15k	60.53	0.24	0.48
	WN18	0.56	0.62	0.81
Distmult	FB15k	80.30	0.22	0.44
	WN18	0.54	0.55	0.81
SimplE	FB15k	-	0.66	0.83
	WN18	-	0.94	0.94

models for 300 epochs with our set of hyperparameters (based on the original implementations of the papers). The results are shown in Table 3.

The whole idea for this experiment set was to analyze the improvements in performance of models for the Link Prediction task, and to create a testing harness for the same, to allow others to do the same.

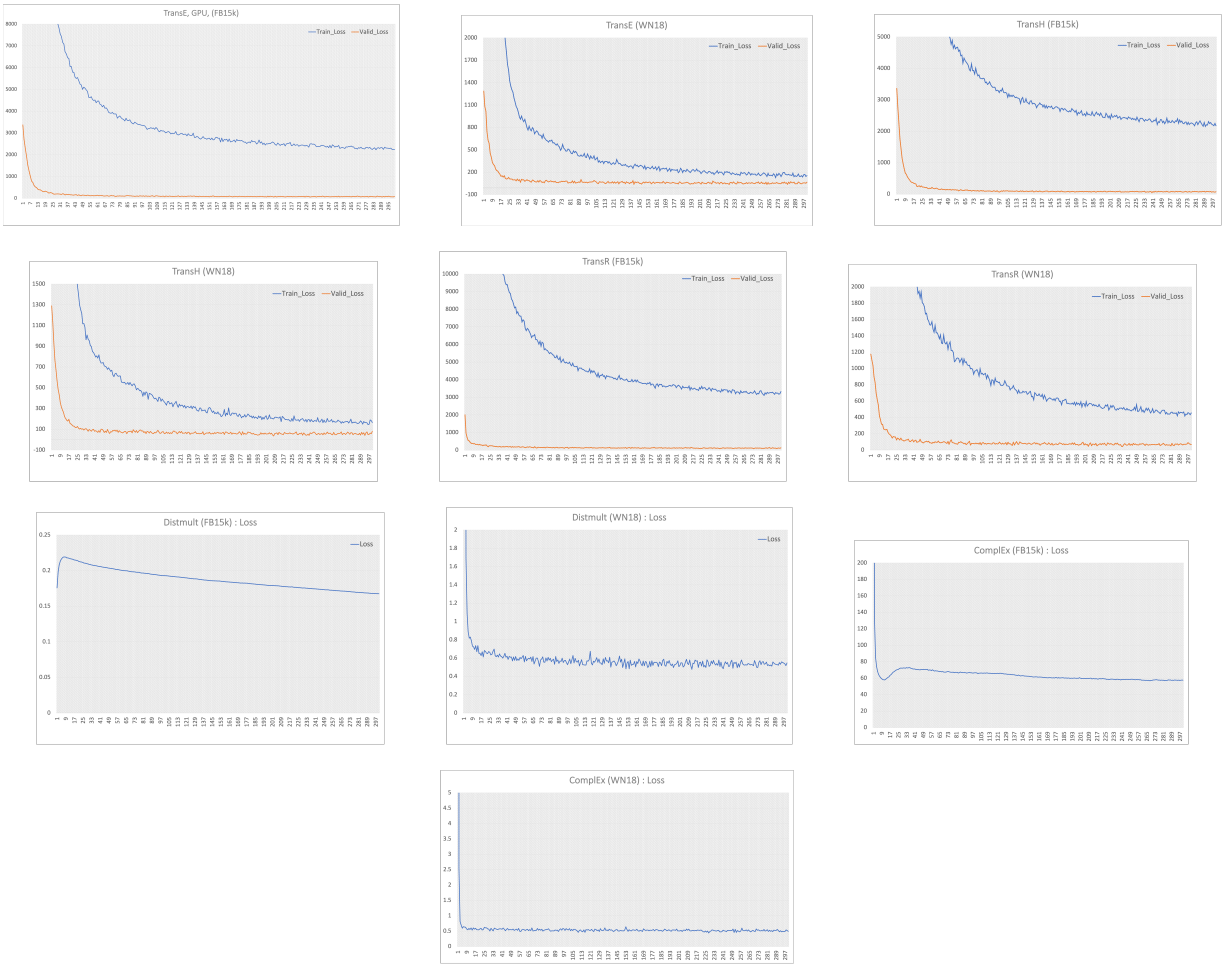
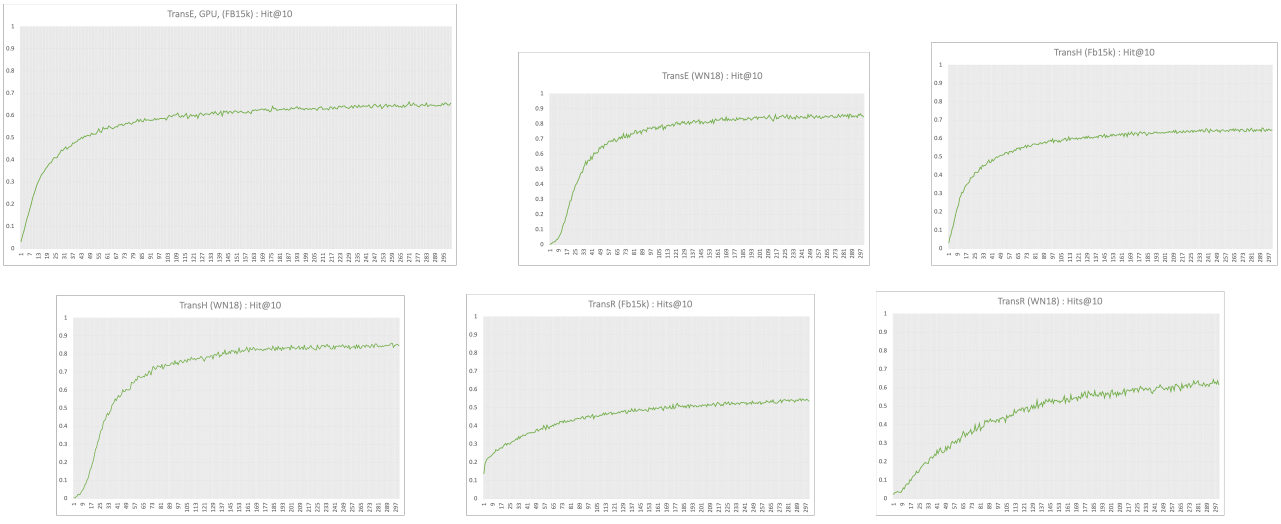


Figure 17: Comparison between the losses of various models during the training cycle



References

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings*

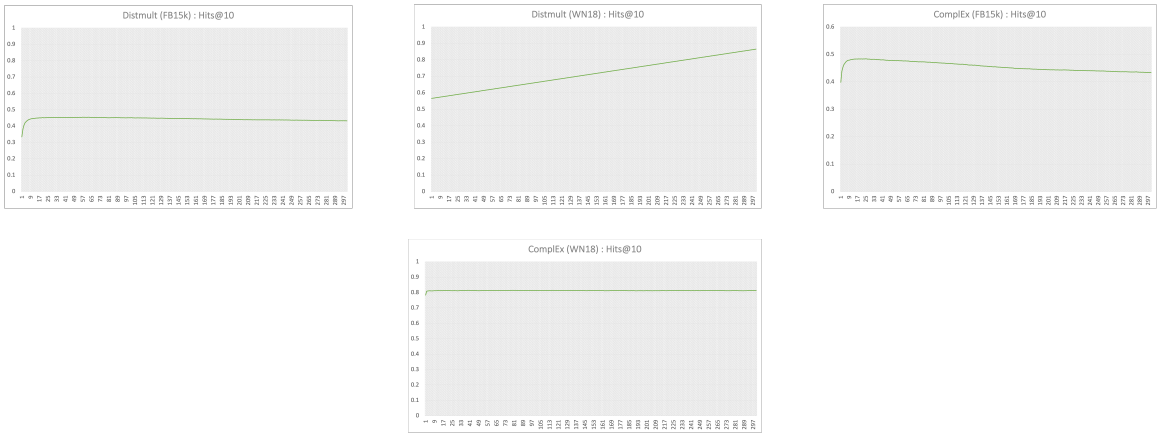


Figure 18: Comparison between the Hits@10 scores of various models during the training cycle

of the 2008 ACM SIGMOD international conference on Management of data, pages 1247–1250, 2008.

[2] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[4] Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[5] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems*, 31, 2018.

[6] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[7] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[8] Nasheen Nur, Noseong Park, Kookjin Lee, Hyunjoong Kang, and Soonhyeon Kwon. Two problems in knowledge graph embedding: Non-exclusive relation categories and zero gradients. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1181–1186. IEEE, 2019.

[9] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26, 2013.

[10] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[11] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[12] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[13] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3065–3072, 2020.