Knowledge Graphs in the Age of Language Models and Neuro-Symbolic AI
A. Salatino et al. (Eds.)

doi:10.3233/SSW240010

TWIG-I: Embedding-Free Link Prediction and Cross-KG Transfer Learning Using a Small Neural Architecture

Jeffrey SARDINA[a,1], Alok DEBNATH[a], John D. KELLEHER[a] and Declan O'SULLIVAN[q]
[a]*Trinity College Dublin*
ORCiD ID: Jeffrey Sardina https://orcid.org/0000-0003-0654-2938, Alok Debnath https://orcid.org/0000-0002-1270-369X, John D. Kelleher https://orcid.org/0000-0001-6462-3248, Declan O'Sullivan https://orcid.org/0000-0003-1090-3548

Abstract. Knowledge Graphs (KGs) are relational knowledge bases that represent facts as a set of labelled nodes and the labelled relations between them. Their machine learning counterpart, Knowledge Graph Embeddings (KGEs), learn to predict new facts based on the data contained in a KG – the so-called link prediction task. To date, almost all forms of link prediction for KGs rely on some form of embedding model, and KGEs hold state-of-the-art status for link prediction. In this paper, we present TWIG-I (Topologically-Weighted Intelligence Generation for Inference), a novel link prediction system that can represent the features of a KG in latent space without using node or edge embeddings. TWIG-I shows mixed performance relative to state-of-the-art KGE models – at times exceeding or falling short of baseline performance. However, unlike KGEs, TWIG-I can be natively used for transfer learning across distinct KGs. We show that using transfer learning with TWIG-I can lead to increases in performance in some cases both over KGE baselines and over TWIG-I models trained without finetuning. While these results are still mixed, TWIG-I clearly demonstrates that structural features are sufficient to solve the link prediction task in the absence of embeddings. Finally, TWIG-I opens up cross-KG transfer learning as a new direction in link prediction research and application.

Keywords. Link Prediction, Knowledge Graph Embeddings, Graph Topology

1. Introduction

Knowledge Graphs (KGs) are knowledge bases that represent data as (s, p, o) triples, where s and o are nodes and p describes the directed, labelled edge between them [1].

This graphical format has allowed KGs to naturally represent a large variety of real-world data, including social networks, computer networks, biological networks, linguistic data, climate data, general knowledge, and much more [2,3,4,1,5,6]. Recently, larger

[1]Corresponding Author: Jeffrey Sardina, sardinaj@tcd.ie

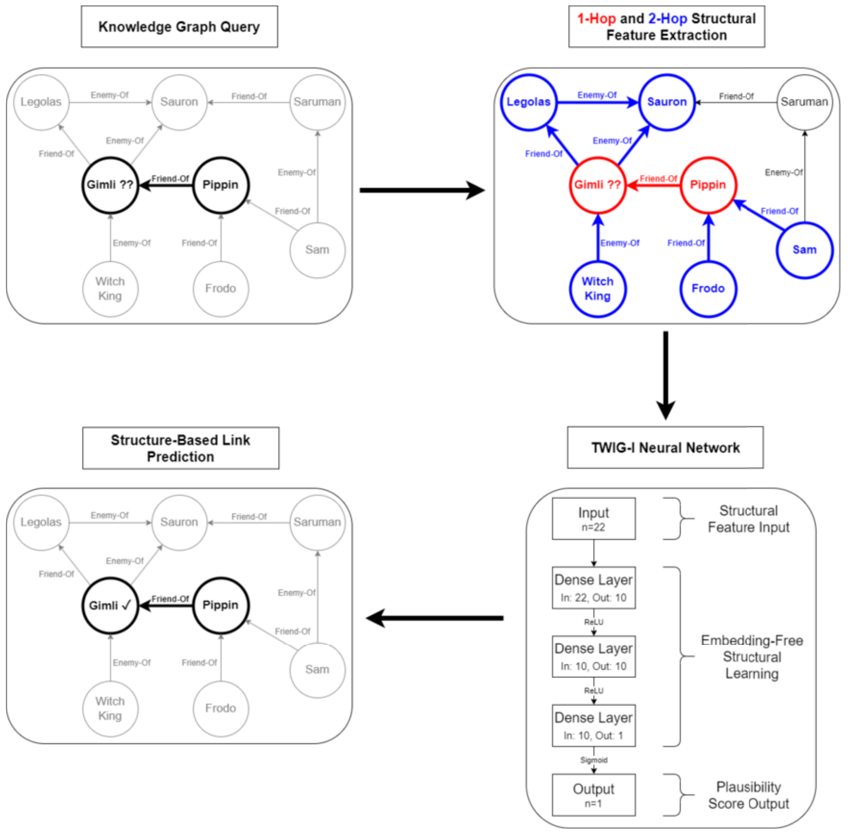

Knowledge Graph Query
1-Hop and 2-Hop Structural Feature Extraction
Structure-Based Link Prediction
TWIG-I Neural Network
Legolas
Sauron
Saruman
Gimli ??
Gimli ✓
Pippin
Sam
Witch King
Frodo
Enemy-Of
Friend-Of
Input n=22
Dense Layer In: 22, Out: 10
ReLU
Dense Layer In: 10, Out: 10
Dense Layer In: 10, Out: 1
Sigmoid
Output n=1
Structural Feature Input
Embedding-Free Structural Learning
Plausibility Score Output

Figure 1. A pictorial overview of the TWIG-I structure-based link predictor.

and larger KGs have been published, leading to substantial growth in big graph data and KGs that can contain millions or even hundreds of millions of triples [7,8,9,10].

The result of the increasing size and scale of KGs is a need for improved learning systems on these graphs. As a response to this, Knowledge Graph Embeddings (KGEs) have been developed to model the latent properties and meaning contained in KGs. KGEs learn to represent every node and every edge in a KG as a unique embedded vector. These embeddings are then used to predict new statements that belong to the graph to facilitate searching and exploring its data [1,11,12]. This is called the “Link Prediction task”, and KGEs currently are considered state-of-the-art in link prediction [13,14,15].

Previous work has shown that KGEs, despite their success, have limitations – that node/edge embeddings are learned largely from a very short, often 1-hop, range [16, 17,18,19], and that the output of KGEs can be understood and replicated using only the structure of the KG with no accounting for its internal semantics or meaning [16]. Further, all KGE systems known to the authors are made to predict statements on only one graph at a time, and therefore by design do not support transfer learning [13,14,15].

In response to this, recent literature has proposed a structure-based, rather than embedding-based, approach to the link prediction task [16]. In particular, the Topologically-Weighted Intelligence Generation (TWIG) model works by using a set of 1-hop and 2-hop structural features around a triple to predict the rank a KGE system will assign to it –

a task it can achieve with high accuracy [16]. The authors used this evidence to propose that KG learning and link prediction may be possible using only structural features [16].

In this paper, we follow up on this research and demonstrate that structure-only, embedding-free KG learning is a viable direction for link prediction. Since our structure-based learning model is based on a fixed set of structural features [16], we are able to natively and directly apply TWIG-I in the transfer-learning setting across different KGs. Such transfer learning is something that state-of-the-art KGE systems cannot do. Using TWIG-I in transfer learning can (but does not universally) improve performance over KGE baselines and over TWIG-I models trained without transfer learning. All code and data are available here: `https://github.com/Jeffrey-Sardina/TWIG-I-v1.0`.

2. Preliminaries

2.1. Learning Knowledge Graphs

Most learning on KGs is done using Knowledge Graph Embeddings [1,11,12,20,14]. KGEs assign vector embeddings to each node and relationship such that new, likely-true triples can be predicted using the mathematical properties of the embeddings themselves; this is called the "link prediction task" [1,11,12,20,14]. KGE models are composed of three core components: the scoring function, the negative sampler, and the loss function.

The scoring function, denoted $f(e_s, e_p, e_o) \rightarrow R$ is a function that takes as input the embeddings of a triple (i.e. of its subject, predicate, and object) and converts them into a scalar-valued score. Higher scores indicate that the input triple is considered more plausible [11,12,14]. The negative sampler is used during training to provide "negative" or "corrupted" triples in the form (s', p, o) or (s, p, o'), where s' and o' are sampled entities from the Knowledge Graph [11,12,14]. The scores of all positive and negative triples are used as inputs to the loss function, which aims to reward the KGE system when true triples have higher scores than negative triples, and to penalise it if not [11,12,14].

This standard KGE pipeline has reached state-of-the-art results, most notably in the KGE model ComplEx [13]. The DistMult and TransE models, based on dot products and vector translation for scoring respectively, are common baselines [11,12] but (especially in the case of TransE) tend to lag behind the performance of ComplEx [13,21,15].

Other methods for learning on KGs exist; most notably, there are Graph Neural Networks (GNNs) [22,23,24,25,26], logic- / rule- based methods [27,28], and sequence-based approaches [29,30]. Of these methods, only logic-based methods can operate in the embedding-free manner as they produce predictions by using inductive learning to construct logical rules that apply to entity and relationship types in general. Overall, however, the most commonly used, most studied, and highest-performance techniques for learning KGs tend to come from the KGE family [14,13,15,1].

2.2. Motivation for TWIG-I

Existing literature on KGEs (and most other graph learning methods) use a form of learned embedding to represent the nodes and edges of a KG. This means that even nodes with very low information content, of which there are typically many, are assigned the same number of latent features as high-information-context nodes [31,32].

In other words, a huge amount of the parameter space of KGE models is filled with low-information-content / noisy data, which can lead to lowered performance [31,32].

Existing literature on KGEs show that the output of KGEs can be predicted with high fidelity via simulation of the KGE process in which the only input data is localised KG structure [16]. This approach, called Topologically-Weighted Intelligence Generation (TWIG), suggests that it is possible to replace attempts at embedding-based learning of KGs with structural learning of KGs (based on TWIG-like neural network). Other work on KGE poisoning attacks [17,18] and structure-based KG analysis of KGs and KGEs [32,31,19] provide theoretical and empirical evidence for the success of such a structure-based approach – they all document (from different perspectives) the fact that KGEs learn to embed nodes and edges based almost exclusively on the properties of neighbour nodes in a 1- to 2- hop distance around each node and edge. Other related works suggest that the optimal choice of some elements of KGE models, such as the negative sampler and loss function, are functions of KG structure [33,34,35,36].

Taken together, this suggests that applying what is called the "twiggy methodology" [16] to link prediction itself, rather than simulation of KGEs, could result in a significantly more powerful link predictor with only a tiny fraction of the parameter cost.

Finally, to the knowledge of the authors, no existing KGE models can be fine-tuned to predict links on new KGs not seen during pre-training (the so-called "transfer learning" setting); existing models assume that link prediction will be done on only one KG at a time [11,20,14,37,13]. We highlight that this limitation comes from the creation of embeddings for the nodes and edges of a specific graph, which cannot be readily transferred to another KG. Since TWIG-I takes a fixed set of structural features as input, we highlight that TWIG-I natively supports transfer learning across diverse KGs.

3. Methodology

In this section, we explain in detail how TWIG-I operates. We first examine how we selected structural features from a KG that it can use to learn, and then document its neural architecture and learning system in full. Finally, we explain the KG datasets and baselines we used to evaluate TWIG-I.

3.1. *Selection of Structural Features*

The selection of structural features was guided principally by existing research that suggests that a 1- to 2- hop range around a triple is most important for how it is learned [31,19,17,18,16]. Specifically, we adopt the same structural features used by the original TWIG paper [16]. For reference, the canonical TWIG features for every triple in the KG come in two main types – "fine-grained" features at the 1-hop range (the triple in question) and "coarse-grained" features at the 2-hop range (all triples connecting to the triple in question). Fine-grained features focus on the properties of the triple elements themselves – such as the degree of the subject and object nodes in the triple. Coarse-grained features collect aggregate statistics of the triples surrounding the triple in question – for example, the degrees of the highest- and lowest- degree nodes connected to the triple.

There are a total of 6 fine-grained features and 16 coarse-grained features, for a total of 22 features. A full enumeration of all features and what they represent, delimited by

Table 1. A summary of all input features used, and their definitions, in TWIG-I. All features values are calculated using the structure of the training set only.

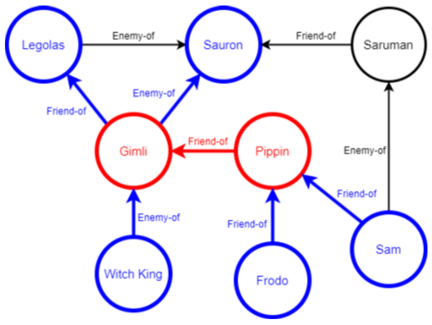
Feature	**Meaning**
fine-grained fts	
s_deg	The degree of the subject node
o_deg	The degree of the object node
p_freq	The frequency of the predicate
s-p cofreq	The number of times the given subject and predicate co-occur
o-p cofreq	The number of times the given object and predicate co-occur
s-o cofreq	The number of times the given subject and object co-occur
coarse-grained fts	
s min deg neighbour	The degree of the lowest-degree neighbour of the subject
s max deg neighbour	The degree of the highest-degree neighbour of the subject
s mean deg neighbour	The degree of the mean-degree neighbour of the subject
o min deg neighbour	The degree of the lowest-degree neighbour of the object
o max deg neighbour	The degree of the highest-degree neighbour of the object
o mean deg neighbour	The degree of the mean-degree neighbour of the object
s num neighbours	The total number of neighbours the subject has
o num neighbours	The total number of neighbours the object has
s min freq edge	The frequency of the least-frequent edge linked to the subject
s max freq edge	The frequency of the most-frequent edge linked to the subject
s mean freq edge	The mean frequency of edges linked to the subject
o min freq edge	The frequency of the least-frequent edge linked to the object
o max freq edge	The frequency of the most-frequent edge linked to the object
o mean freq edge	The mean frequency of edges linked to the object
s num edges	The total number of edges incident on the subject
o num edges	The total number of edges incident on the object

type, is given in Table 1. Figure 2a gives a pictorial overview of the regions of a graph that are samples for fine-grained (highlighted in red) or coarse-grained (highlighted in blue) features, as well as regions of the graph (in black) that are not sampled for features.

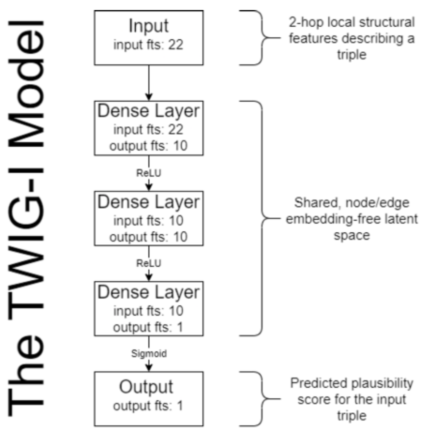
3.2. *The TWIG-I Model*

Neural Architecture. TWIG-I is a simple neural network with three dense layers, structured as shown in Figure 2b. Input data is a feature vector representing a triple, with 22 structural features as noted in the previous section. All input data is z-score normalised before being input to TWIG. In the TWIG neural network, this input data is first passed through the two dense layers of constant size. The third and final dense layer outputs a single scalar value that is treated as a plausibility score. This plausibility score is finally passed through a sigmoid layer so that all scores lie on the range $[0,1]$.

Negative Sampling. When training TWIG-I, we follow the standard KGE practice of negative sampling, which involves creating synthetic, likely-false triples as counterexamples during training by randomly replacing the subject or the object of the triple with another node from the graph [11,12,14]. This process can be concisely phrased as the following: for every triple (s,p,o), generate a set of triples in the form (s',p,o) and (s,p,o') where s' and o' are randomly chosen entities from within the KG. When scoring negative triples, their feature vectors are calculated exactly as is done for true triples.



(a) Sample depiction of the structural features used by TWIG-I. Elements shown in red are are a 1-hop distance from the triple; those shown in blue are at a 2-hop distance from the triple; all other elements are unused.



(b) Depiction of the TWIG-I link prediction model.

Figure 2. An overview of TWIG-I system, including a input data (left) and neural architecture (right).

Loss Function and Learning. Every time we evaluate TWIG-I on a batch, we generate negatives for all triples in the batch. All of these triples (positive and negative) are then scored using the TWIG neural network. With this set of scores of true triples and of negative triples, we compute a loss value via the Margin Ranking Loss (sometimes called "Pairwise Hinge Loss") loss function, a loss function that has shown considerable success with KGEs on the link prediction task [34,14,35].

The Margin Ranking Loss (MRL) loss function is defined as follows:

$$MRL = \max((score\_neg - score\_pos + margin), 0)$$

where $score\_neg$ is the plausibility score of a negative triple output by TWIG-I, $score\_pos$ is the plausibility score of the positive triple from which the negative was generated, and *margin* is the hyperparameter that defines what margin the model should attempt to enforce between the scores of positive and negative triples. For all of our experiments, we use a *margin* value of 0.1. The minimisation of this loss requires that TWIG-I learn to assign higher scores to true triples, and lower scores to negatives.

3.3. Evaluation Protocol and Metrics

For evaluation, we used the standard link prediction metric Mean Reciprocal Rank (MRR), which is defined in terms of *ranked output* for a link predictor [14,35]. To produce this ranked output in the evaluation phase, negatives are generated for an input triple by corrupting the subject and object, one at a time, with all the other entities in the graph. Any generated triples that happen to also be true (i.e. those that were in the original KG) are filtered out. All of the remaining negatives, and the original true triples, are passed through TWIG-I to be given plausibility scores. These scores are then sorted, and the position of the true triple among its negatives is referred to as its "rank". 1-indexed ranking is used, which means the best (lowest) possible rank is 1.

Mean Reciprocal Rank calculates, for all triples in the testing set, the arithmetic mean of the reciprocal of the ranks assigned to all true triples. This can be written as:

$$MRR = \frac{1}{n}\left(\sum_{i=1}^{n} \frac{1}{rank_i}\right)$$

where n is the total number of ranks of true triples in the test set. MRR is bounded on $(0,1]$, where higher valued indicate better performance.

3.4. Datasets Used

When evaluating TWIG-I, we use four different benchmark KG datasets – FB15k-237, WN18RR, CoDExSmall, and DBpedia50 [4,38,39]. We choose FB15k-237 and WN18RR because these are the standard benchmark datasets for the link prediction task in KGE and link prediction literature [35,14,4]. We use CoDExSmall and DBpedia50, which are smaller KGs, to evaluate how TWIG-I in the context of more diverse KGs.

A brief overview of these datasets, and their basic structural statistics, is given in Table 2. All datasets, their standard train-test-valid splits, and their statistics on number of triples, nodes, and predicates are obtained from the PyKEEN KGE repository [40].

Table 2. An overview of the 4 benchmark datasets used in this study.

Dataset	#Nodes	#Predicates	#Triples	Reference
FB15k-237	14505	237	310079	Toutanova et al. [4]
WN18RR	40559	11	92583	Toutanova et al. [4]
CoDExSmall	2034	42	36543	Safavi et al. [38]
DBpedia50	24624	351	34421	Shi et al. [39]

3.5. Hyperparameter Search for TWIG-I

For each KG we run TWIG-I on, we do an independent hyperparameter grid search to select optimal hyperparameter values for TWIG-I. For each dataset, each candidate set of hyperparameter values is trained for 20 epochs on the dataset's standard training split. The resultant model is then evaluated on the dataset's validation set in terms of MRR. The grid searched on depends on the size of the dataset in question – for larger datasets (i.e. FB15k-237 and WN18RR) we use a smaller grid since they are more computationally demanding. This hyperparameter grid is given in Table 3. For smaller datasets (CoDExSmall and DBpedia50) we run on a larger hyperparameter grid, given in Table 4.

Table 3. The hyperparameter grid used for small datasets (i.e. CoDExSmall and DBpedia50).

Hyperparameter	Values searched
negatives per positive	30, 100, 500
learning rate	5e-3, 5e-4, 5e-5
batch size	64, 128, 256
margin	0.1 (constant)

Table 4. The hyperparameter grid used for larger datasets (i.e. FB15k-237 and WN18RR).

Hyperparameter	**Values searched**
negatives per positive	30, 100, 500
learning rate	5e-3, 5e-4, 5e-5
batch size	128 (constant)
margin	0.1 (constant)

The optimal hyperparameter combinations obtained from all of the hyperparameter searches for both experiment settings (standard learning and transfer learning) are reported alongside evaluation results in the Results section.

3.6. TWIG-I Evaluation

3.6.1. Evaluation in the Standard Learning Setting

In the so-called "standard" learning setting, we do not use transfer learning, but rather train a full TWIG-I model from scratch. In this case, we use the optimal hyperparameters obtained from the hyperparameter search to create a new TWIG-I model and train it for 100 epochs on FB15k-237 and WN18RR. We then evaluate the model to produce its final MRR score. All of these results are presented in the results section.

3.6.2. Evaluation in the Transfer Learning Setting

To evaluate TWIG-I's ability to transfer learn from one KG to another, we take our pretrained TWIG-I models that were trained on FB15k-237 and WN118RR respectively and finetune both of them on CoDExSmall and DBpedia50. We further finetune our pretrained FB15k-237 model to WN18RR, and our pretrained WN18RR to FB15k-237.

For all finetuning experiments, we repeat a hyperparameter search. This hyperparameter search uses the same hyperparameter grids as before; these are given in Tables 3 and 4. Finetuning is run for 20 epochs on the training set, and then evaluated on the testing set to produce final evaluation metrics. The results of all these experiments are shown in the Results section.

3.7. KGE Baselines

For baselines, we compare to the state-of-the-art ComplEx model [13], as well as DistMult [41] and TransE [42], which together are the literature-standard baselines [11,12,14,13,21,15]. We take optimal model configurations and final KGE performance values on large datasets (FB15k-237 and WN18RR) from Ruffinelli et al.'s KGE benchmarking experiments [35] as performing them locally was computationally infeasible.

For smaller datasets (CoDExSmall and DBpedia50) we run a hyperparameter grid search for 100 epochs and train them on optimal hyperparameters for 1000 epochs before evaluating on the test set. The hyperparameter grid used is given in Table 5. For more information on KGE hyperparameters, see [14].

KGE model implementations from PyKEEN were used for these experiments [40]. In all cases, the literature standard train-test-validation splits are used for all KGs [40].

Table 5. The hyperparameter grid used for KGEs on the smaller KGs (i.e. CoDExSmall and DBpedia50).

Hyperparameter	**Values Searched**
loss function	Margin Ranking, Binary Cross Entropy, Cross Entropy
negative sampler	Basic, Bernoulli, Pseudo-typed
learning rate	$1e$-2, $1e$-4, $1e$-6
regulariser coefficients	$1e$-2, $1e$-4, $1e$-6
negatives per positive	5, 25, 125
margin	0.5, 1, 2
embedding dim	50, 100, 250

3.8. TWIG-I Baselines for Finetuning

Tr to assess the impact of finetuning rather than training a new TWIG-I model from scratch, we further evaluate two TWIG-I baselines. For the first baseline, which we call the "from scratch" baseline, TWIG-I is trained for 20 epochs exactly as the fine-tuned models are. However, it is not initialised from a pre-trained model. As such, this baseline presents how much TWIG-I can learn in a reduced number of epochs without any transfer learning. The second baseline, which we call the "full-training" baseline, is trained for the full 100 epochs we use in the standard learning protocol. This baseline represents how much TWIG-I can learn when it is given a larger computational budget, but is not able to take advantage of any knowledge from a pre-trained TWIG-I model.

4. Results

4.1. Hyperparameter Searches

Hyperparameter search on TWIG-I was done in three principle cases: for training TWIG-I models without fine-tuning, for finetuning TWIG-I onto a new dataset from a pre-trained FB15k-237 TWIG-I model, and for finetuning TWIG-I onto a new dataset from a pretrained WN18RR TWIG-I model. The optimal hyperparameters for training TWIG-I from scratch (without finetuning) are given in Table 6. The optimal hyperparameters for finetuning TWIG-I onto a new dataset using a TWIG-I model pretrained on FB15k-237 are given in Table 7. The optimal hyperparameters for finetuning TWIG-I onto a new dataset using a TWIG-I model pretrained on WN18RR are given in Table 8.

Table 6. Hyperparameters selected for TWIG-I on each dataset for the standard training protocol.

Dataset	**negatives per positive**	**learning rate**	**batch size**	**margin**
FB15k-237	100	5e-4	128	0.1
WN18RR	500	5e-3	128	0.1
CoDExSmall	100	5e-3	64	0.1
DBpedia50	30	5e-3	128	0.1

In the case of KGE models, the hyperparameter search results for CoDExSmall and DBpedia50 on all three KGEs tested are given in Table 9.

Table 7. Hyperparameters for finetuning to each KG from a pretrained FB15k-237 TWIG-I model.

Dataset	**negatives per positive**	**learning rate**	**batch size**	**margin**
WN18RR	500	5e-4	128	0.1
CoDExSmall	100	5e-3	64	0.1
DBpedia50	30	5e-4	128	0.1

Table 8. Hyperparameters for finetuning to each KG from a pretrained WN18RR TWIG-I model.

Dataset	**negatives per positive**	**learning rate**	**batch size**	**margin**
FB15k-237	100	5e-4	128	0.1
CoDExSmall	500	5e-3	128	0.1
DBpedia50	30	5e-4	64	0.1

Table 9. Hyperparameter values selected for each KGE model on each dataset. BCE = Binary Cross Entropy Loss; CE = Cross Entropy Loss; MRL = Margin Ranking Loss; npp = negatives per positive; lr = learning rate; reg coeff = regulariser coefficient; dim = embedding dimension

	Loss Function	**Negative Sampler**	**lr**	**reg coeff**	**npp**	**margin**	**dim**
ComplEx							
CoDExSmall	MRL	Bernoulli	1e-2	1e-2	125	2	100
DBpedia50	BCE	Basic	1e-2	1e-2	25	N/A	100
DistMult							
CoDExSmall	CE	Basic	1e-2	1e-2	125	N/A	250
DBpedia50	CE	Bernoulli	1e-2	1e-2	125	N/A	250
TransE							
CoDExSmall	MRL	Bernoulli	1e-2	1e-6	125	2	50
DBpedia50	CE	Bernoulli	1e-2	1e-2	125	N/A	250

4.2. Standard Evaluation Setting

In order to benchmark TWIG-I against state-of-the-art KGE models in the standard learning setting (i.e. without transfer learning), we first show evaluation results of TWIG-I and all baselines on the standard link prediction benchmark KGs FB15k-237 and WN18RR [4], as well as on the smaller KGs CoDExSmall and DBpedia50 [38,39].

As outlined in our methodology section, results on KGE models (ComplEx, DistMult, and TransE) for FB15k-237 and WN18RR are taken from a previous benchmarking study by Ruffinelli at al. [35]. Results for KGE models on smaller datasets (CoDExSmall and DBpedia50) were run locally on their optimal hyperparameters. For all evaluations here, KGE models were run for 1000 epochs and TWIG-I was run for 100 epochs. Link prediction performance results are shown in Table 10.

We further provide a comparison of the calculated parameter cost of our models versus the calculated parameter cost of KGE models; this can be found in Table 11.

Looking at TWIG-I's predictive performance, we see that TWIG-I has mixed performance relative to KGE baselines. While it can, for example, substantially outperforms KGE-based methods on CoDExSmall and generally matches their performance on DBpedia50, it lags notably behind state-of-the-art on the standard benchmark FB15k-237,

Table 10. MRR performance of KGEs vs TWIG-I in link prediction. The best results are shown in bold.

Dataset	FB15k-237	WN18RR	CoDExSmall	DBpedia50
ComplEx	**0.35**	**0.48**	0.39	0.36
DistMult	0.34	0.45	0.34	**0.39**
TransE	0.31	0.23	0.28	0.31
TWIG-I	0.20	0.06	**0.61**	0.38

Table 11. Sum of all learnable parameters and the number of input features in TWIG-I vs KGE models.

Model	FB15k-237	WN18RR	CoDExSmall	DBpedia50
ComplEx	7,547,904	10,385,920	415,200	4,995,000
DistMult	7,547,904	20,771,840	519,000	6,243,750
TransE	7,547,904	20,771,840	103,800	6,243,750
TWIG-I	11,973,411	3,821,091	1,447,423	1,417,283

and fails to significantly learn on WN18RR. The difference in the predictive performance of TWIG-I on these various KGs is discussed in detail in the Discussion section.

Looking at the parameter and feature cost of TWIG-I versus KGEs, we consider the combined cost of learnable parameters and input features. This means that when counting parameters for TWIG we include not only the learnable weights of its neural network (of which there are only 351), but also the number of features present in all input vectors in the training set. Since in KGEs all vectors are learned, not given, we report for KGE models only the number of learnable parameters, calculated from their embedding dimension and the number of nodes and edges they embed for each KG.

The aggregate parameter-and-feature usage of TWIG-I versus KGE models on each dataset is given in in Table 11. We observe that TWIG-I can lead to either an increase or a decrease in total parameter usage. This effect is very simple – TWIG-I produces input features based on each *triple* in a KG, whereas KGEs produce input features based on each *node and edge* in a KG. This means that in more dense KGs (having more triples relative to the number of nodes/edges), TWIG-I requires more parameters – we see this in FB15k-237 and CoDExSmall. However, in more sparse datasets (such as WN18RR and DBpedia50), TWIG requires fewer parameters than KGE models do.

4.3. *Transfer Learning Setting*

We further present our evaluation of TWIG-I's efficacy in the transfer learning setting – a task that KGEs cannot perform as their learned embeddings are KG-specific. The hyperparameter search results for TWIG-I are shown in Table 7 and Table 8, the hyperparameter results for KGE baseline are shown in Table 9, and the final evaluation results of the transfer learning experiments are given in Table 12.

Looking at DBpedia50, the results indicate that using TWIG-I with transfer learning can allow it to match or beat the performance of KGE baselines. It achieves MRR values of 0.45 and 0.37 on DBpedia50 when pretrained on FB15k-237 and WN18RR respectively. The KGE baselines' performance lies in a similar, but lower, range from 0.31 (with TransE) to 0.39 (with DistMult). Finally, we see that the TWIG-I model trained for the full 100 epochs did not perform differently to the TWIG-I model trained for 20 epochs, suggesting that finetuning enables a boost in predictive performance that cannot be achieved even with increased training time without transfer learning.

Table 12. Evaluation of TWIG-I for transfer learning. All results are MRR values.

Model	**CoDExSmall**	**DBpedia50**	**FB15k-237**	**WN18RR**
ComplEx	0.39	0.36	**0.35**	**0.48**
DistMult	0.34	0.39	0.34	0.45
TransE	0.28	0.31	0.31	0.23
TWIG-I (20 epochs from scratch)	0.52	0.38	0.19	0.03
TWIG-I (100 epochs full-training)	**0.61**	0.38	0.20	0.06
TWIG-I finetune (from FB15k237)	0.44	**0.45**	N/A	0.01
TWIG-I finetune (from WN18RR)	0.60	0.37	0.21	N/A

On CoDExSmall, the performance of the TWIG-I model (pretrained on either FB15k-237 or WN18RR) remains notably stronger than the performance of KGE baselines, reaching MRR values of 0.44 and 0.60, whereas the KGE models' performance lies in between 0.28 and 0.39. Without finetuning, TWIG-I achieves MRR scores of 0.52 and 0.61 in the 20-epoch and 100-epoch settings. This indicates that using finetuning (from a TWIG-I model pretrained on WN18RR) can match the performance of longer-epoch training paradigms on CoDExSmall by using structural knowledge from a different KG.

Similar to our standard evaluation results, TWIG-I lags behind state-of-the-art KGEs on both FB15k-237 (where it has an MRR of 0.21 at best) and WN18RR (where its MRR is near 0). From these results, it is unclear whether finetuning from WN18RR to FB15k-237 leads to a significant change in performance. However, it seems that finetuning from FB15k-237 to WN18RR leads to reduced performance, from 0.06 to 0.01,indicating that using finetuning with TWIG-I does not always lead to increased predictive performance.

The theoretical and practical implications and the limitations of TWIG-I-based transfer learning across KGs are examined further in the Discussion section.

5. Discussion

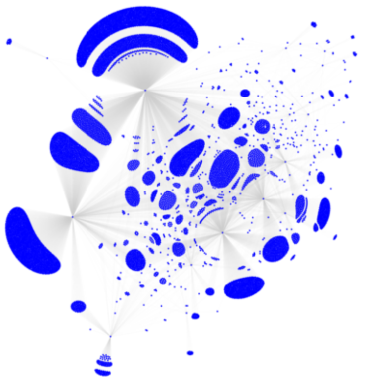
TWIG-I uses a simple neural network to learn the structural features of a knowledge graph. We identify two core insights of the TWIG-I learning paradigm:

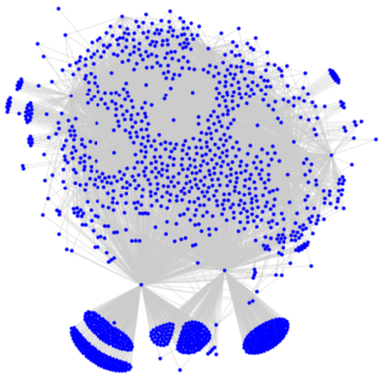
1. TWIG-I allows link prediction to be explicitly correlated with the topological properties of the KG it is learning, and
2. TWIG-I provides a way to achieve transfer learning across knowledge graphs which can lead to increased predictive performance in *some* (but not all) experimental settings.

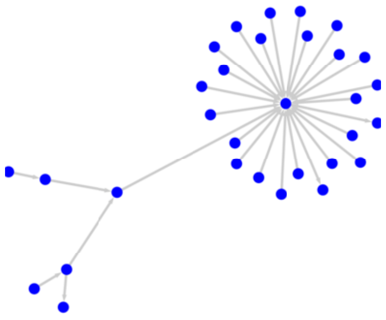
In this section, we discuss these insights in greater detail and show that the link prediction task can be achieved by learning the knowledge graph structure, as opposed to traditional embedding-based KGE approaches.

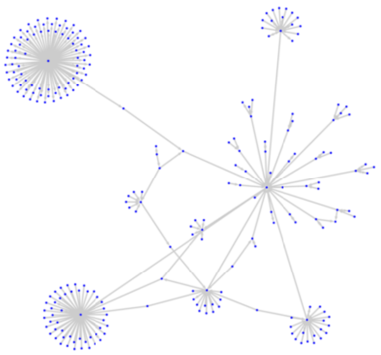
5.1. Learnability, Graph Topology, and Model Performance

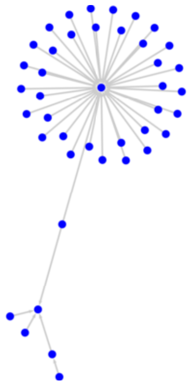
TWIG-I introduces a learning approach leveraging graph structural information, focusing on structural features within a triple and its adjacent triples. In other words, we examine


(a) A subset of FB15k-237, in a 2-hop radius from a median-degree node.


(b) A subset of CoDExSmall, in a 2-hop radius from a median-degree node.


(c) A subset of WN18RR, in a 2-hop radius from a median-degree node.


(d) A subset of WN18RR, in a 3-hop radius from a median-degree node.


(e) A subset of DBpedia50, in a 4-hop radius from a median-degree node.

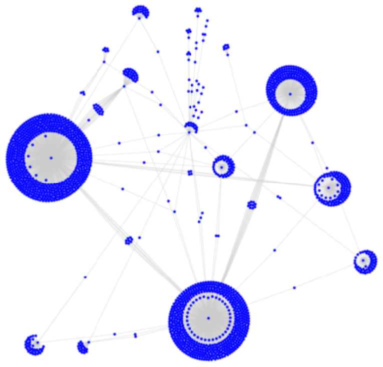

(f) A subset of DBpedia50, in a 6-hop radius from a median-degree node.

Figure 3. Structurally-representative subsets of FB15k-237, WN18RR CoDExSmall, and DBpedia50.

Table 13. Degree distribution statistics for the training sets of all KGs tested.

Degree	CoDExSmall	DBpedia50	FB15k-23	WN18RR
minimum	10	1	1	1
25th percentile	15	1	11	2
median	17	1	22	3
75th percentile	25	2	41	5
maximum	1008	781	7614	482

link prediction as a solely structural task – the model's learning objective is attuned towards learning structural variations in the local subgraph around a triple.

To more fully explain the mechanism by which TWIG-I learns (or fails to learn), we analyze its performance regarding representative subsets of all KGs it was tested on, presented in Figure 3. To further highlight structural disparities, we also provide higher n-hop neighbourhood subgraphs for WN18RR and DBpedia50. In all cases, the subgraph is the region around an arbitrary node of median degree. Finally, we present the general connectivity statistics of each graph in Table 13.

We see that FB15k-237 and CoDExSmall are more connected graphs with a richer local graph topological structure and with easily identifiable clusters of separated and interconnected nodes. We further note that the large median degree of these graphs leads to many more triples being in the local neighbourhood of other triples. Together, this leads to large and structurally diverse subgraphs. We hypothesise that this may lead to better parameterisation and distinguishability by TWIG-I's structural features. In particular, this could explain TWIG-I's high performance on CoDExSmall, and why its performance on FB15k-237 exceeds its performance on WN18RR.

WN18RR and DBpedia50 are much more sparse knowledge graphs: they tend to contain hyperlocalised subgraphs with a "hub-and-spoke" edge distribution. The relatively uniform graph structure, combined with the generally smaller size of the subgraphs around each triple, could possibly provide less topological information that can be represented or learned by TWIG-I. While this hypothesis explains TWIG-I's poor performance on WN18RR, it does not explain why TWIG-I does comparatively well on DBpedia50 – a question that is left open for future research.

5.2. *Transfer Learning Across Knowledge Graphs*

As shown in Table 12, finetuning results in an increase in performance compared to the state-of-the-art in *some*, but not all, cases. The results indicate that TWIG-I performs differently in transfer learning depending on both the source graph (used to pretrain TWIG-I) and the destination graph (on which link prediction was performed).

Looking at the connectivity statistics in Table 13 and the subgraph visualisation in Figure 3, we see that CoDExSmall and FB15k-237 have a generally comparable structure. Similarly, DBpedia50 and WN18RR have comparable structures. Surprisingly, however, TWIG-I models pretrained on FB15k-237 perform better when finetuned on DBpedia50 than on CoDExSmall, and TWIG-I models pretrained on WN18RR perform better when finetuned on CoDExSmall than on DBpedia50. While this initially suggests there may be a preference for diversity in structure between pretraining and finetuning, more research is needed before any definitive conclusions can be drawn. e

5.3. Future Directions

We see TWIG-I as a primarily expository work in structure-based / embedding free graph learning. TO the extent of the knowledge of the authors, it is the first relational learning model to directly and natively enable cross-KG transfer learning. These properties leave us with several future directions, which we enumerate below.

First, we highlight a need to refine (and ablate) the structural features used for learning in TWIG-I to determine which features drive performance and whether new features could be of additional use in learning. We particularly highlight that moving beyond a 2-hop window to take into account a larger subgraph is an important direction, especially for graphs such as WN18RR where the local neighbourhood around a triple tends to be fairly homogeneous and inexpressive.

Second, we highlight that the existing co-frequency structural features (see Table 1 indicate the possibility to examine the performance of TWIG-I in terms of elements of KG ontology, such as domain and range, that these statistics directly encode. This has the interesting implication that TWIG-I maybe be able to model parts of ontologies directly as a result of its structural modelling approach.

Finally, we highlight that further research into TWIG-I's wide variance in performance across KGs, in both the standard learning and the transfer learning settings, merits continued research.

6. Conclusion

In this paper we present TWIG-I, a novel link prediction model that uses a fixed set of structural features rather than node/edge embeddings to perform the link prediction task. As a result of this shift in link prediction problem formulation, TWIG-I is the first relational learning model known to the authors that natively enables transfer learning across diverse KGs. TWIG-I's performance is mixed in both the standard learning and the transfer learning settings, but results indicate that it can approach (or even beat, in some cases) the performance of KGE models. It must, however, be mentioned that it under-performs relative to KGE models on the standard KGE benchmarks FB15k-237 and WN18RR.

As TWIG-I is based on graph structure alone, we hypothesise that further work in graph-structure-first learning models is merited to assess the ability of TWIG-I-like models to generalise and transfer knowledge across KGs from diverse domains.

Acknowledgements

This research was conducted with the financial support of Science Foundation Ireland D-REAL CRT under Grant Agreement No. 18/CRT6225 at the ADAPT SFI Research Centre at Trinity College Dublin, together with the sponsorship of Sonas Innovation Ireland. The ADAPT SFI Centre for Digital Content Technology is funded by Science Foundation Ireland through the SFI Research Centres Programme and is co-funded under the European Regional Development Fund (ERDF) through Grant # 13/RC/2106_P2.

References

[1] Hogan A, Blomqvist E, Cochez M, D'amato C, Melo GD, Gutierrez C, et al. Knowledge Graphs. ACM Comput Surv. 2021 7;54(4). Available from: https://doi.org/10.1145/3447772.

[2] McCray AT, Burgun A, Bodenreider O. Aggregating UMLS semantic types for reducing conceptual complexity. Studies in health technology and informatics. 2001;84(0 1):216.

[3] Kemp C, Tenenbaum JB, Griffiths TL, Yamada T, Ueda N. Learning systems of concepts with an infinite relational model. In: AAAI. vol. 3; 2006. p. 5.

[4] Toutanova K, Chen D. Observed versus latent features for knowledge base and text inference. In: Allauzen A, Grefenstette E, Hermann KM, Larochelle H, Yih SWt, editors. Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality. Beijing, China: Association for Computational Linguistics; 2015. p. 57-66. Available from: https://aclanthology.org/W15-4007.

[5] Lewis D, Keeney J, O'Sullivan D, Guo S. Towards a managed extensible control plane for knowledge-based networking. In: Large Scale Management of Distributed Systems: 17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2006, Dublin, Ireland, October 23-25, 2006. Proceedings 17. Springer; 2006. p. 98-111.

[6] Wu J, Orlandi F, O'Sullivan D, Dev S. An ontology model for climatic data analysis. In: 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS. IEEE; 2021. p. 5739-42.

[7] Chandak P, Huang K, Zitnik M. Building a knowledge graph to enable precision medicine. Scientific Data. 2023;10(1):67.

[8] Mahdisoltani F, Biega J, Suchanek FM. A knowledge base from multilingual Wikipedias–yago3. Technical report, Telecom ParisTech. 2014.

[9] Dumontier M, Callahan A, Cruz-Toledo J, Ansell P, Emonet V, Belleau F, et al. Bio2RDF release 3: a larger connected network of linked data for the life sciences. In: Proceedings of the 2014 international conference on posters & demonstrations track. vol. 1272. Citeseer; 2014. p. 401-4.

[10] Speer R, Chin J, Havasi C. Conceptnet 5.5: An open multilingual graph of general knowledge. In: Proceedings of the AAAI conference on artificial intelligence. vol. 31; 2017. .

[11] Wang Q, Mao Z, Wang B, Guo L. Knowledge Graph Embedding: A Survey of Approaches and Applications. IEEE Transactions on Knowledge and Data Engineering. 2017;29(12):2724-43.

[12] Nickel M, Murphy K, Tresp V, Gabrilovich E. A Review of Relational Machine Learning for Knowledge Graphs. Proceedings of the IEEE. 2016;104(1):11-33.

[13] Lacroix T, Usunier N, Obozinski G. Canonical tensor decomposition for knowledge base completion. In: International Conference on Machine Learning. PMLR; 2018. p. 2863-72.

[14] Ali M, Berrendorf M, Hoyt CT, Vermue L, Galkin M, Sharifzadeh S, et al. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021;44(12):8825-45.

[15] Jain P, Rathi S, Chakrabarti S, et al. Knowledge base completion: Baseline strikes back (again). arXiv preprint arXiv:200500804. 2020.

[16] Sardina J, Kelleher JD, O'Sullivan D. TWIG: Towards pre-hoc Hyperparameter Optimisation and Cross-Graph Generalisation via Simulated KGE Models. In: 2024 IEEE 18th International Conference on Semantic Computing (ICSC); 2024. p. 122-9.

[17] Zhang H, Zheng T, Gao J, Miao C, Su L, Li Y, et al. Data poisoning attack against knowledge graph embedding. arXiv preprint arXiv:190412052. 2019.

[18] Bhardwaj P, Kelleher J, Costabello L, O'Sullivan D. Adversarial attacks on knowledge graph embeddings via instance attribution methods. arXiv preprint arXiv:211103120. 2021.

[19] Sadeghi A, Collarana D, Graux D, Lehmann J. Embedding Knowledge Graphs Attentive to Positional and Centrality Qualities. In: Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II. Berlin, Heidelberg: Springer-Verlag; 2021. p. 548–564. Available from: https://doi.org/10.1007/978-3-030-86520-7_34.

[20] Rossi A, Barbosa D, Firmani D, Matinata A, Merialdo P. Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. ACM Transactions on Knowledge Discovery from Data. 2021 01;15:1-49.

[21] Kadlec R, Bajgar O, Kleindienst J. Knowledge base completion: Baselines strike back. arXiv preprint arXiv:170510744. 2017.

[22] Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, et al. Graph neural networks: A review of methods and applications. AI Open. 2020;1:57-81. Available from: https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[23] Sanchez-Lengeling B, Reif E, Pearce A, Wiltschko AB. A Gentle Introduction to Graph Neural Networks. Distill. 2021. Https://distill.pub/2021/gnn-intro.

[24] Jiang X, Ji P, Li S. CensNet: Convolution with Edge-Node Switching in Graph Neural Networks. In: IJCAI; 2019. p. 2656-62.

[25] Jiang X, Zhu R, Ji P, Li S. Co-embedding of Nodes and Edges with Graph Neural Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020.

[26] Kipf T, Welling M. Variational Graph Auto-Encoders. ArXiv. 2016;abs/1611.07308. Available from: https://api.semanticscholar.org/CorpusID:14249137.

[27] Galárraga L, Teflioudi C, Hose K, Suchanek FM. Fast rule mining in ontological knowledge bases with AMIE++. The VLDB Journal. 2015;24(6):707-30.

[28] Bühmann L, Lehmann J, Westphal P. DL-Learner—A framework for inductive learning on the Semantic Web. Journal of Web Semantics. 2016;39:15-24. Available from: https://www.sciencedirect.com/science/article/pii/S157082681630018X.

[29] Ristoski P, Paulheim H. Rdf2vec: Rdf graph embeddings for data mining. In: The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part I 15. Springer; 2016. p. 498-514.

[30] Portisch J, Paulheim H. The RDF2vec family of knowledge graph embedding methods. Semantic Web. 2024. Available from: https://api.semanticscholar.org/CorpusID:255752765.

[31] Bonner S, Kirik U, Engkvist O, Tang J, Barrett IP. Implications of topological imbalance for representation learning on biomedical knowledge graphs. Briefings in Bioinformatics. 2022 07;23(5). Bbac279. Available from: https://doi.org/10.1093/bib/bbac279.

[32] Liu Z, Zhang W, Fang Y, Zhang X, Hoi SC. Towards locality-aware meta-learning of tail node embeddings on networks. In: Proceedings of the 29th ACM international conference on information & knowledge management; 2020. p. 975-84.

[33] Kotnis B, Nastase V. Analysis of the impact of negative sampling on link prediction in knowledge graphs. arXiv preprint arXiv:170806816. 2017.

[34] Mohamed SK, Novácek V, Vandenbussche PY, Muñoz E. Loss Functions in Knowledge Graph Embedding Models. In: DL4KG@ESWC; 2019. .

[35] Ruffinelli D, Broscheit S, Gemulla R. You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings. In: ICLR; 2020. .

[36] Mai G, Janowicz K, Yan B. Support and centrality: Learning weights for knowledge graph embedding models. In: Knowledge Engineering and Knowledge Management: 21st International Conference, EKAW 2018, Nancy, France, November 12-16, 2018, Proceedings 21. Springer; 2018. p. 212-27.

[37] Galkin M, Denis E, Wu J, Hamilton WL. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. arXiv preprint arXiv:210612144. 2021.

[38] Safavi T, Koutra D. Codex: A comprehensive knowledge graph completion benchmark. arXiv preprint arXiv:200907810. 2020.

[39] Shi B, Weninger T. Open-world knowledge graph completion. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32; 2018. .

[40] Ali M, Berrendorf M, Hoyt CT, Vermue L, Sharifzadeh S, Tresp V, et al. PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. Journal of Machine Learning Research. 2021;22(82):1-6. Available from: http://jmlr.org/papers/v22/20-825.html.

[41] Yang B, Yih Wt, He X, Gao J, Deng L. Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:14126575. 2014.

[42] Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O. Translating embeddings for modeling multi-relational data. Advances in neural information processing systems. 2013;26.