Research Article

Understanding the performance of knowledge graph embeddings in drug discovery

Stephen Bonner [a],[*], Ian P. Barrett [a], Cheng Ye [a], Rowan Swiers [a], Ola Engkvist [b], Charles Tapley Hoyt [c], William L. Hamilton [d],[e]

[a] *Data Sciences and Quantitative Biology, Discovery Sciences, R&D, AstraZeneca, Cambridge, UK*

[b] *Molecular AI, Discovery Sciences, R&D, AstraZeneca, Gothenburg, Sweden*

[c] *Laboratory of Systems Pharmacology, Harvard Medical School, Boston, USA*

[d] *School of Computer Science, McGill University, Montreal, Canada*

[e] *Mila - Quebec AI Institute, Montreal, Canada*

ARTICLE INFO

ABSTRACT

Knowledge Graphs (KG) and associated Knowledge Graph Embedding (KGE) models have recently begun to be explored in the context of drug discovery and have the potential to assist in key challenges such as target identification. In the drug discovery domain, KGs can be employed as part of a process which can result in lab-based experiments being performed, or impact on other decisions, incurring significant time and financial costs and most importantly, ultimately influencing patient healthcare. For KGE models to have impact in this domain, a better understanding of not only of performance, but also the various factors which determine it, is required.

In this study we investigate, over the course of many thousands of experiments, the predictive performance of five KGE models on two public drug discovery-oriented KGs. Our goal is not to focus on the best overall model or configuration, instead we take a deeper look at how performance can be affected by changes in the training setup, choice of hyperparameters, model parameter initialisation seed and different splits of the datasets. Our results highlight that these factors have significant impact on performance and can even affect the ranking of models. Indeed these factors should be reported along with model architectures to ensure complete reproducibility and fair comparisons of future work, and we argue this is critical for the acceptance of use, and impact of KGEs in a biomedical setting.

1. Introduction

The task of discovering effective and safe drugs is a complex and interdisciplinary one, with many drugs failing in clinical trials before being able to help patients [1]. Thus, the field is looking to leverage the large quantities of available data and information, much of which is inherently interconnected, to help improve the chances of a successful drug making it to market. Consequently, over recent years, an increasing number of Knowledge Graphs (KG) suitable for use within the drug discovery domain have been created [2,3], where drugs, genes and diseases are used as entities, with the interactions between them captured as relations. Several fundamental tasks within drug discovery can then be thought of as predicting the missing links between these entities – for example, drug repurposing can be considered as predicting missing links between drug and disease entities [4] or target discovery as identifying missing links between genes and diseases [5].

Naturally the family of Knowledge Graph Embedding (KGE) models (approaches which learn low dimensional representation of entities and relations which are trained to predict the plausibility of a given triple) have begun to be employed for these tasks. Perhaps unlike other domains, the predictions from such models are part of processes which can result in physical real-world experimentation being performed, and ultimately even clinical trials being undertaken – both with significant financial, regulatory and time costs associated, and more importantly impacting on the efforts to improve patient health. Therefore, there is an obvious need to ensure not only accurate predictions are being made but also that there is clear understanding of the various factors that can affect model predictive performance so that such predictions can be used effectively to derive impact and value. One other interesting aspect to consider is that these KGE models are typically not designed or tested against drug discovery datasets, so there is limited understanding about how they should be expected to perform in such cases. Indeed, recent

work [6] has shown that biomedical knowledge graphs have markedly different topological structure to typical KG benchmark datasets such as FB15K-237 [7] or WN18RR [8], displaying much greater average connectivity. Thus adding motivation for understanding how KGE models perform on such datasets.

In this paper, we perform a detailed experimental study of various factors that can affect KGE model performance, using five model architectures (ComplEx [9], DistMult [10], RotatE [11], TransE [12] and TransH [13]) and two real-world drug discovery oriented KGs (Hetionet [2] and BioKG [3]) with the goal of aiding better understanding, evaluation practices and reproducibility in the domain. The factors we investigate are the training setup of the model, the impact of changes in model hyperparameters, how model performance can be affected by both different random initialisations and changes in the train/test dataset splits and assessing performance on a domain specific task. All experiments are performed under a unified and consistent evaluation framework, on public data sources, using known best practices to ensure fair and reproducible comparisons. Additionally we release code to replicate our results.[1]

Lastly, we note that the contribution of this work is not to achieve state-of-the-art results on a given dataset or even to decide upon the definitive choices for the various factors we are investigating, rather it is to highlight how these can affect overall predictive performance and to encourage further research and attention on such foundational topics. Indeed, to the best of our knowledge, this is the first work to specifically focus on KGE model performance factors in the drug discovery domain.

2. Related work

2.1. Knowledge graph embeddings

Knowledge graphs A Knowledge Graph is a heterogeneous, multi-relation and directed graph, containing information about a set of entities $\mathcal{E}$ and a set of relationships between them $\mathcal{R}$, defined as $\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ [14]. A Knowledge Graph is often considered as a series of triples $(h, r, t) \in \mathcal{K}$, where $h, t \in \mathcal{E}$ are the head and tail entities connected via the relationship $r \in \mathcal{R}$. A hypothetical triple from a drug discovery knowledge graph could be $(Drug_1, treats, Disease_2)$, where the entities $Drug_1$ and $Disease_2$ are connected via the relationship *treats*. In many real-world knowledge graphs, the set of triples is known to be noisy and incomplete [15]. Thus numerous techniques have emerged which attempt to complete the missing knowledge based on the existing data in $\mathcal{K}$ through multi-relation link prediction [16]. Such techniques consider the partial triple $(Drug_1, treats, ?)$ and attempt to predict the correct tail entity, or be given $(?, treats, Disease_2)$ and attempt to predict the correct head entity.

Knowledge graph embeddings A growing number of approaches have been proposed in the literature which attempt to perform this knowledge graph completion task. In this work we focus upon the family of Knowledge Graph Embedding (KGE) techniques [17,18]. Typically, a KGE model learns a low-dimensional representation of each entity and relation in the graph. These embeddings are combined in various ways to produce a scalar value representing a measure of how likely that triple is to be true, with a larger score typically implying a more plausible triple [15]. More concretely, a model $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$, calculates a scalar value representing the plausibility for each potential triple $(h, r, t) \in \mathcal{K}$. For KGE approaches, f is typically a learned model which operates only with the embeddings of the elements in the triples, $f(\mathbf{h}, \mathbf{r}, \mathbf{t})$, where $\mathbf{h}, \mathbf{t} \in \mathbb{R}^k$ and $\mathbf{r} \in \mathbb{R}^j$. The values of k and j represent the dimension of the entity and relation embedding respectively.[2]

2.2. Understanding knowledge graph embedding model performance

Over recent years, there has been increasing interest in machine learning with graph structured data, with approaches created for homogeneous graph embeddings [19], graph-specific neural models [20] and knowledge graph embedding [17]. Whilst there have been numerous new model architectures proposed in the literature, there has been less work performed on understanding how these models are affected by the rest of the choices made in the machine learning pipeline, for example, how robust they are across hyperparameter values and model initialisations, or how performance changes across dataset splits. However, the work that does exist demonstrates some interesting observations.

For example, several graph neural network approaches were compared under a fair evaluation procedure [21], which showed that a change in train/test split would drastically alter the ranking of the models and that simpler baseline approaches, with correctly tuned hyperparameters, could outperform more complex models. The performance of different graph neural networks for graph-level classification has also been compared [22], with results showing that baselines approaches not using the graph structure can outperform those that do. The need for consistent, rigorous and reproducible benchmarks for graph machine learning is also an area of increasing research interest [23,24].

Evaluation of knowledge graph embeddings A study comparing seven different knowledge graph embedding techniques under a consistent evaluation framework on the non-biomedical benchmark datasets FB15K-237 [7] and WNRR [8] has been performed [16]. The authors observe that as new models are introduced, they are often accompanied with new training regimes or objective functions, making assessing the value of the new model architecture alone challenging. They undertake a detailed comparison across combinations of models, training paradigms and hyperparameters, using a Bayesian search approach. They find that earlier and comparatively simpler models, are very competitive when trained using modern techniques [16]. However, the study did not consider how model initialisation or dataset splits can affect performance.

In a similar study, 19 knowledge graph embedding approaches, implemented in the PyKEEN framework [25], are compared across eight different benchmark datasets [15]. One of the aims of the study was to investigate whether original published results could be reproduced, a task they found challenging. Additionally they perform detailed experiments over models and training paradigm combinations, searching over the hyperparameter space for a maximum of 24 h or 100 training repeats. Again they find that suitably tuned simple models can out-perform complex ones. The study does not consider drug discovery datasets specifically and does not assess how models perform across model seeds or dataset splits.

Biomedical domain specific evaluations The use of various homogeneous graph embedding techniques has been assessed across a range of biomedical tasks such as drug-drug and protein-protein interactions [26]. Whilst not exploring knowledge graph embedding techniques, the work explores how various hyperparameters affect predictive performance. They explore random walk and neural network based techniques including DeepWalk [27] and Graph Convolution based auto-encoders [28], using various task specific homogeneous graphs. An additional review compares both graph and knowledge graph specific approaches and their use in the biomedical world, however no experimental comparisons are made between the different approaches [29].

The performance of five knowledge graph embedding approaches (TransE, ComplEx, DistMult, SimplE and RotatE) have been compared on a knowledge graph constructed from the SNOMED resource [30]. The models are assessed on the tasks of link prediction, visualisation and entity classification, with a limited grid-search being performed to choose the hyperparameters.

Work has assessed the performance of knowledge graph embedding approaches for tasks within the drug discovery domain such as predicting drug-target interactions [31]. A different, often task-specific, graph is used for each of these experimental setups with much of the data

---

[1] https://github.com/AstraZeneca/kgem-in-drug-discovery.

[2] In practice, these are often set as the same value.

being taken from the BioSNAP repository [32]. However often these graphs are less complex than resources like Hetionet, with a typically limited number of entity and relationship types being present. Results use k-fold cross validation to assess performance variability over dataset splits, with a grid-search over a range of hyperparameters also being performed.

The effect of different data splitting strategies for predicting drug-drug interaction using graph-based methods (including TransE and TransD) has been investigated [33]. The work argues that realistic data splits should be used in order to avoid over-optimistic results, with several domain specific and time-based splits being assessed. Additionally the work claims that the tuning of various hyperparameters had little impact on the overall model performance.

2.3. Knowledge graphs in drug discovery

Recently, approaches exploiting knowledge graphs are being leveraged within the drug discovery domain to solve key tasks [34,35]. In a drug discovery knowledge graph, entities often represent key elements such as genes, disease or drugs, whilst the relations between them capture interactions. Many important tasks in drug discovery can then be considered as predicting missing links between these entities. For example, performing drug target identification, the process of finding genes involved in the mechanism of a given disease, has been addressed as link prediction between gene and disease entities using the ComplEx model on a drug discovery graph [5].

There are increasing numbers of public knowledge graphs suitable for use in drug discovery [34]. One of the first such graphs was Hetionet [2], originally created for drug purposing through the use of knowledge graph-based approaches. Since its introduction, other datasets have been released including the Drug Repurposing Knowledge Graph (DRKG) [36], OpenBioLink [37] and BioKG [3].

3. Experimental setup

In this section, we give an overview of the models, datasets and evaluation protocol used for our experimentation.

3.1. Models

As detailed in Section 2.1, many knowledge graph specific embedding models have been introduced, with the primary differentiator between them being how they score the plausibility of a given triple. Here we briefly detail the models utilised in this study, but interested readers are referred to larger reviews for context and comparisons with other approaches [17,18,38]. The models we have selected are popular approaches from the literature, cover a range of different methodologies and have begun to be explored in the context of drug discovery [5,39].

DistMult A simplification of the earlier RESCAL model, DistMult uses a vector for each relation type (represented as a diagonal square matrix to greatly reduce the number of required parameters) but is limited to learning only symmetric relations [10]. The score function used by DistMult to evaluate each triple is thus as follows:

$$f(h, r, t) = \mathbf{h}^\top \mathrm{diag}(\mathbf{r})\mathbf{t}$$

ComplEx To help overcome the ability to only learn symmetric relations, an extension of DistMult called ComplEx has been introduced [9]. The entity and relation embeddings for the ComplEx model are not real valued, unlike many other approaches, instead they are complex valued such that $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^k$. The score function for ComplEx then becomes:

$$f(h, r, t) = Re(\mathbf{h} \odot \mathbf{r} \odot \mathbf{t}),$$

where $\odot$ is the Hadamard product and $Re()$ is real value only from the complex number.

TransE One of the first models to use translational distance to learn embeddings such that relations are used to translate in latent space is

Table 1
Biomedical knowledge graphs used in this study.

KG dataset	$|\mathcal{E}|$	$|\mathcal{K}|$	$\mathcal{E}$ types	$\mathcal{R}$ types
Hetionet v1.0 [2]	47 K	2.2 M	11	24
BioKG v1.0 [3]	105 K	2 M	10	17

TransE [12]. In TransE, the relation embedding is added to the head entity such that the result lies close to the tail embedding, where the score function can be thought of as:

$$f(h, r, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||_F,$$

where F is typically either the l1 or l2 norm. One well known limitation of TransE is that is cannot correctly account for one-to-many, many-to-one or many-to-many relations being present in a knowledge graph [38].

TransH To help address the issues of TransE, another translational distance based model entitled TransH has been introduced [13]. TransH allows for entity embeddings to be given a different context depending upon the relation used in certain triple. This is achieved by modelling each relation r as a hyperplane, with the head and tail entity embeddings first being transformed using a normal vector of the hyperplane $\mathbf{w}_r \in \mathbb{R}^k$ as follows:

$$\mathbf{h}_r = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r,$$

$$\mathbf{t}_r = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r.$$

The score function is similar to the one used with TransE:

$$f(h, r, t) = -||\mathbf{h}_r + \mathbf{d}_r - \mathbf{t}_r||_2^2,$$

where $\mathbf{d}_r$ is a vector that lies in relations r hyperplane.

RotatE Combining ideas from different existing models, RotatE uses complex valued embeddings for entities and relations and works such that relations rotate head to tail entities [11]. Given complex embeddings for head, relation and tail $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^k$ (RotatE limits the complex elements of the relation embeddings to have a modulus of one: $|\mathbf{r}| = 1$) the score function then becomes:

$$f(h, r, t) = -||\mathbf{h} \odot \mathbf{r} - \mathbf{t}||,$$

with $\odot$ again being the Hadamard product. This allows RotatE to deal with symmetry/antisymmetry, inversion, and composition relation types [11].

3.2. Datasets

Throughout this work we employ two publicly available knowledge graphs suitable for use within the drug discovery domain - Hetionet [2] and BioKG [3], which are detailed in Table 1.

Both these datasets contain information about the key elements of drug discovery: genes, diseases and drugs, whilst capturing the interactions between them. Both are constructed from various high-quality public sources of biological and chemical information [34], with the major differences being in the complexity of the relationships captured (BioKG only uses a single relationship type between two entities, whereas Hetionet has up to three).

It has been observed that biomedical knowledge graphs can exhibit a different topological structure than the benchmark datasets against which the models are typically tested [6]. For example, Hetionet has a higher average degree than datasets like FB15K-237 [7] and WN18RR [8]. However it remains unknown how this impacts KGE model performance and we leave a detailed comparison between knowledge graphs from the biomedical and other domains for further work.

3.3. Evaluation protocol

The evaluation of KGE models is typically performed by measuring how likely the model ranks a holdout set of triples from the original

graph. However there are a series of choices one can make in this evaluation process which can drastically alter the results, thus ultimately making direct comparisons between published results challenging [15]. Hence the evaluation protocol is one of the crucial aspects for reproducibility, as the choices made can have a large impact on comparative performance. Here we describe our own evaluation procedure in full which closely follows the one established in Ali et al. [15].

Given a set of test triples $\mathcal{K}_{test} \subseteq \mathcal{K}$, we perform *both* left and right side evaluation where the head and tail entities are removed in turn and the model is evaluated based on how well these missing entities can be predicted given the partial triple. To do this, for each triple in the test set $(h, r, t) \in \mathcal{K}_{test}$, two corrupted sets are constructed: the set where the head entity has been corrupted with every possible entity $\mathcal{H}' = \{(h', r, t) \mid h' \in \mathcal{E}\}$ and likewise for the tail entity $\mathcal{T}' = \{(h, r, t') \mid t' \in \mathcal{E}\}$. The goal then is to have the original true triple given a higher score by the model than these corrupted triples. One decision that needs to be taken is if any true triples, those that are already part of $\mathcal{K}$, in the corrupted sets are removed before scoring. Following prior work [12,15], we use the filtered evaluation setting where we remove any corrupted triple which is already in the graph, as their presence can skew the results.

It is possible that two or more triples in the test set are given the same score by the model when all are being ranked and how this situation is handled can also affect the results. One can assume the extremes, where the true triple is assumed to be at the start or end of the ranked list. For this work we present the mean of the rank using these two assumptions.

Metrics We employ commonly used knowledge graph performance metrics including Mean Reciprocal Rank (MRR) and Hits@k (see [15] for definitions). Additionally we use the recently introduced Adjusted Mean Rank (AMR) [40] owing to its ability to allow comparison between graphs of different sizes. However we would like to highlight that using metrics alone, especially for use cases like drug discovery where model predictions will often result in real-world lab-based experiments being performed, perhaps should not be the sole way in which models are judged.

3.4. Implementation details

All work has been performed using the PyKEEN framework [25], a python library for knowledge graph embeddings built on top of PyTorch [41]. Additionally we use the Optuna library to perform the hyperparameter optimisation [42]. All experiments were performed on machines with Intel(R) Xeon(R) Gold 5218 CPUs and NVIDIA(R) V100 32 GB GPUs. Additionally, we kept the software environment consistent throughout all experimentation using python 3.8, CUDA 10.1, PyTorch 1.7, Optuna 2.3 and PyKEEN 1.0.6.

Table 2
Default model training setups and hyperparameters.

Parameter	Value by approach				
	ComplEx	DistMult	RotatE	TransE	TransH
Embedding Dim	50	50	200	50	50
Num Epochs			500		
Learning Rate			0.02		
Num Negatives			1		
Optimiser			Adagrad		
Inverse Relations			False		
Loss Function		Margin Ranking Loss (Margin 1.0)			

4. Results

We now present the results of our experimental evaluation. All the results presented are taken from a random 10% holdout set of test triples, unseen during the training process. Unless otherwise stated, these test triples comprise all entity and relation types. Throughout we will make use of the default set of training setup choices and hyperparameters detailed in Table 2.

4.1. Training setup study

We begin by assessing the impact of various categorial choices about the training setup of the models, evaluating several common options. Specifically, we vary the optimiser (from a choice of Stochastic Gradient Descent (SGD), Adam and AdaGrad), training objective function (from a choice of Binary Cross Entropy (BCEL), Softplus (SPL), Margin Ranking (MRL) and the self adversarial loss (called NSSA henceforth) from Sun et al. [11]) and with/without adding inverse relationships into the graph (the process of adding a copy of each triple during training with an inverse relation [43]). Throughout all of these runs, the model initialisation seed, dataset split and hyperparameters were kept constant and are detailed in Table 2.

Fig. 1 presents the distribution of Hits@10 scores on the test set across training setup choices for all models and both datasets, enabling a global view of how the models respond to changes in training setup. It can be seen that for Hetionet, many of the models have a similar range of performance, with DistMult standing out as having poor performance no matter the training setup, a trend which can also be seen on the BioKG dataset.

To further investigate the impact of different training setup choices, Fig. 2 highlights the distributions of model performance separated by the different choices for both datasets in which some interesting trends emerge. Perhaps the most striking observation can be seen in Fig. 2e and f, which shows that adding inverse relationships to the training KG
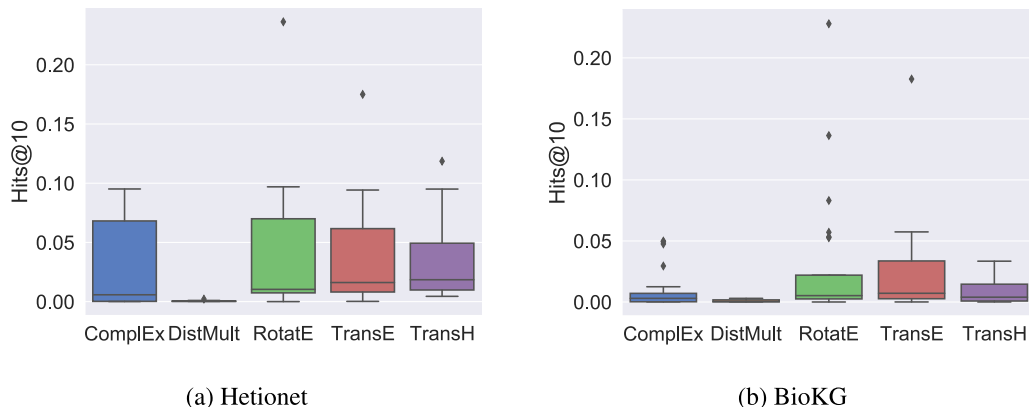

(a) Hetionet (b) BioKG

Fig. 1. Distribution of the Hits@10 scores across all categorial training setup choices.

(a) Hetionet Opt

(b) BioKG Opt

(c) Hetionet Loss

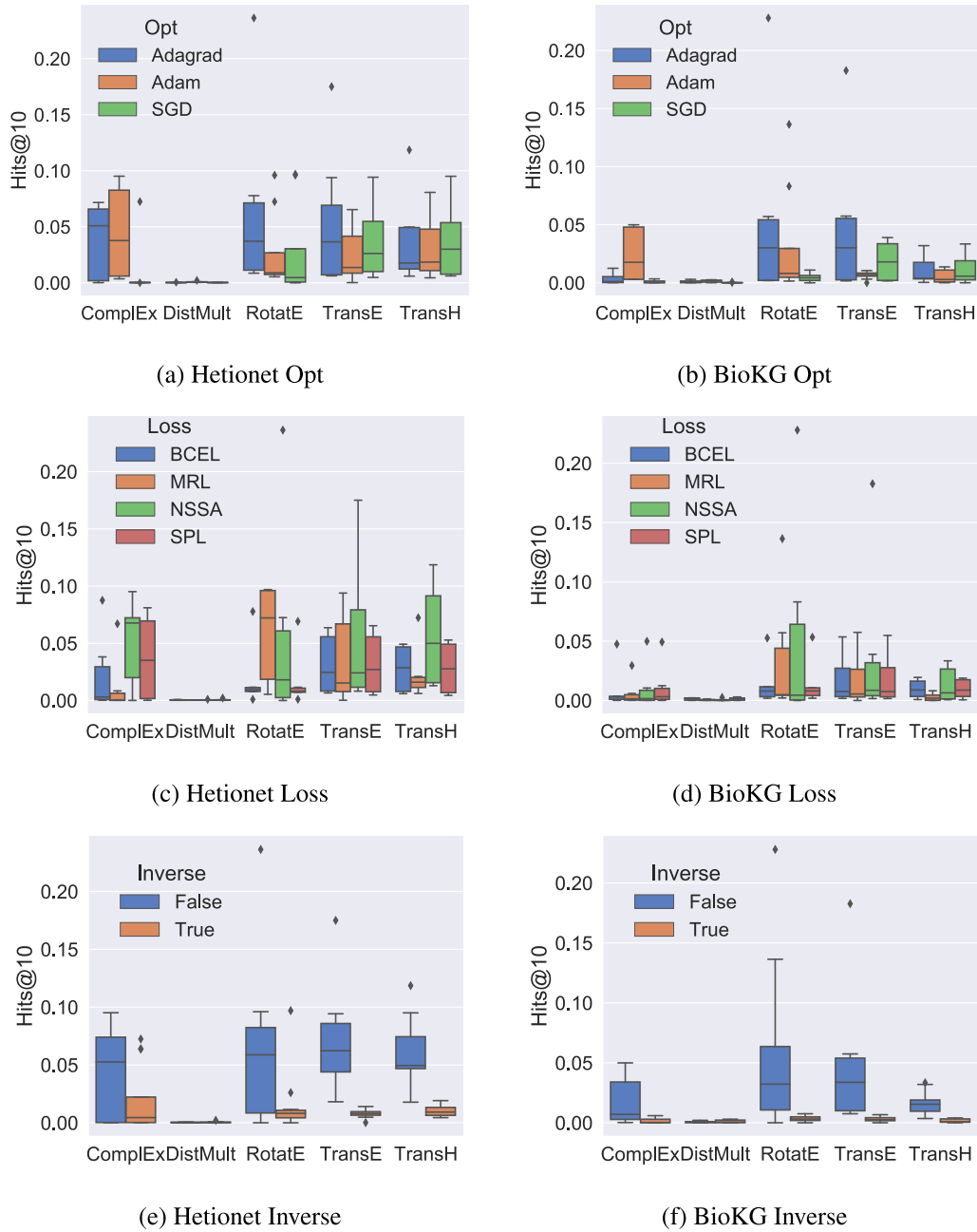(d) BioKG Loss

(e) Hetionet Inverse

(f) BioKG Inverse

Fig. 2. The effect of different training setup choices across all models and both datasets.

almost always performs worse on average than not including them – this is in contrast to recent experimental evidence from non-biomedical domains [15]. Other observations include that, as shown in Fig. 2c and d, using the NSSA loss function typically results in models having the highest peak predictive performance and that Fig. 2a and b show that the Adagrad optimiser is typically used in the best performing training setup.

However, overall Fig. 2 reveals how multifactorial the problem of choosing the training setup can be – clearly users must experiment to discover the most performant combination. The figure also highlights how improvements in the training setups have driven performance increases, perhaps more so than improvements in model architectures. For example, the RotatE model, given the correct training setup, is shown to perform best overall for both datasets, however it can also be outperformed by older approaches like TransE if suboptimal choices were made.

4.2. Hyperparameter optimisation

Even with the correct training setup, values given to key hyperparameters can have a significant impact on overall performance. It is common for various strategies to be employed to search for the best set of hyperparameters, a process called HyperParameter Optimisation (HPO) [42]. In this section, we perform a detailed HPO search using two sampling strategies across all models and datasets. The two sampling strategies employed are the Bayesian *Tree-structured Parzen Estimator Approach (TPE)* [44], an approach which creates a model to approximate the performance of a given hyperparameter set using historical information and a *random* search [45] in which hyperparameters are sampled for each run independent of any previous ones. For each combination of search strategy, model and dataset we run 100 different experiments (with no time limit) to determine the hyperparameters - with the model seed, training setup (as detailed in Table 2) and dataset split being fixed.
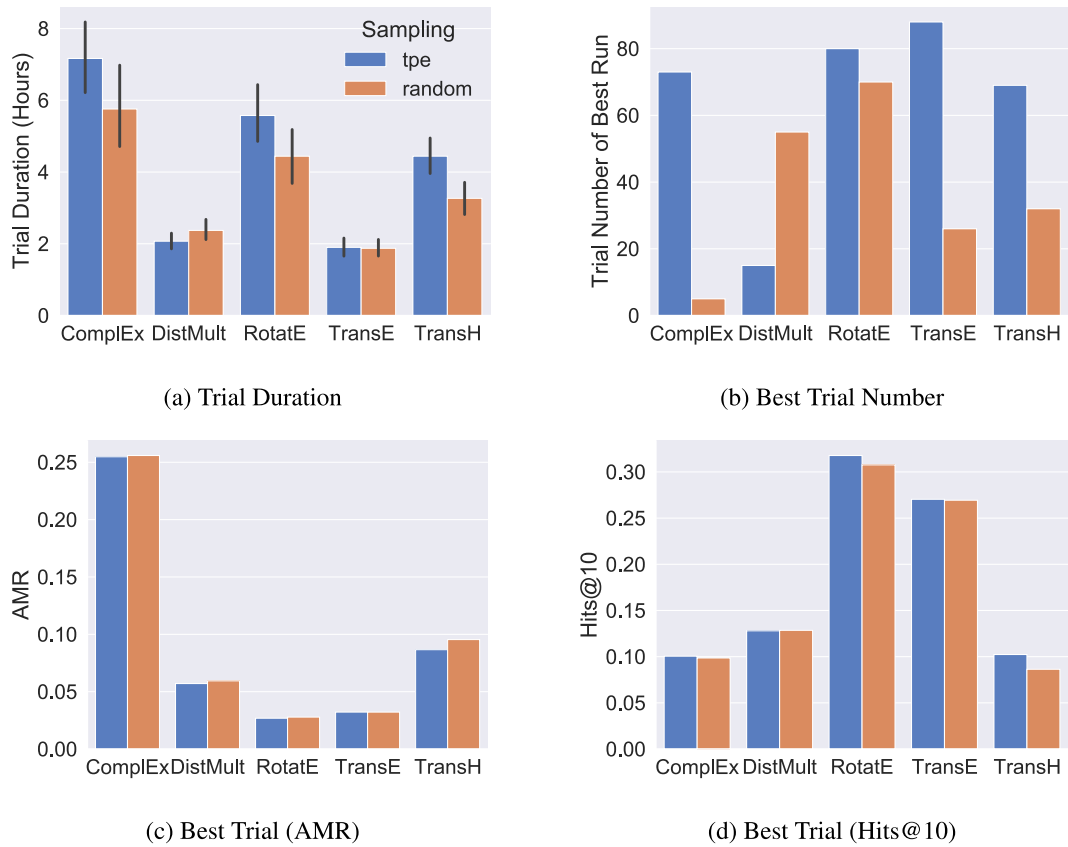
(a) Trial Duration

(b) Best Trial Number

(c) Best Trial (AMR)

(d) Best Trial (Hits@10)

Fig. 3. Performance of the two sampling algorithms after 100 trials across a range of metrics with all models on Hetionet.

Table 3
Range of search for parameter values, presented as min, max and step value.

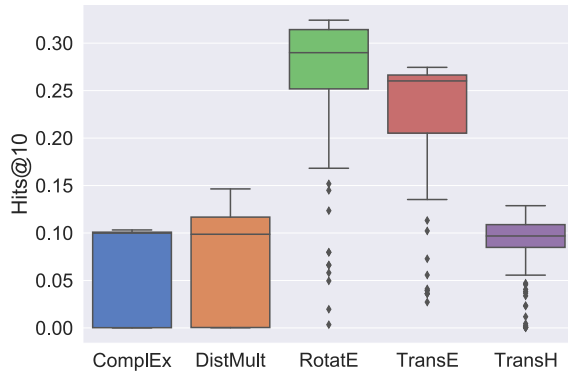| Parameter	Value range
Embedding Dim	$[16 \dots 512, 16]$
Num Epochs	$[100 \dots 1000, 100]$
Learning Rate	$[0.001 \dots 0.1, \log]$
Num Negatives	$[1 \dots 100, 10]$

The ranges of values searched over is detailed in Table 3, this range is the same for all models, datasets and sampling strategies. A given set of hyperparameters were assessed using the AMR metric for both search strategies on a holdout set of triples. This holdout set is a randomly selected, but fixed across all trails and repeats, set of triples comprising 10% of all in the respective KGs.

Fig. 3 presents an overview of the HPO experiments comparing the two sampling strategies, focusing on the Hetionet dataset for brevity (the results for BioKG demonstrated very similar patterns). Firstly, Fig. 3a displays the mean runtime of all models for both sampling strategies, showing that the TPE approach generated longer trial durations on average. This is likely as it tuned parameters which can increase runtime, embedding dimension, number of negative samples and number of epochs, in the pursuit of additional predictive performance. Fig. 3b shows on which of the 100 experimental trials the best performing model was produced, with the TPE approach generally displaying its best performance close to the maximum number of repeats. Fig. 3c and d display the predictive performance of the approaches as measured by AMR and Hits@10 respectively. One surprising observation is how close the performance is of the TPE and random sampling strategies across all models – with TPE only producing models which are marginally better than those trained on random hyperparameter choices. This highlights how

given enough repeats, a random search can happen upon near optimal parameters by chance, at least for these models and datasets. Indeed given the additional average runtime incurred by TPE, a random search may be the better balance of runtime and performance. One final observation is the negative correlation between the AMR and Hits@10 performance, which is encouraging to see as the HPO search was only optimising the AMR value.

In Fig. 4, the distribution of Hits@10 scores across all models and 100 trials using TPE sampling is presented. The figure shows the importance of correct hyperparameter selection, as all models are shown to be sensitive to them. Indeed the selection of hyperparameters could change the ordering of ranked model performance, with the best performing model RotatE demonstrating performance below others given suboptimal parameter choices. Additionally one should consider that older approaches like TransE can still perform very competitively given appropriate time is spent tuning them. This is important to consider as new models continue to be proposed, and we hope that authors make the effort to appropriately tune baseline approaches for fairer comparison.

Fig. 5 highlights the percentage in improvement across different Hits@k levels for the best set of parameters from the TPE sampling strategy over the baseline ones (i.e., models trained using the setup defined in Table 2). It can be seen that the majority of models clearly benefit from a detailed hyperparameter search, albeit to varying degrees. For example, those approaches which have worst performance overall such as DistMult benefit most by having their parameters tuned, conversely the higher performing approaches like RotatE benefit to a lesser degree. One observation seen in Fig. 5b is that the best set of parameters produced via the HPO process fails to exceed the performance of the baseline at lower values of k for the ComplEx model. However, overall these experiments demonstrate that time should be taken to fine-tune hyperparameters as the performance increase over even sensible defaults can be dramatic.
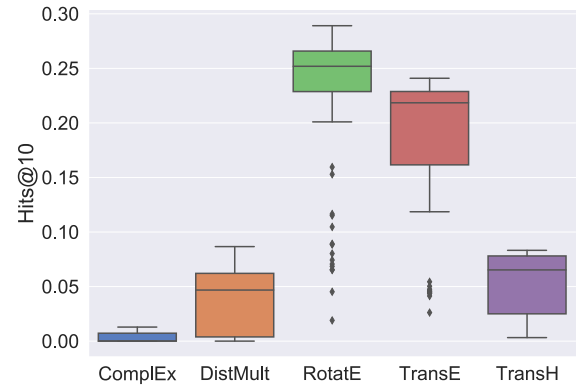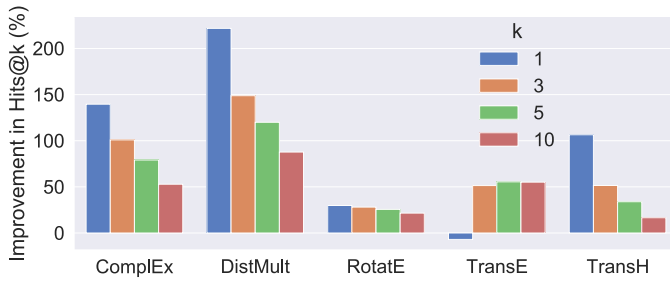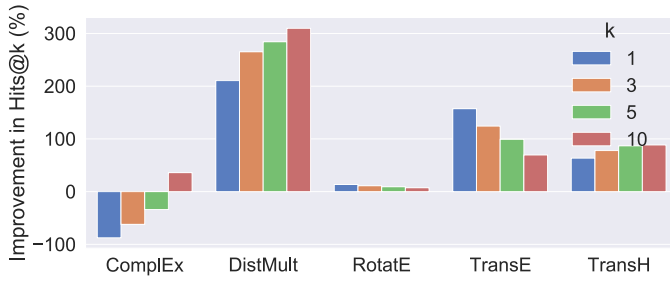
Fig. 4. Distribution of the Hits@10 scores across all 100 runs of different parameters.

(a) Hetionet

(b) BioKG

Fig. 5. Percentage improvement for the best parameter configurations over baseline values.

Fig. 6. Performance changes with embedding dimension for RotatE on Hetionet, whilst fixing all other parameters.

4.2.1. Hyperparameter values

To investigate the values taken by hyperparameters and how they correlate with performance, we averaged hyperparameter values for the five best and worst configurations (as determined by the Hits@10 metric) from the HPO runs for both search approaches on the Hetionet dataset – which can be seen in Fig. 7. Fig. 7 reveals some interesting trends, for example it can seen that there is a large variance in many hyperparameter values even among just five examples, indicating that there are multiple permutations of hyperparameter values which perform to a similar level. It is also surprising how close the range of values for many parameters are for the five best and worst configurations, perhaps revealing how multifactorial the problem of choosing hyperparameters to be – performance seems to be driven by specific combinations of values given to multiple parameters. This indicates that researchers should search over multiple parameters simultaneously, rather than focusing on a single one to achieve the best overall configuration (Although Fig. 6 shows that model performance can be increased via
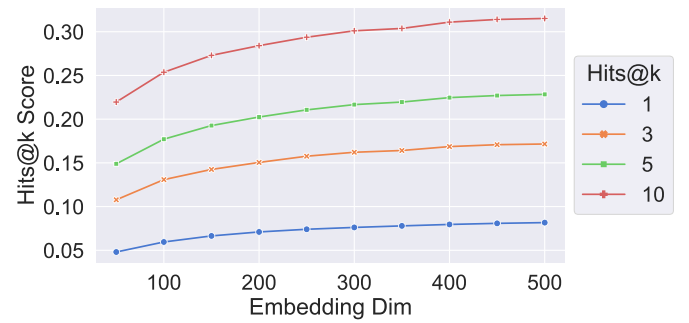
changes in a single parameter, although it does not explain all). However, there are examples in the figure of clear differences between best and worst configurations, with RotatE performing better on average with larger embeddings, more training epochs and a larger learning rate.

The average parameter values over the five best runs for both dataset is displayed in Table 4. This shows that, despite the datasets being from the same domain, parameter values are not similar, highlighting that values need to be optimised on a per-dataset basis. Comparing the values for the two KGs some trends do emerge however, for example all models on BioKG tend to require a larger learning rate and have a broader range of embedding dimensions which perform well.

In Table 5 we report the specific set of hyperparameters that resulted in the best overall performing models across both Hetionet and BioKG. These are compared with hyperparameters, taken from the original publications, for two non-biomedical KGs: FB15K [12] and WN18 [12]. The table highlights how the parameters across the two domains are rarely comparable in value. For example, typically all models require a larger embedding size and more negative samples to achieve the highest level of performance using biomedical KGs. This further reinforces the idea that one cannot assume hyperparameters will generalise across domains, or even across datasets from within a domain. This can be seen in the difference in parameters for the same model across Hetionet and BioKG, despite the two comprising similar entity and relation types.

4.2.2. Task specific HPO

In many applications, achieving good performance on specific relation types is the ultimate goal, with the rest of the graph hopefully being used to improve this specific task. One interesting aspect to consider is if tuning the HPO search to focus on just the relations of interest generates better performing models than tuning on all possible relation types. For
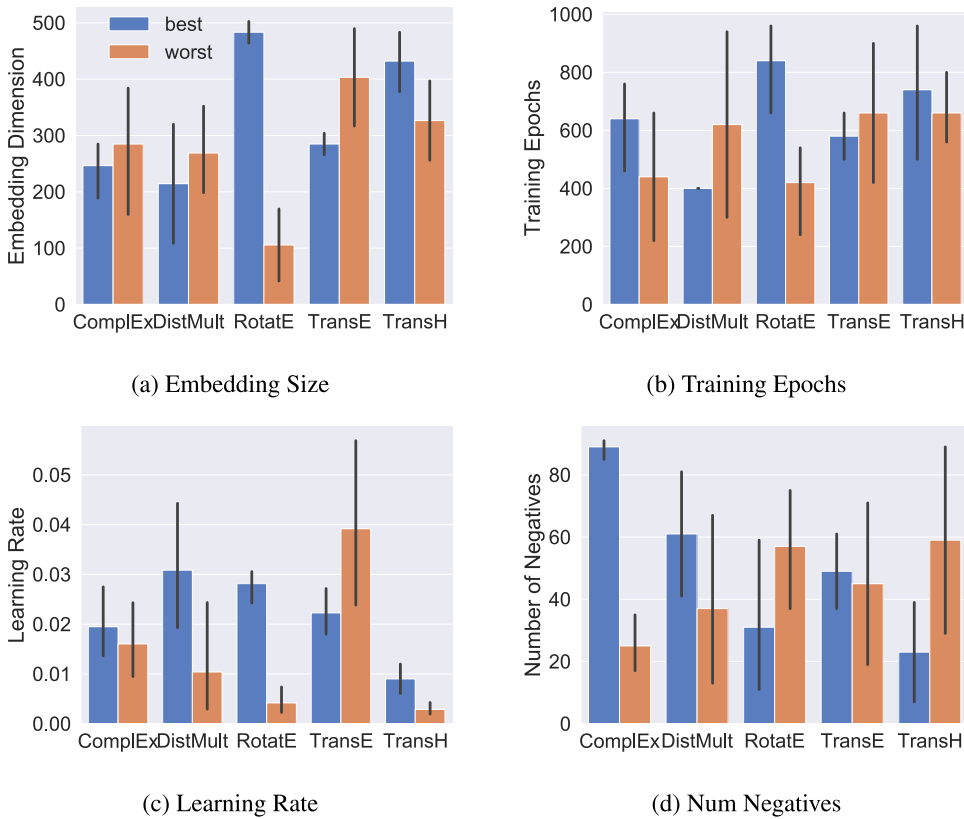
(a) Embedding Size

(b) Training Epochs

(c) Learning Rate

(d) Num Negatives

Fig. 7. The mean hyperparameter values of the top five best and worst configurations by model on the Hetionet dataset.

Table 4
The top five best performing hyperparameter configurations (includes models from both TPE and random search) for all models and datasets. Values presented as mean and standard deviation.

Dataset	Approach	Top five configs				
		Emb size	Num epochs	Learning rate	Num neg	Hits@10 ↑
Hetionet	ComplEx	246 ± 61	640 ± 195	0.019 ± 0.009	89 ± 4	0.102 ± 0.001
	DistMult	214 ± 141	400 ± 000	0.030 ± 0.015	61 ± 27	0.140 ± 0.004
	RotatE	483 ± 26	840 ± 194	0.028 ± 0.004	31 ± 30	0.322 ± 0.001
	TransE	285 ± 26	580 ± 110	0.022 ± 0.006	49 ± 16	0.272 ± 0.001
	TransH	432 ± 67	740 ± 297	0.008 ± 0.004	23 ± 20	0.129 ± 0.076
BioKG	ComplEx	422 ± 96	600 ± 187	0.058 ± 0.030	81 ± 10	0.012 ± 0.001
	DistMult	349 ± 180	100 ± 000	0.039 ± 0.006	71 ± 0	0.082 ± 0.002
	RotatE	413 ± 51	760 ± 89	0.055 ± 0.020	25 ± 9	0.286 ± 0.001
	TransE	378 ± 115	540 ± 134	0.078 ± 0.019	91 ± 0	0.239 ± 0.001
	TransH	317 ± 114	920 ± 130	0.067 ± 0.018	27 ± 9	0.08 ± 0.000

example, if we were specifically interested in being able to predict gene to disease edges, would tuning the HPO process on just these edges yield better overall performance on this task? To investigate this, we repeated the HPO search using the TPE method on Hetionet across both RotatE (generally the best performing model thus far) and ComplEx (typically one of the lower performing models), where parameters were tuned via model performance only on gene to disease edges. This is in contrast to the previous HPO search, where parameters were tuned via model performance measured on a random holdout set comprised, potentially, of all relation types.

This comparison is presented in Fig. 8, where RotatE and ComplEx were evaluated using the Hits@1 and Hits@10 metrics on a random selection of gene-disease edges using hyperparameters optimised for this edge type, compared with those optimised on all edges. The figure shows that for ComplEx, which performs poorly on gene-disease edges when using the hyperparameters tuned on the full set of edges, its predictive performance increases when using the edge-tuned hyperparameters. The gene-disease edges represent a small fraction of total in Hetionet at less than 1%, which seems to impact the two models in different ways. Com-

pared to ComplEx, RotatE demonstrates a decrease in performance using the gene-disease tuned hyperparameters, suggesting it is seemingly not being able to discover an optimal set of hyperparameters using the much sparser signal offered by the reduced edge counts. It is also possible that, given the much smaller size of the validation set used for hyperparameter tuning, that the model was over-fitting to these limited edges and not being able to generalise to unseen examples. Overall these experiments suggest that, when a model performs well overall, hyperparameters learned across all relation types can still be optimal for use in a relation type specific setting.

4.3. Model initialisation random seed

We now assess how model predictive performance changes as the random seed used to initialise model parameters is varied. Many architectures are known to be impacted by the initial conditions of the parameters as determined by the random seed, with performance not being consistent over seeds [46]. However, thus far, the impact of model random seed has not been thoroughly investigated in the biomedical KG
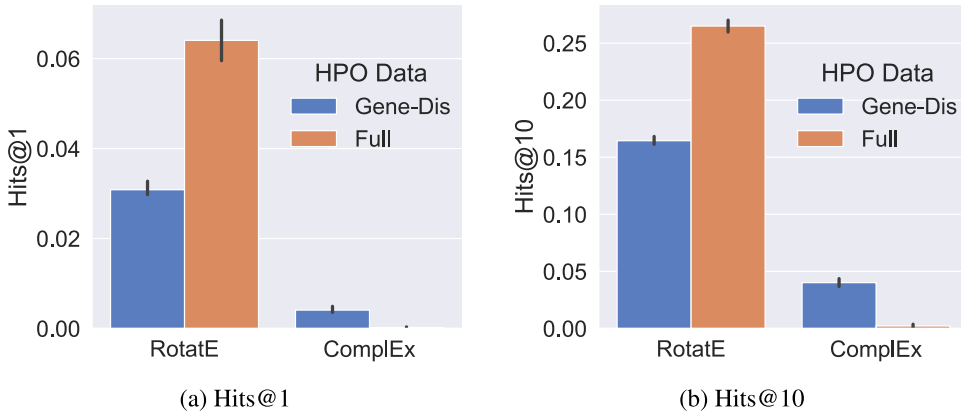
(a) Hits@1
(b) Hits@10

Fig. 8. Comparison between models trained on Hetionet using HPO values taken from optimising with TPE across all relation types (Full) and just Gene-Disease relations (Gene-Dis). Hits@1 and Hits@10 scores are presented on a holdout testset comprised only of Gene-Disease relations.

Table 5
The hyperparameters from the best performing run. Results from two non-biomedical KGs (FB15k and WN18) are included for comparison purposes.

Approach	Dataset	Best config			
		Emb size	Num epochs	Learning rate	Num neg
ComplEx	Hetionet	272	700	0.03	91
	BioKG	464	600	0.09	91
	FB15K	200	1000	0.5	10
	WN18	150	1000	0.5	1
DistMult	Hetionet	80	400	0.02	41
	BioKG	480	100	0.05	71
	FB15K	100	100	0.1	2
	WN18	100	300	0.1	2
RotatE	Hetionet	512	500	0.03	41
	BioKG	448	900	0.06	31
	FB15K	1000	1000	0.0001	128
	WN18	500	1000	0.0001	1024
TransE	Hetionet	304	500	0.02	61
	BioKG	448	600	0.1	91
	FB15K	50	1000	0.01	1
	WN18	20	4000	0.01	1
TransH	Hetionet	480	800	0.005	1
	BioKG	368	900	0.06	31
	FB15K	100	500	0.005	1
	WN18	50	500	0.01	1

(a) Hetionet
(b) BioKG

Fig. 9. Distribution of AMR score obtained by repeating the experiment whilst varying the model random seed.

(a) Hetionet
(b) BioKG

Fig. 10. Distribution of AMR score obtained by repeating the experiment whilst varying the data split randomly.

domain, with results in the literature rarely being presented as averaged over different random seeds.

To assess the impact of model initialisation, in this experiment, we keep all other experiment variables constant (fixing the hyperparameters, training setup and dataset split) and run repeats over a fixed set of 20 random seed for all models and across both datasets. The overall results of this are presented in Table 6, which shows that the majority of the models are relatively robust to the random seed used for parameter initialisation. The one clear exception is the performance of the DistMult model, which demonstrates a large sensitivity to the random seed with certain metrics.

To investigate this further, the distribution of the AMR score across the random seeds is presented in Fig. 9, showing that DistMult's average performance is skewed by a few outlying poor results at certain seed values. Although it can be seen that, whilst ignoring the outliers for Hetionet brings DistMult's performance more inline with the other models, the majority of the runs for the BioKG dataset are still significantly worse than other models.

4.4. Dataset splits

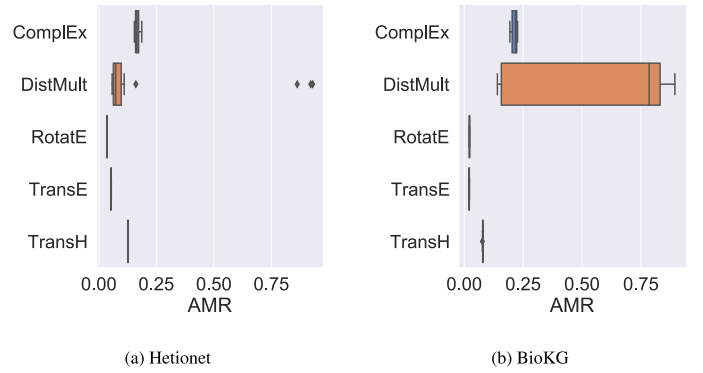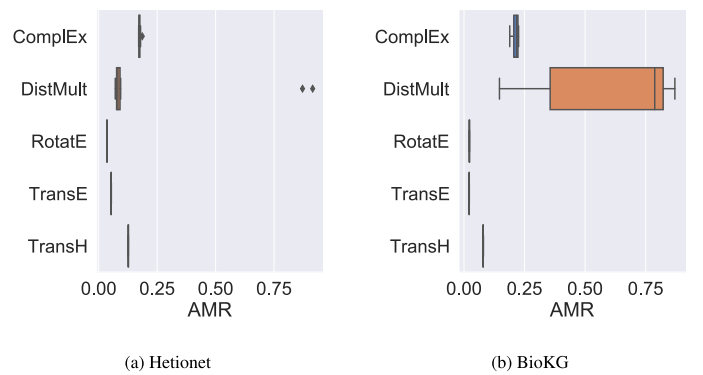We now explore how model performance can be affected depending upon the dataset splits.

Random splits The performance of machine learning models is known to vary over different dataset splits. However, many benchmark datasets are provided with predetermined train/test splits, which has resulted, in the case of graph-based models, in approaches over-fitting to validation sets and not generalising to other random splits [21].

In this experiment, we assess how models respond to changes in the train/test split of the underlying datasets as other variables (hyperparameters, models seeds and training setup) are kept constant, with each dataset being split randomly 10 times. Here 10% of the triples are used for test, with the remainder used for training. Table 7 displays the results from this experiment and shows that most models have very consistent performance across the different dataset splits. Indeed many of the values are similar to those from the model seed experiments, indicating that they do reflect the models true performance. However, Fig. 10 shows

Table 6

The mean performance with standard deviation over 10 fixed random seeds of all models and both datasets as measured by various metrics. Here only the random seed used to initialise the model parameters is changed, whilst all over variables are kept fixed.

| Dataset	Approach	Metric			
		AMR ↓	MRR ↑	Hits@1 ↑	Hits@10 ↑
Hetionet	ComplEx	0.167 ± 0.009	0.026 ± 0.009	0.008 ± 0.003	0.064 ± 0.024
	DistMult	0.201 ± 0.303	0.036 ± 0.019	0.012 ± 0.007	0.079 ± 0.045
	RotatE	$\mathbf{0.035 \pm 0.000}$	$\mathbf{0.127 \pm 0.000}$	$\mathbf{0.063 \pm 0.000}$	$\mathbf{0.262 \pm 0.001}$
	TransE	0.053 ± 0.000	0.079 ± 0.001	0.034 ± 0.001	0.117 ± 0.002
	TransH	0.126 ± 0.000	0.033 ± 0.001	0.007 ± 0.000	0.088 ± 0.002
BioKG	ComplEx	0.213 ± 0.011	0.008 ± 0.001	0.003 ± 0.000	0.008 ± 0.003
	DistMult	0.560 ± 0.339	0.015 ± 0.003	0.007 ± 0.001	0.027 ± 0.006
	RotatE	0.022 ± 0.000	$\mathbf{0.123 \pm 0.000}$	$\mathbf{0.059 \pm 0.000}$	$\mathbf{0.240 \pm 0.001}$
	TransE	$\mathbf{0.021 \pm 0.000}$	0.062 ± 0.000	0.019 ± 0.000	0.134 ± 0.001
	TransH	0.078 ± 0.001	0.022 ± 0.000	0.008 ± 0.000	0.042 ± 0.001

Table 7

The mean performance with standard deviation over 10 fixed random dataset splits of all models and both datasets as measured by various metrics. Here only the random seed used to initialise the model parameters is changed, whilst all other variables are kept fixed.

Dataset	Approach	Metric			
		AMR ↓	MRR ↑	Hits@1 ↑	Hits@10 ↑
Hetionet	ComplEx	0.174 ± 0.004	0.024 ± 0.006	0.007 ± 0.002	0.059 ± 0.016
	DistMult	0.242 ± 0.343	0.028 ± 0.016	0.008 ± 0.006	0.061 ± 0.036
	RotatE	$\mathbf{0.035 \pm 0.000}$	$\mathbf{0.126 \pm 0.000}$	$\mathbf{0.135 \pm 0.001}$	$\mathbf{0.262 \pm 0.001}$
	TransE	0.053 ± 0.000	0.081 ± 0.000	0.035 ± 0.001	0.117 ± 0.001
	TransH	0.126 ± 0.000	0.032 ± 0.000	0.006 ± 0.000	0.087 ± 0.001
BioKG	ComplEx	0.212 ± 0.012	0.008 ± 0.001	0.003 ± 0.001	0.009 ± 0.004
	DistMult	0.627 ± 0.303	0.014 ± 0.003	0.007 ± 0.001	0.026 ± 0.004
	RotatE	0.022 ± 0.000	$\mathbf{0.123 \pm 0.000}$	$\mathbf{0.060 \pm 0.000}$	$\mathbf{0.241 \pm 0.001}$
	TransE	$\mathbf{0.021 \pm 0.000}$	0.063 ± 0.000	0.021 ± 0.000	0.135 ± 0.001
	TransH	0.079 ± 0.001	0.023 ± 0.001	0.008 ± 0.000	0.043 ± 0.001
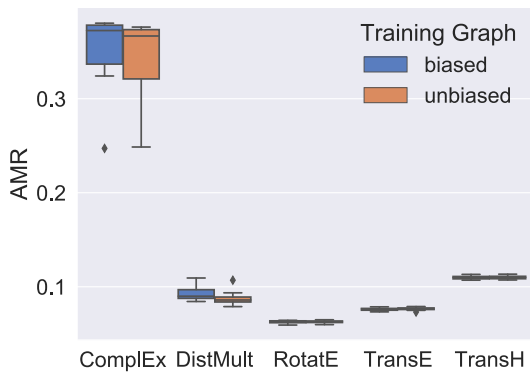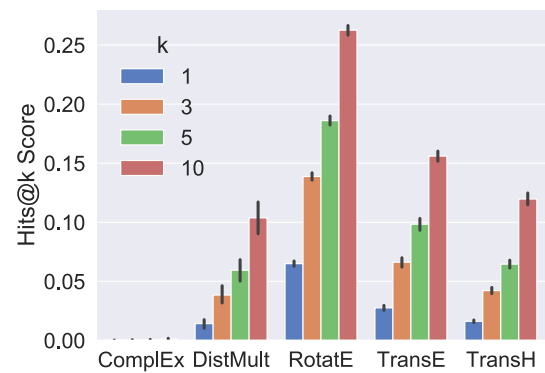
(a) Bias Vs. Unbiased   (b) Hits@k

Fig. 11. Assessing model performance at gene-disease prioritisation on Hetionet.

that once again, that DistMult has much greater performance variability over the dataset splits which affects its average ranking compared with other approaches. This reinforces the notion that models should be tested both on different dataset splits and with different random initialisations to assess their true generalisability.

Domain specific splits - gene-disease prioritisation One area where the use of knowledge graphs has great potential to be used within the drug discovery domain is gene-disease prioritisation [5]. This crucial step in the drug discovery pipeline involves attempting to identify the causally implicated biological entity (typically a gene or protein) for a certain disease – which can be thought of as predicting a link between a gene and disease entity in a knowledge graph. Drugs can then be developed to modulate this target entity and ultimately treat or cure the disease. As such, this task better reflects a real-world use case of knowledge graphs within the drug discovery domain.

In order to conduct this experiment, we focus upon the Hetionet dataset and evaluate performance at predicting links between gene and disease entities. Hetionet has three distinct relation types connecting genes and diseases: associates (12 K examples), downregulates (7k examples) and upregulates (7k examples). For this work we, focus on the relation with the largest number of examples: associates. We partition the associates' edges using 10-fold cross validation, where 9 folds are used as part of the training data (along with the rest of the Hetionet dataset) and the remaining split used for solely test, with results presented as the average over all folds. One interesting aspect to consider is whether this could lead to trivial examples being present during training time, with which models could effectively cheat [47]. This is because if a gene and disease pair are linked via either a downregulates or an upregulates relation, they can also be linked via an associates relation (although this is only the case in a small number of pairs for the Hetionet

dataset), meaning that the model could see gene and disease entity pairs both in the train and test set. To assess the impact of this, we create two versions of the training dataset, a biased version (where gene-disease pairs from the train set linked via other relations in the train set are not removed) and an unbiased version (where the training set contains no gene-disease pairs linked via other relations in the test set).

Fig. 11 a highlights the distribution of AMR scores on the test set over the 10-folds after training on the biased and unbiased datasets. Interestingly the figure shows that the potential bias introduced by trivial examples does little to impact performance in this instance. This could well be due to the small number of both gene-disease pairs and the even smaller number which are linked by more than one relation type – however the wider issues of potential trivial examples in the context of drug discovery knowledge graphs is an area deserving continued attention.

Fig. 11 b shows the average Hits@k score over all folds using the unbiased data. One clear trend is that ComplEx performs poorly at the gene-disease prioritisation task, both compared to the other approaches and against its own performance in the general dataset split experiments. This may indicate ComplEx is more sensitive to low data quantity setups such as this. However, most models achieve performance on this gene-disease restricted setup which is comparable to that when run on the whole graph – further reinforcing the notion that the crucial task of gene-disease prioritisation should continue to be investigated in the context of KGs.

5. Discussion and conclusion

Overall observations Our results show RotatE is often the best performing model of the five on both datasets and throughout all the experiments. This reinforces previous similar findings [15,30] and shows RotatE to be a strong baseline in the context of drug discovery. Our results also highlight that older approaches like TransE can still be very competitive given an optimised training and hyperparameter setup. Regarding training setup choices, we found that NSSA and AdaGrad were often the best performing approaches and could serve as starting points for further comparisons. More generally, it can seen that KGE models are more than just architectures and should be considered in combination with their training setup and hyperparameter values.

This study has further demonstrated the effect of hyperparameters, with tuning providing a potentially large increase in performance over default baseline choices. After performing a detailed parameter search using two sampling methods, it can be seen that there is still a large variance in parameter values, even among top performing configurations for a given model. This highlights how performance seems rarely to be driven by single parameters, rather a nuanced combination of values is often required. This may suggest hand-crafted tuning is unlikely to result in optimal choices and a HPO strategy should be employed. There was also a marked difference in hyperparameters between the two KGs, despite them being from the same domain and containing similar entities – revealing how dataset dependent parameters can be. Additionally, we found a random search of parameter space (given enough repeats) to yield configurations which perform very closely to those from more principled approaches, whilst taking less time to do so.

We assessed model performance at target discovery by predicting links specifically between gene and disease entities. This showed that, although predictive performance was comparable to when measured on relations of all types, there were some differences. This suggests that researchers should not assume that model performance at the general link prediction task is indicative of performance in a more focused application. Additionally, we highlighted that performing HPO to optimise a single relation type can actually hurt downstream performance on that type in a limited data setting. The issue of potentially trivial examples being present deserves continued attention, especially in the context of drug discovery where the complexity of the underlying data could amplify the risks.

Reproducibility & fair comparison This work has shown how knowing model architecture alone, without training setup, hyperparameters and dataset splits, is probably insufficient to replicate results. To improve reproducibility, these should be more prominently reported alongside performance metrics. Additionally, to allow for fair comparisons to be made, new models should be assessed against well-tuned baseline approaches, with any changes introduced in training setup also being applied to competing methods. Also with the goal of fairer comparisons in mind, performance metrics should also be presented as averages over different random seeds and dataset splits in order to assess robustness.

Conclusions KGs are increasingly being used in the field of drug discovery to help address key challenges such as gene-disease prioritisation. In this work we have assessed how various factors, including training setup, hyperparameter choices, model parameter initialisation and different dataset splits, affect the performance of five KGE models on two real-world biomedical datasets. However, ultimately for these approaches to impact drug discovery, they need to be used in associated decision making. This in turn depends on the level of trust and understanding of the approaches. We hope that increased attention on such foundational aspects will improve rigour, reproducibility and understanding of factors influencing different tasks and contexts, thereby maximising the potential to improve drug discovery efforts.

Future work We plan to expand the set of models we assess to include a more diverse set of approaches, as well as exploring other suitable drug discovery datasets and focusing on more domain specific tasks such as drug repurposing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

References

[1] Morgan P, Brown DG, Lennard S, Anderton MJ, Barrett JC, Eriksson U, et al. Impact of a five-dimensional framework on R&Dproductivity at AstraZeneca. Nat Rev Drug Discov 2018;17(3):167.

[2] Himmelstein DS, Lizee A, Hessler C, Brueggeman L, Chen SL, Hadley D, Green A, Khankhanian P, Baranzini SE. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. Elife 2017;6:e26726.

[3] Walsh B, Mohamed SK, Nováček V. Biokg: a knowledge graph for relational learning on biological data. In: Proceedings of the 29th ACM international conference on information & knowledge management; 2020. p. 3173–80.

[4] Malas TB, Vlietstra WJ, Kudrin R, Starikov S, Charrout M, Roos M, et al. Drug prioritization using the semantic properties of a knowledge graph. Sci Rep 2019;9(1):1–10.

[5] Paliwal S, de Giorgio A, Neil D, Michel JB, Lacoste AM. Preclinical validation of therapeutic targets predicted by tensor factorization on heterogeneous graphs. Sci Rep 2020;10(1):1–19.

[6] Liu Y, Hildebrandt M, Joblin M, Ringsquandl M, Raissouni R, Tresp V. Neural multi-hop reasoning with logical rules on biomedical knowledge graphs. In: European semantic web conference. Springer; 2021. p. 375–91.

[7] Toutanova K, Chen D. Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd workshop on continuous vector space models and their compositionality; 2015. p. 57–66.

[8] Dettmers T, Minervini P, Stenetorp P, Riedel S. Convolutional 2D knowledge graph embeddings. In: Proceedings of the AAAI conference on artificial intelligence, vol. 32; 2018.

[9] Trouillon T, Welbl J, Riedel S, Gaussier É, Bouchard G. Complex embeddings for simple link prediction. In: International conference on machine learning (ICML); 2016.

[10] Yang B, Yih SWT, He X, Gao J, Deng L. Embedding entities and relations for learning and inference in knowledge bases. In: Proceedings of the international conference on learning representations (ICLR) 2015. Cornell University; 2015.

[11] Sun Z, Deng Z-H, Nie J-Y, Tang J. Rotate: knowledge graph embedding by relational rotation in complex space. In: International conference on learning representations; 2019.

[12] Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O. Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems; 2013. p. 2787–95.

[13] Wang Z, Zhang J, Feng J, Chen Z. Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI conference on artificial intelligence; 2014.

[14] Zhang C, Song D, Huang C, Swami A, Chawla NV. Heterogeneous graph neural network. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining; 2019. p. 793–803.

[15] 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 Trans Pattern Anal Mach Intell 2021. doi:10.1109/TPAMI.2021.3124805.

[16] Ruffinelli D, Broscheit S, Gemulla R. You can teach an old dog new tricks! on training knowledge graph embeddings. In: International conference on learning representations; 2019.

[17] Ji S, Pan S, Cambria E, Marttinen P, Philip SY. A survey on knowledge graphs: representation, acquisition, and applications. IEEE Trans Neural Netw Learn Syst 2022;33(2):494–514. doi:10.1109/TNNLS.2021.3070843.

[18] Wang Q, Mao Z, Wang B, Guo L. Knowledge graph embedding: a survey of approaches and applications. IEEE Trans Knowl Data Eng 2017;29(12):2724–2743.

[19] Zhang D, Yin J, Zhu X, Zhang C. Network representation learning: a survey. IEEE Trans Big Data 2018;6(1):3–28.

[20] Hamilton WL, Ying R, Leskovec J. Representation learning on graphs: methods and applications. IEEE Data Eng Bull 2017;40(3):52–74.

[21] Shchur O., Mumme M., Bojchevski A., Günnemann S. Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811058682018.

[22] Errica F, Podda M, Bacciu D, Micheli A. A fair comparison of graph neural networks for graph classification. In: International conference on learning representations (ICLR 2020); 2020.

[23] Dwivedi V.P., Joshi C.K., Laurent T., Bengio Y., Bresson X.. Benchmarking graph neural networks. arXiv preprint arXiv:200300982 2020.

[24] Hu W, Fey M, Zitnik M, Dong Y, Ren H, Liu B, et al. Open graph benchmark: datasets for machine learning on graphs. Adv Neural Inf Process Syst 2020;33:22118–33.

[25] 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. J Mach Learn Res 2021;22(82):1–6.

[26] Yue X, Wang Z, Huang J, Parthasarathy S, Moosavinasab S, Huang Y, Lin SM, Zhang W, Zhang P, Sun H. Graph embedding on biomedical networks: methods, applications and evaluations. Bioinformatics 2020;36(4):1241–51.

[27] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining; 2014. p. 701–10.

[28] Kipf T.N., Welling M.. Variational graph auto-encoders. arXiv preprint arXiv:161107308 2016.

[29] Su C, Tong J, Zhu Y, Cui P, Wang F. Network embedding in biomedical data science. Brief Bioinform 2020;21(1):182–97.

[30] Chang D, Balazevic I, Allen C, Chawla D, Brandt C, Taylor RA. Benchmark and best practices for biomedical knowledge graph embeddings. In: BioNLP; 2020. p. 167.

[31] Mohamed SK, Nounu A, Nováček V. Biological applications of knowledge graph embedding models. Brief Bioinform 2021;22(2):1679–93. doi:10.1093/bib/bbaa012.

[32] Zitnik M., Sosic R., Leskovec J.. BioSNAP datasets: stanford biomedical network dataset collection. http://snap.stanford.edu/biodata; 2018.

[33] Celebi R, Uyar H, Yasar E, Gumus O, Dikenelli O, Dumontier M. Evaluation of knowledge graph embedding approaches for drug-drug interaction prediction in realistic settings. BMC Bioinform 2019;20(1):1–14.

[34] Bonner S., Barrett I.P., Ye C., Swiers R., Engkvist O., Bender A., Hoyt C.T., Hamilton W.. A review of biomedical datasets relating to drug discovery: a knowledge graph perspective. arXiv preprint arXiv:210210062 2021.

[35] Gaudelet T, Day B, Jamasb AR, Soman J, Regep C, Liu G, et al. Utilizing graph machine learning within drug discovery and development. Brief Bioinform 2021;22(6):bbab159.

[36] Ioannidis V.N., Song X., Manchanda S., Li M., Pan X., Zheng D., Ning X., Zeng X., Karypis G.. Drkg - drug repurposing knowledge graph for COVID-19. https://github.com/gnn4dr/DRKG/; 2020.

[37] Breit A, Ott S, Agibetov A, Samwald M. Openbiolink: a benchmarking framework for large-scale biomedical link prediction. Bioinformatics 2020;36(13):4097–8. doi:10.1093/bioinformatics/btaa274.

[38] Rossi A, Barbosa D, Firmani D, Matinata A, Merialdo P. Knowledge graph embedding for link prediction: acomparative analysis. ACM Trans Knowl Discov Data (TKDD) 2021;15(2):1–49.

[39] Zheng S, Rao J, Song Y, Zhang J, Xiao X, Fang EF, et al. Pharmkg: a dedicated knowledge graph benchmark for bomedical data mining. Brief Bioinform 2020;22(4). doi:10.1093/bib/bbaa344.

[40] Berrendorf M., Faerman E., Vermue L., Tresp V.. On the ambiguity of rank based evaluation of entity alignment or link prediction methods. arXiv preprint arXiv:200206914 2020.

[41] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: an imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 2019;32:8026–37.

[42] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining; 2019. p. 2623–31.

[43] Kazemi SM, Poole D. Simple embedding for link prediction in knowledge graphs. In: Proceedings of the 32nd international conference on neural information processing systems; 2018. p. 4289–300.

[44] Bergstra J, Bardenet R, Bengio Y, Kégl B. Algorithms for hyper-parameter optimization. 25th annual conference on neural information processing systems (NIPS 2011), vol 24. Neural Information Processing Systems Foundation; 2011.

[45] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J Mach Learn Res 2012;13(2):281–305.

[46] Madhyastha PS, Jain R. On model stability as a function of random seed. In: Proceedings of the 23rd conference on computational natural language learning (CoNLL); 2019. p. 929–39.

[47] Rossi A, Matinata A. Knowledge graph embeddings: are relation-learning models learning relations?. EDBT/ICDT workshops; 2020.