Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models under a Unified Framework

Ali, Mehdi; Berrendorf, Max; Hoyt, Charles Tapley; Vermue, Laurent; Galkin, Mikhail; Sharifzadeh, Sahand; Fischer, Asja; Tresp, Volker; Lehmann, Jens

Published in:
Ieee Transactions on Pattern Analysis and Machine Intelligence

Link to article, DOI:
10.1109/TPAMI.2021.3124805

Publication date:
2022

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Galkin, M., Sharifzadeh, S., Fischer, A., Tresp, V., & Lehmann, J. (2022). 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*, *44*(12), 8825 - 8845. https://doi.org/10.1109/TPAMI.2021.3124805

Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models under a Unified Framework

Mehdi Ali, Max Berrendorf†, Charles Tapley Hoyt†, Laurent Vermue†, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann

Abstract—The heterogeneity in recently published knowledge graph embedding models' implementations, training, and evaluation has made fair and thorough comparisons difficult. To assess the reproducibility of previously published results, we re-implemented and evaluated 21 models in the PyKEEN software package. In this paper, we outline which results could be reproduced with their reported hyper-parameters, which could only be reproduced with alternate hyper-parameters, and which could not be reproduced at all, as well as provide insight as to why this might be the case.

We then performed a large-scale benchmarking on four datasets with several thousands of experiments and 24,804 GPU hours of computation time. We present insights gained as to best practices, best configurations for each model, and where improvements could be made over previously published best configurations. Our results highlight that the combination of model architecture, training approach, loss function, and the explicit modeling of inverse relations is crucial for a model's performance and is not only determined by its architecture. We provide evidence that several architectures can obtain results competitive to the state of the art when configured carefully. We have made all code, experimental configurations, results, and analyses available at https://github.com/pykeen/pykeen and https://github.com/pykeen/benchmarking

Index Terms—Knowledge Graph Embeddings, Link Prediction, Reproducibility, Benchmarking

1 Introduction

As the usage of knowledge graphs (KGs) becomes more widespread, their inherent incompleteness can pose a liability for typical downstream tasks that they support, e.g., question answering, dialogue systems, and recommendation systems [1]. Knowledge graph embedding models (KGEMs) present an avenue for predicting missing links. However, the following two major challenges remain in their application.

First, the reproduction of previously reported results turned out to be a major challenge — there are even examples of different results reported for the same combinations of KGEMs and datasets [2]. In some cases, the lack of availability of source code for KGEMs or the usage of different frameworks and programming languages inevitably introduces variability. In other cases, the lack of a precise specification of hyper-parameters introduces variability.

Second, the verification of the novelty of previously reported results remains difficult. It is often difficult to attribute the incremental improvements in performance reported with each new state of the art model to the model's architecture itself or instead to the training approach, hyper-parameter values, or specific prepossessing steps, e.g., the explicit modeling of inverse relations. It has been shown that baseline models can achieve competitive performance to more sophisticated ones when optimized appropriately [3], [2]. Additionally, the variety of implementations and interpretations of common evaluation metrics for link prediction makes a fair comparison to previous results difficult [4].

This paper makes two major contributions towards addressing these challenges:

1) We performed a reproducibility study in which we tried to replicate reported experimental results in the original papers (when sufficient information was provided).
2) We performed an extensive benchmark study on 21 KGEMs over four benchmark datasets in which we evaluated the models based on different hyper-parameter values, training approaches (i.e. training under the *local closed world assumption* and *stochastic local closed world assumption*), loss functions, optimizers, and the explicit modeling of inverse relations.

Previous studies have already investigated important aspects for a subset of models: Kadlec *et al.* [3] showed that a fine-tuned baseline (DistMult [5]) can outperform

†*Equal contribution.*
Mehdi Ali is affiliated with Smart Data Analytics (University of Bonn), Germany, & Fraunhofer IAIS, Sankt Augustin and Dresden, Germany.
Max Berrendorf is affiliated with Ludwig-Maximilians-Universität München, Munich, Germany.
Charles Tapley Hoyt is affiliated with Laboratory of Systems Pharmacology, Harvard Medical School, Boston, USA.
Laurent Vermue is affiliated with the Technical University of Denmark, Kongens Lyngby, Denmark.
Mikhail Galkin is affiliated with Mila & McGill University, Montreal, Canada
Sahand Sharifzadeh is affiliated with Ludwig-Maximilians-Universität München, Munich, Germany.
Asja Fischer is affiliated with the Ruhr University Bochum, Germany.
Volker Tresp is affiliated with Ludwig-Maximilians-Universität München & Siemens AG, Munich, Germany.
Jens Lehmann is affiliated with Smart Data Analytics (University of Bonn), Bonn, Germany, & Fraunhofer IAIS, Sankt Augustin and Dresden Germany.

more sophisticated models on FB15K. Akrami *et al.* [2], [6] examined the effect of removing faulty triples from KGs on the model's performance. Mohamed *et al.* [7] studied the influence of loss functions on the models' performances for a set of KGEMs. Concurrent to the work on this paper, Rufinelli *et al.* [8] performed a benchmarking study in which they investigated five knowledge graph embedding models. After describing their benchmarking [8], they called for a larger study that extends the search space and incorporates more sophisticated models. Our study answers this call and realizes a fair benchmarking by completely re-implementing KGEMs, training pipelines, loss functions, and evaluation metrics in a unified, open-source framework. Inspired by their findings, we have also included the cross entropy loss (CEL) function, which has been previously used by Kadlec *et al.* [3]. Our benchmarking can be considered as a superset of many previous benchmarkings — to the best of our knowledge, there exists no study of comparable breadth or depth. A further interesting study with a different focus is the work of Rossi *et al.* [9] in which they investigated the effect of the structural properties of KGs on models' performances, instead of focusing on the combinations of different model architectures, training approaches, and loss functions.

This article is structured as follows: in Section 2, we introduce our notation of KG and the link prediction task and introduce an exemplary KG to which we refer in examples throughout this paper. In Section 3, we present our definition of a KGEM and review the KGEMs that we investigated in our studies. In Section 4, we describe and discuss established evaluation metrics as well as a recently proposed one [10]. In Section 5, we introduce the benchmark datasets on which we conducted our experiments. In Section 6 and Section 7, we present our respective reproducibility and benchmarking studies. In Section 8, we investigate how well the investigated KGEMs can model symmetry, anti-symmetry, and composition patterns. Finally, we provide a discussion and an outlook for our future work in Section 9.

2 Knowledge Graphs

For a given set of entities $\mathcal{E}$ and set of relations $\mathcal{R}$, we consider a knowledge graph $\mathcal{K} \subseteq \mathbb{K} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ as a directed, multi-relational graph that comprises triples $(h, r, t) \in \mathcal{K}$ in which $h, t \in \mathcal{E}$ represent a triples' respective head and tail entities and $r \in \mathcal{R}$ represents its relationship. Figure 1 depicts an exemplary KG. The direction of a relationship indicates the roles of the entities, i.e., head or tail entity. For instance, in the triple *(Sarah, CEO_Of, Deutsche_Bank)*, *Sarah* is the head and *Deutsche_Bank* is the tail entity. KGs usually contain only true triples corresponding to available knowledge.

In contrast to triples in a KG, there are different philosophies, or *assumptions*, for the consideration of triples *not* contained in a KG [11], [12]. Under the closed world assumption (CWA), all triples that are not part of a KG are considered as false. Based on the example in Figure 1, the triple *(Sarah, lives_in, Germany)* is a false fact under the CWA since it is not part of the KG. Under the open world assumption (OWA), it is considered unknown as to whether triples that are not part of the KG are true or false. The construction of KGs

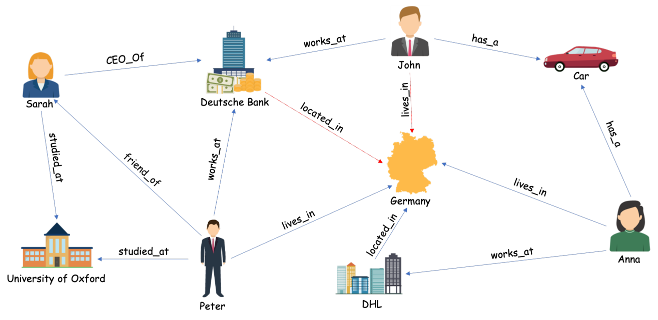

Fig. 1. Exemplary KG: nodes represent entities and edges their respective relations.

under the principles of the semantic web (and RDF) rely on the OWA as well as most of the relevant works to this paper [13], [11].

Because KGs are usually incomplete and noisy, several approaches have been developed to predict new links. In particular, the task of link prediction is defined as predicting the tail/head entities for $(h, r)/(r, t)$ pairs. For instance, given queries of the form *(Sarah, studied_at, ?)* or *(?, CEO_of, Deutsche Bank)*, the task is the correctly detect the entities that answer the query, i.e. *(Sarah, studied_at, **University of Oxford**)* and *(**Sarah**, CEO_of, Deutsche Bank)*. While classical approaches have relied on domain-specific rules to derive missing links, they usually require a large number of user-defined rules in order to generalize [11]. Alternatively, machine learning approaches learn to predict new links based on the set of existing ones. It has been shown that especially relational-machine learning methods are successful in predicting missing links and identifying incorrect ones, and recently knowledge graph embedding models have gained significant attention [11].

3 Knowledge Graph Embedding Models

Knowledge graph embedding models (KGEMs) learn latent vector representations of the entities $e \in \mathcal{E}$ and relations $r \in \mathcal{R}$ in a KG that best preserve its structural properties [1], [11], [14]. Besides for link prediction, they have been used for tasks such as entity disambiguation, and clustering as well as for downstream tasks such as question answering, recommendation systems, and relation extraction [1]. Figure 2 shows an embedding of the entities and relations in $\mathbb{R}^2$ from the KG from Figure 1.

Here, we define a KGEM as four components: an *interaction model*, a *training approach*, a *loss function*, and its usage of *explicit inverse relations*. This abstraction enables investigation of the effect of each component individually and in combination on each KGEMs' performance. Each are described in detail in their following respective subsections 3.1, 3.2, 3.3, and 3.4. We focus on *shallow* embedding approaches [15] in this work, i.e., matrix lookups represent the entity and relation encoders. Recently, several graph neural network (GNN)-based approaches for learning representations of KGs have been developed. GNNs encode entities and relations by neighbor aggregation. We refer interested readers to [14], [15]. Furthermore, learning

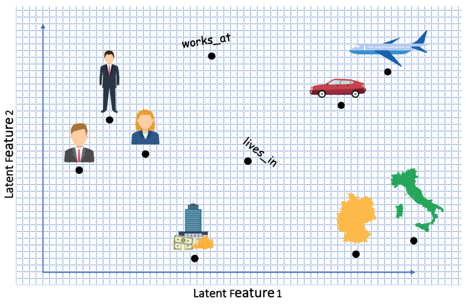

Fig. 2. An example embedding of the entities and relations from the knowledge graph portrayed by Figure 2

representation for temporal KGs has gained increased interest. Because learning representation for temporal KGs is a distinct line of research with its own benchmarking datasets, we do not discuss temporal KGEMs in this work. Instead, we refer interested readers to [16].

In this paper, we use a boldface lower-case letter $\mathbf{x}$ to denote a vector, $\|\mathbf{x}\|_p$ to represent its l_p norm, a boldface upper-case letter $\mathbf{X}$ to denote a matrix, and a fraktur-font upper-case letter $\mathfrak{X}$ to represent a three-mode tensor. Furthermore, we use $\odot$ to denote the Hadamard product $\odot : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$:

$$[\mathbf{a} \odot \mathbf{b}]_i = \mathbf{a}_i \cdot \mathbf{b}_i \tag{1}$$

Finally, we use $\overline{x}$ to denote the conjugate of a complex number $x \in \mathbb{C}$.

3.1 Interaction Models

An interaction model $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$ computes a real-valued score representing the plausibility of a triple $(h, r, t) \in \mathbb{K}$ given the embeddings for the entities and relations. In general, a larger score indicates a higher plausibility. The interpretation of the score value is model-dependent, and usually, it cannot be directly interpreted as a probability. We follow [1], [14] and categorize interaction models into *translational distance* based and *semantic matching* based interaction models. Translational distance interaction models compute the plausibility of triples based on a distance function, e.g., Euclidean distance between (projected) entities, and semantic similarity matching models exploit the similarity of the latent features usually induced by inner a product formulation.

3.1.1 Translational Distance Interaction Models

Unstructured Model The Unstructured Model (UM) [17] scores a triple by computing the distance between the head and tail entity

$$f(h,t) = -\|\mathbf{h} - \mathbf{t}\|_2^2 \ , \tag{2}$$

where $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$ are the embeddings of head and tail entity, respectively. A small distance between these embeddings indicates a plausible triple. In the UM, relations are not considered, and therefore, it cannot distinguish between different relationship types. However, the model can be beneficial for learning embeddings for KGs that contain only a single relationship type or only equivalent relationship types, e.g. *GrandmotherOf* and *GrandmaOf*. Moreover, it may serve as a baseline to interpret the performance of relation-aware models.

Structured Embedding Structured Embedding (SE) [18] models each relation by two matrices $\mathbf{M}_r^h, \mathbf{M}_r^t \in \mathbb{R}^{d \times d}$ that perform relation-specific projections of the head and tail embeddings:

$$f(h,r,t) = -\|\mathbf{M}_r^h \mathbf{h} - \mathbf{M}_r^t \mathbf{t}\|_1 \ . \tag{3}$$

As before, $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$ are the embeddings of head and tail entity, respectively. By employing different projections for the embeddings of the head and tail entities, SE explicitly distinguishes between the subject- and object-role of an entity.

TransE TransE [19] models relations as a translation of head to tail embeddings, i.e. $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. Thus, the interaction model is defined as:

$$f(h,r,t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_p \ , \tag{4}$$

with $p \in \{1, 2\}$ is a hyper-parameter. A major advantage of TransE is its computational efficiency which enables its usage for large scale KGs. However, it inherently cannot model 1-N, N-1, and N-M relations: assume $(h, r, t_1), (h, r, t_2) \in \mathcal{K}$, then the model adapts the embeddings in order to ensure $\mathbf{h} + \mathbf{r} \approx \mathbf{t}_1$ and $\mathbf{h} + \mathbf{r} \approx \mathbf{t}_2$ which results in $\mathbf{t}_1 \approx \mathbf{t}_2$.

TransH TransH [20] is an extension of TransE that specifically addresses the limitations of TransE in modeling 1-N, N-1, and N-M relations. In TransH, each relation is represented by a hyperplane, or more specifically a normal vector of this hyperplane $\mathbf{w}_r \in \mathbb{R}^d$, and a vector $\mathbf{d}_r \in \mathbb{R}^d$ that lies in the hyperplane. To compute the plausibility of a triple $(h, r, t) \in \mathbb{K}$, the head embedding $\mathbf{h} \in \mathbb{R}^d$ and the tail embedding $\mathbf{t} \in \mathbb{R}^d$ are first projected onto the relation-specific hyperplane: $\mathbf{h}_r = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r$ and $\mathbf{t}_r = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r$. Then, the projected embeddings are used to compute the score for the triple (h, r, t):

$$f(h,r,t) = -\|\mathbf{h}_r + \mathbf{d}_r - \mathbf{t}_r\|_2^2 \ . \tag{5}$$

TransR TransR [21] is an extension of TransH that explicitly considers entities and relations as different objects and therefore represents them in different vector spaces. For a triple $(h, r, t) \in \mathbb{K}$, the entity embeddings, $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$, are first projected into the relation space by means of a relation-specific projection matrix $\mathbf{M}_r \in \mathbb{R}^{k \times d}$: $\mathbf{h}_r = \mathbf{M}_r \mathbf{h}$ and $\mathbf{t}_r = \mathbf{M}_r \mathbf{t}$. Finally, the score of the triple (h, r, t) is computed:

$$f(h,r,t) = -\|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_2^2 \tag{6}$$

where $\mathbf{r} \in \mathbb{R}^k$.

TransD TransD [22] is an extension of TransR that, like TransR, considers entities and relations as objects living in different vector spaces. However, instead of performing the same relation-specific projection for all entity embeddings, entity-relation-specific projection matrices $\mathbf{M}_{r,h}, \mathbf{M}_{t,h} \in \mathbb{R}^{k \times d}$ are constructed. To do so, all head entities, tail entities, and relations are represented by two vectors, $\mathbf{h}, \mathbf{h}_p, \mathbf{t}, \mathbf{t}_p \in \mathbb{R}^d$ and $\mathbf{r}, \mathbf{r}_p \in \mathbb{R}^k$, respectively. The first set of embeddings is used for calculating the entity-relation-specific projection matrices: $\mathbf{M}_{r,h} = \mathbf{r}_p \mathbf{h}_p^T + \tilde{\mathbf{I}}$ and $\mathbf{M}_{r,t} = \mathbf{r}_p \mathbf{t}_p^T + \tilde{\mathbf{I}}$, where $\tilde{\mathbf{I}} \in \mathbb{R}^{k \times d}$ is a $k \times d$ matrix with ones on the diagonal and zeros elsewhere. Next, $\mathbf{h}$ and $\mathbf{t}$ are projected into the relation space by means of the constructed projection matrices: $\mathbf{h}_r = \mathbf{M}_{r,h}\mathbf{h}$ and $\mathbf{t}_r = \mathbf{M}_{r,t}\mathbf{t}$. Finally, the plausibility score for $(h, r, t) \in \mathbb{K}$ is given by:

$$f(h, r, t) = -\|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_2^2 \ . \qquad (7)$$

RotatE RotatE [23] models relations as rotations from head to tail entities in the complex space: $\mathbf{t} = \mathbf{h} \odot \mathbf{r}$, where $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$ and $|r_i| = 1$, that is the complex elements of $\mathbf{r}$ are restricted to have a modulus of one. Because of the latter, r_i can be represented as $e^{i\theta_{r,i}}$, which corresponds to a counterclockwise rotation by $\theta_{r,i}$ radians. The interaction model is then defined as:

$$f(h, r, t) = -\|\mathbf{h} \odot \mathbf{r} - \mathbf{t}\| \ , \qquad (8)$$

which allows to model *symmetry*, *antisymmetry*, *inversion*, and *composition* [23].

MuRE MuRE [24] is the Euclidean counterpart of MuRP, a hyperbolic interaction model that is capable of effectively modeling hierarchies in KG. Its interaction model involves a distance function:

$$f(h, r, t) = -\|\mathbf{R}\mathbf{h} - \mathbf{t} + \mathbf{r}\|_2^2 + \mathbf{b_h} + \mathbf{b_t} \qquad (9)$$

where the head entity is transformed by the diagonal matrix $\mathbf{R} \in \mathbf{R}^{d\times d}$ and the tail entity by the relation r. $\mathbf{b_h}$ and $\mathbf{b_t}$ represent scalar offsets.

KG2E KG2E [25] aims to explicitly model (un)certainties in entities and relations (e.g. influenced by the number of triples observed for these entities and relations). Therefore, entities and relations are represented by probability distributions, in particular by multi-variate Gaussian distributions $\boldsymbol{\mathcal{N}}_i(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ where the mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ denotes the position in the vector space and the diagonal variance $\boldsymbol{\Sigma}_i \in \mathbb{R}^{d\times d}$ models the uncertainty. Inspired by the TransE model, relations are modeled as transformations from head to tail entities: $\boldsymbol{\mathcal{H}} - \boldsymbol{\mathcal{T}} \approx \boldsymbol{\mathcal{R}}$ where $\boldsymbol{\mathcal{H}} \sim \boldsymbol{\mathcal{N}}_h(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$, $\boldsymbol{\mathcal{H}} \sim \boldsymbol{\mathcal{N}}_t(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, $\boldsymbol{\mathcal{R}} \sim \boldsymbol{\mathcal{P}}_r = \boldsymbol{\mathcal{N}}_r(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$ and $\boldsymbol{\mathcal{H}} - \boldsymbol{\mathcal{T}} \sim \boldsymbol{\mathcal{P}_e} = \boldsymbol{\mathcal{N}_{h-t}}(\boldsymbol{\mu_h} - \boldsymbol{\mu_t}, \boldsymbol{\Sigma_h} + \boldsymbol{\Sigma_t})$ (since head and tail entities are considered to be independent with regards to the relations). The interaction model measures the similarity between $\boldsymbol{\mathcal{P}_e}$ and $\boldsymbol{\mathcal{P}_r}$ by means of the Kullback-Leibler (KL) divergence:

$$\begin{aligned} f(h, r, t) &= \mathcal{D}_{\mathcal{KL}}(\boldsymbol{\mathcal{P}}_e, \boldsymbol{\mathcal{P}}_r) \\ &= \frac{1}{2}\Big\{tr(\boldsymbol{\Sigma}_r^{-1}\boldsymbol{\Sigma}_e) + (\boldsymbol{\mu}_r - \boldsymbol{\mu}_e)^T\boldsymbol{\Sigma}_r^{-1}(\boldsymbol{\mu}_r - \boldsymbol{\mu}_e) \\ &\quad -log(\frac{det(\boldsymbol{\Sigma}_e)}{det(\boldsymbol{\Sigma}_r)}) - d\Big\} \ . \end{aligned} \qquad (10)$$

Besides the asymmetric KL divergence, the authors propose a symmetric variant which uses the expected likelihood.

3.1.2 *Semantic Matching Interaction Models*

RESCAL RESCAL [26] is a bilinear model that models entities as vectors and relations as matrices. The relation matrices $\mathbf{W}_r \in \mathbb{R}^{d\times d}$ contain weights $w_{i,j}$ that capture the amount of interaction between the i-th latent factor of $\mathbf{h} \in \mathbb{R}^d$ and the j-th latent factor of $\mathbf{t} \in \mathbb{R}^d$ [11], [26]. Thus, the plausibility score of $(h, r, t) \in \mathbb{K}$ is given by:

$$f(h, r, t) = \mathbf{h}^T\mathbf{W}_r\mathbf{t} = \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}^{(r)} h_i t_j \qquad (11)$$

DistMult DistMult [5] is a simplification of RESCAL where the relation matrices $\mathbf{W}_r \in \mathbb{R}^{d\times d}$ are restricted to diagonal matrices:

$$f(h, r, t) = \mathbf{h}^T\mathbf{W}_r\mathbf{t} = \sum_{i=1}^{d} \mathbf{h}_i \cdot diag(\mathbf{W}_r)_i \cdot \mathbf{t}_i \ . \qquad (12)$$

Because of its restriction to diagonal matrices DistMult is computational more efficient than RESCAL, but at the same time less expressive. For instance, it is not able to model anti-symmetric relations, since $f(h, r, t) = f(t, r, h)$.

ComplEx ComplEx [27] is an extension of DistMult that uses complex valued representations for the entities and relations. Entities and relations are represented as vectors $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$, and the plausibility score is computed using the Hadamard product:

$$f(h, r, t) = Re(\mathbf{h} \odot \mathbf{r} \odot \mathbf{t}) \qquad (13)$$

where $Re(\mathbf{x})$ denotes the real component of the complex valued vector $\mathbf{x}$. Because the Hadamard product is not commutative in the complex space, ComplEx can model anti-symmetric relations in contrast to DistMult.

QuatE QuatE [28] learns hypercomplex valued representations (quaternion embeddings) for entities and relations, i.e., $\mathbf{e_i}, \mathbf{r_j} \in \mathbb{H}^d$. Hypercomplex representations extend complex representations by representing each number with one real and three imaginary components. In QuatE, relations are modelled as rotations in the hypercomplex space. More precisely, the relation is used to rotate the head entity: $\mathbf{h_r} = \mathbf{h} \otimes \mathbf{r}$, where in this context $\otimes$ represents the Hamilton product. The final score is obtained by computing the inner product between the rotated head and the the tail entity:

$$f(h, r, t) = \mathbf{h_r} \cdot \mathbf{t} \qquad (14)$$

In contrast to ComplEx, QuatE is capable of modeling *composition* patterns.

SimplE SimplE [29] is an extension of **canonical polyadic (CP)** [29], one of the early tensor factorization approaches. In CP, each entity $e \in \mathcal{E}$ is represented by two vectors $\mathbf{h}_e, \mathbf{t}_e \in \mathbb{R}^d$ and each relation by a single vector $\mathbf{r} \in \mathbb{R}^d$. Depending whether an entity participates in a triple as the head or tail entity, either $\mathbf{h}_e$ or $\mathbf{t}_e$ is used. Both entity representations are learned independently, i.e. observing a triple (e_1, r, e_2), the method only updates $\mathbf{h}_{e_1}$ and $\mathbf{t}_{e_2}$. In contrast to CP, SimplE introduces for each relation r the inverse relation r', and formulates the interaction model based on both:

$$f(h, r, t) = \frac{1}{2}\left(\langle \mathbf{h}_{e_i}, \mathbf{r}, \mathbf{t}_{e_j} \rangle + \langle \mathbf{h}_{e_j}, \mathbf{r}', \mathbf{t}_{e_i} \rangle\right) \ . \qquad (15)$$

Therefore, for each triple $(e_1, r, e_2) \in \mathbb{K}$, both $\mathbf{h}_{e_1}$ and $\mathbf{t}_{e_2}$ as well as $\mathbf{h}_{e_2}$ and $\mathbf{t}_{e_1}$ are updated [29].

TuckER TuckER [30] is a linear model that is based on the tensor factorization method Tucker [31] in which a three-mode tensor $\mathfrak{X} \in \mathbb{R}^{I\times J\times K}$ is decomposed into a set of factor matrices $\mathbf{A} \in \mathbb{R}^{I\times P}$, $\mathbf{B} \in \mathbb{R}^{J\times Q}$, and $\mathbf{C} \in \mathbb{R}^{K\times R}$ and a core tensor $\mathfrak{Z} \in \mathbb{R}^{P\times Q\times R}$ (of lower rank): $\mathfrak{X} \approx \mathfrak{Z} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$, where $\times_n$ is the tensor product, with n denoting along which mode the tensor product is computed. In TuckER, a KG is considered as a binary tensor which is factorized using the Tucker factorization where

$\mathbf{E} = \mathbf{A} = \mathbf{C} \in \mathbb{R}^{n_e \times d_e}$ denotes the entity embedding matrix, $\mathbf{R} = \mathbf{B} \in \mathbb{R}^{n_r \times d_r}$ represents the relation embedding matrix, and $\mathfrak{W} = \mathfrak{Z} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is the *core tensor* that indicates the extent of interaction between the different factors. The interaction model is defined as:

$$f(h,r,t) = \mathfrak{W} \times_1 \mathbf{h} \times_2 \mathbf{r} \times_3 \mathbf{t} \ , \tag{16}$$

where $\mathbf{h}, \mathbf{t}$ correspond to rows of $\mathbf{E}$ and $\mathbf{r}$ to a row of $\mathbf{R}$.

ProjE ProjE [32] is a neural network-based approach with a *combination* and a *projection* layer. The interaction model first combines h and r by a combination operator [32]: $\mathbf{h} \otimes \mathbf{r} = \mathbf{D}_e\mathbf{h} + \mathbf{D}_r\mathbf{r} + \mathbf{b}_c$, where $\mathbf{D}_e, \mathbf{D}_r \in \mathbb{R}^{k \times k}$ are diagonal matrices which are used as shared parameters among all entities and relations, and $\mathbf{b}_c \in \mathbb{R}^k$ represents the candidate bias vector shared across all entities. Next, the score for the triple $(h,r,t) \in \mathbb{K}$ is computed:

$$f(h,r,t) = g(\mathbf{t}\ z(\mathbf{h} \otimes \mathbf{r}) + \mathbf{b}_p) \ , \tag{17}$$

where g and z are activation functions, and $\mathbf{b}_p$ represents the shared projection bias vector.

HolE Holographic embeddings (HolE) [33] make use of the circular correlation operator to compute interactions between latent features of entities and relations:

$$f(h,r,t) = \sigma(\mathbf{r}^T(\mathbf{h} \star \mathbf{t})) \ . \tag{18}$$

where the circular correlation $\star : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is defined as $[\mathbf{a} \star \mathbf{b}]_i = \sum_{k=0}^{d-1} \mathbf{a}_k * \mathbf{b}_{(i+k)\ mod\ d}$. By using the correlation operator each component $[\mathbf{h} \star \mathbf{t}]_i$ represents a sum over a fixed partition over pairwise interactions. This enables the model to put semantic similar interactions into the same partition and share weights through $\mathbf{r}$. Similarly irrelevant interactions of features could also be placed into the same partition which could be assigned a small weight in $\mathbf{r}$.

ERMLP ERMLP [34] is a multi-layer perceptron based approach that uses a single hidden layer and represents entities and relations as vectors. In the input-layer, for each triple the embeddings of head, relation, and tail are concatenated and passed to the hidden layer. The output-layer consists of a single neuron that computes the plausibility score of the triple:

$$f(h,r,t) = \mathbf{w}^T g(\mathbf{W}[\mathbf{h};\mathbf{r};\mathbf{t}]), \tag{19}$$

where $\mathbf{W} \in \mathbb{R}^{k \times 3d}$ represents the weight matrix of the hidden layer, $\mathbf{w} \in \mathbb{R}^k$, the weights of the output layer, and g denotes an activation function such as the hyperbolic tangent.

Neural Tensor Network The Neural Tensor Network (NTN) [35] uses a bilinear tensor layer instead of a standard linear neural network layer:

$$f(h,r,t) = \mathbf{u}_r^T \cdot \tanh(\mathbf{h}\mathfrak{W}_r\mathbf{t} + \mathbf{V}_r[\mathbf{h};\mathbf{t}] + \mathbf{b}_r) \ , \tag{20}$$

where $\mathfrak{W}_r \in \mathbb{R}^{d \times d \times k}$ is the relation specific tensor, and the weight matrix $\mathbf{V}_r \in \mathbb{R}^{k \times 2d}$, the bias vector $\mathbf{b}_r$, and the weight vector $\mathbf{u}_r \in \mathbb{R}^k$ are the standard parameters of a neural network, which are also relation specific. The result of the tensor product $\mathbf{h}\mathfrak{W}_r\mathbf{t}$ is a vector $\mathbf{x} \in \mathbb{R}^k$ where each entry x_i is computed based on the slice i of the tensor $\mathfrak{W}_r$: $\mathbf{x}_i = \mathbf{h}\mathfrak{W}_r^i\mathbf{t}$ [35]. As indicated by the interaction model, NTN defines for each relation a separate neural network which makes the model very expressive, but at the same time computationally expensive.

ConvKB ConvKB [36] uses a convolutional neural network (CNN) whose feature maps capture global interactions of the input. Each triple $(h,r,t) \in \mathbb{K}$ is represented as a input matrix $\mathbf{A} = [\mathbf{h};\mathbf{r};\mathbf{t}] \in \mathbb{R}^{d \times 3}$ in which the columns represent the embeddings for h, r and t. In the convolution layer, a set of convolutional filters $\boldsymbol{\omega}_i \in \mathbb{R}^{1 \times 3}, i = 1, \ldots, \tau$, are applied on the input in order to compute for each dimension global interactions of the embedded triple. Each $\boldsymbol{\omega}_i$ is applied on every row of $\mathbf{A}$ creating a feature map $\mathbf{v}_i = [v_{i,1}, ..., v_{i,d}] \in \mathbb{R}^d$:

$$\mathbf{v}_i = g(\boldsymbol{\omega}_j \mathbf{A} + \mathbf{b}) \ , \tag{21}$$

where $\mathbf{b} \in \mathbb{R}$ denotes a bias term and g an activation function which is employed element-wise. Based on the resulting feature maps $\mathbf{v}_1, \ldots, \mathbf{v}_\tau$, the plausibility score of a triple is given by:

$$f(h,r,t) = [\mathbf{v}_i; \ldots; \mathbf{v}_\tau] \cdot \mathbf{w} \ , \tag{22}$$

where $[\mathbf{v}_i; \ldots; \mathbf{v}_\tau] \in \mathbb{R}^{\tau d \times 1}$ and $\mathbf{w} \in \mathbb{R}^{\tau d \times 1}$ is a shared weight vector. ConvKB may be seen as a restriction of ER-MLP with a certain weight sharing pattern in the first layer.

ConvE ConvE [37] is a CNN-based approach. For each triple (h,r,t), the input to ConvE is a matrix $\mathbf{A} \in \mathbb{R}^{2 \times d}$ where the first row of $\mathbf{A}$ represents $\mathbf{h} \in \mathbb{R}^d$ and the second row represents $\mathbf{r} \in \mathbb{R}^d$. $\mathbf{A}$ is reshaped to a matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$ where the first $m/2$ half rows represent $\mathbf{h}$ and the remaining $m/2$ half rows represent $\mathbf{r}$. In the convolution layer, a set of *2-dimensional* convolutional filters $\Omega = \{\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_i \in \mathbb{R}^{r \times c}\}$ are applied on $\mathbf{B}$ that capture interactions between $\mathbf{h}$ and $\mathbf{r}$. The resulting feature maps are reshaped and concatenated in order to create a feature vector $\mathbf{v} \in \mathbb{R}^{|\Omega|rc}$. In the next step, $\mathbf{v}$ is mapped into the entity space using a linear transformation $\mathbf{W} \in \mathbb{R}^{|\Omega|rc \times d}$, that is $\mathbf{e}_{h,r} = \mathbf{v}^T\mathbf{W}$. The score for the triple $(h,r,t) \in \mathbb{K}$ is then given by:

$$f(h,r,t) = \mathbf{e}_{h,r}\mathbf{t} \ . \tag{23}$$

Since the interaction model can be decomposed into $f(h,r,t) = \langle f'(\mathbf{h},\mathbf{r}), \mathbf{t} \rangle$, the model is particularly designed to 1-N scoring, i.e. efficient computation of scores for (h,r,t) for fixed h, r and many different t.

3.2 Training Approaches

Because most KGs contain only positive examples, we require training approaches involving techniques such as negative sampling to avoid over-generalization to true facts. Here, we describe two common training approaches found in the literature: the local closed world assumption (LCWA) and the stochastic local closed world assumption (sLCWA). It should be noted that the LCWA and the sLCWA do not affect the evaluation.

3.2.1 *Local closed world assumption*

The LCWA was introduced by [34] and used in subsequent works as an approach to generate negative examples during training [37], [30]. In this setting, for any triple $(h,r,t) \in \mathcal{K}$ that has been observed, a set $\mathcal{T}^-(h,r)$ of negative examples is created by considering all triples $(h,r,t_i) \notin \mathcal{K}$ as false. Therefore, for our exemplary KG (Figure 1) for the

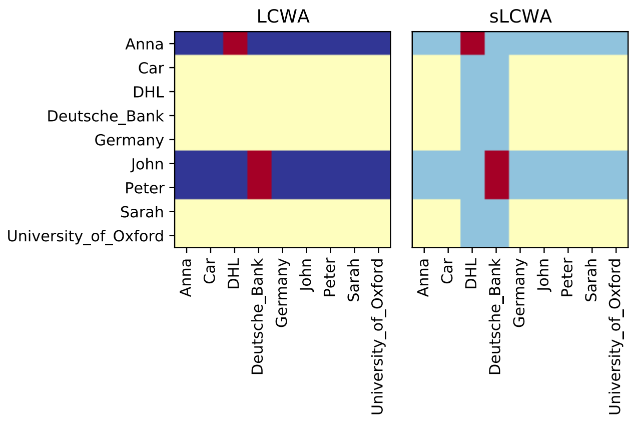

Fig. 3. Visualization of different training approaches for the relation `works_at` in the KG in Figure 1. Red color indicates positive examples, i.e. true triples present in the KG. Dark blue color denotes triples used as negative examples in LCWA. Light blue color sampling candidates for negative examples in sLCWA. Yellow color indicates triples that are not considered.

pair *(Peter, works_at)*, the triple *(Peter, works_at, DHL)* is a false fact since for this pair only the triple *(Peter, works_at, Deutsche Bank)* is part of the KG. Similarly, we can construct $\mathcal{H}^-(r,t)$ based on all triples $(h_i,r,t) \notin \mathcal{K}$, or $\mathcal{R}^-(h,t)$ based on the triples $(h,r_i,t) \notin \mathcal{K}$. Constructing $\mathcal{R}^-(h,t)$ is a popular choice in visual relation detection domain [38], [39]. However, most of the works in knowledge graph modeling construct only $\mathcal{T}^-(h,r)$ as the set of negative examples, and in the context of this work refer to $\mathcal{T}^-(h,r)$ as the set of negatives examples when speaking about LCWA.

3.2.2 *Stochastic local closed world assumption*

Under the stochastic local closed world assumption (sLCWA), instead of considering all possible triples $(h,r,t_i) \notin \mathcal{K}$, $(h_i,r,t) \notin \mathcal{K}$ or $(h,r_i,t) \notin \mathcal{K}$ as false, we randomly take samples of these sets.

Two common approaches for generating negative samples are uniform negative sampling (UNS) [19] and Bernoulli negative sampling (BNS) [20] in which negative triples are created by corrupting a positive triple $(h,r,t) \in \mathcal{K}$ by replacing either h or t. We denote with $\mathcal{N}$ the set of all potential negative triples:

$$\mathcal{T}(h,r) = \{(h,r,t') \mid t' \in \mathcal{E} \wedge t' \neq t\} \tag{24}$$
$$\mathcal{H}(r,t) = \{(h',r,t) \mid h' \in \mathcal{E} \wedge h' \neq h\} \tag{25}$$
$$\mathcal{N} = \bigcup_{(h,r,t)\in\mathcal{K}} \mathcal{T}(h,r) \cup \mathcal{H}(r,t) \ . \tag{26}$$

Theoretically, we would need to exclude all positive triples from this set of candidates for negative triples, i.e., $\mathcal{N}^- = \mathcal{N} \setminus \mathcal{K}$. In practice, however, since usually $|\mathcal{N}| \gg |\mathcal{K}|$, the likelihood of generating a false negative is rather low. Therefore, the additional filter step is often omitted to lower computational cost. It should be taken into account that a corrupted triple that is *not part* of the KG can represent a true fact.

UNS and BNS differ in the way they define sample weights for (h',r,t) or (h,r,t'):

Uniform negative sampling With uniform negative sampling (UNS) [19], the first step is to randomly (uniformly) determine whether h or t shall be corrupted for a positive triple $(h,r,t) \in \mathcal{K}$. Afterwards, an entity $e \in \mathcal{E}$ is uniformly sampled and selected as the corrupted head/tail entity.

Bernoulli negative sampling With Bernoulli negative sampling (BNS) [20], the probability of corrupting h or t in $(h,r,t) \in \mathcal{K}$ is determined by the property of the relation r: if the relation is a *one-to-many* relation (e.g. *motherOf*), BNS assigns a higher probability to replace h, and if it is a *many-to-one* relation (e.g. *bornIn*) it assigns a higher probability to replace t. More precisely, for each relation $r \in \mathcal{R}$ the average number of tails per head (*tph*) and heads per tail (*hpt*) are first computed. These statistics are then used to define a Bernoulli distribution with parameter $\frac{tph}{tph+hpt}$. For a triple $(h,r,t) \in \mathcal{K}$ the head is corrupted with probability $\frac{tph}{tph+hpt}$ and the tail with probability $\frac{hpt}{tph+hpt}$. The described approach reduces the chance of creating corrupted triples that represent true facts [20].

3.3 Loss Functions

The loss function can have a significant influence on the performance of KGEMs [7]. In the following, we describe *pointwise*, *pairwise*, and *setwise* loss functions that have been frequently be used within KGEMs. For additional discussion and a slightly different categorization we refer to the work of Mohamed *et al.* [7].

3.3.1 *Pointwise Loss Functions*

Let f denote the interaction model of a KGEM. With t_i, we denote a triple (i.e. $t_i \in \mathbb{K}$), and with $l_i \in \{0,1\}$ or $\hat{l}_i \in \{-1,1\}$ its corresponding label, where 1 corresponds to the label of the positive triples, and 0 / -1 to the label of the negative triples. Pointwise loss functions compute an independent loss term for each triple-label pair, i.e. for a batch $B = \{(t_i,l_i)\}_{i=1}^{|B|}$, the loss is given as

$$\mathcal{L} = \frac{1}{|B|} \sum_{(t_i,l_i)\in B} L(t_i,l_i) \tag{27}$$

In the following, we describe four different pointwise losses: The *square error loss*, *binary cross entropy loss (BCEL)*, *pointwise hinge loss*, and *logistic loss*.

Square Error Loss The square error loss function computes the squared difference between the predicted scores and the labels $l_i \in \{0,1\}$ [7]:

$$L(t_i,l_i) = \frac{1}{2}(f(t_i) - l_i)^2 \tag{28}$$

The squared error loss strongly penalizes predictions that deviate considerably from the labels, and is usually used for regression problems. For simple models it often permits more efficient optimization algorithms involving analytical solutions of sub-problems, e.g. the Alternating Least Squares algorithm used by [26].

Binary cross entropy loss The binary cross entropy loss is defined as [37]:

$$\begin{aligned} L(t_i,l_i) = &-(l_i \cdot \log(\sigma(f(t_i))) \\ &+ (1-l_i) \cdot \log(1-\sigma(f(t_i)))), \end{aligned} \tag{29}$$

where $l_i \in \{0,1\}$ and σ represents the logistic sigmoid function. Thus, the problem is framed as a binary classification problem of triples, where the model's outputs are regarded as logits. The loss is not well-suited for translational distance models because these models produce a negative distance as score and cannot produce positive model outputs. ConvE and TuckER were originally trained in a multi-class setting using the binary cross entropy loss where each (h,r)-pair has been classified against $e \in \mathcal{E}$ simultaneously, i.e., if $|\mathcal{E}| = n$, the label vector for each (h,r)-pair has n entries indicating whether the triple (h,r,e_i) is (not) part of the KG, and along each dimension of the label vector a binary classification is performed. It should be noted that there exist different implementation variants of the binary cross entropy loss that address numerical stability. ConvE and TuckER employed a numerically unstable variant, and in the context of this work, we refer to this variant when referring to the binary cross entropy loss.

Pointwise Logistic Loss/Softplus loss An alternative, but equivalent formulation of the binary cross entropy loss is the pointwise logistic loss (or Softplus loss (SPL)):

$$L(t_i, l_i) = \log(1 + \exp(-\hat{l}_i \cdot f(t_i)) \tag{30}$$

where $\hat{l}_i \in \{-1,1\}$ [7]. It has been used to train ComplEx, ConvKB, and SimplE. We consider both variants separately because both have been used in different model implementations, and their implementation details might yield different results (e.g., to numerical stability).

Pointwise Hinge Loss The pointwise hinge loss sets the score of positive examples larger than a margin parameter λ while reducing the scores of negative examples to values below $-\lambda$:

$$L(t_i, l_i) = \max(0, \lambda - \hat{l}_i \cdot f(t_i)) \tag{31}$$

where $\hat{l}_i \in \{-1,1\}$. The loss penalizes scores of positive examples which are smaller than λ, but does not impose any restriction on values $> \lambda$. Similarly, negative scores larger than $-\lambda$ contribute to the loss, whereas all values smaller than $-\lambda$ do not have any loss contribution [7]. Thereby, the model is not encouraged to further optimize triples which are already predicted well enough (according to the margin parameter λ).

3.3.2 *Pairwise Loss Functions*

Next, we describe widely applied pairwise loss functions that are used within KGEMs, namely the *pairwise hinge loss* and the *pairwise logistic loss*. They both compare the scores of a positive triple t^+ and a negative triple t^-. The negative triple in a pair is usually obtained by corrupting the positive one. Thus, the pairs often share common head or tail entities and relations. For a batch of pairs $B = \{(t_i^+, t_i^-)\}_{i=1}^{|B|}$, the loss is given as

$$\mathcal{L} = \frac{1}{|B|} \sum_{(t_i^+, t_i^-) \in B} L(f(t_i^-) - f(t_i^+)) \ . \tag{32}$$

Hence, the loss function evaluates the difference in scores $\Delta = f(t_i^-) - f(t_i^+)$ between a positive and a negative triple, rather than their absolute scores. This is in accordance to the OWA assumption, where we do not assume to have negative labels, but just "less positive" ones.

Pairwise Hinge Loss/Margin ranking loss The pairwise hinge loss or margin ranking loss (MRL) is given by

$$L(\Delta) = \max(0, \lambda + \Delta) \ . \tag{33}$$

Pairwise Logistic Loss The pairwise logistic loss is defined as [7]:

$$L(\Delta) = \log(1 + \exp(\Delta)) \ . \tag{34}$$

Thus, it can be seen as a soft-margin formulation of the pairwise hinge loss with a margin of zero.

3.3.3 *Setwise Loss Functions*

Setwise loss functions neither compare individual scores, or pairs of them, but rather more than two triples' scores. Here, we describe the self-adversarial negative sampling loss (NSSAL) and the cross entropy loss (CEL) as examples of such loss functions that have been applied within KGEMs [23], [7].

Self-adversarial negative sampling loss The Self-adversarial negative sampling loss (NSSAL) addresses the limitation that many negative examples are trivial and do not provide helpful information. The authors of [23] propose to overcome this limitation by sampling negative samples according to the scores predicted by the interaction model [23]:

$$p((h'_i, r, t'_i)|(h_i, r_i, t_i)) = \frac{\exp(\alpha f(h'_i, r, t'_i))}{\sum_{j=1}^{n} \exp(\alpha f(h'_j, r, t'_j))} \ , \tag{35}$$

where $(h_i, r_i, t_i) \in \mathcal{K}$ denotes a true triple, $\{(h'_i, r, t'_i)\}_{i=1}^{K}$ it's set of negative samples generated, and $\alpha \in \mathbb{R}$ a temperature parameter. Because sampling from this distribution may be computationally expensive, the probabilities obtained by Equation 35 are used to weight the generated negative examples in the loss function [23].

$$\begin{aligned} \mathcal{L} = & -\log(\sigma(\gamma + f(h,r,t))) \\ & - \sum_{i=1}^{K} p((h',r,t')) \cdot \log(\sigma(-(\gamma + f(h'_i, r, t'_i)))) \ . \end{aligned} \tag{36}$$

Thus, negative samples for which the model predicts a high score relative to other samples are weighted stronger.

Cross entropy loss The cross entropy loss (CEL) has been successfully applied together with 1-N scoring, i.e., predicting for each (h,r)-pair simultaneously a score for each possible tail entity, and framing the problem as a multi-class classification problem [3], [8]. To apply the CEL, first, the labels are normalized in order to form a proper probability distribution. Second, the predicted scores for the tail entities of (h,r)-pair are normalized by a softmax:

$$p(t \mid h, r) = \frac{\exp(f(h,r,t))}{\sum\limits_{t' \in \mathcal{E}} \exp(f(h,r,t'))} \ . \tag{37}$$

Finally, the cross entropy between the distribution of the normalized scores and the normalized label distribution is computed:

$$\mathcal{L} = -\sum_{t' \in \mathcal{E}} \mathbb{I}[(h,r,t') \in \mathcal{K}] \cdot \log(p(t \mid h, r)) \ , \tag{38}$$

where $\mathbb{I}$ denotes the indicator function. Note that this loss differs from the multi-class binary cross entropy as it applies a softmax normalization implying that this is a *single-label* multi-class problem.

3.4 Explicitly Modeling Inverse Relations

Inverse relations introduced by [29] and [40] are explicitly modeled by extending the set of relations $\mathcal{R}$ by a set of inverse relations $r_{inv} \in \mathcal{R}_{inv}$ with $\mathcal{R}_{inv} \cap \mathcal{R} = \emptyset$. This is achieved by training an inverse triple (t, r_{inv}, h) for each triple $(h, r, t) \in \mathcal{K}$. Equipping a KGEM with inverse relations implicitly doubles the relation embedding space of any model that has relation embeddings. The goal is to alter the scoring function, such that the task of predicting the head entities for (r, t) pairs becomes the task of predicting tail entities for (t, r_{inv}) pairs. The explicit training of the implicitly known inverse relations can lead to better model performance [40] and can for some models increase the computational efficiency [37].

4 Evaluation Metrics for KGEMs

KGEMs are usually evaluated based on link prediction, which is on KG defined as predicting the tail/head entities for $(h, r)/(r, t)$ pairs. For instance, given queries of the form *(Sarah, studied_at, ?)* or *(?, CEO_of, Deutsche Bank)* the capability of a link predictor to predict the correct entities that answer the query, i.e. *(Sarah, studied_at, **University of Oxford**)* and *(**Sarah**, CEO_of, Deutsche Bank)* is measured.

However, given the fact that usually true negative examples are not available, both the training and the test set contain only true facts. For this reason, the evaluation procedure is defined as a ranking task in which the capability of the model to differentiate corrupted triples from known true triples is assessed [19]. For each test triple $t^+ = (h, r, t) \in \mathcal{K}_{test}$ two sets of corrupted triples are constructed:

1) $\mathcal{H}(r, t) = \{(h', r, t) \mid h' \in \mathcal{E} - \{h\}$ which contains all the triples where the head entity has been corrupted, and
2) $\mathcal{T}(h, r) = \{(h, r, t') \mid t' \in \mathcal{E} - \{t\}\}$ that contains all the triples with corrupted tail entity.

For each t^+ and its corresponding corrupted triples, the scores are computed and the entities sorted accordingly. Next, the rank of every t^+ among its corrupted triples is determined, i.e. the position in the score-sorted list.

Among the corrupted triples in $\mathcal{H}(r, t)$ / $\mathcal{T}(h, r)$, there might be true triples that are part of the KG. If these false negatives are ranked higher than the current test triple t^+, the results might get distorted. Therefore, the *filtered* evaluation setting has been proposed [19], in which the corrupted triples are filtered to exclude known true facts from the train and test set. Thus, the rank does not decrease when ranking another true entity higher.

Moreover, we want to draw attention to the fact that the metrics can be further be distorted by *unknown false negatives*, i.e., true triples that are contained in the set of corrupted triples but are not part of the KG (and therefore cannot be filtered out). Therefore, it is essential to investigate the predicted scores of a KGEM and not solely rely on the computed metrics.

Based upon these individual ranks, the following measures are frequently used to summarize the overall performance:

Mean rank The mean rank (MR) represents the average rank of the test triples, i.e.

$$\text{MR} = \frac{1}{|\mathcal{K}_{test}|} \sum_{t \in \mathcal{K}_{test}} rank(t) \tag{39}$$

Smaller values indicate better performance.

Adjusted mean rank Because the interpretation of the MR depends on the number of available candidate triples, comparing MRs across different datasets (or inclusion of inverse triples) is difficult. This is sometimes further exacerbated in the filtered setting because the number of candidates varies. Therefore, with fewer candidates available, it becomes easier to achieve low ranks. The adjusted mean rank (AMR) [10] compensates for this problem by comparing the mean rank against the expected mean rank under a model with random scores:

$$\text{AMR} = \frac{MR}{\frac{1}{2} \sum\limits_{t \in \mathcal{K}_{test}} (\xi(t) + 1)} \tag{40}$$

where $\xi(t)$ denotes the number of candidate triples against which the true triple $t \in \mathcal{K}_{test}$ is ranked. In the unfiltered setting we have $\xi(t) = |\mathcal{E}| - 1$ for all $t \in \mathcal{K}_{test}$. Thereby, the measure also adjusts for chance, as a random scoring achieves an expected adjusted mean rank of 1. The AMR has a fixed value range from 0 to 1, where smaller values (AMR $\ll 1$) indicate better performance.

Mean reciprocal rank The mean reciprocal rank (MRR) is defined as:

$$\text{MRR} = \frac{1}{|\mathcal{K}_{test}|} \sum_{t \in \mathcal{K}_{test}} \frac{1}{rank(t)} \tag{41}$$

where $\mathcal{K}_{test}$ is a set of test triples, i.e. the MRR is the mean over reciprocal individual ranks. However, the MRR is flawed since the reciprocal rank is an ordinal scale and not an interval scale, i.e. computing the arithmetic mean is statistically incorrect [41], [42]. Still, it is often used for early stopping since it is a smooth measure with stronger weight on small ranks, and less affected by outlier individual ranks than the mean rank. The MRR has a fixed value range from 0 to 1, where larger values indicate better performance.

Hits@K Hits@K denotes the ratio of the test triples that have been ranked among the top *k* triples, i.e.,

$$\text{Hits@k} = \frac{|\{t \in \mathcal{K}_{test} \mid rank(t) \leq k\}|}{|\mathcal{K}_{test}|} \tag{42}$$

Larger values indicate better performance.

Additional Metrics Further metrics that might be relevant are the area under the Receiver Operating Characteristic curve (AUC-ROC) and the area under the precision-recall curve (AUC-PR) [11]. However, these metrics require the number of true positives, false positives, true negatives, and false negatives, which in most cases cannot be computed since the KGs are usually incomplete.

5 Existing Benchmark Datasets

In this section, we describe the benchmark datasets that have been established to evaluate KGEMs. A summary is also given in Table 1.

FB15K Freebase is a large cross-domain KG consisting of around 1.2 billion triples and more than 80 million entities. Bordes *et al.* [19] extracted a subset of Freebase, which is used as a benchmark dataset and named it FB15K. It contains 14,951 entities, 1,345 relations, as well as more than half a million triples describing facts about movies, actors, awards, sports, and sports teams [37].

FB15K-237 FB15K has a test-leakage, i.e. a major part of the test triples (∼81%) are inverses of triples contained in the training set: for most of the test triples of the form (h, r, t), there exists a triple (h, r', t) or (t, r', h) in the training set. Therefore, Toutanova and Chen [43] constructed FB15K-237 in which inverse relations were removed [43]. FB15K-237 contains 14,541 entities and 237 relations.

WN18 WordNet[1] is a lexical knowledge base in which entities represent terms and are called *synsets*. Relations in WordNet represent conceptual-semantic and lexical relationships (e.g. hyponym). Bordes *et al.* [17] extracted a subset of WordNet named WN18 that is frequently used to evaluate KGEMs. It contains 40,943 synsets and 18 relations.

WN18RR Similarly to FB15K, WN18 also has a test-leakage (of approximately 94%) [43]. For instance, for most of the test triples of the form *(h, hyponym, t)*, there exists a triple *(t, hypernym, o)* in the training set. Dettmers *et al.* [37] have shown that a simple rule-based system can obtain results competitive to the state of the art results on WN18. For this reason, they constructed WN18RR by removing inverse relations similarly to the procedure applied to FB15K. WN18RR contains 40,943 entities and 11 relations.

Kinships The Kinships [44] dataset describes relationships between members of the Australian tribe *Alyawarra* and consists of 10,686 triples. It contains 104 entities representing members of the tribe and 26 relationship types that represent kinship terms such as *Adiadya* or *Umbaidya* [17].

Nations The Nations [45] dataset contains data about countries and their relationships with other countries. Exemplary relations are *economic_aid* and *accusation* [17].

Unified Medical Language System The Unified Medical Language System (UMLS) [46] is an ontology that describes relationships between high-level concepts in the biomedical domain. Examples of contained concepts are *Cell*, *Tissue*, and *Disease*, and exemplary relations are *part_of* and *exhibits* [17], [46].

YAGO3-10 Yet Another Great Ontology (YAGO) [47] is a KG containing facts that have been extracted from Wikipedia and aligned with WordNet in order to exploit the large amount of information contained in Wikipedia and the taxonomic information included in WordNet. It contains general facts about public figures, geographical entities, movies, and further entities, and it has a taxonomy for those concepts. YAGO3-10 is a subset of YAGO3 [48] (which is an extension of YAGO) that contains entities associated with at least ten different relations. In total, YAGO3-10 has 123,182 entities and 37 relations, and most of the triples describe attributes of persons such as citizenship, gender, and profession [37].

1. https://wordnet.princeton.edu/

TABLE 1
Existing Benchmark Datasets.

Dataset	Triples	Entities	Relations
FB15K	592,213	14.951	1,345
FB15K-237	272,115	14,541	237
WN18	151,442	40,943	18
WN18RR	93,003	40,943	11
Kinships	10,686	104	26
Nations	11,191	14	56
UMLS	893,025	135	49
YAGO3-10	1,079,40	132,182	37

6 Reproducibility Studies

The goal of the reproducibility studies was to investigate whether it is possible to replicate experiments based on the information provided in each model's accompanying paper. If specific information was missing, such as the number of training epochs, we tried to find this information in the accompanying source code if it was accessible. For our study, we focused on the two most frequently used benchmark datasets, FB15K and WN18, as well as their respective subsets FB15K-237 and WN18RR. Table 5 (Appendix A1) illustrates for which models results were reported (in the accompanying publications) for the considered datasets. A checkmark denotes that results were reported, and green background indicates that the entire experimental setup for the corresponding dataset was described. Results have not been reported for every model for every dataset because some of the benchmark datasets were created after the models were published. Therefore, these models have been excluded from our reproducibility study.

Experimental Setup For each KGEM, we applied identical training and evaluation settings as described in their concomitant papers. We ran each experiment four times with random seeds to measure the variance in the obtained results. We evaluated the models based on the ranking metrics MR, AMR, MRR, and Hits@K. As discussed in [4], [10], the exact computation of ranks differs across different codebases, and can lead to significant differences [4]. We follow the nomenclature of Berrendorf *et al.* [10], and report scores based on the optimistic, pessimistic, and realistic rank definitions.

Tables 8-11 (Appendix A3-A4) represent the results for FB15K, FB15K-237, WN18, and WN18RR where experiments highlighted in black were reproducible, in blue soft-reproducible experiments (i.e., could be reproduced by a margin $\leq 5\%$), and experiments highlighted in orange could not be reproduced. In the following, we discuss the observations that we made during our experiments.

6.1 Reproductions Requiring Alternate Hyper-Parameters

One of the observations we made is that for some experiments, results could only be reproduced with a different set of hyper-parameter values. For instance, the results for TransE could only be reproduced by adapting the batch size and the number of training epochs. We trained TransE on WN18 for 4000 epochs compared to a reported number of 1000 epochs in order to obtain comparable results. Furthermore, for RotatE on FB15K and WN18, we received better results when adapting the learning rate. The reason for these differences might be explained by the implementation details of the underlying frameworks which have been used to train the models. Authors of early KGEMs often implemented their training algorithms themselves or used frameworks that were popular at the respective time but are not used anymore. Therefore, differences between the former and current frameworks may require an adaption of the hyper-parameter values. Even within the same framework, bug fixes or optimizations of the framework can lead to different results based on the used version. Our benchmarking study highlights that with adapted settings, results can be reproduced and even improved.

6.2 Unreported Hyper-parameters Impedes Reproduction

Some experiments did not report the full experimental setup impeding the reproduction of results. For example, the embeddings in the ConvKB experiments have been pre-trained based on TransE. However, the batch size for training TransE has not been reported, which can significantly affect the results, as previously discussed. Furthermore, we obtained a high deviation for the reported results for HolE on FB15K. The apparent reason is that we could not find the hyper-parameter setting for FB15K, such that we used the same setting as for WN18, which we found in the accompanying implementation.

6.3 Two Perspectives: Publication versus Implementation

While preparing our experiments, we observed that for some experiments, essential aspects, which are part of the released source code, have not been discussed in the paper. For instance, in the publication describing ConvE, it is not mentioned that inverse triples have been added to the KGs in a pre-processing step. This step seems to be essential to reproduce the results. A second example is SimplE, for which the predicted scores have been clamped to the range of $[-20, 20]$. This step was not mentioned in the publication, but it can have a significant effect when the model is evaluated based on an optimistic ranking approach, which is the case for SimplE.

6.4 Lack of Official Implementations Impedes Reproduction

During our experiments, we observed that for DistMult and TransD, we were able to reproduce the results on WN18, but not on FB15K. A reason might be differences in the implementation details of the frameworks used to train and evaluate the models. For example, the initialization of the embeddings or the normalization of the loss values could have an impact on the performance. Since there exists no official implementation (see Table 5 in Appendix A1) for DistMult and TransD, it is not possible to check the above-mentioned aspects. Furthermore, we were not able to reproduce the results for TransH for which also no official implementation is available. There exist reference implementations[2], which slightly differ from the model initially proposed.

6.5 Reproducibility is Dependent on The Ranking Approach

As discussed in [4], [10], the ranking metrics have been implemented differently by various authors. In our experiments, we report results based on three common implementations of the ranking metrics: i.) realistic, ii.) optimistic and iii.) pessimistic ranking (Section 4). If a model predicts the same score for many triples, there will be a large discrepancy between the three ranking approaches. We could observe such a discrepancy for SimplE for which the results on FB15K (Table 8 in Appendix A3) and WN18 (Table 10 in Appendix A4) were almost 0% based on the realistic ranking approach, but were much higher based on the optimistic ranking approach. Similar observations for other KGEM have been made in [4].

7 Benchmarking

In our benchmarking studies, we evaluated a large set of different combinations of interaction models, training approaches, loss functions, and the effect of explicitly modeling inverse relations. Additionally, we evaluated how well the interaction models can model symmetry, anti-symmetry and composition patterns (Appendix 8.1). In particular, we investigated 21 interaction models, two training approaches, and five loss functions on four datasets. We refer to a specific combination of interaction model, training approach, loss function, and whether inverse relations are explicitly modeled as a *configuration*, e.g., RotatE + LCWA + SPL + inverse relations. We do *not* refer to different hyper-parameter values such as batch size or learning rate when we use the term configuration. For each configuration, we used random search to perform the hyper-parameter optimizations over all other hyper-parameters and applied early stopping on the validation set. Each hyper-parameter optimization experiment lasted for a maximum of 24 hours or 100 iterations, in which new hyper-parameters have been sampled in each iteration. Overall, we performed individual hyper-parameter optimizations for more than 1,000 configurations. We retrain the model with the best hyper-parameter setting and report evaluation results on the test set.

Before presenting our results, we provide an overview of the experimental setup, comprising the investigated interaction models, training approaches, loss functions, negative samplers, and datasets. We used the sLCWA and LCWA as training approaches. For the sLCWA we applied a *1:k*-Scoring as usually done throughout the literature [19], [27], where k denotes the number of negative examples for each positive. For the LCWA, we applied a *1:N*-Scoring, i.e., we sample each batch against all negatives examples as typically done for training with the LCWA [37]. Table 6 (Appendix A1) shows the hyper-parameter ranges for the sLCWA and the LCWA assumptions.

2. https://github.com/thunlp/OpenKE

Datasets We performed experiments on the following four datasets: WN18RR, FB15K-237, Kinships and YAGO3-10. We selected WN18RR and FB15K-237 since they are widely applied benchmarking datasets. We chose Kinships and YAGO3-10 to investigate the performance of KGEMs on a small and a larger dataset.

Interaction Models We investigated all interaction models described in Section 3.1. Because of our vast experimental setup and the size of YAGO3-10, we restricted the number of interaction models on YAGO3-10 as otherwise, the computational effort would be prohibitive. Based on their variety of model types as described in Section 3.1, we selected the following interaction models: ComplEx, ConvKB, DistMult, ERMLP, HolE, MuRE, QuatE, RESCAL, RotatE, SE, TransD, and TransE.

Training Approaches We trained the interaction models based on the sLCWA (Section 3.2.2) and the LCWA (Section 3.2.1) training approaches. Due of the extent of our benchmarking study and the fact that YAGO3-10 contains more than 132,000 entities, which makes the training based on the LCWA with 1-n scoring expensive, we restricted the training approach to the sLCWA for YAGO3-10.

Loss Functions We investigated MRL, BCEL, SPL, NSSAL, and CEL since they represent the variety of types described in Section 3.3 and because they have been previously shown to yield good results. MRL has not been historically used in the 1-N scoring setting likely due to the fact that in 1-N scoring, the number of positive and negative scores in each batch is not known in advance and dynamic. Thus, the number of possible pairs varies as well ranging from $N-1$ to $(N/2)^2$ for each (h,r) combination. The accompanying variance in memory requirements for each batch thus poses practical challenges. Therefore, we did not use the MRL in combination with the 1-N scoring setting.

Negative Sampler When using the sLCWA, we generated negative samples with UNS. When training with the LCWA and 1-N scoring, no explicit negative sampling was required.

Early Stopping We evaluated each model every 50 epochs and performed early stopping with a patience of 100 epochs on all datasets except for YAGO3-10. There, considering the larger number of triples seen in each epoch we evaluated each model every 10 epochs and performed early stopping with a patience of 50 epochs.

Below, we describe the results of our benchmarking study. In the four following subsections, we summarize the results for each dataset (i.e., Kinships, WN18RR, FB15K-237, YAGO3-10) along with a discussion of the effect of the models' individual components (i.e., training approaches, loss functions, the explicit modeling of inverse relations) and optimizers on the performance. Finally, we compare the model complexity versus performance. In the appendix, we provide further results. In particular, we provide for each model the results of all tested combinations of interaction model, training approach, and loss function.

7.1 Results on the Kinships Dataset

Investigating the model performances on Kinhsips is interesting because it is a comparatively small KG and thus permits for each configuration a large number of HPO iterations for all interaction models. Figure 4 provides a general overview of the results, i.e., performance of the interaction models, loss functions, training approach, the effect of modeling inverse relations, and the effect of the optimizers. Overall, it can be observed that for most interaction models, several well-performing configurations can be determined. However, some interaction models heavily depend on specific configurations such as KG2E and QuatE. Although link prediction on Kinships seems to be relatively easy, there are several translational distance-based interaction models that perform relatively poor (i.e., TransD, TransE, TransH, TransR, and UM). The poor performance of UM is not surprising considering that it omits the multi-relational information of the data. Finally, the results illustrate that Adam outperforms Adadelta (in many cases with high margin). Therefore, we decided to progress only with Adam as optimizer for the remaining datasets in order to reduce the computational costs.

Impact of the Training approach Figure 5 depicts the effect of the training approaches. We focus only on the BCEL and the SPL (which is equivalent to BCEL, but numerical more stable, see Section 3.3.1) since they have been trained with both training approaches. It can be observed that some interaction models such as MuRE perform equally well on both training approaches on Kinships whereas others such as RESCAL benefit from one of the training approaches (in this case from the sLCWA).

Impact of the Loss Function

Figure 4 highlights that selecting the appropriate loss function is crucial also for relatively small dataset such as Kinships. Although all five loss functions achieve high performance, all except the MRL exhibit high variance. Comparing an interaction model that has been trained with the MRL with an interaction model that has been trained with a different loss function can lead to misleading conclusions since finding a suitable configuration for the loss functions except for the MRL is more difficult.

Impact of Explicitly Modeling Inverse Relations

Figures 4 and 6 present the effect of explicitly modeling inverse relations. Overall, explicitly modeling inverse relations results in less variance across the investigated configurations (Figure 4). Further investigating the effect of modeling of inverse relations on the different loss functions and training approaches (Figure 6), it can be observed that in general, the LCWA benefits from explicit usage of inverse relations in terms of robustness. This is to be expected since, in the LCWA, the model only learns to perform tail predictions, and without explicitly modeling inverse relations, the model might have difficulties in correctly predicting head entities. However, when explicitly modeling inverse relations, the head predictions are obtained by predicting the tail entities of the corresponding inverse triples (see Section 3.4)

Interestingly, MRL and NSSAL-based configurations, which are both only trained with the sLCWA (i.e., the model already learns to perform head and tail predictions) are more robust when trained with inverse relations. Therefore, depending on the dataset, it might be helpful to employ inverse relation for these loss functions even though they might be trained with sLCWA.

Model Complexity versus Performance Figure 17 (Appendix A9) plots the model size against the obtained per-

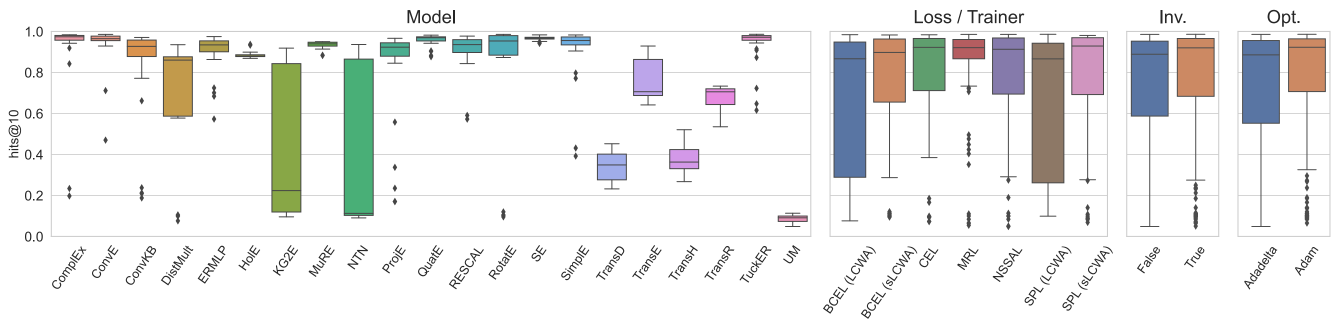

Fig. 4. Overall hits@10 results for Kinships where box-plots summarize the best results across different configurations, i.e., combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

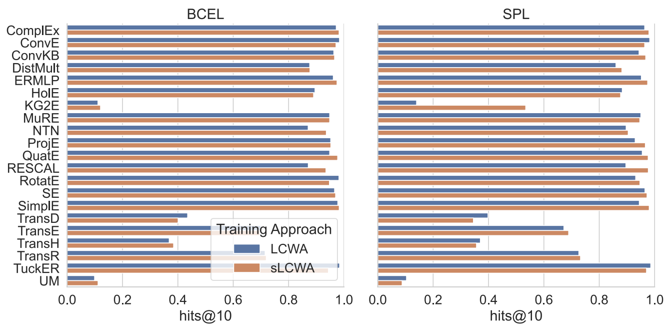

Fig. 5. Impact of training approach on the performance for a fixed interaction model and loss function for the Kinships dataset based on Adam.

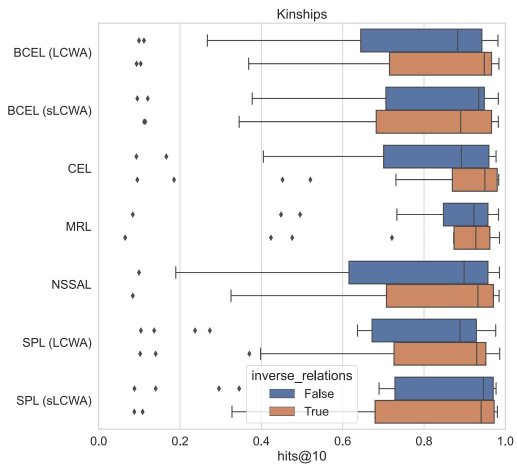

Fig. 6. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the Kinships dataset.

formance. The results highlight that there is no strong correlation between model size and performance, i.e., models with a small number of parameters can perform equally well as large models on the Kinships data set. The skyline comprises small UM models, some intermediate HolE and ProjE models, and larger RotatE and TuckER models. A full list is provided in Table 14 in Appendix A6.

7.2 Results on the WN18RR Dataset

Figure 7 depicts the overall results over WN18RR. A detailed overview of all configurations can be found in Figure 20 in Appendix A12. The results highlight that there are several combinations of interaction models, loss functions, and training approaches that obtain hits@10 results that are competitive with state-of-the-art results[3]. In particular, ComplEx (53.74%), ConvE (56.33% compared to 52.00% in the original paper [37]), DistMult (52.62%), MuRE (57.90% compared to 55.50% in the original paper [24]), KG2E (52.30%), ProjE (51,73%), TransE (56.98%), RESCAL (53.92%), RotatE (60.09% compared to 56.61% in the original paper [23]), SimplE (50.89%), and TuckER (56.09% compared to 52.6% in the original paper [30]) obtained high performance. Especially the result obtained by TransE is impressive since with a suitable configuration, it beats most of the published state-of-the-art results. The results highlight that determining an appropriate combination of interaction model, loss function, training approach, and the decision to explicitly modeling inverse relation is fundamental since many interaction models such as ConvE and KG2E reveal a high variance across different configurations. The results for ComplEx and RESCAL further underpin this observation. They reveal competitive results with very specialized configurations that represent outliers. Another interesting observation is the performance of UM, which does not model relations, but can still compete with some of the other interaction models on WN18RR. This observation might indicate that the relational patterns in WN18RR are not too diverse across relations.

Impact of the Training Approach Figures 7 and 8 depict the impact of the training approach. Again, we focus only on BCEL and SPL since they have been trained under both the sLCWA and LCWA. The figures highlight that for both realizations of the binary cross entropy loss, the LCWA achieves higher maximum performance, but at the same time, it reveals a larger variance on both loss functions. Consequently, it may be more difficult to find configurations that obtain high performance. The overall lower variance of SPL can be explained by the fact that it is numerically more stable than the BCEL.

Figure 8 shows the impact of the training approaches for fixed interaction models and used loss functions. The

3. https://paperswithcode.com/sota/link-prediction-on-wn18rr

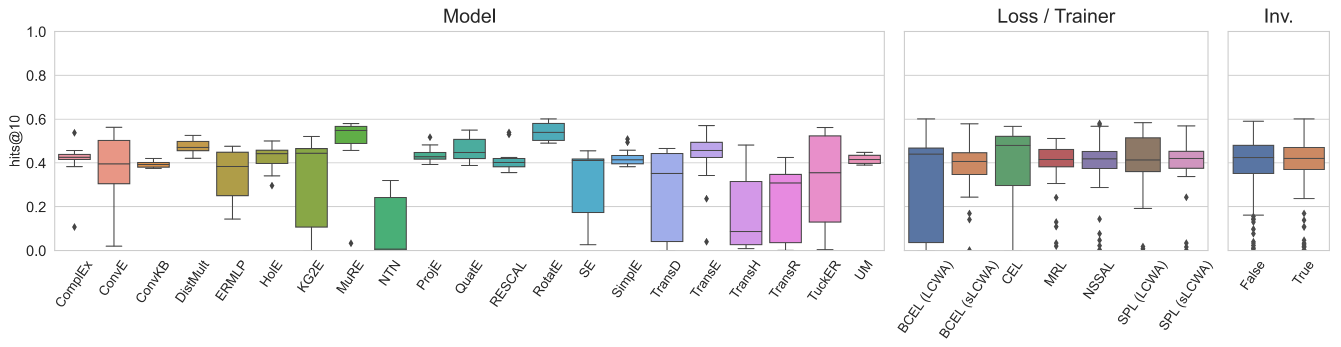

Fig. 7. Overall hits@10 results for WN18RR where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

results indicate that for some combinations of interaction models and loss functions, the training approach's choice has a significant impact on the results. For instance, ConvE, RotatE, TransE and TuckER reveal stronger performance when trained with the LCWA whereas TransH suffer under the LCWA.

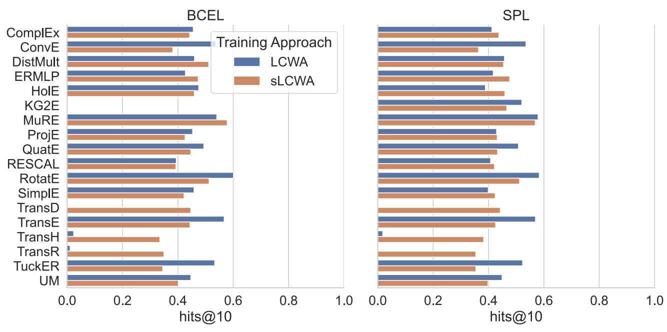

Fig. 8. Impact of training approach on the performance for a fixed interaction model and loss function for the WN18RR dataset.

Impact of the Loss Function Figure 7 depicts the performance of the different loss functions. State-of-the-art results for WN18RR are currently between 50% and 60%, and for each loss function, at least 50% could be achieved (Figure 20 in Appendix A12). However, the MRL is comparably less competitive than the other loss functions. This observation is especially important considering that early KGEMs have often been trained with the MRL. The results highlight that there is a trade-off between highest performance and robustness, i.e., SPL and BCEL achieve the highest performance (when trained under the LCWA), but also have high variance across different configurations (especially BCEL + LCWA).

Figure 24 (Appendix A16) reveals that some interaction models can obtain a further performance boost when configured with specific loss functions. For instance, the performance of ComplEx, ProjE and RESCAL can be increased by a significant margin when composed together with the CEL.

Impact of Explicitly Modeling Inverse Relations Figure 9 illustrates that it is easier to find a strong performing sLCWA-configurations when trained without inverse relations. Surprising is that for LCWA based configurations, the interaction models are still competitive when trained without inverse relations. This observation is surprising because KGEMs that are configured with the LCWA and without inverse relations are not explicitly trained to predict the head entities of triples.

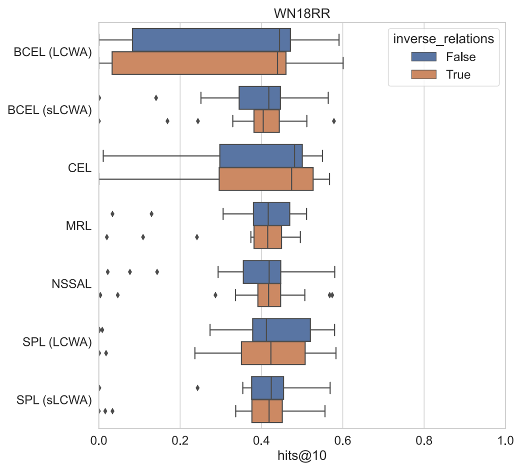

Fig. 9. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the WN18RR dataset.

Model Complexity vs. Performance Figure 17 (Appendix A9) highlights that there is no significant correlation between model size and performance. Instead, the results show that with an appropriate configuration, the model complexity can be significantly reduced (Table 15 in Appendix A6). For instance, for RotatE, several high-performing configurations have been found (Figure 20 in the Appendix A12), and the second-best configuration achieved a hits@10 value of 58.33% while trained with an embedding dimension of *64* (in the complex space). This is especially interesting considering that RotatE originally obtained a performance of 57.1% hits@10 [23] with an embedding dimension of 500 (in the complex space) using the sLCWA as training approach and the NSSAL as loss function [4]. By changing the training approach and the loss function, the embedding dimension could be reduced significantly while getting at the same time an improvement in the hits@10 score.

4. https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding

7.3 Results on the FB15K-237 Dataset

Figure 10 provides an overall overview of the results obtained on FB15K-237. For the results for each individual configuration, we refer to Figure 21 in Appendix A13. We can observe that TuckER outperforms the other interaction models followed by RotatE. DistMult again obtains surprisingly good results (Table 21 in Appendix A13) considering that the interaction model enforces symmetric relations. The results illustrate again that choosing a suitable composition is essential for the performance of an interaction model. For instance, TuckER and QuatE perform well only with dedicated compositions. A further example is DistMult, which again obtains surprisingly good results (Table 21 in Appendix A13) considering that the interaction model enforces symmetric relations. DistMult, however, achieves a strong performance only when composed with the LCWA and the CEL (Table 17 in Appendix A7), highlighting that a simple interaction model can obtain strong performance when composed beneficially.

Impact of the Training Approach Figure 10 shows that for both, BCEL and SPL, the LCWA obtains significantly higher results, but they express a high variance at the same time. Figures 11 and 25 (Appendix A17) illustrate that some interaction models are extremely sensitive to the choice of the training approaches. For instance, it can be observed that RotatE, TransE, and TuckER suffer when trained together with the sLCWA for both loss functions. Table 17 (Appendix A7) shows that most of the interaction models obtain their best performance on FB15K-237 when trained together with the LCWA.

Impact of the Loss Function Figure 10 illustrates that the BCEL and SPL outperform the other loss functions, but they also exhibit higher variance. Figure 25 (Appendix A17) expresses that some interaction models seem to be more sensitive to the usage of different loss function. For instance, ConvE and TuckER suffer from the MRL and the NSSAL, DistMult together with the CEL outperforms the other loss functions. However, TransE performs similarly for all loss functions except the NSSAL.

Impact of Explicitly Modeling Inverse Relations Figure 12 reveals, as for the previous datasets, that in general, the usage of inverse relations is crucial for the training based on the LCWA approach. Different from the results obtained for WN18RR, the LCWA is not competitive when trained without inverse relations.

Model Complexity vs. Performance Figure 17 (Appendix A9) illustrates that for FB15K-237, there is no clear correlation between model size and performance. Tiny models can already obtain similar performance as larger models. The skyline comprises an intermediate UM, TransE and DistMult models, and a larger TuckER model. A full list is provided in Table 13 (Appendix A6).

7.4 Results on the YAGO3-10 Dataset

YAGO3-10 is the largest benchmark dataset in our study. Therefore, it is of interest to investigate how the different interaction models perform on a larger KG. As mentioned in the introduction of this chapter, we reduced the experimental setup for YAGO3-10 in order to reduce the computational complexity of our entire study. Figure 13 depicts the overall results obtained for YAGO3-10. Detailed results for all configurations are illustrated in Figure 22 in Appendix A14.

The results highlight the previous observation that the performance of many KGEMs heavily depends on the choice of its components and is dataset-specific. For instance, MuRE, the best-performing interaction model, and especially RotatE, which is among the top-performing interaction models, exhibit high variance across their configurations. TransE, which was among the top-performing interaction models on WN18RR, performed poorly on YAGO3-10. One might conclude that TransE performs better on smaller KGs, but the results obtained on Kinships do not support this assumption. It should be taken into account that some interaction models might benefit from being trained with the LCWA on YAGO3-10 as observed for TransE on WN18RR. Therefore, TransE might perform much better when trained with the LCWA approach. Remarkably, ComplEx and QuatE seem to be robust for all sLCWA configurations. With regards to the loss functions, all loss functions except MRL obtain comparable results. Though, the MRL is more robust than other loss functions.

Impact of the Loss Function Figure 13 shows again that the choice of the loss functions has an import impact on the models' performance: the margin ranking loss and the self-adversarial negative sampling loss are less competitive than the binary cross entropy loss/Softplus loss. Figure 22 (Appendix A 14) highlights that some interaction models are susceptible to the choice of the loss function. For instance, RotatE and TransE suffer when trained with BCEL and SPL whereas ERMLP suffers when trained with the MRL.

Impact of Explicitly Modeling Inverse Relations Figure 14 shows the effect of explicitly modeling inverse relations for fixed loss functions (it should be noted that the results are obtained based only on the sLCWA training approach). In contrast to the results observed for WN18RR and FB15K-237, the MRL benefits from explicitly modeling inverse relations. Furthermore, also the SPL obtains its best performance with inverse inverse relations.

Model Complexity vs. Performance Figure 17 (Appendix A9) expresses that there is a low correlation between model size and performance for YAGO3-10. However, the improvement is tiny compared to the differences in model size. It should be taken into account that for KGEMs, the model size is usually dependent on the number of entities and relations. Therefore, dependent on the space complexity of the interaction model (Table 4 in Appendix A1), the size can grow fast for large KGs. The skyline comprises an intermediate TransE, DistMult and ConvKB model, and a larger MuRE model. A full list is provided in Table 16 (Appendix A6).

8 Relational Pattern Analysis

Knowledge graphs exhibit relational patterns such as symmetry (e.g., the relation *marriedTo*), and the performance of KGEMs depend on how well these patterns can be modeled. Four major relational patterns that have been investigated in the literature are *symmetry*, *anti-symmetry*, *inversion*, and *composition* [23], [27], [43]. Here, we provide a large-scale

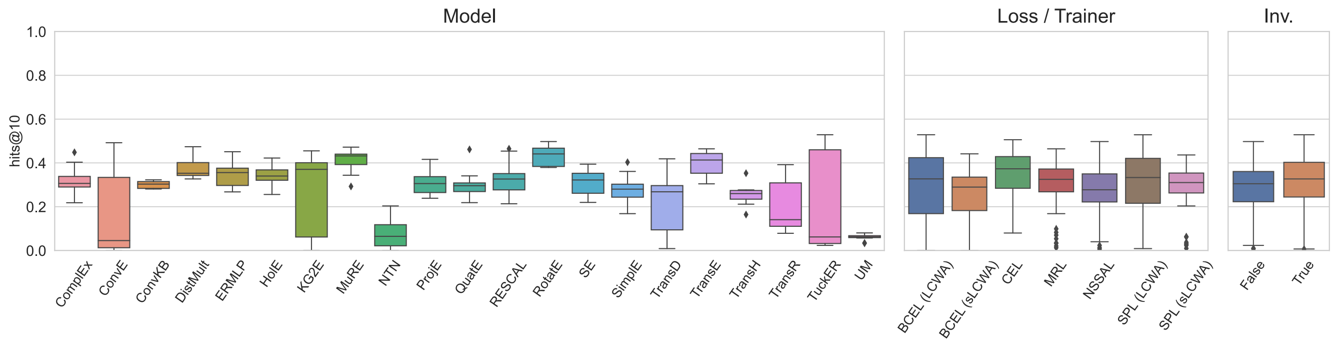

Fig. 10. Overall hits@10 results for FB15K-237 where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

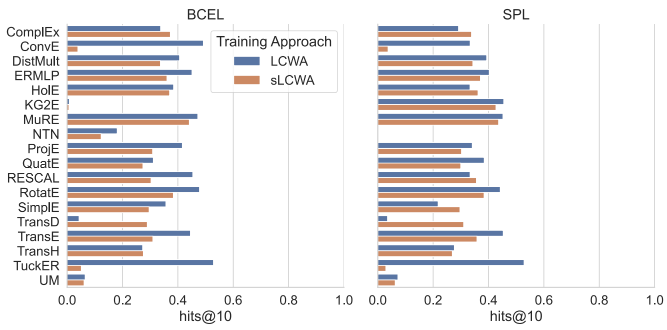

Fig. 11. Impact of training approach on the performance for a fixed interaction model and loss function for the FB15K-237 dataset.

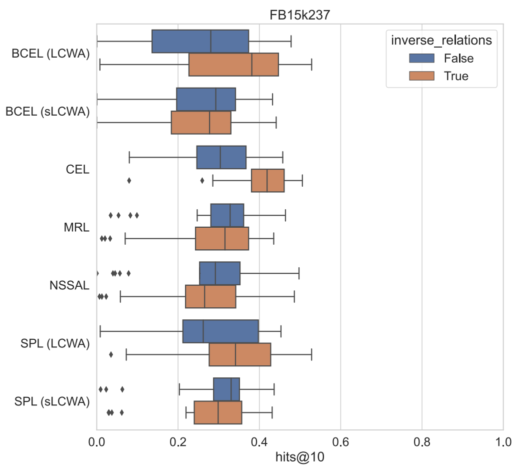

Fig. 12. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the FB15K-237 dataset.

performance analysis of our investigated KGEMs in modeling *symmetry*, *anti-symmetry*, and *composition* patterns for the datasets FB15k-237, WN18RR, and YAGO3-10. First, we provide statistics about the *support* and *confidence* of the symmetry, anti-symmetry, inversion, and composition patterns in the FB15k-237, WN18RR and YAGO3-10 datasets. Next, we describe our experimental setup. Finally, we present the results of our relational pattern analysis.

8.1 Relational Patterns and their Detection

Here, we formally define the relational patterns symmetry, anti-symmetry, inversion, and composition patterns according to [23], the measures *support* and *confidence*, and provide an overview of the *support* and *confidence* of the these patterns in the FB15k-237, WN18RR and YAGO3-10 datasets.

Definition 1 (Symmetric Relation). A relation $r \in \mathcal{R}$ is **symmetric**, if $(h, r, t) \in \mathcal{T} \implies (t, r, h) \in \mathcal{T}$

Definition 2 (Anti-Symmetric Relation). A relation $r \in \mathcal{R}$ is **anti-symmetric**, if $(h, r, t) \in \mathcal{T} \implies (t, r, h) \notin \mathcal{T}$

Definition 3 (Inverse Relation). A relation $r \in \mathcal{R}$ is **inverse** to $r_{inv} \in \mathcal{R}$, if $(h, r, t) \in \mathcal{T} \implies (t, r_{inv}, h) \in \mathcal{T}$. If there exists a $r' \in \mathcal{R}$ with $r' \neq r$ and r' is inverse to r, then we call r an inverse relation.

Definition 4 (Composite Relation). A relation $r \in \mathcal{R}$ is a **composition** of two relations $r_1, r_2 \in \mathcal{R}$, if $(a, r_1, b) \in \mathcal{T} \wedge (b, r_2, c) \in \mathcal{T} \implies (a, r, c) \in \mathcal{T}$. We call r a composite relation, if such two relations exist.

Since KGs are known to be incomplete, a false antecedent, i.e., right-hand side of a rule, may not only be caused by the relation not being of the relation type of interest, but also originate from the KG's incompleteness. Thus, we detect relation types using a support and confidence threshold, defined akin to the concepts of association rule mining.

The *support* of one of the aforementioned patterns p for a relation r indicates the number of different assignments of entities such that the precedent, i.e., the left-hand side of a rule, holds. For most of the simple rules this is equivalent to the relation frequency, but, e.g., for composite relations, we need to consider all pairs of triples with matching the candidate relations r_1, r_2 and being linked by the intermediate entity b.

The *confidence* of a relational pattern is the number of times the right-hand side holds divided by the support. Thus, it can be interpreted as an estimate of the the conditional probability of the antecedent, given the precedent holds.

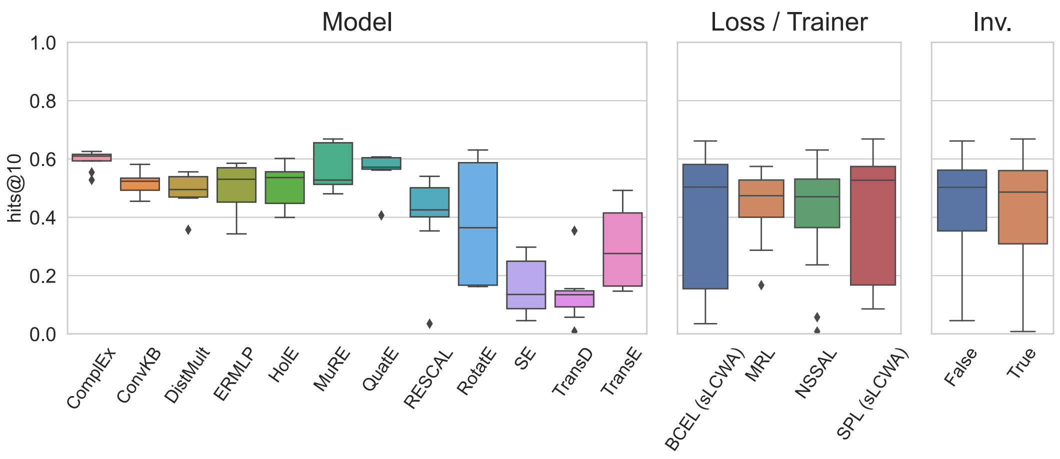

Fig. 13. Overall hits@10 results for YAGO3-10 where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations. In contrast, to the previous datasets, the models have only been trained based on the stochastic local closed world assumption.

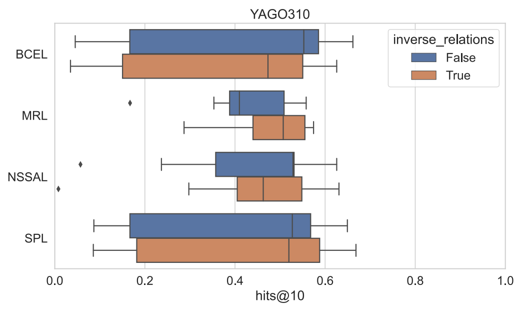

Fig. 14. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the YAGO3-10 dataset.

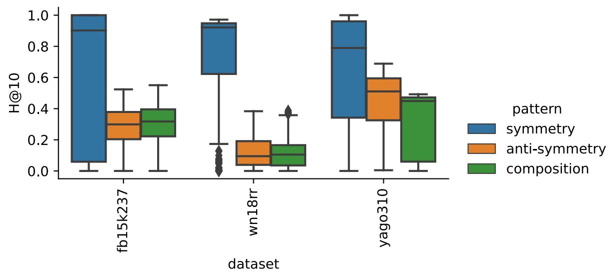

Fig. 15. Performance Distribution of all best models per configuration in H@10.

TABLE 2
Frequency of detected relation patterns across the benchmark datasets.

pattern dataset	anti-symmetry	composition	symmetry
fb15k237	205	147	3
wn18rr	7	1	3
yago310	30	3	2

8.2 Relation Patterns in Benchmark Datasets

Table 2 shows the frequency of the detected pattern types for the three studied benchmark datasets. Similar to related work we used a confidence threshold of 97% [43]. Note that we did not detect a single inverse relation, since FB15k-237 and WN18RR have been explicitly preprocessed to remove such.

8.3 Experimental Setup

To measure the performance of the investigated KGEMs in modeling symmetry, anti-symmetry, and composition patterns, we slightly adapted the standard link prediction evaluation procedure (Section 4). Instead of computing the metrics based on all test triples, we extracted for each relational pattern all test triples that contain the associated relations, aggregated the single ranks obtained of each triple in the subset, and computed the hits@10 metric for each subset. Therefore, we can express how well a KGEM can model a specific relational pattern.

8.4 Results

Figure 15 shows the overall performance on pattern types per dataset. We show the distribution of best models' performance for each configuration in terms of H@10. We generally observe a tendency that symmetric relations are easier to model than anti-symmetric and composite relations, which seem to be equally challenging.

Figure 16 (Appendix A2) shows the performance of best models' for each configuration for each dataset and pattern type, grouped by interaction function. For the most simple pattern, symmetry, almost all interaction functions can obtain strong results on WN18RR, with NTN, TransD and SE slightly falling behind. For FB15k237, we observe similar results, except that SimplE and KG2E fail to capture this pattern (while performing still sufficiently good on other patterns). On YAGO3-10, translation-based methods such as TransE or TransD cannot match the performance of, ComplEx, RotatE and DistMult, with ER-MLP's performance in between.

On the more difficult anti-symmetry and composition patterns, the differences are more pronounced. Overall, RotatE and TransE obtain the best results, whereas UM and NTN cannot obtain good results.

9 Discussion & Future Work

Table 7 (Appendix A1) illustrates the extent of our studies and Table 3 (Appendix 18) summarizes the main findings our work. Although the re-implementation of all machine learning components into a unified, fully configurable framework was a major effort, we believe it is essential to analyze reproducibility and obtain fair results on benchmarking. In particular, we were able to address the issue of incompatible evaluation procedures and preprocessing steps in previous publications that are not obvious. We highlighted that the evaluation metrics, which usually are utilized to evaluate the performance of knowledge graph embedding models, are realized differently depending on the definition of the **rank**. Specifically, three major rank definitions are employed: *optimistic*, *realistic*, and *pessimistic* ranking. Because the optimistic and pessimistic ranking can lead to distorted conclusions in cases where a KGEM predicts the same score for many triples, we recommend evaluating knowledge graph embedding models based on the realistic ranking approach.

During our reproducibility study, we found that the reproduction of experiments is a major challenge and, in many cases, not possible with the available information in current publications. In particular, we observed the following four main aspects:

- For a set of experiments, the results can sometimes only be reproduced with a different set of hyper-parameter values.
- For some experiments, the entire experimental setup was not provided, impeding the reproduction of experiments.
- The lack of an official implementation hampers the reproduction of results.
- Some results are dependent on the utilized ranking approach (average, optimistic, and pessimistic ranking approach). For example, the optimistic rank may lead to incorrect conclusions about the model's performance.

Our benchmarking study shows that the term KGEM should be used with caution and should be differentiated from the actual interaction model since our results highlight that the specific *combination* of the interaction model, training approach, loss function, and the usage of explicit inverse relations is often fundamental for the performance.

No configuration performs best across all datasets. Depending on the dataset, several configurations can be found that achieve comparable results (Tables 17-20 in Appendix A7-A8, and Figures 19-22 in Appendix A11-A14). Moreover, with an appropriate configuration, the model size can significantly be compressed (see Pareto-optimal configurations in Tables 13-16 in Appendix A6) that has especially a practical relevance when looking for a trade-off between required memory and performance.

The results also highlight that even interaction models such as TransE that have been considered as baselines can outperform state-of-the-art interaction models when trained with an appropriate training approach and loss function. This raises the question of the necessity of the vast number of available interaction models. However, for some interaction models such as RotatE, MuRE or TuckER, we can observe a good performance across all datasets (note: TuckER has not been evaluated on YAGO3-10). For RotatE, we even obtained the state-of-the-art results on WN18RR (similar results were obtained by Graph Attenuated Attention Networks [49]), and for ConvE, MuRE, and TuckER, we obtained results superior to the originally published ones. ComplEx proved to be a very robust interaction model across different configurations. This can, in particular, be observed from the results obtained on YAGO3-10 (Figure 13).

We discovered that no loss function consistently achieves the best results. Instead it can be seen that with different loss functions, such as the BCEL, NSSAL, and SPL, good results can be obtained across all datasets. Remarkably, the MRL is overall the worst-performing loss function. However, one might argue that the MRL is the most compatible loss function with the sLCWA since it does not assume artificially generated negative examples to be actually false in contrast to the other loss functions used. The MRL only learns to score positive examples higher than *corresponding* negative examples, but it does not ensure that a negative example is scored lower than every other positive example. Thus, the absolute score values are not interpretable and cannot be used to compare triples without common head/tail entities. They can only be interpreted relatively, and only when comparing scores for triples with the same $(hr)/(rt)$. Although loss functions such as BCEL or SPL treat generated negative triples as true negatives that actually contain also unknown positive examples, they obtain good performance. This might be explained by the fact that usually the set of unknown triples are dominated by false triples. Therefore, it is likely that a major part of the generated triples are actually negative. Consequently, the KGEM learns to distinguish better positive from negative examples.

Considering the explicit usage of inverse relations, we found out that the impact of inverse relations can be significant, especially when the interaction model is trained under the LCWA. This might be explained by the fact that based on the LCWA-training, the KGEM only learns to perform *one-side predictions* (i.e., it learns to either predict head or tail entities), but during the evaluation, it is asked to perform *both-side predictions*. Through the inclusion of inverse relations, the model learns to perform both-side predictions based on one side, i.e., $(*, r, t)$ can be predicted through $(t, r_{inverse}, *)$. Overall, our results indicate that further investigations on FB15K-237 and YAGO3-10 might lead to results that are competitive to the state-of-the-art.

Looking forward, it would be of great interest to re-investigate previously performed studies that analyze the relationship between the performance of KGEMs and the properties of the underlying KGs to verify that their findings indeed can be attributed to the *interaction model* alone, rather than the exact configuration including the loss function, the training approach and the explicit modeling of inverse relations. Further, the effect of explicitly modeling inverse

TABLE 3
Summary of main insights over all datasets. Each component (i.e., interaction model, loss function, and training approach) is considered to be among the top-ten performing configurations when they occur at least once in the top-ten performing configurations. Note that a single component is part of several configurations, and therefore, can occur multiple times in the top-ten performing configurations.

Interaction Models	
RotatE	Among top-ten-performing interaction models across all datasets.
MuRE	Among top-ten-performing interaction models on WN18RR, FB15K-237, and YAGO3-10.
ConvE	Among top-ten-performing interaction models on Kinships and FB15K-237 (has not been evaluated on YAGO3-10).
ComplEx	Among top-ten-performing interaction models on Kinships and YAGO3-10.
TuckER	Among top-ten-performing interaction models for Kinships, and FB15K-237 (has not been evaluated on YAGO3-10).
DistMult	Among top-ten-performing interaction models on FB15K-237.
QuatE	Among top-ten-performing interaction models on YAGO3-10.
TransE	Among top-ten-performing interaction models on WN18RR.
SE	Among top-ten-performing interaction models on Kinships.
Loss Functions	
BCEL	Among top-ten-performing loss functions across all datasets.
NSSAL	Among top-ten-performing loss functions across all datasets.
SPL	Among top-ten-performing loss functions across all datasets.
CEL	Among top-ten-performing loss functions on Kinships and FB15K-237 (has not been evaluated on YAGO3-10).
MRL	Among top-ten-performing loss functions on Kinships.
Training Approaches	
sLCWA	Among top-ten-performing training approaches across all datasets.
LCWA	Among top-ten-performing training approaches on Kinships, WN18RR and FB15K-237 (has not been evaluated on YAGO3-10).
Explicit Modeling of Inverse Relations	
	Is usually beneficial in combination with the local closed world assumption.
Configurations	
Performance	Appropriate combination of interaction model, training assumption, loss function, choice of explicitly modeling inverse relations is crucial for the performance, e.g., TransE can compete when with several state-of-the-art interaction models on WN18RR when appropriate configuration is selected. There is no single best configuration that works best for all dataset.
Variance	Some interaction models exhibit a high variance across different configurations, e.g., RotatE on YAGO3-10 (Figure 13 on page 16)
Pareto-Optimal Configurations	Tables 13-16 in Appendix A6 describe Pareto-optimal configurations. It can be seen that there are configurations that require fewer parameters while obtaining almost the same performance. In some cases, for the same interaction model, the model can be significantly compressed.
Reproducibility	
Results	For FB15K, four out of 13, for WN18, five out of 13, for FB15K-237, two out of three, and for WN18RR, three out of five experiments can be categorized as soft-reproducible.
Code	For four out of 15 models, no official implementation was available.
Parameters	For six out of 15 papers, source code was available and full experimental setup was precisely described.
General Insights	
SOTA	For WN18RR, we achieve based on a RotatE-configuration (together with Graph Attenuated Attention Networks [49]) state-of-the-art results in terms of hits@10 through our study (60.09% Hits@10). Furthermore, we found a TransE configuration that achieves high performance beating most of the published SOTA results (56.98% Hits@10). Based on our results, we emphasize to further investigate the hyper-parameters space for the most promising configurations for the remaining benchmarking datasets.
Improvements	For ConvE (56.33% compared to 52.00% [37]), MuRE (57.90% compared to 55.50% [24]) and TuckER (56.09% compared to 52.6% [30]), we are beating the reported results in the original papers due selecting appropriate configurations and hyper-parameters on WN18RR.

relations has not been analyzed in depth, in particular how the learned representations of a relation and its inverse are related to each other. Ultimately, we believe our work provides an empirical foundation for such studies and a practical tool to execute them.

ACKNOWLEDGMENT

We want to thank the Center for Information Services and High Performance Computing (ZIH) at TU Dresden for generous allocations of computer time and the Technical University of Denmark for providing us access to their DTU Compute GPU cluster that enabled us to conduct our studies. This work was funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A and Grant No. 01IS18050D (project "MLWin"), the Innovation Fund Denmark with the Danish Center for Big Data Analytics driven Innovation (DABAI), and the Defense Advanced Research Projects Agency (DARPA) Automating Scientific Knowledge Extraction (ASKE) program under grant HR00111990009.

REFERENCES

[1] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Trans.*

Knowl. Data Eng., vol. 29, no. 12, pp. 2724–2743, 2017.

[2] F. Akrami, L. Guo, W. Hu, and C. Li, "Re-evaluating embedding-based knowledge graph completion methods," in *CIKM*. ACM, 2018, pp. 1779–1782.

[3] R. Kadlec, O. Bajgar, and J. Kleindienst, "Knowledge base completion: Baselines strike back," in *Rep4NLP@ACL*. Association for Computational Linguistics, 2017, pp. 69–74.

[4] Z. Sun, S. Vashishth, S. Sanyal, P. P. Talukdar, and Y. Yang, "A re-evaluation of knowledge graph completion methods," in *ACL*. Association for Computational Linguistics, 2020, pp. 5516–5522.

[5] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *ICLR (Poster)*, 2015.

[6] F. Akrami, M. S. Saeef, Q. Zhang, W. Hu, and C. Li, "Realistic re-evaluation of knowledge graph completion methods: An experimental study," in *SIGMOD Conference*. ACM, 2020, pp. 1995–2010.

[7] S. K. Mohamed, V. Novácek, P. Vandenbussche, and E. Muñoz, "Loss functions in knowledge graph embedding models," in *DL4KG@ESWC*, ser. CEUR Workshop Proceedings, vol. 2377. CEUR-WS.org, 2019, pp. 1–10.

[8] D. Ruffinelli, S. Broscheit, and R. Gemulla, "You CAN teach an old dog new tricks! on training knowledge graph embeddings," in *ICLR*. OpenReview.net, 2020.

[9] A. Rossi, D. Firmani, A. Matinata, P. Merialdo, and D. Barbosa, "Knowledge graph embedding for link prediction: A comparative analysis," *CoRR*, vol. abs/2002.00819, 2020.

[10] M. Berrendorf, E. Faerman, L. Vermue, and V. Tresp, "Interpretable and fair comparison of link prediction or entity alignment methods with adjusted mean rank," *CoRR*, vol. abs/2002.06914, 2020.

[11] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proc. IEEE*, vol. 104, no. 1, pp. 11–33, 2016.

[12] B. Kotnis and V. Nastase, "Analysis of the impact of negative sampling on link prediction in knowledge graphs," *CoRR*, vol. abs/1708.06816, 2017.

[13] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, "Amie: association rule mining under incomplete evidence in ontological knowledge bases," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 413–422.

[14] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[15] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, 2017.

[16] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation learning for dynamic graphs: A survey," *J. Mach. Learn. Res.*, vol. 21, pp. 70:1–70:73, 2020.

[17] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation," *Mach. Learn.*, vol. 94, no. 2, pp. 233–259, 2014.

[18] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *AAAI*. AAAI Press, 2011.

[19] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013, pp. 2787–2795.

[20] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *AAAI*. AAAI Press, 2014, pp. 1112–1119.

[21] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*. AAAI Press, 2015, pp. 2181–2187.

[22] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *ACL (1)*. The Association for Computer Linguistics, 2015, pp. 687–696.

[23] Z. Sun, Z. Deng, J. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *ICLR (Poster)*. OpenReview.net, 2019.

[24] I. Balazevic, C. Allen, and T. M. Hospedales, "Multi-relational poincaré graph embeddings," in *NeurIPS*, 2019, pp. 4465–4475.

[25] S. He, K. Liu, G. Ji, and J. Zhao, "Learning to represent knowledge graphs with gaussian embedding," in *CIKM*. ACM, 2015, pp. 623–632.

[26] M. Nickel, V. Tresp, and H. Kriegel, "A three-way model for collective learning on multi-relational data," in *ICML*. Omnipress, 2011, pp. 809–816.

[27] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, 2016, pp. 2071–2080.

[28] S. Zhang, Y. Tay, L. Yao, and Q. Liu, "Quaternion knowledge graph embeddings," in *NeurIPS*, 2019, pp. 2731–2741.

[29] S. M. Kazemi and D. Poole, "Simple embedding for link prediction in knowledge graphs," in *NeurIPS*, 2018, pp. 4289–4300.

[30] I. Balazevic, C. Allen, and T. M. Hospedales, "Tucker: Tensor factorization for knowledge graph completion," in *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 5184–5193.

[31] L. R. Tucker *et al.*, "The extension of factor analysis to three-dimensional matrices," *Contributions to mathematical psychology*, vol. 110119, 1964.

[32] B. Shi and T. Weninger, "Proje: Embedding projection for knowledge graph completion," in *AAAI*. AAAI Press, 2017, pp. 1236–1242.

[33] M. Nickel, L. Rosasco, and T. A. Poggio, "Holographic embeddings of knowledge graphs," in *AAAI*. AAAI Press, 2016, pp. 1955–1961.

[34] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: a web-scale approach to probabilistic knowledge fusion," in *KDD*. ACM, 2014, pp. 601–610.

[35] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *NIPS*, 2013, pp. 926–934.

[36] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," *arXiv preprint arXiv:1712.02121*, 2017.

[37] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *AAAI*. AAAI Press, 2018, pp. 1811–1818.

[38] H. Zhang, Z. Kyaw, S. Chang, and T. Chua, "Visual translation embedding network for visual relation detection," in *CVPR*. IEEE Computer Society, 2017, pp. 3107–3115.

[39] S. Sharifzadeh, M. Berrendorf, and V. Tresp, "Improving visual relation detection using depth maps," *CoRR*, vol. abs/1905.00966, 2019.

[40] T. Lacroix, N. Usunier, and G. Obozinski, "Canonical tensor decomposition for knowledge base completion," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 2869–2878.

[41] N. Fuhr, "Some common mistakes in IR evaluation, and how they can be avoided," *SIGIR Forum*, vol. 51, no. 3, pp. 32–41, 2017.

[42] S. S. Stevens, "On the theory of scales of measurement," *Science*, vol. 103, no. 2684, pp. 677–680, 1946.

[43] K. Toutanova and D. Chen, "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, pp. 57–66.

[44] W. W. Denham, "The detection of patterns in alyawara nonverbal behavior," Ph.D. dissertation, University of Washington, Seattle., 1973.

[45] R. J. Rummel, *The dimensionality of nations project: attributes of nations and behavior of nations dyads, 1950-1965*. Inter-university Consortium for Political Research, 1976, no. 5409.

[46] A. T. McCray, "An upper-level ontology for the biomedical domain," *International Journal of Genomics*, vol. 4, no. 1, pp. 80–84, 2003.

[47] T. Rebele, F. M. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum, "YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames," in *International Semantic Web Conference (2)*, ser. Lecture Notes in Computer Science, vol. 9982, 2016, pp. 177–185.

[48] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "YAGO3: A knowledge base from multilingual wikipedias," in *CIDR*. www.cidrdb.org, 2015.

[49] R. Wang, B. Li, S. Hu, W. Du, and M. Zhang, "Knowledge graph embedding via graph attenuated attention networks," *IEEE Access*, vol. 8, pp. 5212–5224, 2020.

Mehdi Ali Mehdi Ali received his M.Sc. degree in Computer Science with a focus on intelligent systems from the University of Bonn. Currently, he is a Ph.D. candidate at the computer science department of the University of Bonn and a research associate at the Fraunhofer Institute IAIS. In his Ph.D., he focuses on machine learning models for (knowledge) graphs, multi-modal models that combine graph and textual information, and reproducibility in the field of knowledge graph embedding models.

Max Berrendorf Max Berrendorf received his B.Sc and M.Sc. degree in Computer Science with a minor in Mathematics from RWTH Aachen University. Currently he is pursuing a Ph.D. degree at the chair of Database Systems and Data Mining at Ludwig-Maximilians-Universität München. In his research, he focuses on machine learning on graphs, in particular knowledge graphs, graph matching problems, and reproducibility in machine learning.

Charles Tapley Hoyt Dr. Charles Tapley Hoyt completed his Ph.D. in Computational Life Sciences from the University of Bonn in 2019 and is now affiliated with the Laboratory of Systems Pharmacology at Harvard Medical School, Boston, USA. His interests are in the biological applications of knowledge graph embedding models towards proteochemometrics, target prioritization, drug repositioning, predictive toxicology, and precision medicine.

Laurent Vermue Laurent Vermue received his M.Sc. degree in Industrial Engineering and Management at the Technical University of Berlin and MMSc. degree in Management Science and Engineering at the Tongji University. Currently he is a Ph.D. student at the Section for Statistics and Data Analysis and the Section for Cognitive Systems at DTU Compute, Technical University of Denmark. His research interests include machine learning, complex network modeling and open research software.

Mikhail Galkin Dr. Mikhail Galkin received his Ph.D. degree in Computer Science from the University of Bonn in 2018 studying knowledge graphs, their creation, integration, and querying. Currently, he is a postdoctoral fellow at Montreal Institute for Learning Algorithms (Mila) and McGill University. His interests include applications of knowledge graphs and graph representation learning to neural reasoning and natural language processing.

Sahand Sharifzadeh Sahand Sharifzadeh received his M.Sc degree from Technical University of Munich majoring in Computer Vision and Artificial Intelligence. Currently, he is a Ph.D. candidate at Ludwig-Maximilians-Universität München. In his research, he focuses on extracting graphs from images and text, as well as knowledge graph modeling. He often collaborates with biologists, physicists and robotic engineers as interdisciplinary machine learning research is one of his interests.

Asja Fischer Asja Fischer is professor for machine learning at Ruhr University Bochum. Her research interests are focus on the development, analysis, and application of deep learning models and methods. Before becoming a professor in Bochum she was assistant professor at Bonn university, and a post-doctoral researcher at the Montreal Institute for Learning Algorithms (MILA). Between 2010 and 2015, she was employed both at the Institute for Neural Computation at the Ruhr University Bochum and the Department of Computer Science at the University of Copenhagen working on her PhD, which she defended in Copenhagen in 2014. Before, she studied Biology, Bioinformatics, Mathematics, and Cognitive Science at the Ruhr-University Bochum, the Universidade de Lisboa, and the University of Osnabrück.

Volker Tresp Volker Tresp received the Diploma degree from the University of Goettingen, Germany, in 1984 and the M.Sc. and Ph.D. degrees from Yale University, New Haven, CT, USA, in 1986 and 1989, respectively. Since 1989, he has been the head of various research teams in machine learning at Siemens, Research and Technology, Munich, Germany. He filed more than 70 patent applications and was inventor of the year of Siemens in 1996. He has published more than 100 scientific articles and administered over 20 Ph.D. dissertations. The company Panoratio is a spin-off out of his team. His research focus in recent years has been machine learning in information networks for modeling knowledge graphs, medical decision processes, and sensor networks. He is the coordinator of one of the first nationally funded big data projects for the realization of precision medicine. In 2011, he became a Honorary Professor at the Ludwig Maximilian University of Munich, Germany, where he teaches an annual course on machine learning.

Jens Lehmann Prof. Dr. Jens Lehmann leads the "Smart Data Analytics" research group at the University of Bonn and Fraunhofer IAIS with 40 researchers. His research interests involve knowledge graphs, machine learning, question answering, distributed computing and knowledge representation. He is particularly excited about the combination of data- and knowledge-driven AI methods. Prof. Lehmann won more than 10 international awards for his research work. He is founder, leader or contributor of several community research projects, including SANSA, DL-Learner, DBpedia and LinkedGeoData. Previously, he completed his PhD with "summa cum laude" at the University of Leipzig with visits to the University of Oxford. He studied Computer Science at the Technical University of Dresden.