

Investigating Graph Representation Learning Methods For Link Prediction in Knowledge Graphs

Kumulative Dissertation

zur Erlangung des Doktorgrades (Dr. rer. nat.)
der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

MEHDI ALI

aus Bonn

Bonn, 2022

Angefertigt mit Genehmigung
der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. rer. nat. Jens Lehmann
 2. Gutachter: Univ.-Prof. Dr. rer. nat. Stefan Wrobel
- Tag der Promotion: 06.09.2023
Erscheinungsjahr: 2023

Abstract

Knowledge graphs (KGs) have become a fundamental approach to represent structured data and are employed in academic and industrial applications. KGs are used in various machine learning applications, such as question answering, dialogue systems, and recommendation systems. Although real-world KGs contain up to billions of links, they are usually still incomplete, which can severely impact downstream applications.

Link prediction in KGs is the task of predicting missing links and can be performed in a transductive or inductive setting. In the past, a wide range of link prediction approaches have been proposed, encompassing rule-based and machine learning-based approaches. One promising line of research has been link prediction based on graph representation learning methods. In particular, a large number of knowledge graph embedding models (KGEMs) have been proposed and recently, also graph neural network (GNN) based approaches are used for link prediction within KGs. Despite the intensive research efforts in KGEMs, their capabilities are often not transparent. It has been shown that baseline models can obtain competitive results to the state-of-the-art models when configured appropriately, indicating that the performance of a KGEM may not merely depend on its model architecture, but on the interplay of various components. Link prediction within KGs has been investigated mainly within the transductive setting, prohibiting inference over unseen entities. However, lately, inductive link prediction approaches have obtained increased attention since they are capable of predicting links involving unseen entities.

In this thesis, we propose an extensive ecosystem for investigating the performance of KGEM-based link prediction. We used the developed ecosystem to first perform a reproducibility study in which we investigated the reproducibility crisis of KGEM-based link prediction experiments. Second, we performed the most extensive KGEM-based link prediction study in which we investigated whether incremental performance improvements reported for KGEMs can solely be attributed to the model architectures or the combination of the KGEM's components. After providing an in-depth analysis of transductive link prediction within triple-based KGs, we focus on inductive link prediction within hyper-relational KGs. We bridge the concepts of inductive link prediction and hyper-relational KGs and demonstrate that hyper-relational information improves semi- and fully-inductive link prediction. Finally, we demonstrate the effectiveness of knowledge graph representation learning for addressing biomedical applications.

Declaration of Authorship

I, Mehdi ALI, declare that this thesis titled, “Investigating Graph Representation Learning Methods For Link Prediction in Knowledge Graphs” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Acknowledgements

I want to thank everybody who supported me on my path. First and foremost, I want to thank my parents, who supported me in every aspect of my life.

Next, I want to thank my supervisors, Prof. Dr. Jens Lehmann and Prof. Dr. Asja Fischer. I want to thank Prof. Dr. Jens Lehmann for supporting my scientific career from the very beginning when I joined his research lab as a master thesis student. He ensured that I had all the conditions to grow as a scientist and gave me all the freedom to follow my research interests while always being available for a discussion. I got to know Prof. Dr. Asja Fischer when joining her course (Deep Learning Lab) during my master's studies at the University of Bonn. From the very beginning, she supported me in shaping my scientific career. I want to thank Prof. Dr. Asja Fischer for the great collaboration and all the discussions and advice she gave me during my PhD.

Finally, I want to thank all my friends and colleagues. The path of pursuing a PhD is an amazing journey, and spending time with you/and working with you further made this a unique experience. A special thanks goes to Dr. Daniel Domingo Fernández, Dr. Charles Tapley Hoyt, Dr. Diego Esteves, Dr. Harsh Thakkar, Dr. Gezim Sejdiu, Dr. Denis Lukovnikov, Dr. Sahar Vahdati, Prof. Dr. Maria Maleshkova, Dr. Michael Galkin, Dr. Mohnish Dubey, Endri Kacupaj, Sarah Mubeen, Sina Däubener, Dr. Max Berrendorf, Dr. Laurent Vermue, and Sahand Sharifzadeh.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem Statement & Challenges	2
1.2.1	Problem Statement	2
1.2.2	Challenges	3
	Reproducibility Challenges	3
	Benchmarking Challenges	4
	Hyper-Relational Inductive Link Prediction Challenges	4
1.3	Research Questions & Contributions	5
1.4	List of Publications	7
1.4.1	Thesis Publications	7
1.4.2	Other publications	9
1.5	Thesis Structure	10
1.6	Foundations	11
1.6.1	General Notation	11
1.6.2	Knowledge Graphs	11
1.6.3	Link Prediction	13
	Transductive Link Prediction	13
	Inductive Link Prediction	13
1.6.4	Knowledge Graph Embedding Models	14
	Interaction Models	14
	Translational Distance Interaction Models	15
	Semantic Matching Interaction Models	17
1.6.5	Loss Functions	19
	Pointwise Loss Functions	20
	Pairwise Loss Functions	20
	Setwise Loss Functions	20
1.6.6	Training Approaches	20
	Local closed world assumption	21
	Stochastic local closed world assumption	21
1.6.7	Explicitly Modelling Inverse Relations	21
1.6.8	Graph Neural Networks	22
1.6.9	Evaluation of Link Predictors	22
2	The KEEN Universe: An Ecosystem for Knowledge Graph Embeddings with a Focus on Reproducibility and Transferability	25
3	PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings	29

4	Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models under a Unified Framework	33
5	Improving Inductive Link Prediction Using Hyper-relational Facts	37
6	Applications	41
7	Conclusion & Future Work	47
7.1	Conclusion	47
7.2	Future Work	51
	Bibliography	53
	Appendices	A1
A	The KEEN Universe: An Ecosystem for Knowledge Graph Embeddings with a Focus on Reproducibility and Transferability	A3
B	PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings	A21
C	Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models under a Unified Framework	A29
D	Improving Inductive Link Prediction Using Hyper-relational Facts	A69
E	BioKEEN: a library for learning and evaluating biological knowledge graph embeddings	A89
F	CLEP: a hybrid data- and knowledge-driven framework for generating patient representations	A93

List of Figures

1.1	An exemplary KG. Nodes represent entities and edges their relations.	12
1.2	An exemplary hyper-relational fact. The triple is associated with the qualifier pairs (<i>Degree, Master of Science</i>) and (<i>Academic Major, Computer Science</i>).	12
1.3	Transductive vs. inductive link prediction scenarios. Grey nodes represent entities seen during training, and red nodes are entities seen during inference. Dashed edges represent edges to be predicted during inference.	13

Acronyms

AMR adjusted mean rank

BCEL binary cross entropy loss

BNS Bernoulli negative sampling

CEL cross entropy loss

CP canonical polyadic

CWA closed world assumption

GNN graph neural network

KG knowledge graph

KGEM knowledge graph embedding model

KL Kullback-Leibler

LCWA local closed world assumption

MR mean rank

MRL margin ranking loss

MRR mean reciprocal rank

NSSAL self-adversarial negative sampling loss

NTN Neural Tensor Network

OWA open world assumption

SE Structured Embedding

sLCWA stochastic local closed world assumption

SPL Softplus loss

UM Unstructured Model

UNS uniform negative sampling

Chapter 1

Introduction

1.1 Motivation

Representing structured information in the form of KGs has become a fundamental approach, and KGs are a crucial component in many academic and industrial machine learning applications [1]. Typical downstream machine learning tasks that involve KGs are question answering [2], dialogue systems [3], and recommendation systems [4]. An exemplary domain in which KGs are widely applied is the biomedical domain. For instance, KGs are utilised to model diseases, organisms, and clinical information [5]. Applications involving KGs depend on the completeness of the KGs and the validity of the contained triples. Because of limitations in KG construction approaches and the fast growth of data, real-world KGs such as DBpedia [6], Wikidata [7], Freebase [8], and Bio2RDF [9] that contain millions of facts are usually still incomplete and noisy [10, 11]. Therefore, in the past, various approaches have been proposed to determine missing links by applying, for instance, logic-based reasoning using first-order or description logic and machine-learning-based approaches, such as knowledge graph representation learning approaches [1, 11].

One line of research that has received significant attention in the last decade is KGEM-based link prediction [12], and recently graph neural network-based approaches have received increased attention [13]. Link prediction in KGs has been mainly a transductive task [12, 14], i.e., inference has been restricted to entities and relations that have been seen during training. A vast number of models have been proposed to address transductive link prediction. Often, the reported improvements in the link prediction performance are incremental, and the impact of the proposed contributions is unclear, mainly for two reasons. First, there is a major challenge in reproducing reported link prediction experiments. The reasons for this are manifold: lack of official implementation, employment of slightly different evaluation procedures [15], missing hyper-parameter specifications, and usage of different programming languages and frameworks [14]. Investigating the reproducibility of link prediction experiments under identical conditions provides a better understanding of the performance of the proposed models. More specifically, such a study will highlight experiments that can or cannot be reproduced, adaptations required to reproduce a specific experiment, and potential factors that impede the reproduction of specific experiments.

Second, it has been shown that baseline model architectures/interaction models can obtain competitive results when configured appropriately [16], raising the question of whether the incremental improvements reported in several works are solely due to the proposed interaction models or the impact of other components, such as the training approach and the loss function. In order to answer this question, an extensive study is required in which a large number of different models based on diverse configurations are evaluated. In answering this question, a realistic and holistic overview of the performance of KGEMs for the link prediction task can be established. Furthermore, the answer to this question can significantly impact the field of KGEM research by providing a solid foundation by which the research focus for upcoming studies in this field can be influenced. For instance, the answer can appeal to continue with the research focus of the last years, i.e., focusing mainly on developing novel interaction models or shift the research focus to, e.g., theoretical understanding of KGEMs. Finally, the results of such a study will support practitioners in selecting the appropriate KGEM for their use case, which potentially can have a significant practical impact, e.g., by addressing biomedical use cases.

Lately, inductive link prediction has received significant attention [17]. Inductive link prediction enables inference over unseen entities/reasons, and recent works address different inductive settings. So far, inductive link prediction approaches are restricted to KGs. However, there is an increased usage of *hyper-relational* KGs such as Wikidata [7] that allow modelling edges of a KG together with a set of attributes in the form of key-value pairs that are called *qualifiers* [18]. The hyper-relational information (qualifiers) contained in hyper-relational KGs may improve the generalisation capabilities of inductive link predictors [18]. Besides the academic interest in investigating whether exploring hyper-relational information benefits inductive link prediction, it also has practical relevance since an increasing number of industrial KGs are employing the hyper-relational paradigm [19].

As indicated above, KGs occupy a central role in the biomedical domain [5]. Therefore, it is of huge interest to investigate the effectiveness of KG representation learning and link prediction for biomedical applications.

1.2 Problem Statement & Challenges

This section presents the problem statement underlying this thesis and the challenges related to addressing this problem.

1.2.1 Problem Statement

KGs are widely adopted in academic and industrial applications [13]. Despite their large size, they are often incomplete, impacting downstream applications involving KGs [11]. Therefore, investigating approaches to predict missing links in KGs is crucial. Knowledge graph representation learning, and especially KGEMs, have been heavily investigated for KG transductive

link prediction. However, the performance of KGEMs-based link predictors is often not comprehensible. In particular, the reproduction of various link prediction experiments is not possible, and the impact of the individual components of KGEMs on the model's performance and their interplay is often unclear.

Transductive link prediction approaches are limited to entities (and relations) seen during training. To get reliable predictions involving unseen entities, (generally) a model would need to be retrained, which is impractical for real-world use cases that involve predictions over continuously evolving KGs, such as item recommendations within e-commerce platforms [20, 21]. Lately, inductive link prediction that enables inference over unseen entities has gained increased attention. Because proposed inductive link prediction approaches are designed only for triple-based KGs, but not for hyper-relational KGs despite the increased adoption of hyper-relational KGs [19], it yet not evident how exploiting hyper-relational information impacts inductive link prediction.

KGs occupy an important role in many biomedical applications, and KG representation learning has great potential to be applied in diverse biomedical use cases such as predicting drug-target links [22]. Nonetheless, KG representation learning and link prediction based on KGEMs have been under-explored in the biomedical domain [22, 23].

1.2.2 Challenges

Reproducibility Challenges

Challenge 1: Lack of Information One challenge for the reproduction of link prediction experiments is that for several KGEMs, important information, i.e., official source code or the detailed description of the experimental set-up, is not available [14]. In the case of missing source code, we have to only rely on the information provided in the accompanying paper. However, all necessary implementation details may not be discussed in the paper. Trying to recover missing hyper-parameter values can be a tedious process that does not ensure that appropriate values that ensure the reproduction of published results can be determined.

Challenge 2: Usage of Different Programming Languages & Frameworks Unavoidably, different programming languages and (or) underlying frameworks have been used to implement the KGEMs over the years [14]. This aspect introduces variability and makes the reproduction of published experiments challenging.

Challenge 3: Different Realisation of the Evaluation Metrics It has been shown that the definition of the rank, which is the basis for several link prediction evaluation metrics, has been realised differently [15] which can have a major impact on the link prediction performance [14]. This aspect is especially critical when official source code is lacking.

Challenge 4: Lack of Software Ecosystem To conduct the reproduction experiments, to understand potential reasons that hamper reproduction, and to perform experiments for each model under identical conditions, we require a software ecosystem that implements the KGEMs to investigate. For reproducibility, it is crucial that the ecosystem offers implementations that coincide with the original implementations and does not integrate changes that, for instance, improve the performance of the models. However, the landscape of software ecosystems for KGEMs was limited. Often, software projects for KGEMs are repositories that accompany a publication or cover a small number of KGEMs or provide implementations that do not rely on the original implementation or rely on a single realisation of the evaluation metric, which can hamper reproduction as mentioned above.

Benchmarking Challenges

Challenge 5: Lack of Extensive and Fully Configurable KGEM Software Ecosystem To perform an extensive benchmarking in order to measure the impact of individual components of a KGEM and their interplay under identical conditions, a fully-configurable and extensive software ecosystem is required that enables such an analysis.

Challenge 6: Need for Extensive Computational Resources Besides a comprehensive software ecosystem, we also require extensive computational resources to conduct the benchmarking study. Assuming that we evaluate 21 interaction models on four datasets where, for each interaction model, we evaluate five loss functions, two training approaches, and the effect of (not) modelling inverse relations, we would perform $21 \cdot 4 \cdot 5 \cdot 2 \cdot 2 = 1680$ ablation studies. If we further assume that for each ablation study, we perform a hyperparameter-optimisation that requires, on average, 14 GPU hours, we would require $1680 \cdot 14 = 23520$ GPU hours, which corresponds to ≈ 2.7 years of computing time on a single GPU.

Hyper-Relational Inductive Link Prediction Challenges

Challenge 7: Terminology Gap One challenge in investigating inductive link prediction approaches is the lack of a unified terminology in the inductive link prediction literature. Conceptually similar inductive settings are called differently [17].

Challenge 8: Lack of Benchmark Datasets Because inductive link prediction in hyper-relational KGs has not yet been investigated to the best of our knowledge, a further challenge is the lack of benchmark datasets for evaluating inductive link prediction in hyper-relational KGs.

1.3 Research Questions & Contributions

In this section, we present the research questions we address, the challenges addressed in the context of these research questions, and the contributions we made to answer these questions.

RQ1. *To which extent are published knowledge graph embedding-based link prediction results reproducible?*

This research question addresses the reproducibility of link prediction experiments and covers Challenges 1-4. To answer this question, we made the following contributions. In a first step, we developed PyKEEN/PyKEEN 1.0 (Chapters 2 and 3), an extensive KGEM library for training and evaluating KGEMs-based link predictors under identical conditions. Equipped with PyKEEN 1.0, we performed a reproducibility study involving 34 experiments over 15 KGEMs and four datasets (Chapter 4) in which we demonstrated which results could be reproduced and revealed the existing obstacles in reproducing reported link prediction experiments. We made four key observations: i.) a set of experiments could only be reproduced with an alternative set of hyper-parameter values, ii.) the absence of a detailed experimental description hampered the reproduction, iii.) the absence of an official implementation impeded the reproduction, and iv.) specific results depended on the realised definition of the rank (i.e., depended on the use of the optimistic, average, and pessimistic ranking).

RQ2. *Can the performance of knowledge graph embedding-based link predictors solely be attributed to the interaction model?*

In the context of this research question, we investigate the influence of a KGEM's components on the overall model performance. We seek to answer whether the determining factor for the performance of a KGEM is its interaction model (as postulated in the past) or whether the interplay of various components is crucial for the model performance. While answering this research question, we address Challenges 5-6, and make the following contributions. We performed the most extensive benchmarking study in the field of KGEMs (Chapter 4). We (re-)defined a KGEM as a composition of four components (interaction model, loss function, training approach, and the usage of inverse relations) allowing us to measure the impact of each component on the KGEMs' performance individually. For each interaction model, we evaluated various configurations, where a configuration corresponds to a specific combination of the four components of a KGEM. In particular, our benchmarking study covered 21 interaction models, two training approaches, five loss functions, two optimisers and four datasets. Our study involved several thousands of experiments spanning 24,804 GPU hours to answer the research question.

We empirically show that the performance of a KGEM is not only dependent on its interaction model, but heavily depends on the model's configuration, i.e., the composition of the interaction model, loss function, training

approach, and the decision to explicitly model inverse relations. No composition performs best across all datasets, but the performance is dataset-dependent. We demonstrate that several compositions can compete with the state-of-the-art when configured appropriately. In several cases, previously reported results for an interaction model could be outperformed. Finally, we show that model size can be compressed in certain instances while maintaining competitive performance.

RQ3. *Can hyper-relational information improve inductive link prediction?*

In the context of research questions **RQ1** and **RQ2**, we focused on transductive link prediction approaches that have been applied on triple-based KGs. In the context of this research question, we move one step further and investigate inductive link prediction within hyper-relational KGs. While answering this research question, we address Challenges 7-8. In order to answer the research question, we make the following three contributions. We first address the terminology gap in the literature and propose a theoretical framework to classify inductive link prediction settings (Chapter 5). Our framework distinguishes between semi-inductive and fully-inductive settings and shows that proposed approaches fall in either one of these categories or are a mixture of both. Second, we provide a novel set of benchmark datasets to investigate inductive link prediction in hyper-relational KGs. Third, we adopted two baseline models for the hyper-relational setting and performed extensive quantitative and qualitative experiments for the fully and semi-inductive link prediction settings to investigate the impact of hyper-relational information on the inductive link prediction performance of the models. In summary, we bridged the concepts of inductive link prediction and hyper-relational KGs. We received the Best Research Paper Award at the *International Semantic Web Conference 2021* for our work.

Applications A further contribution of this thesis is the successful utilisation of knowledge graph representation learning for two biomedical applications (Chapter 6).

First, we develop BioKEEN, a software library that facilitates the usage of KGEMs for bioinformaticians without deep knowledge of KGEMs and the implementation of the models. BioKEEN assists users in configuring their link prediction experiments and provides access to numerous biomedical databases. In an exemplary use case, we employed BioKEEN to predict crosstalks and hierarchies between biological pathways in a novel pathway dataset. We could identify that *TGF-beta Receptor Signaling* (*wikipathways: WP560*) is equivalent to *TGF-beta signaling pathway* (*kegg: hsa04350*), and that *Lipoic acid* (*kegg: hsa00785*) is part of *Lipid metabolism* (*reactome: R-HSA-556833*). Both predictions describe novel links demonstrating the effectiveness of BioKEEN in successfully addressing a biomedical link prediction problem.

In a second application, we propose CLEP (CLinical Embedding of Patients), an approach for learning patient embeddings by combining patient-level data and prior knowledge in the form of biological KGs. The patients are incorporated as nodes and linked to the biomedical entities in the KGs

based on their most characteristic features. Based on the enriched KGs, we employed KGEMs to learn knowledge graph embeddings. The learned patient embeddings have been used to train classifiers to discriminate between patients and healthy controls. The results highlight that the investigated classifiers performed better when trained on top of the learned patient embeddings instead of the raw transcriptomics data.

1.4 List of Publications

1.4.1 Thesis Publications

1. **Mehdi Ali**, Hajira Jabeen, Charles Tapley Hoyt, and Jens Lehmann. “The KEEN Universe - An Ecosystem for Knowledge Graph Embeddings with a Focus on Reproducibility and Transferability”. In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*. vol. 11779. Lecture Notes in Computer Science. Springer, 2019, pp. 3–18. DOI: 10.1007/978-3-030-30796-7_1. URL: https://doi.org/10.1007/978-3-030-30796-7_1
Mehdi Ali developed the idea of an ecosystem for KGEMs that focuses on the reproducibility and transferability of KGEM research. Mehdi Ali implemented the major parts of the source code. Charles Tapley Hoyt refactored, and packaged the code, and added additional functionalities. Mehdi Ali wrote the manuscript. All authors revised the manuscript.
2. **Mehdi Ali***, Max Berrendorf*, Charles Tapley Hoyt*, Laurent Vermue*, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. “PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings”. In: *Journal of Machine Learning Research* 22.82 (2021). * equal contribution, pp. 1–6. URL: <http://jmlr.org/papers/v22/20-825.html>
Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Sahand Sharifzadeh, and Laurent Vermue developed the idea of abstracting and implementing KGEMs as the composition of four components that can flexibly be composed after each author identified limitations of existing works restricting the in-depth analysis of KGEMs. Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, and Laurent Vermue implemented the source code. Mehdi Ali wrote the initial manuscript, and all authors extended and finalised it.
3. **Mehdi Ali**, Max Berrendorf*, Charles Tapley Hoyt*, Laurent Vermue*, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. “Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified

Framework". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021). * equal contribution. © 2022 IEEE. Reprinted, with permission, from Mehdi Ali and Max Berrendorf and Charles Tapley Hoyt and Laurent Vermue and Mikhail Galkin and Sahand Sharifzadeh and Asja Fischer and Volker Tresp and Jens Lehmann, *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*, 01/2022. DOI: <https://doi.org/10.1109/TPAMI.2021.3124805>. URL: <https://ieeexplore.ieee.org/abstract/document/9601281>

Mehdi Ali initially discussed the existing research gap with Asja Fischer and Jens Lehmann. Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Sahand Sharifzadeh, and Laurent Vermue developed the idea to systematically address the challenges in reproducing published KGEM experiments and investigate the impact of the single components of a KGEM on its link prediction performance. Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, and Laurent Vermue implemented the source code. Mehdi Ali, Laurent Vermue and Mikhail Galkin performed the reproducibility and benchmarking experiments. In addition, Max Berrendorf implemented and performed the relational pattern analysis. Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt and Laurent Vermue evaluated the results and wrote the initial manuscript. All authors revised the manuscript.

4. **Mehdi Ali***, Max Berrendorf*, Mikhail Galkin, Veronika Thost, Tengfei Ma, Volker Tresp, and Jens Lehmann. "Improving Inductive Link Prediction Using Hyper-relational Facts". In: *Lecture Notes in Computer Science 12922* (2021). * equal contribution. **For this work, we received the Best Research Paper Award.**, pp. 74–92. DOI: [10.1007/978-3-030-88361-4_5](https://doi.org/10.1007/978-3-030-88361-4_5)

The idea to exploit hyper-relational information for inductive link prediction was proposed by Mehdi Ali and further developed together with Max Berrendorf and Mikhail Galkin. Mehdi Ali, Max Berrendorf, Mikhail Galkin, Veronika Thost, and Tengfei Ma developed the theoretical framework for classifying different inductive link prediction scenarios. Mikhail Galkin generated the fully-inductive benchmark datasets, and Mehdi Ali and Max Berrendorf generated based on the fully-inductive datasets, the semi-inductive datasets. Mehdi Ali and Max Berrendorf implemented the main part of the source code. Mehdi Ali conducted the experiments, and Mehdi Ali, Max Berrendorf, and Mikhail Galkin evaluated the results. Mehdi Ali, Max Berrendorf and Mikhail Galkin wrote the initial manuscript. All authors revised the manuscript.

5. **Mehdi Ali**, Charles Tapley Hoyt, Daniel Domingo-Fernández, Jens Lehmann, and Hajira Jabeen. "BioKEEN: a library for learning and

evaluating biological knowledge graph embeddings”. In: *Bioinformatics* 35.18 (Feb. 2019), pp. 3538–3540. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz117. URL: <https://doi.org/10.1093/bioinformatics/btz117>

The idea of BioKEEN and showcasing a concrete biomedical application was developed by Mehdi Ali and further conceptualized together with Charles Tapley Hoyt and Daniel Domingo-Fernández. Mehdi Ali implemented the machine learning functionalities, Charles Tapley Hoyt and Daniel Domingo-Fernández implemented the adapters for integrating the biomedical datasets. Mehdi Ali ran the experiments, and Mehdi Ali, Charles Tapley Hoyt and Daniel Domingo-Fernández evaluated the results. Mehdi Ali, Charles Tapley Hoyt, and Daniel Domingo-Fernández wrote the manuscript, and all authors revised the manuscript.

6. Vinay Srinivas Bharadhwaj, **Mehdi Ali**, Colin Birkenbihl, Sarah Mubeen, Jens Lehmann, Martin Hofmann-Apitius, Charles Tapley Hoyt, and Daniel Domingo-Fernández. “CLEP: a hybrid data- and knowledge-driven framework for generating patient representations”. In: *Bioinform.* 37.19 (2021), pp. 3311–3318. DOI: 10.1093/bioinformatics/btab340. URL: <https://doi.org/10.1093/bioinformatics/btab340>

Daniel Domingo-Fernández and Charles Tapley Hoyt conceived and designed the study. Vinay Srinivas Bharadhwaj implemented CLEP and ran the experiments with supervision and support from Daniel Domingo-Fernández. Mehdi Ali guided and supported the implementation of the knowledge graph embedding generation and classification tasks. Colin Birkenbihl assisted with data and method development. Sarah Mubeen processed the knowledge graph. All the authors contributed to the writing of the manuscript. All authors have read and approved the final manuscript

1.4.2 Other publications

1. **Mehdi Ali**, Sahar Vahdati, Shruti Singh, Sourish Dasgupta, and Jens Lehmann. “Improving Access to Science for Social Good”. In: *Machine Learning and Knowledge Discovery in Databases - International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part I*. vol. 1167. Communications in Computer and Information Science. Springer, 2019, pp. 658–673. DOI: 10.1007/978-3-030-43823-4_52. URL: https://doi.org/10.1007/978-3-030-43823-4_52
2. **Mehdi Ali**, Charles Tapley Hoyt, Daniel Domingo-Fernández, and Jens Lehmann. “Predicting Missing Links Using PyKEEN”. in: *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and*

Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019. Vol. 2456. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 245–248. URL: <http://ceur-ws.org/Vol-2456/paper64.pdf>

3. Veronika Henk, Sahar Vahdati, Mojtaba Nayyeri, **Mehdi Ali**, Hamed Shariat Yazdi, and Jens Lehmann. “Metaresearch recommendations using knowledge graph embeddings”. In: *RecNLP workshop of AAAI Conference*. 2019

1.5 Thesis Structure

In the following, we describe the structure of this thesis, comprised of 7 chapters.

Chapter 1 In Chapter 1, we present the introduction to this thesis covering the motivation, problem statement, challenges, research questions, and contributions made in this thesis. It further lists the publications building the foundation of the thesis and presents its theoretical background.

Chapter 2 In Chapter 2, we present the KEEN Universe, an ecosystem for KGEMs. Each component of the KEEN Universe, i.e., PyKEEN, BioKEEN and the KEEN Model Zoo, is presented. The KEEN Universe builds the foundation for PyKEEN 1.0 and, therefore, the foundation for the research questions **RQ1** and **RQ2**.

Chapter 3 In Chapter 3, we present PyKEEN 1.0, a community effort in which we redesigned and re-implemented PyKEEN to be fully configurable. PyKEEN 1.0 presents the most extensive KGEMs software library and has been used to investigate and answer **RQ1** and **RQ2**.

Chapter 4 In Chapter 4, we present our KGEM-based link prediction reproducibility study and the most extensive benchmarking study performed in order to answer **RQ1** and **RQ2**. We demonstrate which experiments can (cannot) be reproduced and discuss potential factors that hamper reproduction. Then, in the context of our benchmarking study, we highlight that the performance of a KGEM is often not solely dependent on its interaction model but on the interplay of the interaction model, training approach, loss function, and the usage/avoidance of inverse relations.

Chapter 5 In Chapter 5, we present our work addressing **RQ3**. We present our theoretical framework for classifying (existing) inductive link prediction

settings, present a novel set of benchmark datasets for inductive link prediction and our study in which we demonstrate that hyper-relational information improves inductive link prediction.

Chapter 6 In Chapter 6, we present two applications of knowledge graph representation learning for the biomedical domain. First, we present BioKEEN, an extension of PyKEEN that supports bioinformaticians lacking expertise in KGEMs and their implementation to apply KGEMs effectively. We present an application of BioKEEN in which cross-talks and hierarchies are predicted between biological pathways. We highlight that BioKEEN can effectively be applied to predict novel links. Second, we present CLEP, an approach for learning representations of patients that are incorporated into KGs that express prior biomedical knowledge. We show that the learned representations outperform patients’ raw representations in downstream classification tasks.

Chapter 7 In Chapter 7, we summarise the contributions presented in this thesis and provide an outlook for future works.

1.6 Foundations

1.6.1 General Notation

In the context of this thesis, scalars are expressed as lowercase letters x , vectors as bold-face lowercase letters \mathbf{x} , matrices as bold-face uppercase letters \mathbf{X} , three-mode tensors as fraktur font uppercase letters \mathfrak{X} , and the l_p -norm of vectors as $\|\mathbf{x}\|_p$. To express the conjugate of a complex number $x \in \mathbb{C}$, we use \bar{x} . Finally, we use \odot to represent the Hadamard product: $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$: $[\mathbf{x} \odot \mathbf{y}]_i = \mathbf{x}_i \cdot \mathbf{y}_i$ [14].

1.6.2 Knowledge Graphs

A KG is defined as a directed multi-relational graph $\mathcal{K} \subseteq \mathbb{K} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ where \mathcal{E} represents the set of nodes/entities and \mathcal{R} the set of edges/relations. KGs contain triples $(h, r, t) \in \mathcal{K}$ in which $h, t \in \mathcal{E}$ denote the head and tail entities and $r \in \mathcal{R}$ their respective relation [14, 11]. For instance, in the triple $(John, graduated_from, University\ of\ Oxford)$ contained in the exemplary KG depicted in Figure 1.1, *Peter* is the head entity, *University of Oxford* the tail entity, and *graduated_from* represents the relation.

While triples in KGs usually represent true triples, there are different *assumptions* about triples that are not part of a specific KG. According to the closed world assumption (CWA), triples which are not contained in the KGs are considered as non-existing triples, i.e., false triples. For instance, considering our KG in Figure 1.1, the triple $(Gabriele, lives_in, Germany)$ is considered as a false triple according the CWA because it is not contained in the KG. According to the open world assumption (OWA), triples not contained

in the KGs are regarded as triples, which might be true or false. Therefore, the previously mentioned exemplary triple is not considered to be necessarily false.

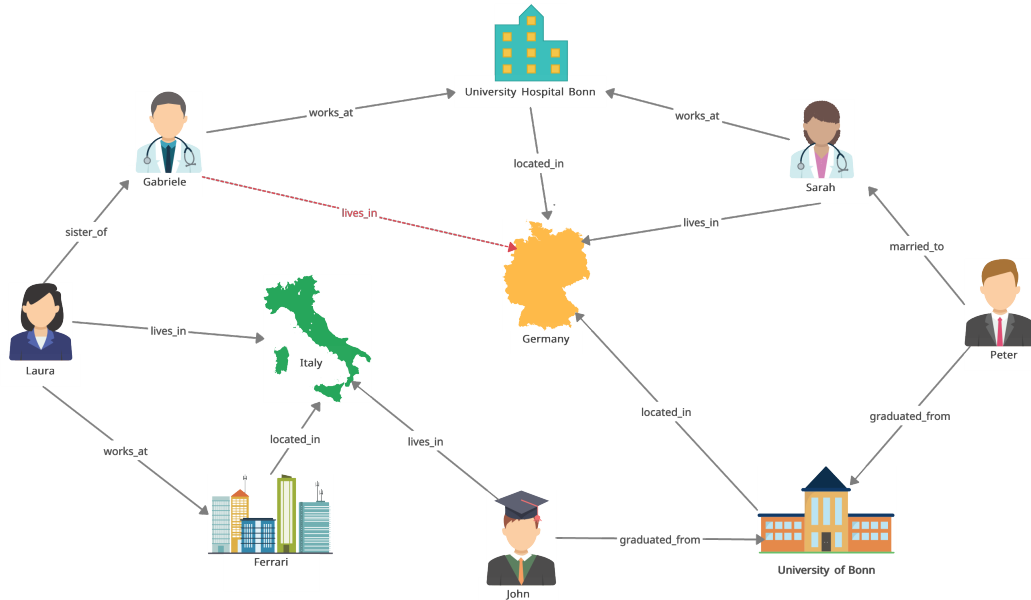


FIGURE 1.1: An exemplary KG. Nodes represent entities and edges their relations.

So far, we have focused on triples. However, KGs such as Wikidata provide for a proportion of triples an additional set of *key-value* pairs which are called qualifiers according to the Wikidata statement model or *triple metadata* in RDF* [18, 30]. Triples associated with qualifiers are called hyper-relational facts. An exemplary hyper-relational fact is depicted in Figure 1.2. The triple is associated with two qualifier pairs: (*Degree*, *Master of Science*) and (*Academic Major*, *Computer Science*).

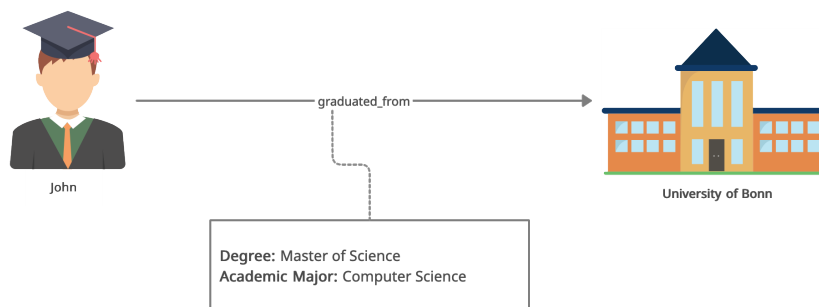


FIGURE 1.2: An exemplary hyper-relational fact. The triple is associated with the qualifier pairs (*Degree*, *Master of Science*) and (*Academic Major*, *Computer Science*).

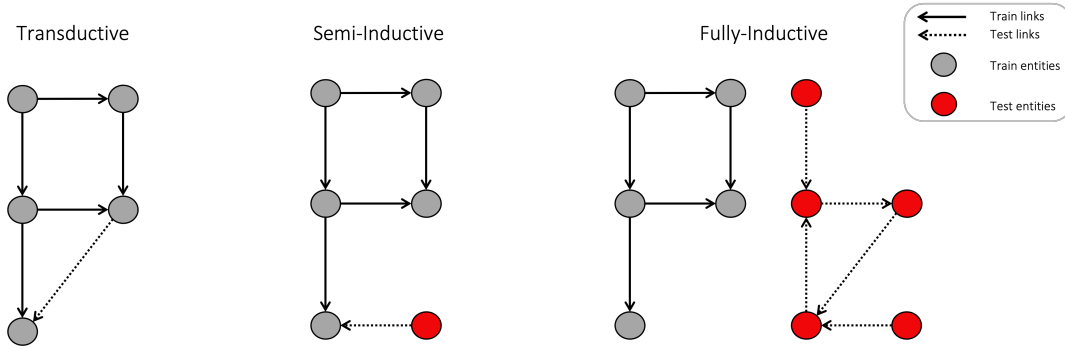


FIGURE 1.3: Transductive vs. inductive link prediction scenarios. Grey nodes represent entities seen during training, and red nodes are entities seen during inference. Dashed edges represent edges to be predicted during inference.

1.6.3 Link Prediction

Since real-world KGs are usually incomplete (and noisy), numerous approaches have been developed to predict missing links. Link prediction is the task of predicting missing tail/head entities of $(h,r)/(r,t)$ -pairs. Let \mathcal{E}_\bullet represent the set of training entities, i.e., they appear as head or tail entities in the set of training triples $\mathcal{T}_{train} \subseteq \mathcal{E}_\bullet \times \mathcal{R} \times \mathcal{E}_\bullet$, and $\mathcal{E}_\circ \subseteq \mathcal{E}$ the set of unseen entities. Depending on whether the evaluation triples (validation and/or test triples) \mathcal{T}_{eval} involve unseen entities, the link prediction setting is classified either as a *transductive* or *inductive* link prediction setting [17].

Transductive Link Prediction

In transductive link prediction, the evaluation triples \mathcal{T}_{eval} involve only seen entities, i.e., $\mathcal{T}_{eval} \subseteq \mathcal{E}_\bullet \times \mathcal{R} \times \mathcal{E}_\bullet$. Let the exemplary KG in Figure 1.1 be the training graph, then predicting the tail entity *Germany*, for the head-relation pair $(\textit{Gabriele}, \textit{lives_in}, ?)$ is an example for predicting a transductive link since both the head and tail entity (and the relation) have been seen during training [14].

Inductive Link Prediction

In contrast to transductive link prediction, inductive link prediction involves entities/relations during inference that are not seen during training (see Figure 1.3). In the context of this work, we focus on unseen entities, but the approaches can be generalised to unseen relations, too. Different inductive link prediction scenarios have been proposed in the literature, and in several cases, the same concepts are named differently. Therefore, we follow the classification presented in [17], which provides a unified categorisation of the approaches. In particular, it classifies all inductive settings into the *semi-inductive* or/and *fully-inductive* setting.

Semi-Inductive Link Prediction In the semi-inductive link prediction setting, links between seen and unseen entities are predicted, i.e., $\mathcal{T}_{eval} \subseteq \mathcal{E}_\bullet \times \mathcal{R} \times \mathcal{E}_\circ \cup \mathcal{E}_\circ \times \mathcal{R} \times \mathcal{E}_\bullet$. The second example in Figure 1.3 depicts the semi-inductive setting. The link between the test entity (red node) and a grey node (training entity) shall be predicted. In the context of this thesis, we focus on the case in which no additional information for unseen entities in the semi-inductive setting is provided. However, the so called *k-shot* learning scenario setting is proposed where for unseen entities, k links to training entities are provided [31, 32, 33].

Fully-Inductive Link Prediction In the fully-inductive link prediction setting, links are predicted within an unseen graph $\mathcal{T}_{inf} \subseteq \mathcal{E}_\circ \times \mathcal{R} \times \mathcal{E}_\circ$ where the set of relations is a subset of the relations of the training graph, but all the entities in the graph are unseen. The last example in Figure 1.3 illustrates the described scenario. The graph comprised of the unseen entities is the inference graph \mathcal{T}_{inf} , and links are only predicted within this graph. Because the relations are shared across the training and the inference graph, the information learned on the training graph can be transferred to the inference graph.

1.6.4 Knowledge Graph Embedding Models

In the preceding section, we introduced the task of link prediction. Different approaches have been proposed to tackle link prediction covering rule-based and machine-learning-based approaches. In the last decade, one research field that received increased popularity is the field of KGEMs [11, 14]. KGEMs learn to embed entities and relations of a KG into a latent feature space/latent feature spaces while best-preserving its inherent structure [11, 12, 14]. We follow the definition presented in [14] and define a KGEM as a composition of an *interaction model* (Section 1.6.4), a *loss function* (Section 1.6.5), a *training approach* (Section 1.6.6), and the usage of *inverse relations* (Section 1.6.7). This definition allows us to dissect the performance of KGEMs based on their individual components as well as their combination. In the following, we describe each of the components in detail.

Interaction Models

An interaction model is a function $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$ of triples that computes real-valued scores representing the plausibility that the triple is true given the embeddings of the head/tail entities and relations [14, 11]. In the context of this thesis, a higher score corresponds to a higher plausibility. Interaction models can be classified into *translational distance* based and *semantic matching* based interaction models [12]. Translational distance interaction models employ distance functions such as the Euclidean distance to compute the plausibility of a triple, and semantic matching interaction models employ similarity-based functions (e.g., inner product) [12].

In the following, we rely on the notation presented in Ali *et al.* [14]. When referring to head/tail entities or relations, we refer to their embeddings \mathbf{h} , \mathbf{t} and \mathbf{r} . If not indicated otherwise, we assume that they are elements of \mathbb{R}^d .

Translational Distance Interaction Models

Unstructured Model Bordes *et al.* [34] introduced the Unstructured Model that computes the Euclidean distance between the head and tail entities omitting the relation to compute the plausibility score [14]:

$$f(h, t) = -\|\mathbf{h} - \mathbf{t}\|_2^2, \quad (1.1)$$

Because the relations are ignored, UM does not address the multi-relational aspect of KGs. For instance, UM would compute for the triples $(A, \text{fatherOf}, B)$ and $(A, \text{motherOf}, B)$ the same score which is clearly a limitation. Nonetheless, the approach can be helpful when the KG is comprised solely of a single relationship type such as *friendOf* in simplified social networks.

Structured Embedding In Structured Embedding [35], each relation is represented by two matrices $\mathbf{M}_r^h, \mathbf{M}_r^t \in \mathbb{R}^{d \times d}$, and are used to project head and tail entity of a triple separately [14]:

$$f(h, r, t) = -\|\mathbf{M}_r^h \mathbf{h} - \mathbf{M}_r^t \mathbf{t}\|_1. \quad (1.2)$$

Therefore, the role of an entity in a triple as a head or tail entity is explicitly addressed.

TransE In TransE [36], relations are used to translate head entities to tail entities by enforcing $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ [14]:

$$f(h, r, t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_p, \quad (1.3)$$

where $p \in \{1, 2\}$ represents a hyper-parameter. Despite its simplicity and efficiency in modeling multi-relational data, it cannot model $1-N$, $N-1$, and $N-M$ relations. Let $(h, r, t_1), (h, r, t_2) \in \mathcal{K}$ be two triples part of a KG. Given the constraint $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, it follows $\mathbf{h} + \mathbf{r} \approx \mathbf{t}_1$ and $\mathbf{h} + \mathbf{r} \approx \mathbf{t}_2$ resulting in $\mathbf{t}_1 \approx \mathbf{t}_2$.

TransH Wang *et al.* [37] addressed the limitation of TransE and proposed TransH. TransH follows the idea of TransE and interprets each relation as a translation from the head to the tail entity. However, the translation is performed in a relation-specific hyperplane [14]:

$$f(h, r, t) = -\|\mathbf{h}_r + \mathbf{d}_r - \mathbf{t}_r\|_2^2. \quad (1.4)$$

where $w_r \in \mathbb{R}^d$ denotes the relation-specific normal vector of a hyperplane and projects the entities into the hyperplane: $\mathbf{h}_r = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h}$, $\mathbf{t}_r = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t}$, and $d_r \in \mathbb{R}^d$ denotes the relation-specific translation vector in the hyperplane.

TransR Lin *et al.* [38] argue that entities and relations are different objects and, therefore, should be represented in distinct spaces. They propose TransR, an extension of TransH that models entities and relations in different vector spaces [14]:

$$f(h, r, t) = -\|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_2^2 \quad (1.5)$$

where $\mathbf{h}_r = \mathbf{M}_r \mathbf{h}$, $\mathbf{t}_r = \mathbf{M}_r \mathbf{t}$, $r \in \mathbb{R}^k$, and $\mathbf{M}_r \in \mathbb{R}^{k \times d}$ represents a relation-specific projection matrix that projects the entities into the relational space.

TransD Ji *et al.* [39] proposed TransD, an extension of TransR that creates entity-relation-specific projection matrices to project entities into the relational space instead of sharing the same projection matrix of a relation r with all entities. The reasoning is that entities are of different types and characterised by distinct attributes. Consequently, not all entities should be treated equally by projecting them based on the same relational projection matrix [39]. Each entity and relation is represented by two vectors, $\mathbf{h}, \mathbf{h}_p, \mathbf{t}, \mathbf{t}_p \in \mathbb{R}^d$ and $\mathbf{r}, \mathbf{r}_p \in \mathbb{R}^k$. One represents its semantics $(\mathbf{h}, \mathbf{t}, \mathbf{r})$, and one is used to construct the projection matrices $(\mathbf{h}_p, \mathbf{t}_p, \mathbf{r}_p)$: $\mathbf{M}_{r,h} = \mathbf{r}_p \mathbf{h}_p^T + \tilde{\mathbf{I}}$ and $\mathbf{M}_{r,t} = \mathbf{r}_p \mathbf{t}_p^T + \tilde{\mathbf{I}}$ where $\mathbf{M}_{r,t}, \mathbf{M}_{r,h} \in \mathbb{R}^{k \times d}$, and $\tilde{\mathbf{I}} \in \mathbb{R}^{k \times d}$ that is comprised with ones on the diagonal and zeros everywhere else. The plausibility score for the triple is computed using the projected head and tail entities [14]:

$$f(h, r, t) = -\|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_2^2 . \quad (1.6)$$

RotatE RotatE [40] is a complex-valued interaction model that models each relation as rotation from the head to tail entity [14], i.e., $\mathbf{t} = \mathbf{h} \odot \mathbf{r}$. The plausibility score for a triple is computed as follows:

$$f(h, r, t) = -\|\mathbf{h} \odot \mathbf{r} - \mathbf{t}\| , \quad (1.7)$$

where $\mathbf{h}, \mathbf{r}, \mathbf{t}$ are complex-valued embeddings, i.e. $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$, and $|r_i| = 1$ allowing to represent r_i as $e^{i\theta_{r,i}}$ representing a counterclockwise rotation by $\theta_{r,i}$ radians. Overall, Rotate is capable of modelling *symmetry/anti-symmetry*, *inversion* and *composition* patterns.

MuRE MuRE [41] is the Euclidean realisation of the hyperbolic interaction model MuRP [41]. MuRE computes the distance between the transformed head and tail entities and adds scalar offsets [14] to the computed distance:

$$f(h, r, t) = -\|\mathbf{R}\mathbf{h} - \mathbf{t} + \mathbf{r}\|_2^2 + \mathbf{b}_h + \mathbf{b}_t \quad (1.8)$$

where $\mathbf{R} \in \mathbb{R}^{d \times d}$ is a diagonal matrix and \mathbf{b}_h and $\mathbf{b}_t \in \mathbb{R}$.

KG2E KG2E [42] explicitly models uncertainties in the entity and relation embeddings by modelling entities and relations as probability distributions. Each entity and relation is modelled by a multi-variate Gaussian $\mathcal{N}_i(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ where the mean indicates the embedding's position in the vector space and the diagonal covariance matrix expresses its uncertainty. A triple (h, r, t) obtains a high plausibility score when the distributions between $\mathcal{P}_e = \mathcal{N}_h(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h) - \mathcal{N}_t(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) = \mathcal{N}_{h-t}(\boldsymbol{\mu}_h - \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_h + \boldsymbol{\Sigma}_t)$ and $\mathcal{P}_r = \mathcal{N}_r(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$ are similar, where \mathcal{N}_h , \mathcal{N}_r and \mathcal{N}_t denote the distributions of the head entity, relation and tail entity. To compute the similarity between \mathcal{P}_e and \mathcal{P}_r , the Kullback-Leibler (KL) divergence between both distribution is computed:

$$\begin{aligned}
f(h, r, t) &= \mathcal{D}_{\mathcal{KL}}(\mathcal{P}_e, \mathcal{P}_r) \\
&= \frac{1}{2} \left\{ \text{tr}(\boldsymbol{\Sigma}_r^{-1} \boldsymbol{\Sigma}_e) + (\boldsymbol{\mu}_r - \boldsymbol{\mu}_e)^T \boldsymbol{\Sigma}_r^{-1} (\boldsymbol{\mu}_r - \boldsymbol{\mu}_e) \right. \\
&\quad \left. - \log \left(\frac{\det(\boldsymbol{\Sigma}_e)}{\det(\boldsymbol{\Sigma}_r)} \right) - d \right\}.
\end{aligned} \tag{1.9}$$

where tr denotes the trace operation. In addition to the asymmetric interaction model based on the KL divergence, a symmetric interaction model based on the expected likelihood is proposed.

Semantic Matching Interaction Models

RESCAL RESCAL [43] is a tensor factorisation approach in which entities are represented as vectors and relations as matrices $\mathbf{W}_r \in \mathbb{R}^{d \times d}$. The relation matrices can be considered as weight matrices that define the extent of interaction between the head and tail entities' latent features in that particular relation [14]:

$$f(h, r, t) = \mathbf{h}^T \mathbf{W}_r \mathbf{t} = \sum_{i=1}^d \sum_{j=1}^d w_{ij}^{(r)} h_i t_j \tag{1.10}$$

DistMult DistMult [44] is similar to RESCAL, but the relation matrices are diagonal. Though, reducing the time complexity of RESCAL from $\mathcal{O}(K^2)$ to $\mathcal{O}(K)$. The interaction model is defined as [45, 14]:

$$f(h, r, t) = \mathbf{h}^T \mathbf{W}_r \mathbf{t} = \sum_{i=1}^d \mathbf{h}_i \cdot \text{diag}(\mathbf{W}_r)_i \cdot \mathbf{t}_i. \tag{1.11}$$

While the simplification of RESCAL introduced through DistMult improves the computational efficiency, it reduces its expressivity since it considers all relations as symmetric relations, i.e., $f(h, r, t) = f(t, r, h)$.

Complex ComplEx [45] addresses the limitation of DistMult by representing entities and relations as complex-valued vectors $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$. It uses the Hadamard product to compute the plausibility scores of triples [14]:

$$f(h, r, t) = \text{Re}(\mathbf{h} \odot \mathbf{r} \odot \mathbf{t}), \tag{1.12}$$

where $\text{Re}(x)$ is the real-valued part of the complex-valued score. Compared to DistMult, ComplEx is capable of modelling anti-symmetric relations since the Hadamard product is not a commutative operation in the complex space, i.e., in general $f(h, r, t) \neq f(t, r, h)$.

QuatE QuatE [46] models entities and relations as hypercomplex-valued vectors, i.e., $\mathbf{e}_i, \mathbf{r}_j \in \mathbb{H}^d$. Each hypercomplex number is represented by a real and three imaginary parts. The interaction model is defined as [14]:

$$f(h, r, t) = \mathbf{h} \otimes \mathbf{r} \cdot \mathbf{t}, \tag{1.13}$$

where \otimes denotes the Hamilton product and $\mathbf{h} \otimes \mathbf{r}$ represents a rotation of the head entity performed by the relation representation.

Simple Simple [47] extends the tensor factorisation approach **canonical polyadic (CP)** [47]. CP models each entity $e_i \in \mathcal{E}$ by two vectors $\mathbf{h}_e, \mathbf{t}_e \in \mathbb{R}^d$ where \mathbf{h}_e is used when e occurs as the head entity in a triple and \mathbf{t}_e when it occurs as the tail entity. The limitation of this approach is that \mathbf{h}_e and \mathbf{t}_e are learned independently, i.e., information learned for the entity $e_i \in \mathcal{E}$ based on $(e_i, r_1, e_j) \in \mathcal{K}$ cannot be exploited when predicting $(e_k, r_2, e_i) \in \mathcal{K}$. To overcome this limitation, Simple add for each relation also its inverse relation r' to the set of relations. The interaction model simultaneously employs the participating relation r of a triple $(h, r, t) \in \mathcal{K}$ and its inverse relation r' (and the corresponding inverse triple) in order to propagate the information through both representations of the participating entities [14]:

$$f(h, r, t) = \frac{1}{2} \left(\langle \mathbf{h}_{e_h}, \mathbf{r}, \mathbf{t}_{e_t} \rangle + \langle \mathbf{h}_{e_h}, \mathbf{r}', \mathbf{t}_{e_t} \rangle \right) . \quad (1.14)$$

where $\langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle = \sum_i^d h_i * r_i * t_i$.

Tucker Tucker [48] relies on the tensor factorisation method Tucker [49], i.e., it decomposes the three-mode adjacency tensor representing a KG into a set of factor matrices and a core tensor of lower rank. The entity matrix $\mathbf{E} = \mathbf{A} = \mathbf{C} \in \mathbb{R}^{n_e \times d_e}$ and the relation matrix $\mathbf{R} = \mathbf{B} \in \mathbb{R}^{n_r \times d_r}$ represent the factor matrices. $\mathfrak{W} = \mathfrak{Z} \in \mathbb{R}^{d_e \times d_r \times d_e}$ denotes the core tensor and expresses the amount of interaction between entities and relations. The plausibility score of a triple is computed as follows [14]:

$$f(h, r, t) = \mathfrak{W} \times_1 \mathbf{h} \times_2 \mathbf{r} \times_3 \mathbf{t} , \quad (1.15)$$

where the entity representations \mathbf{h} and \mathbf{t} are rows from \mathbf{E} , and the relation representation \mathbf{r} is a row from \mathbf{R} . \times_n expresses the tensor product, where n indicates the tensor mode used to compute the product.

ProjE ProjE [50] is characterised by two major operations, a *combination* operation \otimes and a *projection* operation. The combination operator combines the embeddings of the head entities and the relations: $\mathbf{h} \otimes \mathbf{r} = \mathbf{D}_e \mathbf{h} + \mathbf{D}_r \mathbf{r} + \mathbf{b}_c$. The combination operator makes use of $\mathbf{D}_e, \mathbf{D}_r \in \mathbb{R}^{k \times k}$ representing diagonal matrices shared across all entities and relations, and $\mathbf{b}_c \in \mathbb{R}^k$ denoting a shared bias vector. The plausibility score of a triple is computed as [14]:

$$f(h, r, t) = g(\mathbf{t} z(\mathbf{h} \otimes \mathbf{r}) + \mathbf{b}_p) , \quad (1.16)$$

where g and h denote non-linear activation functions, and \mathbf{b}_p the projection bias.

HolE In Holographic embeddings (HolE) [51], the plausibility scores of triples are computed using the circular correlation operator $\star : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ [51], where $\mathbf{a} \star \mathbf{b}$ is computed as $[\mathbf{a} \star \mathbf{b}]_i = \sum_{k=0}^{d-1} \mathbf{a}_k * \mathbf{b}_{(i+k) \bmod d}$. The interaction model is defined as [14]:

$$f(h, r, t) = \sigma(\mathbf{r}^T(\mathbf{h} \star \mathbf{t})) \quad (1.17)$$

ERMLP ERMLP [52] is a feed-forward neural network with a hidden layer $\mathbf{W} \in \mathbb{R}^{k \times 3d}$ that expects as input the concatenation of the head, relation and tail representation of a triple $(h, r, t) \in \mathcal{K}$, and an output layer $\mathbf{w} \in \mathbb{R}^k$ that computes the plausibility score of the triples [14]:

$$f(h, r, t) = \mathbf{w}^T g(\mathbf{W}[\mathbf{h}; \mathbf{r}; \mathbf{t}]), \quad (1.18)$$

where g denotes a non-linear activation function.

Neural Tensor Network The Neural Tensor Network (NTN) [53] represents each entity as a vector, i.e., $\mathbf{e}_i \in \mathbb{R}^d$, and each relation is represented by the parameters of a bilinear tensor network. Therefore, for each relation, a relation specific tensor $\mathfrak{W}_r \in \mathbb{R}^{d \times d \times k}$ is defined that relates the head and tail representations of a triple [14]:

$$f(h, r, t) = \mathbf{u}_r^T \cdot \tanh(\mathbf{h}\mathfrak{W}_r\mathbf{t} + \mathbf{V}_r[\mathbf{h}; \mathbf{t}] + \mathbf{b}_r) , \quad (1.19)$$

where $\mathbf{V}_r \in \mathbb{R}^{k \times 2d}$ and $\mathbf{b}_r \in \mathbb{R}^k$ are the usual linear projection matrix, and the bias vector of a hidden feed-forward network, and $\mathbf{u}_r \in \mathbb{R}^k$ represents the weights of the output layer.

ConvKB ConvKB [54] is a convolutional neural network-based approach. The embeddings of a triple $(h, r, t) \in \mathcal{K}$ are organised column-wise into a matrix $\mathbf{A} = [\mathbf{h}; \mathbf{r}; \mathbf{t}] \in \mathbb{R}^{d \times 3}$ on which convolutional filters $\omega_i \in \mathbb{R}^{1 \times 3}, i = 1, \dots, \tau$ create a set of feature maps. Every convolutional filter convolves row-wise over \mathbf{A} and generates a feature map $\mathbf{v}_i = [v_{i,1}, \dots, v_{i,d}] \in \mathbb{R}^d$ capturing interactions between the embeddings of a triple. In order to compute the plausibility score of a triple, the feature maps are concatenated and the dot-product with a shared weight vector $\mathbf{w} \in \mathbb{R}^{\tau d \times 1}$ is computed [14]:

$$f(h, r, t) = [\mathbf{v}_1; \dots; \mathbf{v}_\tau] \cdot \mathbf{w} , \quad (1.20)$$

where $[\mathbf{v}_1; \dots; \mathbf{v}_\tau] \in \mathbb{R}^{\tau d \times 1}$.

ConvE ConvE [55] is like ConvKB, a convolutional neural network-based interaction model. The head and relation representation of a triple $(h, r, t) \in \mathcal{K}$ are organised row-wise into a matrix $\mathbf{A} \in \mathbb{R}^{2 \times d}$. Before \mathbf{A} is passed to the convolutional-layer, it is reshaped into a matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$. The first $m/2$ half rows of \mathbf{B} correspond to the embedding of h , and the second half of the rows to the embedding of r . \mathbf{B} is processed by a convolutional layer where 2-dimensional convolutional filters $\Omega = \{\omega_i \mid \omega_i \in \mathbb{R}^{r \times c}\}$ generate a set of feature maps that reshaped and concatenated to form the feature vector $\mathbf{v} \in \mathbb{R}^{|\Omega|rc}$. The obtained feature vector \mathbf{v} is transformed into $\mathbf{e}_{h,r} = \mathbf{v}^T \mathbf{W}$ by a linear-transformation $\mathbf{W} \in \mathbb{R}^{|\Omega|rc \times d}$, and used together with the tail representation \mathbf{t} to compute the plausibility score [14]:

$$f(h, r, t) = \mathbf{e}_{h,r} \mathbf{t} . \quad (1.21)$$

1.6.5 Loss Functions

It has been shown that the impact of loss functions on a KGEM's performance can be decisive [56, 14]. Typically, loss functions used to train KGEMs belong

to the class of *pointwise*, *pairwise*, and *setwise* loss functions [14]. We follow the categorisation and notation presented in [14] to describe the loss functions. Though, $t_i \in \mathbb{K}$ represents a positive/negative triple, and f denotes the interaction model of a KGEM.

Pointwise Loss Functions

In pointwise loss functions, the loss term is computed between single triples and their binary labels, indicating whether the triple is correct or false:

$$\mathcal{L} = \frac{1}{|B|} \sum_{(f(t_i), l_i) \in B} L(t_i, l_i) \quad (1.22)$$

where $l_i \in \{0, 1\}$ or $\hat{l}_i \in \{-1, 1\}$ represents triple's i -th label (1 represents the label of positive and 0/-1 of negative triples), and B represents a batch of triples. Typical pointwise loss functions are the *square error loss*, *binary cross entropy loss*, *pointwise logistic loss*, and the *pointwise hinge loss*.

Pairwise Loss Functions

In pairwise loss functions, the loss terms are computed between pairs of triples [14]. Mostly applied pairwise loss functions for training KGEMs are the *pairwise hinge loss* and the *pairwise logistic loss* [14] which pairwise combine scores of positive triples and scores of negatives:

$$\mathcal{L} = \frac{1}{|B|} \sum_{(t_i^+, t_i^-) \in B} L(f(t_i^-) - f(t_i^+)) , \quad (1.23)$$

where $f(t_i^-)$ represent the plausibility score of the positive triple and $f(t_i^+)$ the score of the negative one. The positive and negative triples are usually related to each other, i.e., the negative triple is obtained by corrupting the positive one. Because the loss function does not involve negative labels but compares negative triples to related positive ones, i.e., negative triples are not considered as false but as less positive than related positive triples, the loss function naturally complies with the OWA of KGs.

Setwise Loss Functions

In setwise loss functions, the scores of a set of related triples (usually $n > 2$) are compared. Typical setwise loss functions that are used to training KGEMs are *self-adversarial negative sampling loss (NSSAL)* [40] and the *cross entropy loss (CEL)* [55, 56].

1.6.6 Training Approaches

There are two widely employed training approaches to train KGEMs: the local closed world assumption (LCWA) and the stochastic local closed world

assumption (sLCWA) [14]. Both rely on different assumptions about negative training examples, which we will explain in the following.

Local closed world assumption

Based on the LCWA, each triple $(h, r, t) \in \mathcal{K}$ all triples $(h, r, t_i) \notin \mathcal{K}$ are considered as false triples and represented in a set $\mathcal{T}^-(h, r)$. Analogous to $\mathcal{T}^-(h, r)$, the sets $\mathcal{H}^-(r, t)$ and $\mathcal{R}^-(h, t)$ of negative triples can be created in which all triples $(h_i, r, t) \notin \mathcal{K}$, and $(h, r_i, t) \notin \mathcal{K}$ are contained. Though, to train KGEMs based on the LCWA, usually $\mathcal{T}^-(h, r)$ is considered as the set of negative triples [14].

Stochastic local closed world assumption

Based on the sLCWA, not all triples contained in the sets $\mathcal{T}^-(h, r)$, $\mathcal{H}^-(r, t)$, and $\mathcal{R}^-(h, t)$ are considered to be false. Instead, negative triples are samples from these sets. In practice, negative triples are created by corrupting positive ones, i.e., randomly replacing h , r or t of a triple $(h, r, t) \in \mathcal{K}$. Two widely applied approaches are the uniform negative sampling (UNS) [36] and Bernoulli negative sampling (BNS) [37], which corrupt the tail/head entity a positive triple. The set of all potentially negative triples is defined as [14]:

$$\mathcal{N} = \bigcup_{(h,r,t) \in \mathcal{K}} \mathcal{T}(h,r) \cup \mathcal{H}(r,t), \quad (1.24)$$

where $\mathcal{T}(h,r) = \{(h,r,t') \mid t' \in \mathcal{E} \wedge t' \neq t\}$ denotes all negative triples where for a positive triple $(h,r,t) \in \mathcal{K}$ the tail entity t is corrupted and similarly $\mathcal{H}(r,t) = \{(h',r,t) \mid h' \in \mathcal{E} \wedge h' \neq h\}$ denotes the sets of triples where the head entity h is corrupted [14].

The set \mathcal{N} of potentially negative triples can contain positives, i.e., when corrupting a positive triple $(h,r,t) \in \mathcal{K}$, a positive example $(h,r,t_i) \in \mathcal{K}$ and $(h,r,t) \in \mathcal{K}$ can be sampled. Therefore, theoretically, it would be necessary to remove known false negatives from \mathcal{N} , but because of the compute costs of the filtering step and considering the fact that there is a relatively low probability to sample a true negative since often $|\mathcal{N}| \gg |\mathcal{K}|$, the filtering step is skipped [14].

1.6.7 Explicitly Modelling Inverse Relations

When explicitly modelling inverse relations, for each triple $(h, r, t) \in \mathcal{K}$ an inverse triple (t, r_{inv}, h) is introduced where r_{inv} denotes the inverse relation of r and is added to the set of relations \mathcal{R} [47, 57]. In addition to introducing inverse relations, the behaviour of the interaction model is adapted: predicting the head entities for (r, t) -pairs is performed by predicting the tail entities for (t, r_{inv}) -pairs. It has been shown that involving inverse relations and the adapted behaviour of the interaction model can improve the performance [57] and the computational efficiency of dedicated models [55]

1.6.8 Graph Neural Networks

GNNs represent neural network architectures tailored to graph-structured data. In general, they rely on node features and learn to encode instead of learning fixed embeddings and therefore are inherently inductive compared to KGEMs [13]. Typically applications of GNNs are node classification, graph classification, and link prediction. The key concept of GNNs is *message passing* in which representations/messages are interchanged between neighbouring nodes. In the context of message passing, a node \mathbf{h}_i is updated based on *aggregate* and an *update* function [13]:

$$\mathbf{h}_i^{(k+1)} = \text{update}^{(k)}(\mathbf{h}_i^{(k)}, \text{aggregate}^{(k)}(\{\mathbf{h}_v, \forall v \in \mathcal{N}\})), \quad (1.25)$$

where k denotes the k -th layer of the GNN. The aggregate function summarises the local neighbourhood of a node into a message representation $m_{\mathcal{N}(i)}^{(k)}$, and the update function combines the previous representation \mathbf{h}_i^k of the node i , and $m_{\mathcal{N}(i)}^{(k)}$ to generate the updated representation \mathbf{h}_i^{k+1} . The aggregate and the update functions are differentiable and dependent on the specific GNN. Depending on the number of layers of a GNN, information from nodes further away in the graph is integrated. Precisely, a k -layer GNN aggregates the k -hop neighbourhood of a node i [13].

1.6.9 Evaluation of Link Predictors

Link predictors are usually evaluated based on ranking metrics that measure the model's performance in ranking (known) true triples higher than corrupted triples, i.e., assigning higher scores to true triples. Therefore, the following two sets of corrupted triples are generated for each test triple $(h, r, t) \in \mathcal{K}$. The first set $\mathcal{H}(r, t)$ comprises all triples for which the head entity of the test triple is corrupted, i.e., $\mathcal{H}(r, t) = \{(h', r, t) \mid h' \in \mathcal{E} - \{h\}\}$. Similarly, the second set, $\mathcal{T}(h, r) = \{(h, r, t') \mid t' \in \mathcal{E} - \{t\}\}$, contains all triples for which the tail entity is corrupted. The model predicts the score for the test triple and the corresponding corrupted triples, and the test triple's rank/position is determined in the list of sorted scores.

It should be noticed that the corrupted triples, i.e., in $\mathcal{H}(r, t)$ and $\mathcal{T}(h, r)$ can contain known true triples from the training/validation set, which can distort the ranking in case they are ranked higher than the test triple. Therefore, the filtered setting has been introduced in which known false negatives are removed from the set of corrupted triples [36]. Furthermore, unknown false negatives can be part of the set of corrupted triples that potentially can distort the results.

In the following, we present the usually employed rank-based metrics, i.e., mean rank, mean reciprocal rank, and the hits@k. They all rely on the previously described ranks of the test triples.

The MR, computes the average rank based on all test triples [14]:

$$\text{MR} = \frac{1}{|\mathcal{K}_{test}|} \sum_{t \in \mathcal{K}_{test}} \text{rank}(t) \quad (1.26)$$

A smaller MR corresponds to better model performance. It should be noted that the computed rank of a test triple depends on the number of candidate entities in a dataset. Generally, the ranking becomes easier when a test triple is ranked against a smaller set of candidate entities. Therefore, the MRs computed for different datasets cannot be compared. A novel metric that addresses this limitation is the adjusted mean rank [58].

The MRR computes the average reciprocal rank over all test triples:

$$\text{MRR} = \frac{1}{|\mathcal{K}_{test}|} \sum_{t \in \mathcal{K}_{test}} \frac{1}{\text{rank}(t)} \quad (1.27)$$

A larger MRR corresponds to better model performance.

The hits@k computes the ratio of test triples that obtained a rank among the top k predictions:

$$\text{Hits@k} = \frac{|\{t \in \mathcal{K}_{test} \mid \text{rank}(t) \leq k\}|}{|\mathcal{K}_{test}|} \quad (1.28)$$

A larger hits@10 corresponds to better model performance.

Chapter 2

The KEEN Universe: An Ecosystem for Knowledge Graph Embeddings with a Focus on Reproducibility and Transferability

In this chapter, we present the following publication (see Appendix A):

Mehdi Ali, Hajira Jabeen, Charles Tapley Hoyt, and Jens Lehmann. “The KEEN Universe - An Ecosystem for Knowledge Graph Embeddings with a Focus on Reproducibility and Transferability”. In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*. vol. 11779. Lecture Notes in Computer Science. Springer, 2019, pp. 3–18. DOI: 10 . 1007 / 978 - 3 - 030 - 30796 - 7 _ 1. URL: https://doi.org/10.1007/978-3-030-30796-7_1

Authors’ contributions: Mehdi Ali developed the idea of an ecosystem for KGEMs that focuses on the reproducibility and transferability of KGEM research. Mehdi Ali implemented the major parts of the source code. Charles Tapley Hoyt refactored, and packaged the code, and added additional functionalities. Mehdi Ali wrote the manuscript. All authors revised the manuscript.

Summary

KGs have become a fundamental approach for representing structured information in academic and industrial applications, and link prediction is one of the major tasks within KGs [11, 1]. Therefore, in the last decade, a vast number of KGEMs have been proposed to tackle link prediction [14]. However, the performance of KGEMs is often not comprehensible since the reproduction of KGEM experiments remains a major obstacle [14]. Because KGs are employed in various domains, link prediction becomes consequently relevant for applications within these domains. One exemplary domain is biomedicine [22, 59], and exemplary applications include the prediction of drug-target [22], and protein-protein interactions [60]. Despite the tremendous potential KGEMs offer to address domain-specific use cases, their application outside the KGEM-community is limited because it requires strong expertise in KGEMs and in implementing these approaches.

To tackle the mentioned shortcomings, we developed the KEEN Universe, an ecosystem for KGEMs with a focus on reproducibility and transferability. The KEEN Universe consists of three components: the Python packages PyKEEN, BioKEEN, and the KEEN Model Zoo.

PyKEEN represents the ecosystem's underlying machine learning module (including additional functionalities for preprocessing). It has a modular architecture, and the individual modules are organised into two layers: a configuration layer and a learning layer. In the configuration layer, users can configure their experiments using an interactive command-line interface that ensures that experiments are configured correctly. This is especially helpful for users with limited experience in KGEMs. The learning layer contains the machine learning functionalities, i.e., the training approach, the evaluator, and the hyperparameter optimisation components. PyKEEN is built on top of the machine learning framework PyTorch [61] and currently covers the implementation of 10 KGEMs, the sLCWA training approach, the hits@k and MR as evaluation metrics, and random search as the hyperparameter optimisation approach. Because of the modular architecture of PyKEEN, users can easily integrate novel components, i.e., the extension of PyKEEN is facilitated. PyKEEN relies on several community standards, ensuring the quality of the library. To ensure code quality, we use *flake8*¹, and *pyroma*² to enforce metadata standards, *setuptools*³ to package our library, *sphinx*⁴ to build our documentation, *Read the Docs*⁵ to host our documentation, and *Travis-CI*⁶ as the continuous integration server.

BioKEEN is an extension of PyKEEN that is tailored to the biomedical domain to facilitate bioinformaticians' usage of KGEMs. We will further present BioKEEN in the context of Chapter 6.

¹<https://flake8.pycqa.org/en/latest/>

²<https://github.com/regebro/pyroma>

³<https://setuptools.pypa.io/en/latest/>

⁴<https://www.sphinx-doc.org/en/master/>

⁵<https://readthedocs.org/>

⁶<https://travis-ci.org/>

The KEEN Model Zoo is a platform that allows researchers to share their experimental artefacts, including their trained models, with the community fostering reproducibility in the field of KGEMs. We defined the following five requirements to ensure the quality of the model zoo: i.) the experiments need to be described in a peer-reviewed paper, ii.) all experimental artefacts created by PyKEEN/BioKEEN need to be shared, iii.) the datasets used need to be publicly accessible, iv.) a textual description of the experiment needs to be provided and, v.) and a unit test needs to be implemented ensuring that the published KGEM can be instantiated.

We evaluated the usability of the KEEN Universe by addressing a use case from the biomedical domain [23] and one from the domain of scholarly metadata research [29]. We highlighted that the use cases could be solved with effectiveness, efficiency, and satisfaction.

Concluding, the KEEN Universe represents the first step towards an extensive KGEM ecosystem. With its pivot component PyKEEN, researchers and practitioners can perform analyses of KGEMs and employ them in their downstream tasks. BioKEEN is a successful example of transferring results of KGEM research to the field of bioinformatics. A similar approach can be employed to extend PyKEEN to facilitate the usage of KGEMs in other domains. With the KEEN Universe, and in particular, with PyKEEN, we build the foundation for PyKEEN 1.0 (Chapter 3) and, therefore, the groundwork for investigating the reproducibility crisis of link prediction experiments and the groundwork for performing our large-scale benchmarking study of KGEMs presented in Chapter 4.

Chapter 3

PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings

In this chapter, we present the following publication (see Appendix B):

Mehdi Ali*, Max Berrendorf*, Charles Tapley Hoyt*, Laurent Vermue*, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. “PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings”. In: *Journal of Machine Learning Research* 22.82 (2021). * equal contribution, pp. 1–6. URL: <http://jmlr.org/papers/v22/20-825.html>

Authors’ contributions: Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Sahand Sharifzadeh, and Laurent Vermue developed the idea of abstracting and implementing KGEMs as the composition of four components that can flexibly be composed after each author identified limitations of existing works restricting the in-depth analysis of KGEMs. Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, and Laurent Vermue implemented the source code. Mehdi Ali wrote the initial manuscript, and all authors extended and finalised it.

Summary

In the previous chapter, we presented PyKEEN as part of the KEEN Universe. PyKEEN has been the first effort toward an extensive KGEM library that enables researchers and practitioners to analyse the link prediction performance of KGEMs and apply KGEMs to their downstream tasks. PyKEEN has attracted the attention of researchers, and a community of core developers has organised around the project¹. Although PyKEEN already provided rich functionalities, it had certain limitations. In particular, KGEMs were not fully composable, i.e., the arbitrary composition of an *interaction model*, *training approach*, *loss function*, and the decision to *explicitly model inverse relations* was not possible. However, in order to disentangle the performance of a KGEM based on its single components, a fully composable KGEM-library is essential [62]. Evaluating the effect of the single components on the models' performance allows us to investigate whether the performance of KGEMs is solely dependent on their interaction models. Furthermore, the evaluation of KGEMs was (especially for larger KGs) not computationally efficient.

Therefore, in a community effort, we redesigned and re-implemented PyKEEN and developed PyKEEN 1.0, a fully composable and extensive KGEM-library that enables in-depth analysis of KGEMs-based link predictors. At the time of publishing, PyKEEN 1.0 covered the implementation of 23 interaction models, seven loss functions, two training approaches and enabled users to employ explicit inverse relations. In addition, PyKEEN 1.0 integrates 21 benchmark datasets, six evaluation metrics and provides hyper-parameter optimisation (HPO) routines on-top of Optuna [63] ensuring extensive HPO functionalities.

While developing PyKEEN 1.0, we focused on its entire composability by realising each KGEM's component, i.e., interaction model, loss function, and training approach as independent sub-modules. The modelling of explicit inverse relations has been realised within the interaction models. Because each KGEM's component follows our unified APIs for the sub-modules, i.e., `pykeen.model.Model` for the interaction models, `pykeen.loss.Loss` for the loss functions, and `pykeen.training.TrainingLoop` for the training approaches, we can arbitrarily compose a KGEM based on these modules. The modular architecture of PyKEEN 1.0 facilitates researchers and practitioners to integrate new components into the library and compare, for instance, their KGEM against existing ones. The available modules represent a prime reference for implementing customised modules.

The link prediction performance of KGEMs is usually measured based on ranking metrics. However, it has been highlighted that different definitions of the rank have been employed in the literature, affecting ranking metrics and impeding the reproduction of link prediction experiments (in case no official implementation is provided) and the comparability of experiments [64]. To foster reproduction and comparability of KGEMs-based link prediction experiments, we realise the ranking metrics based on the most prevalent rank definitions, i.e., *optimistic*, *realistic*, and *pessimistic* rank [15].

¹<https://github.com/pykeen/pykeen/graphs/contributors>

When performing extensive benchmarking studies, a large set of heterogeneous model configurations are defined that have different memory requirements. It can quickly happen that the available memory is exceeded during training and evaluation because of specific model configurations. Therefore, we realised an *automatic memory optimisation* that automatically determines the maximum possible batch size before the actual experiment is conducted. Suppose the defined batch size exceeds the available hardware. In that case, the automatic memory optimisation computes the maximum sub-batch size for the training routine and the maximum possible batch size for the evaluation routine.

Similar to PyKEEN, we comply with community standards in PyKEEN 1.0 to ensure the library's quality. We use *flake8*² to ensure the code quality, *PyTest*³ for implementing unit tests, *GitHub Actions*⁴ for continuous integration, built our documentation using *Sphinx*⁵ and hosted it on *Read the Docs*⁶.

Compared to related software packages, PyKEEN 1.0 provides entire composability and extensive functionalities, i.e., a large number of interaction models, evaluation metrics and extensive hyperparameter optimisation functionalities are available. Regarding entire composability, LibKGE [65] is the only comparable library.

The entire composability and extensive functionalities of PyKEEN 1.0 enabled us to perform our reproducibility and the most-extensive KGEM-benchmarking study done to date, which is presented in the following chapter. Furthermore, we utilised certain functionalities of PyKEEN 1.0 for investigating inductive link prediction in hyper-relational KGs as presented in Chapter 5. Finally, PyKEEN 1.0 is used to generate patient representations in the context of the CLEP approach presented in Chapter 6.

²<https://flake8.pycqa.org/en/latest/>

³<https://docs.pytest.org/en/7.2.x/>

⁴<https://github.com/features/actions>

⁵<https://www.sphinx-doc.org/en/master/>

⁶<https://readthedocs.org/>

Chapter 4

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

In this chapter, we present the following publication (see Appendix C):

Mehdi Ali, Max Berrendorf*, Charles Tapley Hoyt*, Laurent Vermue*, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. “Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021). * equal contribution. © 2022 IEEE. Reprinted, with permission, from Mehdi Ali and Max Berrendorf and Charles Tapley Hoyt and Laurent Vermue and Mikhail Galkin and Sahand Sharifzadeh and Asja Fischer and Volker Tresp and Jens Lehmann, 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*, 01/2022. DOI: <https://doi.org/10.1109/TPAMI.2021.3124805>. URL: <https://ieeexplore.ieee.org/abstract/document/9601281>

Authors’ contributions: Mehdi Ali initially discussed the existing research gap with Asja Fischer and Jens Lehmann. Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Sahand Sharifzadeh, and Laurent Vermue developed the idea to systematically address the challenges in reproducing published KGEM experiments and investigate the impact of the single components of a KGEM on its link prediction performance. Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, and Laurent Vermue implemented the source code. Mehdi Ali, Laurent Vermue and Mikhail Galkin performed the reproducibility and benchmarking experiments. In addition, Max Berrendorf implemented and performed the relational pattern analysis. Mehdi Ali, Max Berrendorf,

Charles Tapley Hoyt and Laurent Vermue evaluated the results and wrote the initial manuscript. All authors revised the manuscript.

Summary

In the last decade, many approaches have been proposed to advance KGEM-based link prediction. However, comprehending the performance of KGEM-based link predictors is currently a major challenge, primarily for two reasons. First, the reproduction of reported KGEM-based link prediction experiments remains a major obstacle [66]. Second, it is often not apparent whether the (usually) incremental improvement of a proposed KGEM-based link predictor is caused only by its interaction model (model architecture) or by exchanging the loss function, the training approach, or by explicitly modelling inverse relations. Prior work indicates that baseline interaction models can obtain competitive results to sophisticated ones when properly configured [16, 66]. Moreover, the different realisation of the ranking metrics used for evaluating KGEM-based link predictors hinders a fair comparison of published results [67].

Equipped with PyKEEN 1.0 (Chapter 3), we addressed the reproducibility crisis of link prediction experiments by performing a reproducibility study, in which we tried to reproduce reported link prediction experiments based on the information in the corresponding paper and the accompanying implementation if provided. We performed 34 reproduction experiments involving 15 interaction models (a full list is provided in our paper) and four datasets (FB15K, FB15K-237, WN18, and WN18RR) in the reproducibility study. Using PyKEEN 1.0 as a framework, we ensured that all the models were trained and evaluated under identical conditions. To obtain a reliable overview of the models' performance, we computed the ranking metrics based on the *optimistic*, *realistic*, and *pessimistic* definition of the rank. Our reproduction study revealed that the reproduction of KGEMs-based link predictors is, even with significant effort, often not possible. In the reproduction study, we made the following four main observations. First, for a set of experiments, we could not reproduce the results with the reported hyperparameter values but with an alternative set of hyperparameter values, which we obtained by further investigating the models' performance. Second, a set of results depended on the rank's realised definition (average, optimistic, and pessimistic rank). Third, the absence of an official implementation hinders the reproduction of results. Fourth, omitting details about the experimental setup hampers the reproduction of results.

Next, we performed the most-extensive KGEM-based link prediction benchmarking study to investigate whether the performance of KGEMs-based link predictors is solely attributed to their interaction models. Our benchmarking study involved four benchmark datasets, 21 interaction models, two training approaches, five loss functions, and two optimisers. Furthermore, we measured the effect of explicitly modelling inverse relations. Specifically, our benchmarking study involved the Kinships, FB15k-237, WN18RR, and the YAGO3-10 datasets. Furthermore, we investigated the sLCWA and LCWA training approach. In our benchmarking study, we measured the effect of the margin ranking loss (MRL), binary cross entropy loss (BCEL), Softplus loss (SPL), CEL, and NSSAL loss function on the models'

performance. Finally, we evaluated the Adam [68] and the ADADELTA [69] optimisers. In total, we performed several thousand experiments spanning 24,804 GPU hours. Our study evinces that the performance of a KGEM-based link predictor is not solely determined by its interaction model but often strongly depends on the combination of the interaction model, loss function, training approach, and the decision whether or not to explicitly model inverse relations. Even baseline interaction models such as TransE can outperform state-of-the-art interaction models when composed appropriately. Furthermore, the study revealed that no single configuration performs best across all datasets. Instead, we found that depending on the dataset, several configurations perform comparably well. Finally, we could obtain for RotatE comparable results to Graph Attenuated Attention Networks [70] on the WN18RR dataset representing novel state-of-the-art results and present improved configurations for several interaction models, outperforming the results presented in the accompanying papers.

We believe that our reproducibility and benchmarking study are important contributions to the research field of KGEMs-based link prediction because they provide a holistic and in-depth analysis of the research field and help researchers and practitioners to make guided decisions.

Chapter 5

Improving Inductive Link Prediction Using Hyper-relational Facts

In this chapter, we present the following publication (see Appendix D):

Mehdi Ali^{*}, Max Berrendorf^{*}, Mikhail Galkin, Veronika Thost, Tengfei Ma, Volker Tresp, and Jens Lehmann. “Improving Inductive Link Prediction Using Hyper-relational Facts”. In: Lecture Notes in Computer Science 12922 (2021). * equal contribution. **For this work, we received the Best Research Paper Award.**, pp. 74–92. DOI: 10.1007/978-3-030-88361-4_5

Authors’ contributions: The idea to exploit hyper-relational information for inductive link prediction was proposed by Mehdi Ali and further developed together with Max Berrendorf and Mikhail Galkin. Mehdi Ali, Max Berrendorf, Mikhail Galkin, Veronika Thost, and Tengfei Ma developed the theoretical framework for classifying different inductive link prediction scenarios. Mikhail Galkin generated the fully-inductive benchmark datasets, and Mehdi Ali and Max Berrendorf generated based on the fully-inductive datasets, the semi-inductive datasets. Mehdi Ali and Max Berrendorf implemented the main part of the source code. Mehdi Ali conducted the experiments, and Mehdi Ali, Max Berrendorf, and Mikhail Galkin evaluated the results. Mehdi Ali, Max Berrendorf and Mikhail Galkin wrote the initial manuscript. All authors revised the manuscript.

Summary

Transductive link prediction has been the prevalent link prediction setting in KGs in the past. However, transductive link prediction impedes inference over unseen entities. Typical scenarios involving unseen entities are incremental updates of KGs or the prediction within an entire unseen (sub-)graph with a known set of relations. Recently, the task of inductive link prediction has received increased attention [20, 71, 31, 32, 72]. Different semi-inductive and fully-inductive link prediction approaches have been proposed. However, all proposed approaches have been developed for triple-based KGs so far, and inductive link prediction has not yet been investigated within hyper-relational KGs.

Therefore, we address these limitations and investigate whether hyper-relational information/qualifiers improve inductive link predictors. In particular, we make the following three contributions to answering the research question. First, we address the existing terminology gap and propose a theoretical framework upon which existing inductive scenarios can be classified. Second, we develop a novel set of hyper-relational benchmark datasets allowing us to analyse the performance of inductive link predictors within hyper-relational KGs. Third, we adapt two baseline models to the inductive hyper-relational setting and investigate the influence of hyper-relational information on inductive link prediction performance in the context of the *fully-inductive* and *semi-inductive* setting. We performed quantitative and qualitative experiments involving four datasets of different complexity.

While studying the inductive link prediction scenario, we identified a terminology gap. Different namings for conceptually equivalent inductive settings have been introduced, or the same naming for different inductive settings is employed. In our theoretical framework, we can classify all existing inductive settings that are applicable to triple-based KGs and hyper-relational KGs. We differentiate between the semi-inductive and fully-inductive link prediction scenarios and show that existing approaches are either in one of these two categories or a mixture of both. In the semi-inductive setting, a test statement (or triple) contains exactly one unseen entity that occurs as a head or tail entity. In the fully-inductive setting, both the head and tail entities are unseen for all test statements (or triples). More specifically, the inference is performed within a graph comprised solely of unseen entities. We classified existing approaches based on the set of auxiliary statements (or triples) used during inference, the type of inductive links, i.e., links between seen and unseen entities or links between unseen entities, and the set of entities a test statement (or triple) is scored against.

To investigate whether hyper-relational information improves inductive link prediction, we created a novel set of benchmark datasets based on the WD50K dataset [7]. We provide two semi-inductive and four fully-inductive datasets of varying sizes and complexity. The datasets reveal different ratios of triples with qualifiers ranging from 30% to 100%.

Our results highlight that hyper-relational information improves inductive link prediction in fully-inductive and semi-inductive settings. The composition of the datasets, i.e., the ratio of statements with qualifiers and dataset size, has a major impact on the performance of the different approaches. Furthermore, we could observe that even a small number of statements with hyper-relational information improves the link prediction performance of the models in the semi-inductive setting. In our qualitative analysis, we first investigated the model performance for statements with and without qualifiers. We could observe an increased model performance when more qualifiers are provided. Next, we studied how specific qualifiers affect the model performance. Specifically, we grouped statements based on the qualifying relations and compared the model performance for each group, i.) when retaining the qualifying relation in the inference graph and ii.) when removing the qualifying relation from the inference graph. Our analysis illustrated that certain qualifiers can greatly improve or impair the model's link prediction performance. We hypothesise that specific qualifying entities introduce expressive graph structures enabling the link predictors to generalise better, whereas other entities emerge only as rare qualifying entities impeding the generalisation capabilities of the models. Finally, we studied the effect of removing qualifying relations from the entire graph, i.e., we do not compare the model performance within groups that contain a specific qualifying relation but compare the model performance based on all test statements. We could observe that most qualifying relations improve the model performance for a small dataset with a full qualifier coverage (i.e., each statement contains at least one qualifier pair). However, on a larger dataset with a reduced qualifier coverage, some qualifying relations deteriorate the model performance.

In summary, we have bridged two concepts: hyper-relational KGs and inductive link prediction.

Chapter 6

Applications

In this chapter, we present the following two publications (see Appendices E and F):

- **Mehdi Ali**, Charles Tapley Hoyt, Daniel Domingo-Fernández, Jens Lehmann, and Hajira Jabeen. “BioKEEN: a library for learning and evaluating biological knowledge graph embeddings”. In: *Bioinformatics* 35.18 (Feb. 2019), pp. 3538–3540. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz117. URL: <https://doi.org/10.1093/bioinformatics/btz117>

Authors’ contributions: The idea of BioKEEN and showcasing a concrete biomedical application was developed by Mehdi Ali and further conceptualized together with Charles Tapley Hoyt and Daniel Domingo-Fernández. Mehdi Ali implemented the machine learning functionalities, Charles Tapley Hoyt and Daniel Domingo-Fernández implemented the adapters for integrating the biomedical datasets. Mehdi Ali ran the experiments, and Mehdi Ali, Charles Tapley Hoyt and Daniel Domingo-Fernández evaluated the results. Mehdi Ali, Charles Tapley Hoyt, and Daniel Domingo-Fernández wrote the manuscript, and all authors revised the manuscript.

- Vinay Srinivas Bharadhwaj, **Mehdi Ali**, Colin Birkenbihl, Sarah Mubeen, Jens Lehmann, Martin Hofmann-Apitius, Charles Tapley Hoyt, and Daniel Domingo-Fernández. “CLEP: a hybrid data- and knowledge-driven framework for generating patient representations”. In: *Bioinform.* 37.19 (2021), pp. 3311–3318. DOI: 10.1093/bioinformatics/btab340. URL: <https://doi.org/10.1093/bioinformatics/btab340>

Authors’ contribution: Daniel Domingo-Fernández and Charles Tapley Hoyt conceived and designed the study. Vinay Srinivas Bharadhwaj implemented CLEP and ran the experiments with supervision and support from Daniel Domingo-Fernández. Mehdi Ali guided and supported the implementation of the knowledge graph embedding generation and classification tasks. Colin Birkenbihl assisted with data and method development. Sarah Mubeen processed the knowledge graph. All the authors contributed to the writing of the manuscript. All authors have read and approved the final manuscript

Summary

KGs have been widely adopted in the biomedical domain [9] to model biological systems [73], e.g., describing interactions between biomedical entities or modelling diseases. They are applied in various applications, such as in clinical decision support systems. Many biological KGs are available, such as DrugBank [74], KEGG [75] and Gene Ontology [76] that can be used for the described use cases [73]. Considering the broad adoption of biological KGs, knowledge graph representation learning and KGEM-based link prediction have great potential to be applied in biomedical use cases.

We present two applications of KGEMs that can be effectively applied for biomedical use cases. First, we present BioKEEN, an extension of PyKEEN that facilitates the usage of KGEMs for bioinformaticians without deep knowledge about KGEMs and in implementing these models. We first illustrate the effectiveness of BioKEEN in tackling biomedical link prediction tasks by addressing an exemplary use case in which we predict crosstalks and hierarchies between biological pathways in a novel pathway dataset. Second, we present CLEP (CLinical Embedding of Patients), a new approach for learning patient embeddings by combining patient-level data and prior knowledge in the form of KGs.

BioKEEN As we identified great potential in employing KGEMs in the biomedical domain, we developed BioKEEN [23]. BioKEEN extends PyKEEN such that bioinformaticians can effectively use KGEMs for their use cases without having profound knowledge about these models and their implementation. In BioKEEN, we integrate the Bio2BEL [77] software that provides direct access to a large number of biological databases capturing structured knowledge. BioKEEN is comprised of an adapted *configuration layer*, a *data acquisition and transformation layer*, and a *learning layer*. In the configuration layer, an interactive command-line interface assists users in configuring their experiments, particularly ensuring that correct hyperparameters are chosen for a KGEM. Additionally, the user can provide his dataset or select one of the integrated biological datasets through the configuration layer. Furthermore, the training and evaluation routines are set up in the configuration layer. Here, the user can decide to configure a standard training run or a hyperparameter optimisation run. In the data acquisition and transformation layer, the directly accessible databases are accessed through the Bio2BEL software and provided as input to the subsequent layer. In BioKEEN 14, biological databases are currently directly accessible, covering a wide range of biological knowledge, such as disease-differentially expressed gene interactions, pathway-pathway interactions, and drug-target interactions. Users are not expected to deal with the time-consuming process of pre-processing the data. Through the learning layer, BioKEEN users have full access to the machine learning functionalities of PyKEEN. Consequently, a KGEM can be trained with user-defined hyperparameter values or random search can be

applied to find a suitable set of hyperparameter values to support inexperienced users. The models are evaluated based on the mean rank and the hits@k.

In order to measure the effectiveness of BioKEEN, we applied it to ComPath [78], a novel dataset containing biological pathway mappings. ComPath describes equivalences and hierarchies between pathways and is directly accessible within BioKEEN. We were interested in predicting novel equivalences and hierarchies in this use case. We employed the following five KGEMs in our experiments: TransE, TransH, TransR, DistMult and UM. For each model, we performed a hyperparameter optimisation and demonstrated the sensitivity of using appropriate hyperparameter values and the effectiveness of BioKEEN in assisting in finding suitable hyperparameter values. We selected the best-performing model for predicting novel equivalences and hierarchies between pathways. After filtering out reflexive triples from the set of predicted links, a domain expert analysed the highest-ranked predictions and found two novel links describing *TGF-beta Receptor Signaling* (wikipathways: WP560) is equivalent to *TGF-beta signaling pathway* (kegg: hsa04350), and *Lipoic acid* (kegg: hsa00785) is part of *Lipid metabolism* (reactome: R-HSA-556833). This use case demonstrated the effectiveness of BioKEEN in addressing biomedical applications.

Despite the already rich functionalities of BioKEEN and its effectiveness in tackling biomedical applications, BioKEEN could further be improved with additional functionalities. Considering the heterogeneity of biomedical data, multi-modal KGEMs could further extend BioKEEN's application scope within the biomedical domain. Furthermore, adding negative sampling approaches that ensure the generation of true negatives by incorporating prior knowledge and constraints could further improve the models' performance. Exemplary constraints are type constraints, constraints on the attribute range for relations, or functional constraints such as mutual exclusion [11]. Although the hyperparameter optimisation supports inexperienced users in finding suitable hyperparameter values, choosing a suitable KGEM requires a profound understanding of these models. Therefore, BioKEEN could be improved by additionally assisting in finding appropriate KGEMs by, for instance, defining a set of rules capturing limitations of KGEMs in modelling certain relational patterns (e.g., DistMult is not designed for antisymmetric patterns).

CLEP In the biomedical domain, a large set of heterogeneous data is available, such as patient measurements (e.g., transcriptomic data) and biological networks/KGs modelling interactions between biomedical entities. Leveraging several data sources to obtain more expressive features facilitating classification tasks is still underexplored. Therefore, in a second application, we propose CLEP [26], a new approach for generating patient embeddings by combining patient-level data and KGs expressing prior knowledge. Specifically, in CLEP, transcriptomics data of patients are mapped to a KG containing protein-protein interactions (PPIs) from several biomedical datasets. To

map measured patients' features to entities of the KG, CLEP computes a reference distribution for each measured feature across all patients (or healthy controls), i.e., for n measured features, CLEP will generate n reference distributions. For each reference distribution, CLEP determines the patients lying on the extreme ends of the distribution and adds these patients to the KG by representing them as nodes and linking them to corresponding nodes within the KG that represent the measured features. Patients are added to the KG based on two relationship types: $+1$ indicating that a high value has been measured for a particular feature and -1 indicating that a low value has been measured. If a patient lies on the extreme ends of the reference distributions for all n measured features, the patient-node will be connected to n nodes in the KG. The enriched KG combines prior biological knowledge and patient-level data because of the links between the feature nodes and the links between patient nodes and the feature nodes. Based on the enriched KG, PyKEEN 1.0 is employed to learn knowledge graph embeddings that serve as inputs to classifiers that are trained to discriminate between patients and healthy controls.

In the experiments, the Alzheimer's Disease Neuroimaging Initiative (ADNI) [79] dataset has been used. More specifically, the blood plasma transcriptomic data gathered during the study [80] has been employed. The dataset is comprised of 260 healthy controls and 484 impaired patients with different impairment levels ranging from early mild cognitive impairment to Alzheimer's disease. The second transcriptomics dataset is comprised of 83 healthy controls and 99 patients with psychiatric disorders (major depressive disorder, schizophrenia, or bipolar disorder). For both datasets, a protein-protein interaction KG comprising PPIs from six databases has been utilised. Specifically KEGG [75], Reactome [81], WikiPathways [82], BioGrid [83], IntAct [84], and Pathway Commons [85] have been integrated. A binary classification task has been defined for both settings: *healthy controls vs cognitively impaired* and *healthy controls vs psychiatric disorders*. As a first step, CLEP integrates the patients and the healthy controls into the KG. Next, knowledge graph embeddings based on four KGEMs (RotatE, TransE, ComplEx, and HolE) are computed. For each KGEM, a hyperparameter optimisation has been performed in order to find a suitable set of hyperparameter values. Here, insights obtained from our large-scale benchmarking study (Chapter 4) have been exploited. It is worth mentioning that through the use of KGEMs to learn patient representations, the dimensionality of the patient representation can significantly be reduced. While in the first classification task, i.e., classifying healthy controls vs cognitively impaired patients, the raw transcriptomics data of a patient had more than 40,000 features, the final learned knowledge graph embedding of the patient had a dimension size of 256. Based on the learned patient embeddings, several classifiers (Logistic Regression, Support Vector Machines, Random Forest and XGBoost) have been optimised to discriminate patients from healthy controls. Our results demonstrate that the classifiers obtained better performance when trained on top of the patient embeddings instead of the raw transcriptomics data of the patients. Therefore, CLEP is an excellent example of employing graph

representation learning within biomedical applications.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

In this thesis, we provided in-depth analyses of link prediction in KGs based on graph representation learning approaches and highlighted the effectiveness of KG representation learning for addressing biomedical use cases. Overall, the contributions of this thesis can be split into four major parts: i.) development of an extensive software ecosystem for investigating the performance of KGEM-based link prediction approaches, ii.) an in-depth analysis of KGEM-based link prediction covering a reproducibility study and the most extensive benchmarking study of its kind, iii.) improving inductive link prediction, and iv.) the effective application of KGEMs for predicting novel links between biological pathways. Finally, we demonstrated how to utilise KGEMs to generate expressive patient embeddings that can be employed in downstream classification tasks in biomedicine.

In Chapter 2, we present the KEEN Universe, an ecosystem for KGEMs. The KEEN Universe consists of the KGEM library PyKEEN, its extension BioKEEN, and the KEEN Model Zoo. PyKEEN is the main component of the KEEN Universe and represents the first step towards an extensive library for KGEMs. PyKEEN covered the implementation of 10 KGEMs, as well as a training, an evaluation, a hyperparameter optimisation, and an inference routine. It realised the sLCWA training loop, implemented hits@k and MR as evaluation metrics, and performed hyperparameter optimisation using random search. Because of its modular architecture, the integration of novel components is facilitated. The reproducibility of experiments is ensured by exporting the experimental artefacts. Precisely, the entire experimental setup, the evaluation results, the mappings of entities/relations to their ids and the mappings of entities/relations to the learned embeddings are exported as JSON files. We developed the library in compliance with community standards (*flake8*¹, *setuptools*², *pyroma*³, *sphinx*⁴, *Read the Docs*⁵, and *Travis-CI*⁶) to

¹<https://flake8.pycqa.org/en/latest/>

²<https://setuptools.pypa.io/en/latest/>

³<https://github.com/regebro/pyroma>

⁴<https://www.sphinx-doc.org/en/master/>

⁵<https://readthedocs.org/>

⁶<https://travis-ci.org/>

ensure the high quality of the library. Finally, the library has successfully been applied in several applications [24]. We extended PyKEEN and developed BioKEEN, which facilitates the usage of KGEMs within the bioinformatics community. The KEEN Model Zoo enables researchers to share their experimental artefacts created with PyKEEN or BioKEEN. We defined several requirements for sharing artefacts through the model zoo in order to ensure the quality of the models available in the model zoo. In the subsequent chapter, we focused on PyKEEN, which represents the foundation for the work presented in Chapter 3.

In Chapter 3, we addressed the limitations of PyKEEN and developed in a community effort PyKEEN 1.0. The main limitations of PyKEEN were that KGEMs were not fully configurable, the evaluation procedure was too slow, and it was designed to be used mainly through a command-line interface. We redesigned the library from scratch and addressed all the above-mentioned limitations. Furthermore, we added a large set of novel components. At the time of publishing PyKEEN 1.0 [25], it covered the implementation of 23 interaction models, seven loss functions, two training approaches, 21 integrated benchmark datasets, and six evaluation metrics. In addition, it integrated the framework Optuna [63] ensuring extensive hyper-parameter-optimisation functionalities. One of the primary functionalities of PyKEEN 1.0 is the entire composability of KGEMs, i.e., each KGEM can be composed arbitrarily based on the existing interaction models, loss functions, training approaches, and the decision to model inverse relations explicitly. PyKEEN 1.0 follows unified APIs for defining interaction models, loss functions, and training approaches, ensuring the easy integration of novel components. While developing PyKEEN 1.0, we comply with community standards to ensure code quality. PyKEEN 1.0 has become a community project with more than 30 contributors that received over 1000 stars on GitHub⁷ and has been employed in several applications (e.g., for biomedical applications [26, 86]). For instance, a research lab from the large pharmaceutical company AstraZeneca used PyKEEN 1.0 to investigate KGEMs for drug discovery [86].

In Chapter 4, we presented two major contributions: i.) we addressed the reproducibility crisis of KGEM-based link prediction experiments by performing a reproducibility study, and ii.) we investigated whether the performance of KGEM-based link predictors is solely dependent on their interaction models by performing the most extensive benchmarking study of its type. Both studies have been performed using PyKEEN 1.0, which has been presented in Chapter 3.

In our reproducibility study, we examined which KGEM-based link prediction experiments were reproducible and made four main observations. First, we observed that certain experiments could only be reproduced with an alternative set of hyperparameter values. Second, specific results depend on

⁷<https://github.com/pykeen/pykeen>

the realisation of the ranking metric, i.e., the results for the same experiment heavily vary depending on whether the optimistic, realistic, or pessimistic ranking has been employed. Third, for a set of experiments, no official implementation was provided, hampering their reproduction. Fourth, for a set of experiments, the entire experimental setup had not been described, impeding their reproduction. Overall, 14 out of 34 experiments were soft-reproducible (i.e., could be reproduced up to a margin δ), for four out of 15 investigated KGEMs, no official implementation was available, and for six out of 15 models, the entire experimental setup was described, and source code was provided.

In our benchmarking study, we investigated whether the performance of KGEMs can solely be attributed to their interaction models. We defined a KGEM as a composition of an interaction model, a loss function, a training approach, and the usage of inverse relations. This abstraction emphasises each component’s importance and allows measuring the effect of each component individually on the model’s performance or in combination. Based on this abstraction, we evaluated the performance of different configurations for an KGEM, i.e., the combination of the interaction model, loss function, training approach, and the usage of explicit inverse relations. For each configuration, we performed a hyperparameter optimisation to find suitable hyperparameter values. In our study, we performed several thousands of experiments over 24,804 GPU hours involving four datasets, two optimisers, 21 interaction models, two training approaches, five loss functions, and investigated the effect of explicit modelling inverse relations. Our study revealed that the KGEM’s performance cannot solely be attributed to the interaction model, but is often dependent on the specific composition of the four components of a KGEM. Even interaction models considered as baselines, such as TransE, can outperform state-of-the-art interaction models when composed appropriately. Furthermore, we did not find any configuration that performed best across all datasets. Instead, for certain datasets, we could determine several competitive configurations. We extracted the top-performing configurations for each dataset and found out that several interaction models were part of the top-performing interaction models for different datasets. Moreover, we demonstrated that the MRL was the worst-performing loss function and that both training approaches obtained strong performance. We further showed that the explicit modelling of inverse relations benefits, in particular, the LCWA training approach. We could improve the reported results for a set of interaction models by providing novel configurations. Lastly, we could determine a RotatE-based configuration that performs comparably to Graph Attenuated Attention Networks [70] on the WN18RR dataset representing novel state-of-the-art results.

In Chapter 5, we bridged two concepts, namely hyper-relational KGs and inductive link prediction, and demonstrated that hyper-relational facts improve inductive link prediction. We first addressed the terminology gap existing in the literature by providing a theoretical framework based on which all existing inductive link prediction scenarios can be categorised.

In particular, we differentiate between fully-inductive and semi-inductive link prediction scenarios. Next, we provided novel hyper-relational benchmark datasets of varying size and complexity for fully- and semi-inductive link prediction. In particular, we provide four fully-inductive and two semi-inductive datasets with varying ratios of qualifier pairs per triple. We adopted two baselines for the hyper-relational setting and performed 46 ablation studies in which we investigated the effect of employing two, four, six or no qualifiers per triple. Our ablation studies highlight that employing qualifiers/hyper-relational information improves inductive link prediction. Finally, we performed qualitative analyses and demonstrated that certain qualifiers improve inductive link prediction performance, whereas others deteriorate it.

In Chapter 6, we demonstrated, based on two applications, that knowledge graph representation learning can be effectively employed to address biomedical use cases.

First, we present BioKEEN, an extension of PyKEEN (Chapter 2) that facilitates the usage of KGEMs for bioinformaticians. BioKEEN integrates the Bio2BEL [77] software for directly accessing various biomedical sources. Users can easily configure their experiments, including the dataset selection, through an interactive command-line interface. Because BioKEEN is built on top of PyKEEN, all machine learning functionalities of PyKEEN are available within BioKEEN. Users do not need to focus on re-implementing the KGEMs but can focus on their use case. We used BioKEEN to predict crosstalks and hierarchies in a novel dataset containing biological pathways. In particular, we performed hyperparameter optimisations for five models and selected the best-performing to generate new predictions. We provided the top-ranked predictions to a domain expert who identified the novel crosstalk between *TGF-beta Receptor Signaling* (*wikipathways: WP560*) and *TGF-beta signaling pathway* (*kegg: hsa04350*). Furthermore, we identified that *Lipoic acid* (*kegg: hsa00785*) is a part of *Lipid metabolism* (*reactome: R-HSA-556833*). This application showcased that KGEMs can effectively be used to generate valid novel predictions.

Second, we presented CLEP, a novel approach that combines patient-level data (e.g. transcriptomics data) and prior knowledge in the form of KGs. Patients are integrated as nodes in a KG, and their features are represented as edges to biomedical entities. We used PyKEEN 1.0 to learn knowledge graph embeddings for the enriched KG. The learnt patient (and healthy control) representations are used in downstream classification tasks to discriminate between patients and healthy controls. We were able to show that classifiers trained on top of the learned representations outperform classifiers trained based on the raw data of the patients.

7.2 Future Work

There are several directions to further explore in the context of link prediction in KGs and KG representation learning in general. In this thesis, we provided deep insights into link prediction within KGs based on extensive empirical studies. We highlighted that even baseline (transductive) interaction models obtain state-of-the-art performance. Therefore, further elaborating on the theoretical understanding of link predictors is of major interest. In many applications, the prediction of an incorrect link is less critical. However, there are link prediction applications, such as recommending certain treatments or drugs to patients, where incorrect predictions can cause serious personal damage. Therefore, investigating explainability and uncertainty quantification for link prediction has significant practical relevance. Uncertainty quantification is further relevant in the context of inductive link prediction, where inference over unseen entities is performed. Because inductive link prediction involves uncertainty, quantifying this uncertainty could help obtain reliable predictions.

Bibliography

- [1] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. “A Survey on Knowledge Graphs: Representation, Acquisition, and Applications”. In: *IEEE Trans. Neural Networks Learn. Syst.* 33.2 (2022), pp. 494–514.
- [2] Nilesh Chakraborty, Denis Lukovnikov, Gaurav Maheshwari, Priyansh Trivedi, Jens Lehmann, and Asja Fischer. “Introduction to neural network-based question answering over knowledge graphs”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11.3 (2021), e1389.
- [3] Debanjan Chaudhuri, Md. Rashad Al Hasan Rony, and Jens Lehmann. “Grounding Dialogue Systems via Knowledge Graph Aware Decoding with Pre-trained Transformers”. In: *ESWC*. Vol. 12731. Lecture Notes in Computer Science. Springer, 2021, pp. 323–339.
- [4] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. “A survey on knowledge graph-based recommender systems”. In: *IEEE Transactions on Knowledge and Data Engineering* (2020).
- [5] David N Nicholson and Casey S Greene. “Constructing knowledge graphs and their biomedical applications”. In: *Computational and structural biotechnology journal* 18 (2020), pp. 1414–1428.
- [6] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. “DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia”. In: *Semantic Web 6.2* (2015), pp. 167–195.
- [7] Denny Vrandečić and Markus Krötzsch. “Wikidata: a free collaborative knowledgebase”. In: *Commun. ACM* 57.10 (2014), pp. 78–85.
- [8] Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. “Freebase: a collaboratively created graph database for structuring human knowledge”. In: *SIGMOD Conference*. ACM, 2008, pp. 1247–1250.
- [9] François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. “Bio2RDF: Towards a mashup to build bioinformatics knowledge systems”. In: *J. Biomed. Informatics* 41.5 (2008), pp. 706–716.

- [10] Ling Tian, Xue Zhou, Yan-Ping Wu, Wang-Tao Zhou, Jin-Hao Zhang, and Tian-Shu Zhang. "Knowledge graph and knowledge reasoning: A systematic review". In: *Journal of Electronic Science and Technology* (2022), p. 100159.
- [11] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. "A Review of Relational Machine Learning for Knowledge Graphs". In: *Proc. IEEE* 104.1 (2016), pp. 11–33. DOI: 10.1109/JPROC.2015.2483592. URL: <https://doi.org/10.1109/JPROC.2015.2483592>.
- [12] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. "Knowledge Graph Embedding: A Survey of Approaches and Applications". In: *IEEE Trans. Knowl. Data Eng.* 29.12 (2017), pp. 2724–2743.
- [13] William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020.
- [14] **Mehdi Ali**, Max Berrendorf*, Charles Tapley Hoyt*, Laurent Vermue*, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. "Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021). * equal contribution. © 2022 IEEE. Reprinted, with permission, from Mehdi Ali and Max Berrendorf and Charles Tapley Hoyt and Laurent Vermue and Mikhail Galkin and Sahand Sharifzadeh and Asja Fischer and Volker Tresp and Jens Lehmann, 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*, 01/2022. DOI: <https://doi.org/10.1109/TPAMI.2021.3124805>. URL: <https://ieeexplore.ieee.org/abstract/document/9601281>.
- [15] Max Berrendorf, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. "On the ambiguity of rank-based evaluation of entity alignment or link prediction methods". In: *arXiv preprint arXiv:2002.06914* (2020).
- [16] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. "Knowledge Base Completion: Baselines Strike Back". In: *Rep4NLP@ACL*. Association for Computational Linguistics, 2017, pp. 69–74.
- [17] **Mehdi Ali***, Max Berrendorf*, Mikhail Galkin, Veronika Thost, Tengfei Ma, Volker Tresp, and Jens Lehmann. "Improving Inductive Link Prediction Using Hyper-relational Facts". In: *Lecture Notes in Computer Science* 12922 (2021). * equal contribution. **For this work, we received the Best Research Paper Award.**, pp. 74–92. DOI: 10.1007/978-3-030-88361-4_5.
- [18] Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. "Message Passing for Hyper-Relational Knowledge Graphs". In: *EMNLP (1)*. Association for Computational Linguistics, 2020, pp. 7346–7359.

- [19] Dimitrios Alivanistos, Max Berrendorf, Michael Cochez, and Mikhail Galkin. “Query Embedding on Hyper-Relational Knowledge Graphs”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: <https://openreview.net/forum?id=4rLw09TgRw9>.
- [20] Komal Teru, Etienne Denis, and Will Hamilton. “Inductive Relation Prediction by Subgraph Reasoning”. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 9448–9457.
- [21] Wen Zhang, Shumin Deng, Mingyang Chen, Liang Wang, Qiang Chen, Feiyu Xiong, Xiangwen Liu, and Huajun Chen. “Knowledge Graph Embedding in E-commerce Applications: Attentive Reasoning, Explanations, and Transferable Rules”. In: *IJCKG*. ACM, 2021, pp. 71–79.
- [22] Sameh K. Mohamed, Aayah Nounu, and Vít Nováček. “Biological applications of knowledge graph embedding models”. In: *Briefings Bioinform.* 22.2 (2021), pp. 1679–1693.
- [23] **Mehdi Ali**, Charles Tapley Hoyt, Daniel Domingo-Fernández, Jens Lehmann, and Hajira Jabeen. “BioKEEN: a library for learning and evaluating biological knowledge graph embeddings”. In: *Bioinformatics* 35.18 (Feb. 2019), pp. 3538–3540. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz117. URL: <https://doi.org/10.1093/bioinformatics/btz117>.
- [24] **Mehdi Ali**, Hajira Jabeen, Charles Tapley Hoyt, and Jens Lehmann. “The KEEN Universe - An Ecosystem for Knowledge Graph Embeddings with a Focus on Reproducibility and Transferability”. In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*. Vol. 11779. Lecture Notes in Computer Science. Springer, 2019, pp. 3–18. DOI: 10.1007/978-3-030-30796-7_1. URL: https://doi.org/10.1007/978-3-030-30796-7_1.
- [25] **Mehdi Ali***, Max Berrendorf*, Charles Tapley Hoyt*, Laurent Vermue*, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. “PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings”. In: *Journal of Machine Learning Research* 22.82 (2021). * equal contribution, pp. 1–6. URL: <http://jmlr.org/papers/v22/20-825.html>.
- [26] Vinay Srinivas Bharadhwaj, **Mehdi Ali**, Colin Birkenbihl, Sarah Mubeen, Jens Lehmann, Martin Hofmann-Apitius, Charles Tapley Hoyt, and Daniel Domingo-Fernández. “CLEP: a hybrid data- and knowledge-driven framework for generating patient representations”. In: *Bioinform.* 37.19 (2021), pp. 3311–3318. DOI: 10.1093/bioinformatics/btab340. URL: <https://doi.org/10.1093/bioinformatics/btab340>.

- [27] **Mehdi Ali**, Sahar Vahdati, Shruti Singh, Sourish Dasgupta, and Jens Lehmann. “Improving Access to Science for Social Good”. In: *Machine Learning and Knowledge Discovery in Databases - International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part I*. Vol. 1167. Communications in Computer and Information Science. Springer, 2019, pp. 658–673. DOI: 10.1007/978-3-030-43823-4_52. URL: https://doi.org/10.1007/978-3-030-43823-4_52.
- [28] **Mehdi Ali**, Charles Tapley Hoyt, Daniel Domingo-Fernández, and Jens Lehmann. “Predicting Missing Links Using PyKEEN”. In: *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019*. Vol. 2456. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 245–248. URL: <http://ceur-ws.org/Vol-2456/paper64.pdf>.
- [29] Veronika Henk, Sahar Vahdati, Mojataba Nayyeri, **Mehdi Ali**, Hamed Shariat Yazdi, and Jens Lehmann. “Metaresearch recommendations using knowledge graph embeddings”. In: *RecNLP workshop of AAAI Conference*. 2019.
- [30] Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. “Beyond Triplets: Hyper-Relational Knowledge Graph Embedding for Link Prediction”. In: *WWW*. ACM, 2020, pp. 1885–1896.
- [31] Marjan Albooyeh, Rishab Goel, and Seyed Mehran Kazemi. “Out-of-Sample Representation Learning for Knowledge Graphs”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*. Ed. by Trevor Cohn, Yulan He, and Yang Liu. Association for Computational Linguistics, 2020, pp. 2657–2666.
- [32] Rajarshi Bhowmik and Gerard de Melo. “Explainable Link Prediction for Emerging Entities in Knowledge Graphs”. In: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference*. Ed. by Jeff Z. Pan, Valentina A. M. Tamma, Claudia d’Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal. Vol. 12506. Lecture Notes in Computer Science. Springer, 2020, pp. 39–55.
- [33] Louis Clouâtre, Philippe Trempe, Amal Zouaq, and Sarath Chandar. “MLMLM: Link Prediction with Mean Likelihood Masked Language Model”. In: *ACL/IJCNLP (Findings)*. Vol. ACL/IJCNLP 2021. Findings of ACL. Association for Computational Linguistics, 2021, pp. 4321–4331.
- [34] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. “A semantic matching energy function for learning with multi-relational data - Application to word-sense disambiguation”. In: *Mach. Learn.* 94.2 (2014), pp. 233–259.
- [35] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. “Learning Structured Embeddings of Knowledge Bases”. In: *AAAI*. AAAI Press, 2011.

- [36] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. “Translating Embeddings for Modeling Multi-relational Data”. In: *NIPS*. 2013, pp. 2787–2795.
- [37] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. “Knowledge Graph Embedding by Translating on Hyperplanes”. In: *AAAI*. AAAI Press, 2014, pp. 1112–1119.
- [38] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. “Learning Entity and Relation Embeddings for Knowledge Graph Completion”. In: *AAAI*. AAAI Press, 2015, pp. 2181–2187.
- [39] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. “Knowledge Graph Embedding via Dynamic Mapping Matrix”. In: *ACL (1)*. The Association for Computer Linguistics, 2015, pp. 687–696.
- [40] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. “RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space”. In: *ICLR (Poster)*. OpenReview.net, 2019.
- [41] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. “Multi-relational Poincaré Graph Embeddings”. In: *NeurIPS*. 2019, pp. 4465–4475.
- [42] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. “Learning to Represent Knowledge Graphs with Gaussian Embedding”. In: *CIKM*. ACM, 2015, pp. 623–632.
- [43] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. “A Three-Way Model for Collective Learning on Multi-Relational Data”. In: *ICML*. Omnipress, 2011, pp. 809–816.
- [44] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. “Embedding Entities and Relations for Learning and Inference in Knowledge Bases”. In: *ICLR (Poster)*. 2015.
- [45] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. “Complex Embeddings for Simple Link Prediction”. In: *ICML*. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 2071–2080.
- [46] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. “Quaternion Knowledge Graph Embeddings”. In: *NeurIPS*. 2019, pp. 2731–2741.
- [47] Seyed Mehran Kazemi and David Poole. “Simple Embedding for Link Prediction in Knowledge Graphs”. In: *NeurIPS*. 2018, pp. 4289–4300.
- [48] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. “Tucker: Tensor Factorization for Knowledge Graph Completion”. In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 5184–5193.
- [49] Ledyard R Tucker et al. “The extension of factor analysis to three-dimensional matrices”. In: *Contributions to mathematical psychology* 110119 (1964).

- [50] Baoxu Shi and Tim Wenginger. “ProjE: Embedding Projection for Knowledge Graph Completion”. In: *AAAI*. AAAI Press, 2017, pp. 1236–1242.
- [51] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. “Holographic Embeddings of Knowledge Graphs”. In: *AAAI*. AAAI Press, 2016, pp. 1955–1961.
- [52] Xin Dong, Evgeniy Gabrilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. “Knowledge vault: a web-scale approach to probabilistic knowledge fusion”. In: *KDD*. ACM, 2014, pp. 601–610.
- [53] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. “Reasoning With Neural Tensor Networks for Knowledge Base Completion”. In: *NIPS*. 2013, pp. 926–934.
- [54] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. “A novel embedding model for knowledge base completion based on convolutional neural network”. In: *arXiv preprint arXiv:1712.02121* (2017).
- [55] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. “Convolutional 2D Knowledge Graph Embeddings”. In: *AAAI*. AAAI Press, 2018, pp. 1811–1818.
- [56] Sameh K. Mohamed, Vít Nováček, Pierre-Yves Vandebussche, and Emir Muñoz. “Loss Functions in Knowledge Graph Embedding Models”. In: *DL4KG@ESWC*. Vol. 2377. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 1–10.
- [57] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. “Canonical Tensor Decomposition for Knowledge Base Completion”. In: *ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2869–2878.
- [58] Max Berrendorf, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. “Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods with Adjusted Mean Rank”. In: *CoRR* abs/2002.06914 (2020).
- [59] Sameh K. Mohamed, Vít Nováček, and Aayah Nounu. “Discovering protein drug targets using knowledge graph embeddings”. In: *Bioinform.* 36.2 (2020), pp. 603–610.
- [60] Maxat Kulmanov, Wang Liu-Wei, Yuan Yan, and Robert Hoehndorf. “EL Embeddings: Geometric Construction of Models for the Description Logic EL++”. In: *IJCAI*. ijcai.org, 2019, pp. 6103–6109.
- [61] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems* 32. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.nurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

- [62] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. “You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings”. In: *ICLR*. OpenReview.net, 2020.
- [63] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *KDD*. ACM, 2019, pp. 2623–2631.
- [64] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha P. Talukdar, and Yiming Yang. “A Re-evaluation of Knowledge Graph Completion Methods”. In: *ACL*. Association for Computational Linguistics, 2020, pp. 5516–5522.
- [65] Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. “LibKGE - A knowledge graph embedding library for reproducible research”. In: *EMNLP (Demos)*. Association for Computational Linguistics, 2020, pp. 165–174.
- [66] Farahnaz Akrami, Lingbing Guo, Wei Hu, and Chengkai Li. “Re-evaluating Embedding-Based Knowledge Graph Completion Methods”. In: *CIKM*. ACM, 2018, pp. 1779–1782.
- [67] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha P. Talukdar, and Yiming Yang. “A Re-evaluation of Knowledge Graph Completion Methods”. In: *ACL*. Association for Computational Linguistics, 2020, pp. 5516–5522.
- [68] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *ICLR (Poster)*. 2015.
- [69] Matthew D. Zeiler. “ADADELTA: An Adaptive Learning Rate Method”. In: *CoRR* abs/1212.5701 (2012).
- [70] Rui Wang, Bicheng Li, Shengwei Hu, Wenqian Du, and Min Zhang. “Knowledge Graph Embedding via Graph Attenuated Attention Networks”. In: *IEEE Access* 8 (2020), pp. 5212–5224. DOI: 10.1109/ACCESS.2019.2963367. URL: <https://doi.org/10.1109/ACCESS.2019.2963367>.
- [71] Jinheon Baek, Dong Bok Lee, and Sung Ju Hwang. “Learning to Extrapolate Knowledge: Transductive Few-shot Out-of-Graph Link Prediction”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020.
- [72] Daniel Daza, Michael Cochez, and Paul Groth. *Inductive Entity Representations from Text via Link Prediction*. 2020. arXiv: 2010.03496 [cs.CL].
- [73] Hai-Cheng Yi, Zhu-Hong You, De-Shuang Huang, and Chee Keong Kwoh. “Graph representation learning in bioinformatics: trends, methods and applications”. In: *Briefings Bioinform.* 23.1 (2022).

- [74] David S. Wishart et al. "DrugBank 5.0: a major update to the DrugBank database for 2018". In: *Nucleic Acids Res.* 46.Database-Issue (2018), pp. D1074–D1082. DOI: 10.1093/nar/gkx1037. URL: <https://doi.org/10.1093/nar/gkx1037>.
- [75] Minoru Kanehisa, Miho Furumichi, Mao Tanabe, Yoko Sato, and Kanae Morishima. "KEGG: new perspectives on genomes, pathways, diseases and drugs". In: *Nucleic acids research* 45.D1 (2017), pp. D353–D361.
- [76] "Gene Ontology Consortium: The Gene Ontology (GO) database and informatics resource". In: *Nucleic Acids Res.* 32.Database-Issue (2004), pp. 258–261. DOI: 10.1093/nar/gkh036. URL: <https://doi.org/10.1093/nar/gkh036>.
- [77] Charles Tapley Hoyt, Daniel Domingo-Fernández, Sarah Mubeen, Josep Marin Llaó, Andrej Konotopez, Christian Ebeling, Colin Birkenbihl, Özlem Muslu, Bradley English, Simon Müller, et al. "Integration of structured biological data sources using biological expression language". In: *Biorxiv* (2019), p. 631812.
- [78] Daniel Domingo-Ferández, Charles Tapley Hoyt, Carlos Bobis-Álvarez, Josep Marín-Llaó, and Martin Hofmann-Apitius. "ComPath: an ecosystem for exploring, analyzing, and curating mappings across pathway databases". In: *NPJ systems biology and applications* 4.1 (2018), pp. 1–8.
- [79] Susanne G Mueller, Michael W Weiner, Leon J Thal, Ronald C Petersen, Clifford Jack, William Jagust, John Q Trojanowski, Arthur W Toga, and Laurel Beckett. "The Alzheimer's disease neuroimaging initiative". In: *Neuroimaging Clinics* 15.4 (2005), pp. 869–877.
- [80] Andrew J Saykin, Li Shen, Xiaohui Yao, Sungeun Kim, Kwangsik Nho, Shannon L Risacher, Vijay K Ramanan, Tatiana M Foroud, Kelley M Faber, Nadeem Sarwar, et al. "Genetic studies of quantitative MCI and AD phenotypes in ADNI: Progress, opportunities, and plans". In: *Alzheimer's & Dementia* 11.7 (2015), pp. 792–814.
- [81] Bijay Jassal, Lisa Matthews, Guilherme Viteri, Chuqiao Gong, Pascual Lorente, Antonio Fabregat, Konstantinos Sidiropoulos, Justin Cook, Marc Gillespie, Robin Haw, et al. "The reactome pathway knowledge-base". In: *Nucleic acids research* 48.D1 (2020), pp. D498–D503.
- [82] Denise N Slenter, Martina Kutmon, Kristina Hanspers, Anders Riutta, Jacob Windsor, Nuno Nunes, Jonathan Mélius, Elisa Cirillo, Susan L Coort, Daniela Digles, et al. "WikiPathways: a multifaceted pathway database bridging metabolomics to other omics research". In: *Nucleic acids research* 46.D1 (2018), pp. D661–D667.
- [83] Rose Oughtred, Chris Stark, Bobby-Joe Breitkreutz, Jennifer Rust, Lorie Boucher, Christie Chang, Nadine Kolas, Lara O'Donnell, Genie Leung, Rochelle McAdam, et al. "The BioGRID interaction database: 2019 update". In: *Nucleic acids research* 47.D1 (2019), pp. D529–D541.

- [84] Sandra Orchard, Mais Ammari, Bruno Aranda, Lionel Breuza, Leonardo Briganti, Fiona Broackes-Carter, Nancy H Campbell, Gayatri Chavali, Carol Chen, Noemi Del-Toro, et al. "The MIntAct project—IntAct as a common curation platform for 11 molecular interaction databases". In: *Nucleic acids research* 42.D1 (2014), pp. D358–D363.
- [85] Igor Rodchenkov, Ozgun Babur, Augustin Luna, Bulent Arman Aksoy, Jeffrey V Wong, Dylan Fong, Max Franz, Metin Can Siper, Manfred Cheung, Michael Wrana, et al. "Pathway Commons 2019 Update: integration, analysis and exploration of pathway data". In: *Nucleic acids research* 48.D1 (2020), pp. D489–D497.
- [86] Stephen Bonner, Ian P. Barrett, Cheng Ye, Rowan Swiers, Ola Engkvist, and William L. Hamilton. "Understanding the Performance of Knowledge Graph Embeddings in Drug Discovery". In: *CoRR* abs/2105.10488 (2021).

Appendices

Appendix A

The KEEN Universe: An Ecosystem for Knowledge Graph Embeddings with a Focus on Reproducibility and Transferability



The KEEN Universe

An Ecosystem for Knowledge Graph Embeddings with a Focus on Reproducibility and Transferability

Mehdi Ali^{1,2} (✉), Hajira Jabeen¹, Charles Tapley Hoyt³, and Jens Lehmann^{1,2}

¹ Smart Data Analytics Group, University of Bonn, Bonn, Germany
{mehdi.ali, jabeen, jens.lehmann}@cs.uni-bonn.de

² Department of Enterprise Information Systems, Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), Sankt Augustin and Dresden, Germany
{mehdi.ali, jens.lehmann}@iais.fraunhofer.de

³ Department of Bioinformatics,
Fraunhofer Institute for Algorithms and Scientific Computing (SCAI),
Sankt Augustin, Germany
charles.hoyt@scai.fraunhofer.de

Abstract. There is an emerging trend of embedding knowledge graphs (KGs) in continuous vector spaces in order to use those for machine learning tasks. Recently, many knowledge graph embedding (KGE) models have been proposed that learn low dimensional representations while trying to maintain the structural properties of the KGs such as the similarity of nodes depending on their edges to other nodes. KGEs can be used to address tasks within KGs such as the prediction of novel links and the disambiguation of entities. They can also be used for downstream tasks like question answering and fact-checking. Overall, these tasks are relevant for the semantic web community. Despite their popularity, the reproducibility of KGE experiments and the transferability of proposed KGE models to research fields outside the machine learning community can be a major challenge. Therefore, we present the KEEN Universe, an ecosystem for knowledge graph embeddings that we have developed with a strong focus on reproducibility and transferability. The KEEN Universe currently consists of the Python packages PyKEEN (Python KnowlEdge EmbeddiNgs), BioKEEN (Biological KnowlEdge EmbeddiNgs), and the KEEN Model Zoo for sharing trained KGE models with the community.

Resource Type: Software Framework

License: MIT License

Permanent URL: https://figshare.com/articles/The_KEEN_Universe/7957445.

Keywords: Knowledge graph embeddings · Machine learning · Semantic web

1 Introduction

In the last two decades, representing factual information as knowledge graphs (KGs) has gained significant attention. KGs have been successfully applied to tasks such as link prediction, clustering, and question answering. In the context of this paper, a KG is a directed, multi-relational graph that represents entities as nodes, and their relations as edges, and can be used as an abstraction of the real world. Factual information contained in KGs is represented as triples of the form (h, r, t) , where h and t denote the head and tail entities, and r denotes their respective relation. Prominent examples of KGs are DBpedia [18], Wikidata [25], Freebase [5], and Knowledge Vault [10]. Traditionally, KGs have been processed in their essential form as symbolic systems, but recently, knowledge graph embedding models (KGEs) have become popular that encode the nodes and edges of KGs into low-dimensional continuous vector spaces while best preserving the structural properties of the KGs. The learned embeddings can be used to perform algebraic operations on the corresponding KGs, and common tasks are link prediction and entity disambiguation [26]. Furthermore, we can observe that KGEs are applied in downstream tasks such as question answering (QA) [23].

Although KGEs are becoming popular, the reproducibility of KGE experiments and the transferability of the proposed models to research fields outside the machine learning community such as the semantic web or the biomedical domain remains a challenge. Depending on the used hyper-parameter values and the optimization approach, the model performance can vary significantly. For instance, in the experiments performed by Akrami *et al.* [2] an increase of 14.4% for the TransE model and 23.6% for the DistMult model in the $hits@k$ metric has been reported. However, the reasons for the performance discrepancies are often not discussed in depth [29, 30], impeding the reproducibility of experiments. Furthermore, applying proposed KGE models requires both expertise in KGEs and in implementing these models which can be obstacles for non-machine learning researchers. These are significant shortcomings considering that in research fields like the semantic web or the bioinformatics community, KGs are widely applied, and KGE models might have a strong potential to be used in many tasks. Initiatives like the SIGMOD¹ guidelines defined by the database community or the FAIR data principles [28] highlight that reproducibility and transferability is not only a fundamental challenge inside the research field of KGEs, but it is a cross-domain issue.

In this paper, we describe a software ecosystem that we have developed with a strong emphasis on reproducibility and transferability. Our contribution is the KEEN Universe that currently consists of: (i) PyKEEN (Python Knowledge Graph EmbeddiNgs), a Python package encapsulating the machine learning functionalities, (ii) BioKEEN (Biological KnowlEdge Graph EmbeddiNgs) [3], a Python package specifically developed to facilitate the use of KGEs within the bioinformatics community and (iii) the KEEN Model Zoo, a platform to share

¹ <http://db-reproducibility.seas.harvard.edu/>.

pre-trained KGE models. Furthermore, we evaluate the usability of the KEEN Universe on two case scenarios from the area of scholarly metadata research and bioinformatics.

2 Impact and Use Cases

2.1 Impact

Impact on the KGE Community. By providing an ecosystem that enables researchers to easily share code, experimental set-ups and research results without requiring additional overhead, the KEEN Universe is an essential step in the direction of reproducible KGE research. Specifically, researchers can integrate their new KGE models into our ecosystem to enhance comparability with existing approaches as well as to share their trained models through our model zoo to make it easily accessible for the community. The functionalities provided by the KEEN Universe will save researchers significant amount of time and facilitate the work on complex tasks.

Impact Beyond the KGE Community. KGs have become a standard in representing factual information across different domains. Considering that KGs are often incomplete and noisy, the KEEN Universe can be applied in numerous applications to derive new facts. For instance, the KEEN Universe has been used on scholarly KGs to provide research recommendations [14] and on biomedical KGs to predict associations between biomedical entities [3, 17]. Moreover, it can be used in downstream tasks like QA and dialogue generation [6, 19].

Impact on Industry. KGs are established in several major companies such as Google, Facebook, Bayer, Siemens, and KGEs are for instance used to build KGE based recommender systems [6, 15]. Furthermore, the evolution of industry to *Industry 4.0* paves a new way for KGEs to be applied in the observation of manufacturing processes: (knowledge) graphs are a convenient approach to model the data produced by sensors which can be used to model the status of production pipelines. The encoded information can be fed to machine learning based systems for predictive maintenance. Instead of performing feature engineering which is time-consuming and complex, KGEs can be used to encode the information of KGs [11]. Enterprises could use the KEEN Universe to experiment with KGEs before performing major investments to build their own specialized systems.

Impact on Teaching. The KEEN Universe can be used by students to learn how KGE models and their training and evaluation procedures are implemented which helps them to implement new KGE models that in turn could be integrated into the KEEN Universe. It has been already successfully applied in two master theses and currently, it is being used in a further master thesis to compare link prediction approaches based on handcrafted KG features against KGEs based link prediction approaches. Furthermore, it is used in the Knowledge Graph Analysis Lab (University of Bonn) to introduce KGE models to master students.

2.2 Use Cases

Bioinformatics. Bio2Vec² is a project that aims to provide a platform to enable the development of machine learning and data analytic tools for biological KGs with the goal of discovering molecular mechanisms underlying complex diseases and drugs’ modes of action. This project also aims to provide pre-trained embeddings for existing biological data, and additional data created and produced within this project. BioKEEN and PyKEEN have been applied already within Bio2Vec to predict hierarchies and cross-talks between biological pathways [3] and to predict protein-protein interactions [17]. Furthermore, the model to predict interactions between biological pathways has been shared through the KEEN Model Zoo (https://github.com/SmartDataAnalytics/KEEN-Model-Zoo/tree/master/bioinformatics/ComPath/compath_model_01).

Bayer Crop Science R&D. The department of Computational Life Science (CLS) at Bayer Crop Science R&D³ developed a large knowledge graph to describe field trial experiments in which candidates for crop protection products are tested across many experimental settings. The knowledge graph is augmented with trial properties, wherein each node contains information beyond the graph structure. However, a subgraph of the property graph can be extracted in such a way that only important relationships are preserved between nodes. This subgraph is stored as a collection of subject-predicate-object triples to allow for a range of embedding techniques to be easily applied. Since different use cases may require a different approach to mining the graph structure for suggested links or node similarities, it is necessary to have a framework that can simply consume the same graph data and apply new models without a large time investment.

The modular design of PyKEEN makes it a perfect fit for the needs of Bayer CLS researchers. The knowledge graph contains nodes of various categories and relation types, as well as many-to-one and one-to-many relations, requiring the use of advanced embedding methods. In addition, new embedding algorithms can be simply added to or modified from the existing framework. As an initial use case, Bayer CLS researchers implemented the included TransR embedding method to their subgraph and, with very little effort, produced an embedding space that demonstrated clear clusters between node categories. Additionally, they were easily able to add node category support to PyKEEN in order to extend the functionality of the existing TransD algorithm. The team at Bayer CLS expects to provide insights into field trial design, future field trial planning, and data quality checks using link predictions from graph embeddings trained and optimized within PyKEEN.

3 System Description

To improve the reproducibility of KGE experiments, we have defined the following requirements for our ecosystem: (i) provide users the full control of the experimental setup, (ii) provide transparent training procedure for all KGE models,

² <http://bio2vec.net/>.

³ <https://agrar.bayer.de/>.

and (iii) provide identical evaluation procedure for all KGE models. To enable the transferability of KGE research, we have defined two requirements: (i) enable experts and inexperienced users to use the ecosystem, (ii) easy to specialize for requirements in different domains. In the following, we explain how these requirements are addressed within the KEEN Universe. First, we describe PyKEEN (Sect. 3.1), then we introduce BioKEEN (Sect. 3.2), and finally, we present the KEEN Model Zoo (Sect. 3.3).

3.1 PyKEEN

Here, we present PyKEEN’s software architecture, give an overview of the supported data formats, explain our approach for configuring KGE experiments, describe the training and evaluation procedures, describe which experimental artifacts are exported and finally, we present our inference workflow.

Software Architecture. PyKEEN consists of a *configuration* and a *learning layer* (Fig. 1). In the configuration layer, users can define their experiments, i.e. select the KGE model, its hyper-parameters, and define the evaluation procedure. The experimental setup is saved and passed to the learning layer that executes the experiment. In PyKEEN, a KGE model can be trained based on user defined hyper-parameter values or a hyper-parameter optimization can be performed to find suitable values. Finally, the experimental artifacts are exported.

PyKEEN has a modular architecture (Fig. 2) and depending on the task different modules are executed and interact with each other. The *command line interface (CLI)* module enables users to configure experiments through a terminal, the *Pipeline* module starts and controls the configured experiment, *KGE-Model* modules represent KGE models, the *Training* module is responsible for training a *KGEModel* module and the *Evaluator* module for its evaluation. A *HPOptimizer* module performs the hyper-parameter optimization (currently only *random search* is available). To perform inference the *Inference* module has been developed.

Supported Data Formats. PyKEEN supports KGs represented as RDF, from NDEx [22], and as tab-separated values. We provide support for RDF, because it is an established data format to represent KGs [19]. Examples of popular KGs available as RDF are DBpedia [18] and Bio2RDF [4]. NDEx is an online commons for exchanging biological networks, and of interest for life science researchers. Finally, a tab separated file containing the triples of a KG can also be provided directly to PyKEEN. Overall, by supporting these data formats, many KGs can directly be used, allowing users to focus on their experiments rather than on data pre-processing.

Configuration of Experiments. To provide users full control of the experimental setup we have developed the configuration layer (Fig. 1) that enables users to specify every detail of an experiment, i.e. the datasets, the execution mode (training or HPO mode), the KGE model along with its hyper-parameter values, the details of the evaluation procedure, the seed for the random generator,

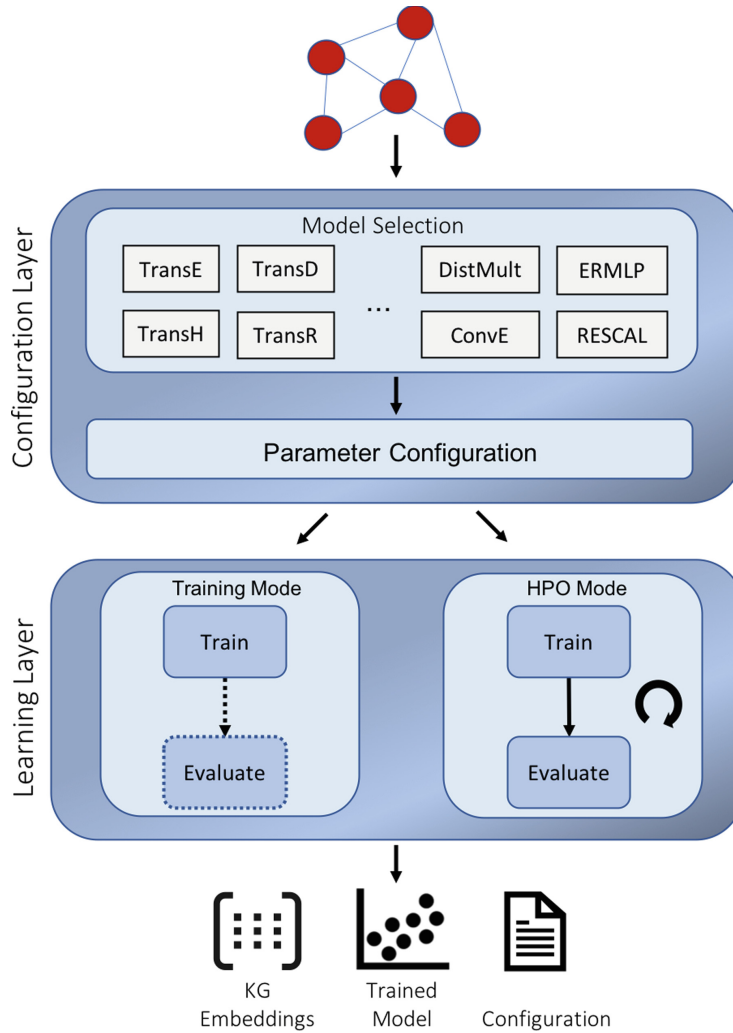


Fig. 1. Software architecture of PyKEEN: (1) the configuration layer assists users to specify experiments and (2) the learning layer trains a model with user-defined hyper-parameters or performs a hyper-parameter search.

and the preferred training device (graphics processing unit (GPU) or CPU). To address experts and inexperienced users, experiments can be either configured through the interactive command line interface (CLI) that assists inexperienced users, or programmatically. The CLI ensures that an experiment is configured correctly. In case that users provide an incorrect value for a hyper-parameter such as a negative number for the embedding dimension, the CLI notifies the users and provides an example of a correct input.

Training of KGE Models. In PyKEEN we have clearly defined training procedures: KGE models are trained based on the *open world assumption* i.e. triples that are not contained in a KG are not considered as non-existing, but as unknowns which might be true or false facts. The models are trained according the algorithm described by Bordes *et al.* [7], and the margin ranking loss and the binary cross entropy are used as loss functions [19]. Selecting suitable hyper-parameter values is fundamental for the model performance and strongly

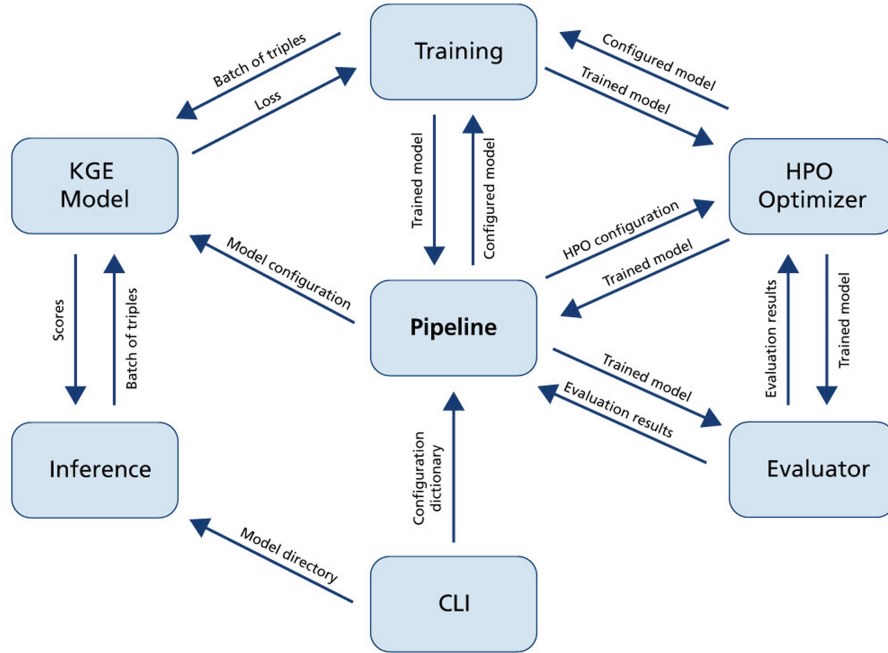


Fig. 2. PyKEEN’s modules and their interactions [3].

depends on the expertise and experience of the users. To address both, experienced and inexperienced users, we have developed the *training* and *hyper-parameter optimization mode (HPO)*. In training mode users provide for each hyper-parameter the corresponding value. Optionally, a trained KGE model can be evaluated in training mode. In HPO mode, users have to define for each hyper-parameter a set of possible values (or single values) and PyKEEN assists users to find suitable hyper-parameter values by applying *random search* [12]. The hyper-parameters obtained by the hyper-parameter optimization can be used later to train the final model in training mode.

Evaluation of KGE Models. Within PyKEEN all the KGE models are evaluated based on the procedure described in Bordes *et al.* [7] and the widely applied metrics *mean rank* and *hits@k* are computed [7]. Users can provide a set of test triples, or they can use PyKEEN to automatically split the input KG into training and test triples based on a user defined splitting ratio. This is especially relevant if a separate test set is not available. Furthermore, users can specify whether they want to compute the mean rank and hits@k in the *raw* or *filtered setting*. In the filtered setting, artificially created negative samples that are contained as positive examples in the training set will be removed [7]. Usually, results for both settings are reported.

Exporting Experimental Artifacts. To ensure the reproducibility of a KGE experiment, we export all relevant experimental artifacts after an experiment is conducted. Specifically, we export a configuration file (JSON) describing the experimental setup, the evaluation results (as JSON file), mappings of entities and relations to unique IDs (JSONs), mappings of entities and relations to

their learned embeddings (JSONs), and the trained model in a serialized format (pickle). The exported artifacts can be distributed by our model zoo.

Inference. Inference can be performed in two ways within PyKEEN. On the one hand, a trained KGE model can be used to provide predictions for a set of triples by calling its *predict* function. On the other hand, we have implemented an inference workflow that provides additional functionalities: for a set of user defined entities and relations, automatically all triple-permutations are created for which predictions are computed. The set of generated triples can be filtered by providing triples that should be removed. This is for instance relevant in a setting, in which predictions for all possible triples except those contained in the training set should be computed. Furthermore, it can be defined that all reflexive triples of the form (e, r, e) should be excluded. The output of the inference workflow is a file containing the triples and their predicted scores where the most plausible triples are located at the beginning of the file.

3.2 BioKEEN

With the development of BioKEEN we demonstrate how KGE research can be transferred to research domains outside the machine learning community. While developing BioKEEN we took into account that expertise in KGE models and in their implementation might be limited in the bioinformatics community. Within BioKEEN we provide direct access to numerous biomedical databases without requiring the user to process them.

Software Architecture. BioKEEN consists of a three-layered architecture (Fig. 3). Its *configuration layer* is an extension of PyKEEN’s configuration layer and enables users to select one of the biomedical databases that are directly accessible through BioKEEN, the *Data Acquisition Layer* provides access to these databases and the learning layer (part of PyKEEN) performs the training of the KGE models.

Easy Access to Numerous Biomedical Databases. Within the biomedical domain, numerous databases containing structured knowledge are available [4]. However, data pre-processing is a time consuming process. For this reason, we have created the *Data Acquisition Layer* that automatically retrieves and converts the content of numerous biomedical databases and makes it available within BioKEEN (a full list is available at https://biokeen.readthedocs.io/en/latest/bio2bel_repositories.html). The data acquisition layer makes use of the Bio2BEL [16] software to access the databases. Bio2BEL is a framework that gathers biological data sources and represents them in the Biological Expression Language (BEL)⁴. By integrating the Bio2BEL software users have direct access to several biomedical databases, can automatically update the database version, and retrieve further databases as they are integrated to Bio2BEL. This functionality allows bioinformaticians to focus on their experiments instead of data pre-processing.

⁴ <http://openbel.org/>.

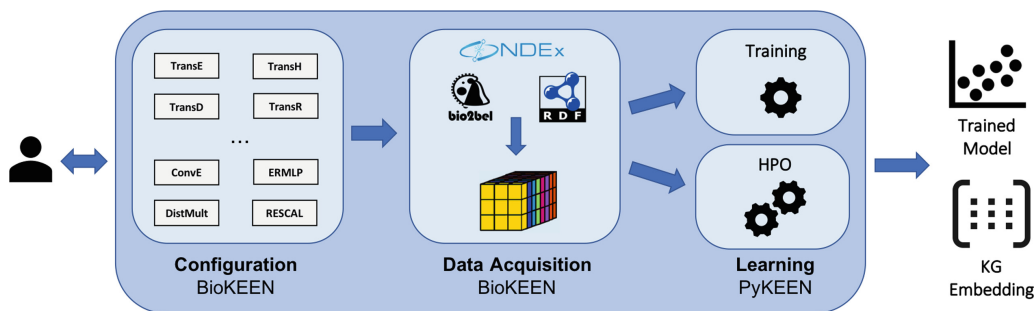


Fig. 3. BioKEEN’s Software Architecture [3].

Overall, the data acquisition layer, the HPO mode and the interactive command line interface are essential features to make KGE research transferable to the domain of bioinformatics considering that the expertise in KGE models and their implementation might be limited.

3.3 KEEN Model Zoo

We have created the KEEN Model Zoo as a GitHub project to provide a platform on which researchers can share their experimental artifacts (i.e. trained KGE models, configuration files, evaluation summaries, etc.) that have been created using components of the KEEN Universe. Providing these artifacts publicly will improve the reproducibility of KGE research, and we aim the community to contribute to this project.

To ensure the quality of the model zoo, we have defined following requirements: (i) conducted experiments must be reported in a scientific paper, (ii) all experimental artifacts that have been created by Py/BioKEEN for an experiment needs to be provided, (iii) the used datasets have to be publicly accessible, (iv) a description of the experiment must be provided, (v) a unit test needs to be implemented checking that the provided model can be instantiated. Within the model zoo, we split experiments based on their research domains (e.g. bioinformatics, scholarly metadata research, etc.), and within each research domain, the experiments are categorized according to the datasets on which the experiments have been conducted.

Researchers that want to share their experimental artifacts are asked to create a *pull request* that will be reviewed and *merged* into the *master branch* if all requirements are fulfilled.

4 Implementation

We have implemented PyKEEN and BioKEEN in Python since it is an established programming language for implementing machine learning models⁵. PyTorch [21] has been used as the underlying machine learning framework,

⁵ <https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning>.

because it provides flexibility in implementing machine learning models, is easy to debug and through its GPU support the training procedure can be accelerated. Furthermore, we make use of the scientific Python stack for scientific computing (NumPy⁶, SciPy⁷, Scikit-Learn⁸, Pandas⁹). Moreover, we apply following community standards: *flake8*¹⁰ to ensure code quality, *setuptools*¹¹ to create distributions, *pyroma*¹² to ensure package metadata standards, *sphinx*¹³ to build our documentation and *Read the Docs*¹⁴ to host it. Finally, *Travis-CI*¹⁵ is used as continuous integration server.

Extensibility. The KEEN Universe can be extended in various ways. New machine learning related components can be added (extension of PyKEEN is required), further data reader can be implemented to load additional data formats (extension of PyKEEN), further components specifically relevant for the bioinformatics community can be integrated (extension of BioKEEN is required), finally extensions of PyKEEN specialized for further research domains can be created. Here, we describe how new machine learning components can be integrated into our ecosystem by extending PyKEEN. Figure 2 depicts the sub-modules of PyKEEN and the most relevant with regards to an extension are the *KGE-Model* and the *HPOOptimizer* modules. The modular architecture of PyKEEN facilitates its extension.

Integration of an Additional KGE Model. Within PyKEEN, a *KGEModel* module interacts with the *Pipeline*, the *Training*, and the *Inference* module (Fig. 2). To ensure that a new KGE model can interact with these modules, it needs to provide implementations of a *forward()* and a *predict()* function. The *forward* should expect two multi-dimensional arrays (tensors) containing the batch of positive and negative training triples (or a batch of training triples and corresponding labels; depends on the KGE model) and return the loss value computed for this batch. The *predict* function should expect a tensor of triples for which predictions should be computed and returned. There are no further constraints for the model implementation.

Integration of an Additional Hyper-Parameter Optimization Algorithm. Currently, random search is applied to perform hyper-parameter optimizations and *RandomSearchHPO* is the corresponding module. It extends our abstract class *AbstractHPOOptimizer* which contains the two abstract functions *optimise_hyperparams* and *sample_parameter_value*, where the former is used to

⁶ <http://www.numpy.org/>.

⁷ <https://www.scipy.org/>.

⁸ <https://scikit-learn.org/stable/>.

⁹ <https://pandas.pydata.org/>.

¹⁰ <http://flake8.pycqa.org/en/latest/>.

¹¹ <https://github.com/pypa/setuptools/tree/master/setuptools>.

¹² <https://github.com/regebro/pyroma>.

¹³ <http://www.sphinx-doc.org/en/master/>.

¹⁴ <https://readthedocs.org/>.

¹⁵ <https://travis-ci.org/>.

initiate the optimization procedure and the latter is called in each optimization iteration to sample new hyper-parameter values. To add a new hyper-parameter optimizer, the respective module has to extend the abstract class *AbstractHPOoptimizer* and provide implementations for its two abstract functions to ensure that the optimizer can interact with the *Pipeline*, the *Training*, and the *Evaluator* module.

Overall, the modular architecture of PyKEEN and the simple API of the KGE and hyper-parameter optimization modules facilitate the integration of new machine learning components to PyKEEN.

5 Availability and Maintenance

Availability. PyKEEN, BioKEEN and the KEEN Model Zoo are available at our GitHub repositories under the MIT License. Furthermore, PyKEEN and BioKEEN are also available through PyPI enabling users to install the software packages easily through **pip**.

Maintenance. We aim that researchers from different communities (e.g., semantic web, machine learning, bioinformatics, crop science) will support us in maintaining and extending the KEEN Universe. Before this state is reached, the maintenance of the KEEN Universe is ensured through the Bio2Vec¹⁶ and the German national funded BmBF project MLwin¹⁷ at least till 2022.

6 Evaluation of the Usability of the KEEN Universe

Usability is defined as the extent a software system can be used to achieve a goal with *effectiveness* (extent to which the tasks can be completed), *efficiency* (resources required to achieve the goals) and *satisfaction* (feeling of the users towards the software) in a specified context [1]. We evaluate these aspects based on two case scenarios: co-author recommendations for a scholarly KG, and the predictions of crosstalks and hierarchies between biological pathways.

6.1 Co-author Recommendations Based on KGEs

In the work of Henk *et al.* [14], PyKEEN has been used to provide co-author recommendations based on KGEs for a scholarly KG. The KG contains the entity types *author*, *paper*, *department* and *event*. Furthermore, it contains the relationship types *isAuthorOf*, *isCoAuthorOf*, *isAffiliatedIn* and *isPublished*. The goal was to evaluate co-author recommendations i.e. triples of the form (*author*, *isCoAuthorOf*, *author*). For additional information including the experimental set-up and the evaluation, we refer to [14] and the final experimental artifacts are available at our model

¹⁶ <http://bio2vec.net/>.

¹⁷ <https://mlwin.de/>.

zoo (https://github.com/SmartDataAnalytics/KEEN-Model-Zoo/tree/master/scholarly_data_related_recommendations/SG4MR/sg4mr_model_01).

Effectiveness. The KEEN Universe provides all components to completely achieve the goal: PyKEEN has been used to train four KGE models (DistMult, TransE, TransH and TransR) on the KG, and through the hyper-parameter optimization mode, a suitable combination of KGE model and hyper-parameter values has been automatically determined. Based on the model that performed best, we have used the inference workflow to provide co-author recommendations which have been manually evaluated by a domain expert that classified the top predictions as valid recommendations.

Efficiency. Considering efficiency with regards to the computation time, we made use of the GPU support of PyKEEN (PyTorch) to reduce the training time. The models have been trained on a single GPU. Efficiency with respect to the time necessary to learn the software to be able to solve the task, the main author could quickly set-up and run her experiments through the command line interface which assisted and ensured that the experiments have been configured correctly. The whole process has been performed without any programming required by the author.

Satisfaction. The main author didn't have any prior knowledge about KGEs and the software ecosystem, but she could easily achieve her goals. This positive experience has helped her to get into the field of KGEs.

6.2 Prediction of Cross-Talks and Hierarchies Between Biological Pathways

In the work of Ali *et al.* [3], BioKEEN has been used to predict novel cross-talks and hierarchies between biological pathways. ComPath [9], a novel database for biological pathways has been used to train the KGE models. ComPath contains two types of relationships: *equivalentTo* expressing that two pathways correspond to the same biological process, and *isPartOf* expressing a hierarchy of pathways. Again, we refer to [3] for additional information and to https://github.com/SmartDataAnalytics/KEEN-Model-Zoo/tree/master/bioinformatics/ComPath/compath_model_01 to access the experimental artifacts of the final model.

Effectiveness. The KEEN Universe provides all components to completely achieve the goal: We have used BioKEEN to train five KGE models (UM, DistMult, TransE, TransH and TransR) on ComPath that is directly accessible through BioKEEN. We performed a hyper-parameter optimization to find the best combination of KGE model and hyper-parameters, showed the sensibility of choosing appropriate hyper-parameter values and the effectiveness of the HPO mode to find suitable hyper-parameter values (performance increase from 19.10% to 63.20% for the hits@k metric). The final model has been used to predict new interactions between pathways based on the inference workflow.

The top predictions have been evaluated by domain experts and we found following novel links that have been added to ComPath: the first link states that the *TGF-beta signaling pathway* is equivalent to the *TGF-beta Receptor Signaling pathway*, and the second link expresses that *Lipoic Acid* is part of *Lipid Metabolism*.

Efficiency. Because ComPath is not a large KG, we trained the KGE models on a single CPU (efficiency regarding computation time). Furthermore, no pre-processing of the dataset was required since it is directly accessible within BioKEEN. Although the primary author has no domain expertise regarding pathway interactions, he effortlessly provided new predictions to domain experts who validated them (efficiency with respect to use the software for solving the task).

Satisfaction. Through BioKEEN the main author was able to get to know a new application area in the field of bioinformatics. Furthermore, researchers from different research fields could work successfully in an interdisciplinary team.

7 Related Work

Supported KGE Models. KGE models can be divided into *translational distance models (TDM)* and *semantic matching models (SMM)* where the former compute the plausibility of a fact by a distance function (e.g. using the Euclidean norm) and the latter apply similarity-based scoring functions (considering the similarity of the latent features of the entities and relations) [26]. Table 1 lists all the KGE models that are currently available within the KEEN Universe.

Existing Ecosystems for KGE Models. The available software for KGE models is limited, and an ecosystem like the KEEN Universe is to the best of our knowledge unique. However, there exist software projects that provide implementations of different KGE models. One of them is `scikit-kge`¹⁸ that provides implementations of three KGE models and different negative sampling approaches. The project doesn't seem to be maintained since the last commit dates back to the year 2016. A recently published framework which enables users to train and evaluate several KGE models is OpenKE [13] that can be compared to PyKEEN (Sect. 3.1). While allowing users to reproduce KGE experiments, we argue that it has not been developed with the goal of making KGE research transferable to domains outside the machine learning community and usable for both, experts and non-experts. For instance, it supports only one data format (a text-file consisting of three columns) whereas within PyKEEN a KG can be provided as tab separated values, RDF and from NDEX (Sect. 3.1). Users without expertise in programming might face difficulties to run the software since it doesn't provide an interactive command line interface, and users without expertise in KGE models might have issues in finding appropriate combinations of

¹⁸ <https://github.com/mnick/scikit-kge>.

Table 1. KGE Models available within the KEEN Universe.

Type	Reference	Model
TDM	[26]	TransE
	[26]	TransH
	[26]	TransR
	[26]	TransD
	[26]	Unstructured Model (UM)
	[26]	Structured Embedding (SE)
SMM	[20]	RESCAL
	[26]	DistMult
	[26]	ERMLP
	[8]	ConvE

KGE models and corresponding hyper-parameter values since it doesn't provide a hyper-parameter optimization procedure. Further software repositories containing implementation for different KGE models can be found at¹⁹ and²⁰.

8 Limitations and Future Work

Currently, all the KGE models available within the KEEN Universe make only use of the triples of a KG. However, several KGs contain additional information such as textual descriptions of entities, images and numerical values which can be used to train multimodal KGE models. Based on multimodal data, KGE models can be developed that are capable of performing inference among different KGs which is currently not possible with models that are trained only based on the entities and relations of a KG [30]. We plan to integrate an additional software package to our ecosystem that contains implementations of multimodal KGE models.

Within PyKEEN, negative samples are created based on the approach described in Bordes *et al.* [7]. However, it has been shown that alternative approaches such as *bern* [27] can yield better performance. Therefore, we aim to implement additional negative sampling approaches.

KGE models are evaluated within our ecosystem based on the widely applied metrics *mean rank* and *hits@k*, but additional metrics such as the *AUC-ROC* and *AUC-PR* curve might be of interest [19]. Furthermore, Sharma *et al.* [24] propose a geometrical analysis of learned embeddings that can provide valuable insights. We plan to implement these additional evaluation metrics within the KEEN Universe.

¹⁹ <https://github.com/bookmanhan/Embedding>.

²⁰ <https://github.com/TimDettmers/ConvE>.

Acknowledgments. This work was partly supported by the KAUST project grant Bio2Vec (grant no. 3454), the European Union’s Horizon 2020 funded project Big-DataOcean (GA no. 732310), the CLEOPATRA project (GA no. 812997), and the German national funded BmBF project MLwin.

References

1. Abran, A., Khelifi, A., Suryn, W., Seffah, A.: Usability meanings and interpretations in ISO standards. *Softw. Qual. J.* **11**(4), 325–338 (2003)
2. Akrami, F., Guo, L., Hu, W., Li, C.: Re-evaluating embedding-based knowledge graph completion methods. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018*, pp. 1779–1782. ACM, New York (2018). <https://doi.org/10.1145/3269206.3269266>
3. Ali, M., Hoyt, C.T., Domingo-Fernández, D., Lehmann, J., Jabeen, H.: BioKEEN: a library for learning and evaluating biological knowledge graph embeddings. *Bioinformatics* (2019). <https://doi.org/10.1093/bioinformatics/btz117>
4. Belleau, F., et al.: Bio2RDF: towards a mashup to build bioinformatics knowledge systems. *J. Biomed. Inform.* **41**(5), 706–716 (2008)
5. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 1247–1250. ACM (2008)
6. Bonatti, P.A., Decker, S., Polleres, A., Presutti, V.: Knowledge graphs: new directions for knowledge representation on the semantic web (dagstuhl seminar 18371). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
7. Bordes, A., et al.: Translating embeddings for modeling multi-relational data. In: *Advances in Neural Information Processing Systems*, pp. 2787–2795 (2013)
8. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. *arXiv preprint [arXiv:1707.01476](https://arxiv.org/abs/1707.01476)* (2017)
9. Domingo-Fernandez, D., Hoyt, C.T., Bobis-Álvarez, C., Marin-Llao, J., Hofmann-Apitius, M.: ComPath: an ecosystem for exploring, analyzing, and curating mappings across pathway databases. *NPJ Syst. Biol. Appl.* **5**(1), 3 (2018)
10. Dong, X., et al.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 601–610. ACM (2014)
11. Garofalo, M., Pellegrino, M.A., Altabba, A., Cochez, M.: Leveraging knowledge graph embedding techniques for industry 4.0 use cases. *arXiv preprint [arXiv:1808.00434](https://arxiv.org/abs/1808.00434)* (2018)
12. Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. MIT Press, Cambridge (2016)
13. Han, X., et al.: OpenKE: an open toolkit for knowledge embedding. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 139–144 (2018)
14. Henk, V., Vahdati, S., Nayyeri, M., Ali, M., Yazdi, H.S., Lehmann, J.: Metaresearch recommendations using knowledge graph embeddings. In: *AAAI 2019 Workshop on Recommender Systems and Natural Language Processing (REC/NLP)* (2019)
15. Hildebrandt, M., Sunder, S.S., Mogoreanu, S., Thon, I., Tresp, V., Runkler, T.: Configuration of industrial automation solutions using multi-relational recommender systems. In: Brefeld, U., et al. (eds.) *ECML PKDD 2018*. LNCS (LNAI),

- vol. 11053, pp. 271–287. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-10997-4_17
16. Hoyt, C.T., et al.: Integration of structured biological data sources using biological expression language. *BioRxiv*, p. 631812 (2019)
 17. Kulmanov, M., Liu-Wei, W., Yan, Y., Hoehndorf, R.: EL embeddings: geometric construction of models for the description logic EL++. *arXiv preprint arXiv:1902.10499* (2019)
 18. Lehmann, J., et al.: DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia. *Semant. Web J.* **6**(2), 167–195 (2015). Outstanding Paper Award (Best 2014 SWJ Paper)
 19. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. *Proc. IEEE* **104**(1), 11–33 (2016)
 20. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, pp. 809–816 (2011)
 21. Paszke, A., et al.: Automatic differentiation in pytorch. In: *NIPS-W* (2017)
 22. Pratt, D., et al.: NDEx, the network data exchange. *Cell Syst.* **1**(4), 302–305 (2015)
 23. Saha, A., Pahuja, V., Khapra, M.M., Sankaranarayanan, K., Chandar, S.: Complex sequential question answering: towards learning to converse over linked question answer pairs with a knowledge graph. *arXiv preprint arXiv:1801.10314* (2018)
 24. Sharma, A., Talukdar, P., et al.: Towards understanding the geometry of knowledge graph embeddings. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 122–131 (2018)
 25. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. *Commun. ACM* **57**(10), 78–85 (2014)
 26. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.* **29**(12), 2724–2743 (2017)
 27. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: *AAAI*, pp. 1112–1119. Citeseer (2014)
 28. Wilkinson, M.D., et al.: The FAIR guiding principles for scientific data management and stewardship. *Sci. Data* **3** (2016)
 29. Xiao, H., et al.: SSP: semantic space projection for knowledge graph embedding with text descriptions. In: *Thirty-First AAAI Conference on Artificial Intelligence* (2017)
 30. Xie, R., et al.: Representation learning of knowledge graphs with entity descriptions. In: *Thirtieth AAAI Conference on Artificial Intelligence* (2016)

Appendix B

PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings

PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings

Mehdi Ali*

Smart Data Analytics Group, University of Bonn & Fraunhofer IAIS

MEHDI.ALI@CS.UNI-BONN.DE

Max Berrendorf*

Ludwig-Maximilians-Universität München

BERRENDORF@DBS.IFI.LMU.DE

Charles Tapley Hoyt*

Enveda Biosciences

CHARLES.HOYT@ENVEDATX.COM

Laurent Vermue*

Technical University of Denmark

LAUVE@DTU.DK

Sahand Sharifzadeh

Ludwig-Maximilians-Universität München

SHARIFZADEH@DBS.IFI.LMU.DE

Volker Tresp

Ludwig-Maximilians-Universität München & Siemens AG

VOLKER.TRESP@SIEMENS.COM

Jens Lehmann

Smart Data Analytics Group, University of Bonn & Fraunhofer IAIS

JENS.LEHMANN@CS.UNI-BONN.DE

Editor: Antti Honkela

Abstract

Recently, knowledge graph embeddings (KGEs) have received significant attention, and several software libraries have been developed for training and evaluation. While each of them addresses specific needs, we report on a community effort to a re-design and re-implementation of PyKEEN, one of the early KGE libraries. PyKEEN 1.0 enables users to compose knowledge graph embedding models based on a wide range of interaction models, training approaches, loss functions, and permits the explicit modeling of inverse relations. It allows users to measure each component's influence individually on the model's performance. Besides, an automatic memory optimization has been realized in order to optimally exploit the provided hardware. Through the integration of Optuna, extensive hyper-parameter optimization (HPO) functionalities are provided.

Keywords: Knowledge Graphs, Knowledge Graph Embeddings, Relational Learning

1. Introduction

Knowledge graphs (KGs) encode knowledge as a set of triples $\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ where \mathcal{E} denotes the set of entities and \mathcal{R} the set of relations. Knowledge graph embedding models (KGEMs) learn representations for entities and relations of KGs in vector spaces while preserving the graph structure. The learned embeddings can support machine learning tasks such as entity clustering, link prediction, entity disambiguation, as well as downstream tasks such

*Equal contribution.

as question answering and item recommendation (Nickel et al., 2015; Wang et al., 2017; Ruffinelli et al., 2020; Kazemi et al., 2020).

Most publications of KGEMs are accompanied by reference implementations, but they are seldomly written for reusability or maintained. Existing software packages that provide implementations for different KGEMs usually lack composability: model architectures (or interaction models), training approaches, loss functions, and the usage of explicit inverse relations cannot arbitrarily be combined. The full composability of KGEMs is fundamental for assessing their performance because it allows the assessment of individual components and not solely the sum of differences in published approaches (Ruffinelli et al., 2020). In most previous libraries, only limited functionalities are provided, e.g., a small number of KGEMs are supported, or functionalities such as hyper-parameter optimization (HPO) are missing. For instance, in PyKEEN (Ali et al., 2019a,b), one of the early software packages for KGEMs, models can only be trained under the stochastic local closed-world approach, the evaluation procedure was too slow for larger KGs, and it was designed to be mainly used through a command-line interface rather than programmatically, in order to facilitate its usage for non-experts. This motivated the development of a reusable software package comprising several KGEMs and related methodologies that is entirely configurable.

Here, we present PyKEEN (Python KnowEdge EmbeddiNGs) 1.0, a community effort in which PyKEEN has been re-designed and re-implemented from scratch to overcome the mentioned limitations, to make models entirely configurable, and to extend it with more interaction models and other components.

2. System Description

In PyKEEN 1.0, a KGEM is considered as a composition of four components that can flexibly be combined: an interaction model (or model architecture), a loss function, a training approach, and the usage of inverse relations. PyKEEN 1.0 currently supports 23 interaction models, seven loss functions, four regularizers, two training approaches, HPO, six evaluation metrics, and 21 built-in benchmarking datasets. It can readily import additional datasets that have been pre-stratified into train/test/evaluation and generate appropriate splits for unstratified datasets. Additionally, we implemented an automatic memory optimization that ensures that the available memory is best utilized.

Composable KGEMs To ensure the composability of KGEMs, the interaction models, loss functions, and training approaches are separated from each other and implemented as independent submodules, whereas the modeling of inverse relations is handled by the interaction models. Our modules can be arbitrarily replaced because we ensured through inheritance that all interaction models, loss functions, and training approaches follow unified APIs, which are defined by `pykeen.model.Model`, `pykeen.loss.Loss`, and `pykeen.training.TrainingLoop`. Currently, we provide implementations of 23 interaction models, the most common loss functions used for training KGEMs including the *binary-cross entropy*, *cross entropy*, *mean square error*, *negative-sampling self-adversarial loss*, and the *softplus loss*, as well as the *local closed-world assumption* (also referred as *KvsAll*) and the *stochastic local closed-world assumption* training approach (also referred as *NegSamp*) (Ruffinelli et al., 2020). In PyKEEN, each interaction model can be trained based on both approaches. To enable users to investigate the effect of explicitly modeling

inverse relations (Lacroix et al., 2018; Kazemi and Poole, 2018) on the model’s performance, each model can be trained with explicit inverse relations in PyKEEN 1.0, i.e., for each relation $r \in \mathcal{R}$ an inverse relation r_{inv} is introduced, and the task of predicting the head entity of a (r, t) -pair becomes the task of predicting the tail entity of the corresponding inverse pair (t, r_{inv}) .

To facilitate the composition of KGE models for non-experts, we provide the `pykeen.pipeline.pipeline()` functions, which provides a high-level entry point into the functionalities of PyKEEN. Users define the components to be used, and the pipeline ensures the correct composition of the KGEM and the correct composition of the training and evaluation workflow.

Evaluation KGEMs are usually evaluated on the task of link prediction. Given (h, r) (or (r, t)), all possible entities \mathcal{E} are considered as tail (or head) and ranked according to the KGEMs interaction model. The individual ranks are commonly aggregated to mean rank, mean reciprocal rank, and hits@k. However, these metrics have been realized differently throughout the literature based on different definitions of the rank, leading to difficulties in reproducibility and comparability (Sun et al., 2019). The three most common rank definitions are the *average rank*, *optimistic rank*, and *pessimistic rank*. In PyKEEN 1.0, we explicitly compute the aggregation metrics for all common rank definitions, *average*, *optimistic*, and *pessimistic*, allowing inspection of differences between them. This can help to reveal cases where the model predicts exactly equal scores for many different triples, which is usually an undesired behavior. In addition, we support the recently proposed *adjusted mean rank* (Berrendorf et al., 2020), which allows the comparison of results across differently sized datasets, as well as offering an interface to use all metrics implemented in scikit-learn (Pedregosa et al., 2011), including AUC-PR and AUC-ROC.

Automatic Memory Optimization Allowing high computational throughput, while ensuring that the available hardware memory is not exceeded during training and evaluation, requires the knowledge of the maximum possible training and evaluation batch size for the current model configuration. However, determining the training and evaluation batch sizes is a tedious process, and not feasible when a large set of heterogeneous experiments are run. Therefore, we implemented an automatic memory optimization step that computes the maximum possible training and evaluation batch sizes for the current model configuration and available hardware before the actual experiment starts. If the user-provided batch size is too large for the used hardware, the automatic memory optimization determines the maximum sub-batch size for the training.

Extensibility Because we defined a uniform API for each interaction model, any new model can be integrated by following the API of the existing models (*pykeen.models*). Similarly, the remaining components, e.g., regularizers, and negative samplers follow a unified API, so that new modules can be smoothly integrated.

Community Standards PyKEEN 1.0 relies on several community-oriented tools to ensure it is accessible, reusable, reproducible, and maintainable. It is implemented for Python 3.7+ using the PyTorch package. It comes with a suite of thorough unit tests that are automated with PyTest, Tox, run in a continuous integration setting on GitHub Actions, and are tracked over time using `codecov.io`. Code quality is ensured with `flake8` and careful

Library	AMO	Models	HPO	ES	Evaluation Metrics	Set TA	Set Inv. Rels.	Set Loss Fct.	MGS	DTR
AmpliGraph (Costabello et al., 2019)	-	6	✓	✓	3	-	✓	✓	-	-
DGL-KE (Zheng et al., 2020)	-	6	-	-	3	-	-	✓	✓	✓
GraphVite (Zhu et al., 2019)	-	6	-	-	4	-	-	-	✓	-
LibKGE (Broscheit et al., 2020)	-	10	✓	✓	3*	✓	✓	✓	-	-
OpenKE (Han et al., 2018)	-	11	-	-	3	-	-	✓	-	-
PyTorch-BigGraph (Lerer et al., 2019)	-	4	-	-	4	-	-	✓	✓	✓
Pykg2vec (Yu et al., 2019)	-	18	✓	✓	2	-	-	-	-	-
PyKEEN (Ali et al., 2019b)	-	10	✓	-	2	-	-	-	-	-
PyKEEN 1.0	✓	23	✓	✓	6*	✓	✓	✓	-	-

Table 1: An overview of the functionalities (determined July 2020) of PyKEEN 1.0 and similar libraries. **AMO** refers to automatic memory optimization, **ES** to early stopping, * indicates that ranking metrics are computed for different definitions of the rank, **Set TA** refers to interchanging the training approach, **Set Inv. Rels.** to the explicit modeling of inverse relations, **MGS** to multi-GPU support, i.e., training a single model across several GPUs, and **DTR** to distributed training.

application of the GitHub Flow development workflow. Documentation is quality checked by doc8, built with Sphinx, and hosted on [ReadTheDocs.org](https://readthedocs.org).

3. Comparison to Related Software

Table 1 depicts the most popular KGE frameworks and their features. It shows that PyKEEN 1.0, in comparison with related software packages, emphasizes on both, full composability of KGEMs and extensive functionalities, i.e., a large number of supported interaction models, and extensive evaluation (several metrics are supported) and HPO functionalities. Concerning the evaluation metrics, PyKEEN and LibKGE are the only libraries that compute the ranking metrics (i.e., *mean rank* and *hits@k*) for different definitions of the rank, which ensures that undesired cases are detected in which the model predicts equal scores for many triples. Finally, PyKEEN 1.0 is the only library that performs an automatic memory optimization that ensures that the memory is not exceeded during training and evaluation. GraphVite, DGL-KE, and PyTorch-BibGraph focus on scalability, i.e., they provide support for multi-GPU/CPU or/and distributed training, but focus less on compositionality and extensibility. For instance, PyTorch-BigGraph supports only a small number of interaction models that follow specific computation blocks.

4. Availability and Maintenance

PyKEEN 1.0 is publicly available under the MIT License at <https://github.com/pykeen/pykeen>, and is distributed through the Python Package Index. It will be maintained by the core developer team that is supported by the Smart Data Analytics research group (University of Bonn), Fraunhofer IAIS, Munich Center for Machine Learning (MCML),

Siemens, and the Technical University of Denmark (section for Cognitive Systems and section for Statistics and Data Analysis). The project is funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A and Grant No. 01IS18050D (project MLWin) as well as the Innovation Fund Denmark with the Danish Center for Big Data Analytics driven Innovation (DABAI) which ensures the maintenance of the project in the next years.

References







- Mehdi Ali, Charles Tapley Hoyt, Daniel Domingo-Fernández, Jens Lehmann, and Hajira Jabeen. Biokeen: a library for learning and evaluating biological knowledge graph embeddings. *Bioinformatics*, 35(18):3538–3540, 2019a.
- Mehdi Ali, Hajira Jabeen, Charles Tapley Hoyt, and Jens Lehmann. The keen universe. In *International Semantic Web Conference*, pages 3–18. Springer, 2019b.
- Max Berrendorf, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. Interpretable and fair comparison of link prediction or entity alignment methods with adjusted mean rank. In *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'20)*. IEEE, 2020.
- Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. Libkge - A knowledge graph embedding library for reproducible research. In *EMNLP (Demos)*, pages 165–174. Association for Computational Linguistics, 2020.
- Luca Costabello, Sumit Pai, Chan Le Van, Rory McGrath, Nicholas McCarthy, and Pedro Tabacof. AmpliGraph: a Library for Representation Learning on Knowledge Graphs, March 2019. URL <https://doi.org/10.5281/zenodo.2595043>.
- Xu Han, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*, 2018.
- Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 4284–4295, 2018.
- Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobzyev, Akshay Sethi, Peter Forsyth, and Pascal Poupert. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.
- Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. *arXiv preprint arXiv:1806.07297*, 2018.
- Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference*, Palo Alto, CA, USA, 2019.
- Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You {can} teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2020.
- Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. A re-evaluation of knowledge graph completion methods. *arXiv preprint arXiv:1911.03903*, 2019.
- Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.
- Shih Yuan Yu, Sujit Rokka Chhetri, Arquimedes Canedo, Palash Goyal, and Mohammad Abdullah Al Faruque. Pykg2vec: A python library for knowledge graph embedding. *arXiv preprint arXiv:1906.04239*, 2019.
- Da Zheng, Xiang Song, Chao Ma, Zeyuan Tan, Zihao Ye, Jin Dong, Hao Xiong, Zheng Zhang, and George Karypis. Dgl-ke: Training knowledge graph embeddings at scale. *arXiv preprint arXiv:2004.08532*, 2020.
- Zhaocheng Zhu, Shizhen Xu, Jian Tang, and Meng Qu. Graphvite: A high-performance cpu-gpu hybrid system for node embedding. In *The World Wide Web Conference*, pages 2494–2504, 2019.

Appendix C

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

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.,

- Mehdi Ali and Jens Lehmann are with Smart Data Analytics, University of Bonn, 53113 Bonn, Germany, and also with Fraunhofer IAIS, 53757 Sankt Augustin, and 01069 Dresden, Germany. E-mail: {mehdi.ali, jens.lehmann}@cs.uni-bonn.de.
- Max Berrendorf is with Ludwig-Maximilians-Universität München, 80539 Munich, Germany. E-mail: berrendorf@dbs.ifi.lmu.de.
- Charles Tapley Hoyt is with the Laboratory of Systems Pharmacology, Harvard Medical School, Boston, MA 02115 USA. E-mail: cthoyt@gmail.com.
- Laurent Vermue is with the Technical University of Denmark, 2800 Kongens Lyngby, Denmark. E-mail: lauved@dtu.dk.
- Mikhail Galkin is with the Mila & McGill University, Montreal, QC H3A 0G4, Canada. E-mail: mikhail.galkin@mila.quebec.
- Sahand Sharifzadeh is with the Ludwig-Maximilians-Universität München, 80539 Munich, Germany. E-mail: sharifzadeh@dbs.ifi.lmu.de.
- Asja Fischer is with Ruhr University Bochum, 44801 Bochum, Germany. E-mail: asja.fischer@rub.de.
- Volker Tresp is with the Ludwig-Maximilians-Universität München, 80539 Munich, Germany, and also with Siemens AG, 80333 Munich, Germany. E-mail: volker.tresp@siemens.com.

Manuscript received 20 Aug. 2021; accepted 21 Oct. 2021. Date of publication 4 Nov. 2021; date of current version 3 Nov. 2022.

This work was supported in part by the German Federal Ministry of Education and Research (BMBF) under Grants 01IS18036A and 01IS18050D under Project MLWin, in part the Innovation Fund Denmark with the Danish Center for Big Data Analytics driven Innovation (DABAI), and in part by the Defense Advanced Research Projects Agency (DARPA) Automating Scientific Knowledge Extraction (ASKE) program under Grant HR00111990009. (Corresponding author: Mehdi Ali.)

Recommended for acceptance by S. Ji.

Digital Object Identifier no. 10.1109/TPAMI.2021.3124805

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 preprocessing 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 [2], [3]. 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 hyperparameter 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 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 Sections 6 and 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. Fig. 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 $(Sarah, \text{CEO_Of}, \text{Deutsche_Bank})$, *Sarah* is the head and *Deutsche_Bank* is the tail entity. KGs usually contain only true triples corresponding to available knowledge.

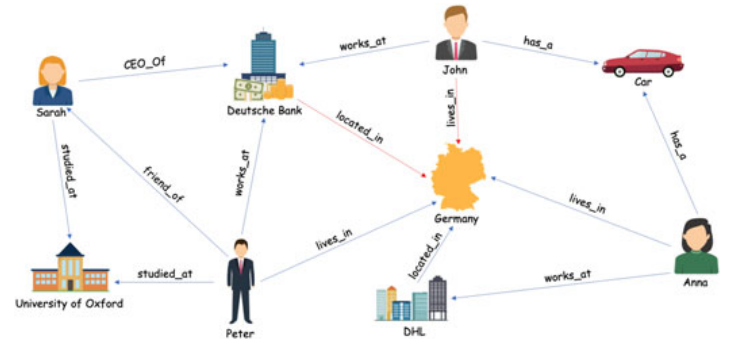


Fig. 1. Exemplary KG: Nodes represent entities and edges their respective relations.

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 Fig. 1, the triple $(Sarah, \text{lives_in}, \text{Germany})$ 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 under the principles of the semantic web (and RDF) rely on the OWA as well as most of the relevant works to this paper [11], [13].

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 $(Sarah, \text{studied_at}, ?)$ or $(?, \text{CEO_of}, \text{Deutsche_Bank})$, the task is the correctly detect the entities that answer the query, i.e. $(Sarah, \text{studied_at}, \text{University of Oxford})$ and $(Sarah, \text{CEO_of}, \text{Deutsche_Bank})$. 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]. Fig. 2 shows an embedding of the entities and relations in \mathbb{R}^2 from the KG from Fig. 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

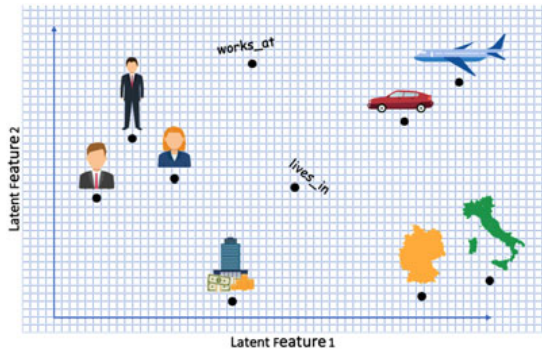


Fig. 2. An example embedding of the entities and relations from the knowledge graph portrayed by Fig. 2.

detail in their following respective Sections 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 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 $\odot: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$:

$$[\mathbf{a} \odot \mathbf{b}]_i = \mathbf{a}_i \cdot \mathbf{b}_i \quad (1)$$

Finally, we use \bar{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

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

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:

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

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:

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

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) :

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

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:

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

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 $\mathbf{M}_{r,h}, \mathbf{M}_{r,t} \in \mathbb{R}^{k \times d}$ 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:

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

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 counter-clockwise rotation by $\theta_{r,i}$ radians. The interaction model is then defined as:

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

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:

$$f(h, r, t) = -\|\mathbf{R}\mathbf{h} - \mathbf{t} + \mathbf{r}\|_2^2 + \mathbf{b}_h + \mathbf{b}_t, \quad (9)$$

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

KG2E KG2E [25] aims to explicitly model (un)certainities 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 $\mathcal{N}_i(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ 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: $\mathcal{H} - \mathcal{T} \approx \mathcal{R}$ where $\mathcal{H} \sim \mathcal{N}_h(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$, $\mathcal{T} \sim \mathcal{N}_t(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, $\mathcal{R} \sim \mathcal{P}_r = \mathcal{N}_r(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$ and $\mathcal{H} - \mathcal{T} \sim \mathcal{P}_e = \mathcal{N}_{h-t}(\boldsymbol{\mu}_h - \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_h + \boldsymbol{\Sigma}_t)$ (since head and tail entities are considered to be independent with regards to the relations). The interaction model measures the similarity between \mathcal{P}_e and \mathcal{P}_r by means of the Kullback-Leibler (KL) divergence:

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

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:

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

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:

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

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:

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

where $\text{Re}(\mathbf{x})$ 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:

$$f(h, r, t) = \mathbf{h}_r \cdot \mathbf{t}. \quad (14)$$

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, SimpleE introduces for each relation r the inverse relation r' , and formulates the interaction model based on both:

$$f(h, r, t) = \frac{1}{2} \left(\langle \mathbf{h}_{e_1}, \mathbf{r}, \mathbf{t}_{e_2} \rangle + \langle \mathbf{h}_{e_2}, \mathbf{r}', \mathbf{t}_{e_1} \rangle \right). \quad (15)$$

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:

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

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:

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

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:

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

where the circular correlation $*$: $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is defined as $[\mathbf{a} * \mathbf{b}]_i = \sum_{k=0}^{d-1} \mathbf{a}_k * \mathbf{b}_{(i+k) \bmod d}$. By using the correlation operator each component $[\mathbf{h} * \mathbf{t}]_i$ 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:

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

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:

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

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 : $x_i = \mathbf{h} \mathfrak{W}_r^i \mathbf{t}$ [35]. As indicated by the interaction model, Neural Tensor Network (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 an 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 $\omega_i \in \mathbb{R}^{1 \times 3}$, $i = 1, \dots, \tau$, are applied on the input in order to compute for each dimension global interactions of the embedded triple. Each ω_i is applied on every row of \mathbf{A} creating a feature map $\mathbf{v}_i = [v_{i,1}, \dots, v_{i,d}] \in \mathbb{R}^d$:

$$\mathbf{v}_i = g(\omega_i \mathbf{A} + \mathbf{b}), \quad (21)$$

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 $\mathbf{v}_1, \dots, \mathbf{v}_\tau$, the plausibility score of a triple is given by:

$$f(h, r, t) = [\mathbf{v}_1; \dots; \mathbf{v}_\tau] \cdot \mathbf{w}, \quad (22)$$

where $[\mathbf{v}_1; \dots; \mathbf{v}_\tau] \in \mathbb{R}^{\tau d \times 1}$ 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 $\Omega = \{\omega_i \mid \omega_i \in \mathbb{R}^{r \times c}\}$ 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:

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

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 local closed world assumption (LCWA) and the stochastic local closed world assumption (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 [30], [37]. 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 (Fig. 1) for the 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:

$$\mathcal{T}(h, r) = \{(h, r, t') \mid t' \in \mathcal{E} \wedge t' \neq t\} \quad (24)$$

$$\mathcal{H}(r, t) = \{(h', r, t) \mid h' \in \mathcal{E} \wedge h' \neq h\} \quad (25)$$

$$\mathcal{N} = \bigcup_{(h,r,t) \in \mathcal{K}} \mathcal{T}(h, r) \cup \mathcal{H}(r, t). \quad (26)$$

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 KGEMs. 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

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

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]:

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

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]:

$$L(t_i, l_i) = -(l_i \cdot \log(\sigma(f(t_i))) + (1 - l_i) \cdot \log(1 - \sigma(f(t_i)))) \quad (29)$$

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)):

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

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 $-\lambda$:

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

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

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

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

$$L(\Delta) = \max(0, \lambda + \Delta). \quad (33)$$

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

$$L(\Delta) = \log(1 + \exp(\Delta)). \quad (34)$$

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 [7], [23].

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]:

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

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].

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

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:

$$p(t|h, r) = \frac{\exp(f(h, r, t))}{\sum_{t' \in \mathcal{E}} \exp(f(h, r, t'))}. \quad (37)$$

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

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

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

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

Smaller values indicate better performance.

Adjusted Mean Rank. Because the interpretation of the mean rank (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:

$$AMR = \frac{MR}{\frac{1}{2} \sum_{t \in \mathcal{K}_{test}} (\xi(t) + 1)} \quad (40)$$

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 adjusted mean rank (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:

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

where \mathcal{K}_{test} is a set of test triples, i.e. the mean reciprocal rank (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.

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
Unified Medical Language System (UMLS)	893,025	135	49
YAGO3-10	1,079,40	132,182	37

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

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

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 ($\sim 81\%$) 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. WordNet¹ 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 $(h, \text{hyponym}, t)$, there exists a triple $(t, \text{hypernym}, o)$ 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].

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, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TPAMI.2021.3124805>.) 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

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

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, available in the online supplemental material) 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 adaptation 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 SimpleE, 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 SimpleE.

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, available in the online supplemental material) 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 SimpleE for which the results on FB15K (Table 8 in Appendix A3, available in the online supplemental material) and WN18 (Table 10 in Appendix A4, available in the online supplemental material) 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, available in the online supplemental material). 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 + Softplus loss (SPL) + inverse relations. We do *not* refer to different hyper-parameter values such as

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

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, available in the online supplemental material) shows the hyper-parameter ranges for the sLCWA and the LCWA assumptions.

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 margin ranking loss (MRL), binary cross entropy loss (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, available in the online supplemental material, 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 Kinships 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. Fig. 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. Fig. 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. Fig. 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.

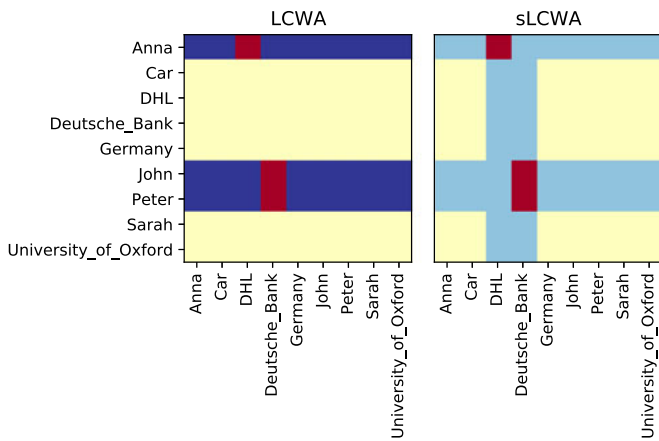


Fig. 3. Visualization of different training approaches for the relation `works_at` in the KG in Fig. 1. Red color indicates positive examples, i.e. true triples present in the KG. Dark blue color denotes triples used as negative examples in LCWA. Light blue color sampling candidates for negative examples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

Impact of Explicitly Modeling Inverse Relations. Figs. 4 and 6 present the effect of explicitly modeling inverse relations. Overall, explicitly modeling inverse relations results in less variance across the investigated configurations (Fig. 4). Further investigating the effect of modeling of inverse relations on the different loss functions and training approaches (Fig. 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. Fig. 17 (Appendix A9, available in the online supplemental material) plots the model size against the obtained performance. The results highlight

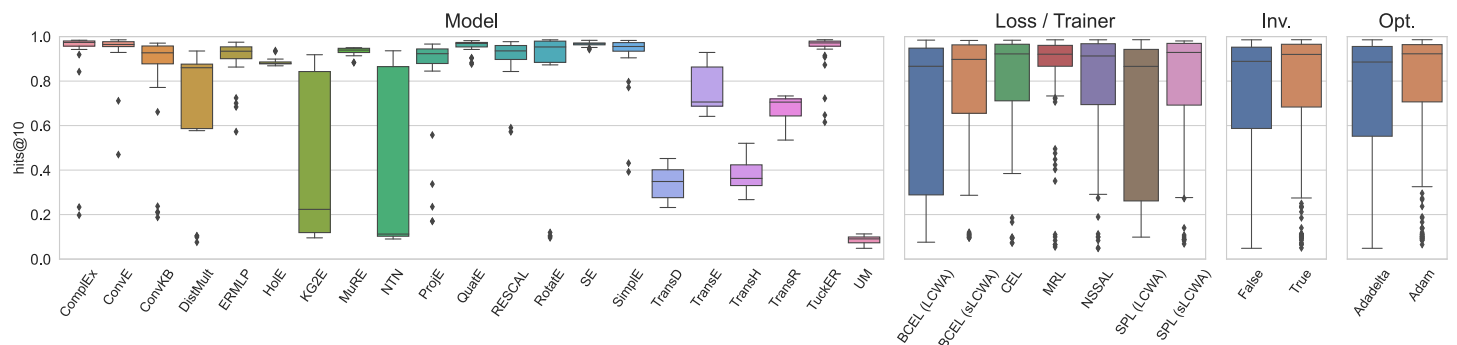


Fig. 4. Overall hits@10 results for Kinships where box-plots summarize the best results across different configurations, i.e., combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

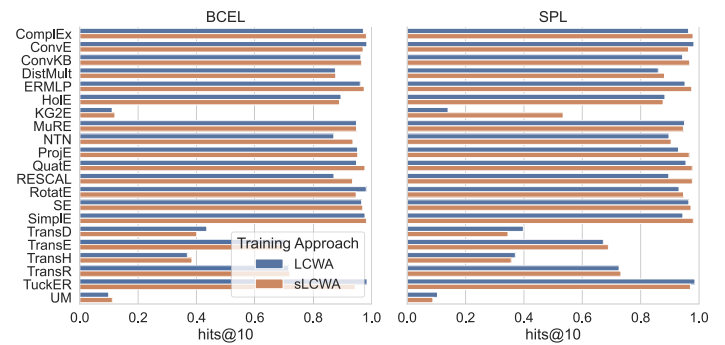


Fig. 5. Impact of training approach on the performance for a fixed interaction model and loss function for the Kinships dataset based on Adam.

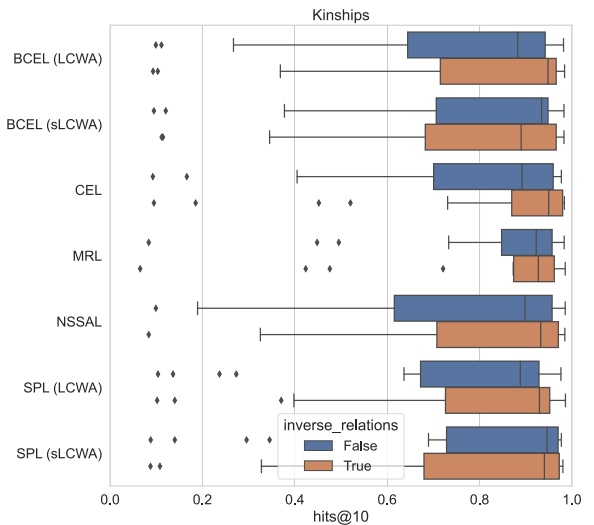


Fig. 6. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the Kinships dataset.

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 HoJE and ProjE models, and larger RotatE and TuckER models. A full list is provided in Table 14 in Appendix A6, available in the online supplemental material.

7.2 Results on the WN18RR Dataset

Fig. 7 depicts the overall results over WN18RR. A detailed overview of all configurations can be found in Fig. 20 in Appendix A12, available in the online supplemental material.

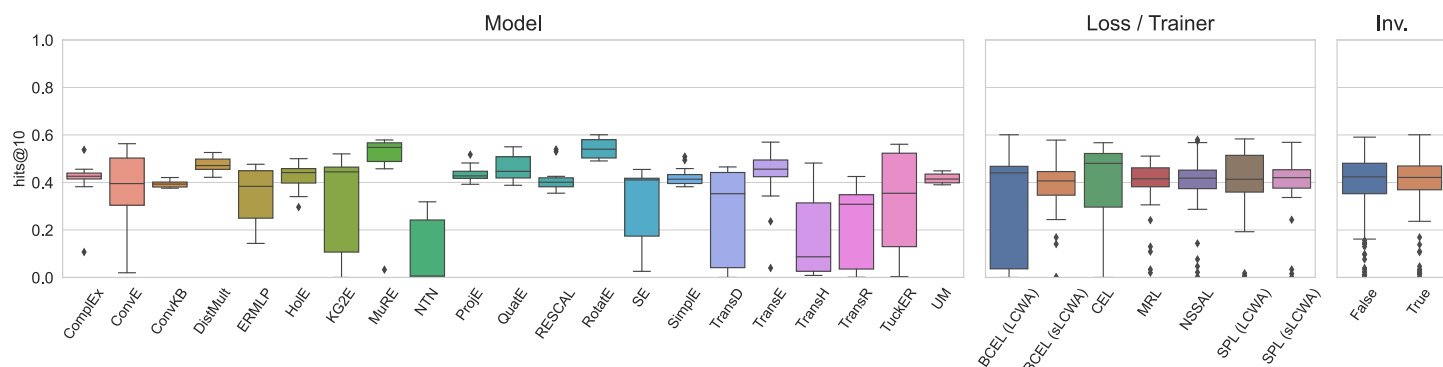


Fig. 7. Overall hits@10 results for WN18RR where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

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]), SimpleE (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. Figs. 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.

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

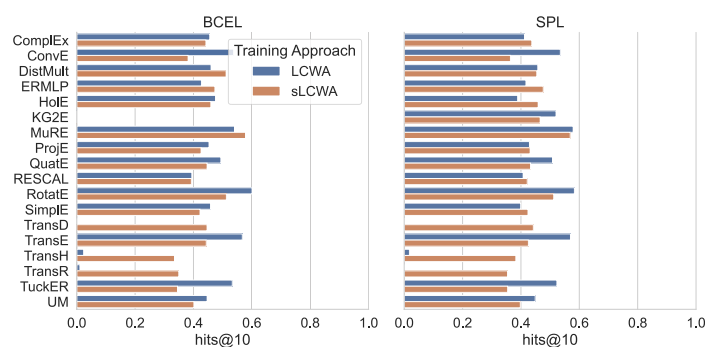


Fig. 8. Impact of training approach on the performance for a fixed interaction model and loss function for the WN18RR dataset.

when trained with the LCWA whereas TransH suffer under the LCWA.

Impact of the Loss Function. Fig. 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 (Fig. 20 in Appendix A12, available in the online supplemental material). 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).

Fig. 24 (Appendix A16, available in the online supplemental material) 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. Fig. 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.

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

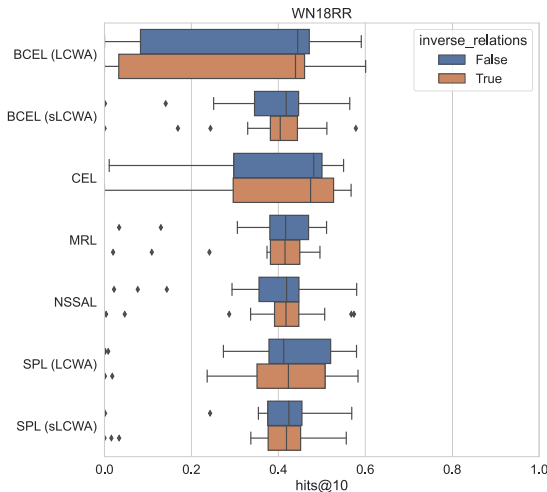


Fig. 9. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the WN18RR dataset.

Model Complexity versus Performance. Fig. 17 (Appendix A9, available in the online supplemental material) 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, available in the online supplemental material). For instance, for RotatE, several high-performing configurations have been found (Fig. 20 in the Appendix A12, available in the online supplemental material), 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.

7.3 Results on the FB15K-237 Dataset

Fig. 10 provides an overall overview of the results obtained on FB15K-237. For the results for each individual configuration, we refer to Fig. 21 in Appendix A13, available in the online supplemental material. We can observe that TuckER outperforms the other interaction models followed by RotatE. DistMult again obtains surprisingly good results (Table 21 in Appendix 13, available in the online supplemental material) 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, available in the online supplemental material) considering that the interaction model enforces symmetric relations. DistMult,

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

however, achieves a strong performance only when composed with the LCWA and the CEL (Table 17 in Appendix A17, available in the online supplemental material), highlighting that a simple interaction model can obtain strong performance when composed beneficially.

Impact of the Training Approach. Fig. 10 shows that for both, BCEL and SPL, the LCWA obtains significantly higher results, but they express a high variance at the same time. Figs. 11 and 25 (Appendix A17, available in the online supplemental material) 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, available in the online supplemental material) 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. Fig. 10 illustrates that the BCEL and SPL outperform the other loss functions, but they also exhibit higher variance. Fig. 25 (Appendix A17, available in the online supplemental material) 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. Fig. 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 versus Performance. Fig. 17 (Appendix A9, available in the online supplemental material) 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, available in the online supplemental material).

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. Fig. 13 depicts the overall results obtained for YAGO3-10. Detailed results for all configurations are illustrated in Fig. 22 in Appendix A14, available in the online supplemental material.

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

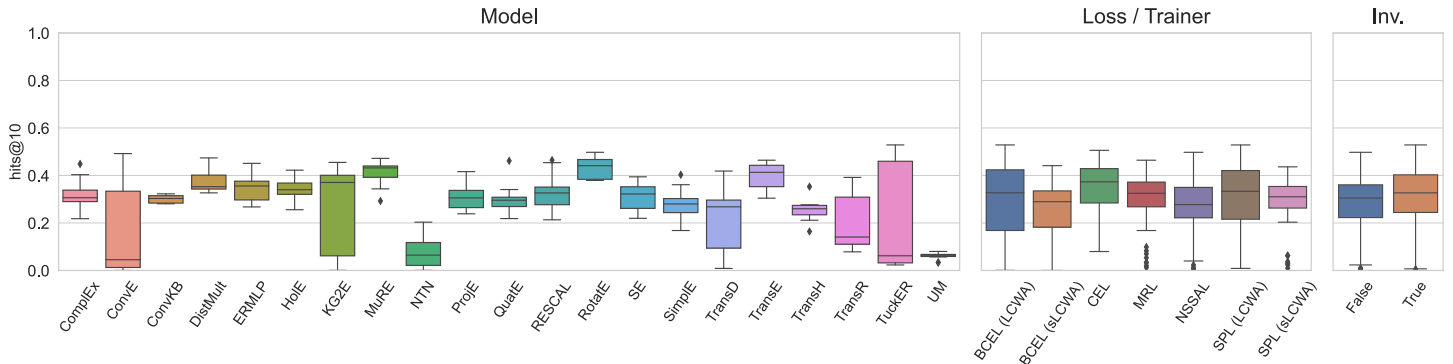


Fig. 10. Overall hits@10 results for FB15K-237 where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

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. Fig. 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 Softplus loss. Fig. 22 (Appendix A 14, available in the online supplemental material) 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. Fig. 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 versus Performance. Fig. 17 (Appendix A9, available in the online supplemental material) expresses that

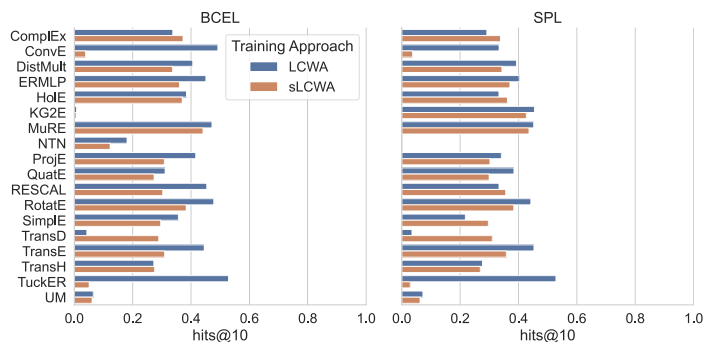


Fig. 11. Impact of training approach on the performance for a fixed interaction model and loss function for the FB15K-237 dataset.

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, available in the online supplemental material), 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, available in the online supplemental material).

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 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

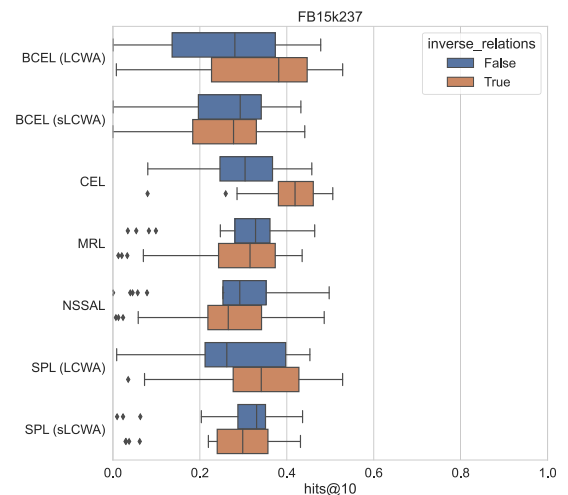


Fig. 12. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the FB15K-237 dataset.

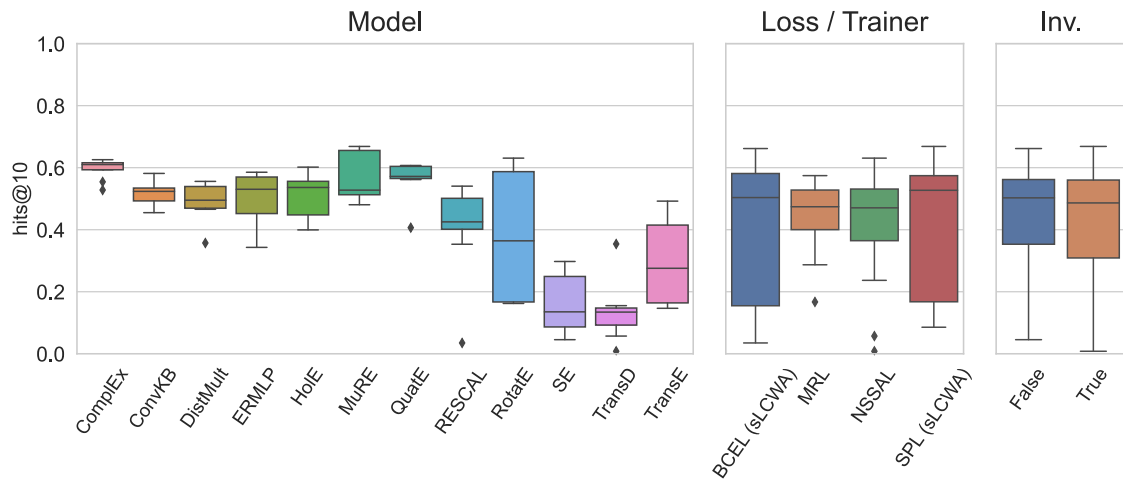


Fig. 13. Overall hits@10 results for YAGO3-10 where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations. In contrast, to the previous datasets, the models have only been trained based on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

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 $(h, r, t) \in \mathcal{T} \Rightarrow (t, r, h) \in \mathcal{T}$

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

Definition 3 (Inverse Relation). A relation $r \in \mathcal{R}$ is inverse to $r_{inv} \in \mathcal{R}$, if $(h, r, t) \in \mathcal{T} \Rightarrow (t, r_{inv}, h) \in \mathcal{T}$. 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 $(a, r_1, b) \in \mathcal{T} \wedge (b, r_2, c) \in \mathcal{T} \Rightarrow (a, r, c) \in \mathcal{T}$. 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 conditional probability of the antecedent, given the precedent holds.

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 knowledge graph embedding model (KGEMs) can model a specific relational pattern.

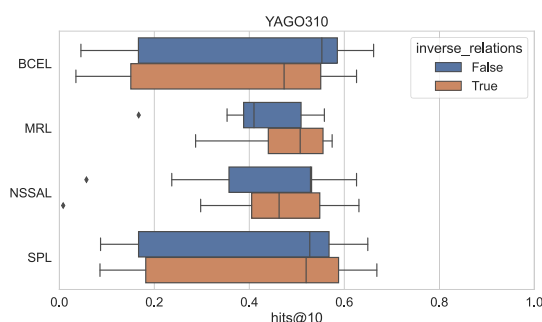


Fig. 14. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the YAGO3-10 dataset.

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

8.4 Results

Fig. 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.

Fig. 16 (Appendix A2, available in the online supplemental material) 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 SimpleE 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, available in the online supplemental material) illustrates the extent of our studies and Table 3 (Appendix 18, available in the online supplemental material) 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 KGEMs 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

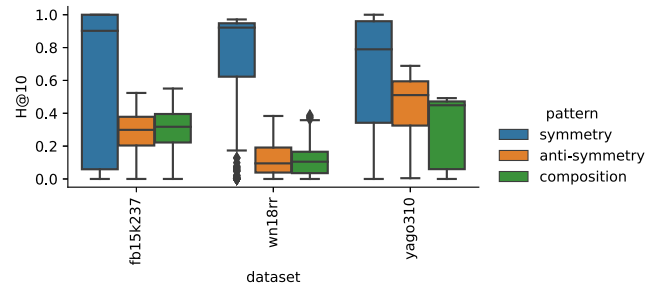


Fig. 15. Performance Distribution of all best models per configuration in H@10.

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 hyperparameter 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 KGEMs 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, available in the online supplemental material, and Figs. 19-22 in Appendix A11-A14, available in the online supplemental material). Moreover, with an appropriate configuration, the model size can significantly be compressed (see Pareto-optimal configurations in Tables 13-16 in Appendix A6, available in the online supplemental material) 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

TABLE 3
Summary of Main Insights Over All Datasets

<i>Interaction Models</i>	
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.
Structured Embedding (SE)	Among top-ten-performing interaction models on Kinships.
<i>Loss Functions</i>	
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.
<i>Training Approaches</i>	
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).
<i>Explicit Modeling of Inverse Relations</i>	
	Is usually beneficial in combination with the local closed world assumption.
<i>Configurations</i>	
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 (Fig. 13 on page 16)
Pareto-Optimal Configurations	Tables 13-16 in Appendix A6, available in the online supplemental material, 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.
<i>Reproducibility</i>	
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.
<i>General Insights</i>	
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.

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.

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 (Fig. 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 KGEMs 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 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.

ACKNOWLEDGMENTS

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. Max Berrendorf, Charles Tapley Hoyt, and Laurent Vermue contributed equally to this work.

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, Dec. 2017.
- [2] F. Akrami, L. Guo, W. Hu, and C. Li, "Re-evaluating embedding-based knowledge graph completion methods," in *Proc. Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 1779–1782.
- [3] R. Kadlec, O. Bajgar, and J. Kleindienst, "Knowledge base completion: Baselines strike back," in *Proc. Rep4NLP@ACL*, 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 *Proc. Assoc. Comput. 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 *Proc. Int. Conf. Learn. Representations*, 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 *Proc. SIGMOD Conf.*, 2020, pp. 1995–2010.
- [7] S. K. Mohamed, V. Nováček, P. Vandembussche, and E. Muñoz, "Loss functions in knowledge graph embedding models," in *Proc. DL4KG@ESWC*, 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 *Proc. Int. Conf. Learn. Representations*, 2020.
- [9] A. Rossi, D. Firmani, A. Matinata, P. Merialdo, and D. Barbosa, "Knowledge graph embedding for link prediction: A comparative analysis," 2020, *arXiv:2002.00819*.
- [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," 2020, *arXiv:2002.06914*.
- [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, Jan. 2016.
- [12] B. Kotnis and V. Nastase, "Analysis of the impact of negative sampling on link prediction in knowledge graphs," 2017, *arXiv:1708.06816*.
- [13] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, "Amie: association rule mining under incomplete evidence in ontological knowledge bases," in *Proc. 22nd Int. Conf. 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 Trans. Neural Netw. Learn. Syst.*, early access, Apr. 26, 2021, doi: 10.1109/TNNLS.2021.3070843.
- [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, Mar. 2020.
- [16] S. M. Kazemi *et al.*, "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 *Proc. AAAI*, 2011, pp. 233–259.
- [19] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Neural Inf. Process. Syst.*, 2013, pp. 2787–2795.
- [20] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. AAAI*, 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 *Proc. AAAI*, 2015, pp. 2181–2187.
- [22] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proc. Assoc. Comput. 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 *Proc. Int. Conf. Learn. Representations*, 2019.
- [24] I. Balazevic, C. Allen, and T. M. Hospedales, "Multi-relational poincaré graph embeddings," in *Proc. Neural Inf. Process. Syst.*, 2019, pp. 4465–4475.

- [25] S. He, K. Liu, G. Ji, and J. Zhao, "Learning to represent knowledge graphs with gaussian embedding," in *Proc. 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 *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 809–816.
- [27] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2071–2080.
- [28] S. Zhang, Y. Tay, L. Yao, and Q. Liu, "Quaternion knowledge graph embeddings," in *Proc. Neural Inf. Process. Syst.*, 2019, pp. 2731–2741.
- [29] S. M. Kazemi and D. Poole, "Simple embedding for link prediction in knowledge graphs," in *Proc. Neural Inf. Process. Syst.*, 2018, pp. 4289–4300.
- [30] I. Balazevic, C. Allen, and T. M. Hospedales, "Tucker: Tensor factorization for knowledge graph completion," in , 2019, pp. 5184–5193. [Online]. Available: <https://dblp.org/rec/conf/emnlp/BalazevicAH19.html?view=bibtex>
- [31] L. R. Tucker *et al.*, "The extension of factor analysis to three-dimensional matrices," *Contributions Math. Psychol.*, vol. 46, pp. 47–57, 1964.
- [32] B. Shi and T. Weninger, "Proje: Embedding projection for knowledge graph completion," in *Proc. AAAI*, 2017, pp. 1236–1242.
- [33] M. Nickel, L. Rosasco, and T. A. Poggio, "Holographic embeddings of knowledge graphs," in *Proc. AAAI*, 2016, pp. 1955–1961.
- [34] X. Dong *et al.*, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proc. Knowl. Discov. Data Mining*, 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 *Proc. Neural Inf. Process. Syst.*, 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," 2017, *arXiv:1712.02121*.
- [37] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2D knowledge graph embeddings," in *Proc. AAAI*, 2018, pp. 1811–1818.
- [38] H. Zhang, Z. Kyaw, S. Chang, and T. Chua, "Visual translation embedding network for visual relation detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3107–3115.
- [39] S. Sharifzadeh, M. Berrendorf, and V. Tresp, "Improving visual relation detection using depth maps," 2019, *arXiv:1905.00966*.
- [40] T. Lacroix, N. Usunier, and G. Obozinski, "Canonical tensor decomposition for knowledge base completion," in *Proc. Int. Conf. Mach. Learn.*, 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 *Proc. 3rd Workshop Continuous Vector Space Models Compositionality*, 2015, pp. 57–66.
- [44] W. W. Denham, "The detection of patterns in alyawara nonverbal behavior," Ph.D. dissertation, Univ. Washington, Seattle, WA, USA, 1973.
- [45] R. J. Rummel, *The Dimensionality of Nations Project: Attributes of Nations and Behavior of Nations Dyads, 1950–1965*. Ann Harbor, MI, USA: Inter-Univ. Consortium Political Res., 1976, no. 5409.
- [46] A. T. McCray, "An upper-level ontology for the biomedical domain," *Int. J. Genomic.*, 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 *Proc. Int. Semantic Web Conf.*, 2016, pp. 177–185.
- [48] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "YAGO3: A knowledge base from multilingual wikipeias," in *Proc. Conf. Innov. Data Syst. Res.*, 2016, pp. 177–185.
- [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 received the MSc degree in computer science with a focus on intelligent systems from the University of Bonn. He is currently working toward the PhD degree with Computer Science Department, University of Bonn. He is currently a research associate with Fraunhofer Institute IAIS. His research interests include 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 received the BSc and MSc degrees in computer science with a minor in mathematics from RWTH Aachen University. He is currently working toward the PhD degree with the chair of Database Systems and Data Mining, Ludwig-Maximilians-Universität München. His research interests include machine learning on graphs, in particular knowledge graphs, graph matching problems, and reproducibility in machine learning.



Charles Tapley Hoyt received the PhD degree in computational life sciences from the University of Bonn in 2019. He is currently affiliated with the Laboratory of Systems Pharmacology, Harvard Medical School, Boston, USA. His interests include the biological applications of knowledge graph embedding models towards proteochemometrics, target prioritization, drug repositioning, predictive toxicology, and precision medicine.



Laurent Vermue received the MSc degree in industrial engineering and management from the Technical University of Berlin and the MMSc degree in management science and engineering from Tongji University. He is currently working toward the PhD degree with the Section for Statistics and Data Analysis and the Section for Cognitive Systems, DTU Compute, Technical University of Denmark. His research interests include machine learning, complex network modeling, and open research software.



Mikhail Galkin received the PhD degree in computer science from the University of Bonn in 2018 studying knowledge graphs, their creation, integration, and querying. He is currently a postdoctoral fellow with the 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 received the MSc degree from the Technical University of Munich in computer vision and artificial intelligence. He is currently working toward the PhD degree with Ludwig-Maximilians-Universität München. His research interests include extracting graphs from images and text and knowledge graph modeling. He often collaborates with biologists, physicists and robotic engineers as interdisciplinary machine learning research is one of his interests.



Asja Fischer is currently a professor for machine learning with Ruhr University Bochum. She was an assistant professor with Bonn university and a postdoctoral researcher with the Montreal Institute for Learning Algorithms. Between 2010 and 2015, she was with the Institute for Neural Computation, Ruhr University Bochum and the Department of Computer Science, University of Copenhagen working on the PhD which she defended in Copenhagen in 2014. She studied Biology, Bioinformatics, Mathematics, and Cognitive Science at the Ruhr-University Bochum, the Universidade de Lisboa, and the University of Osnabrück. Her research interests include the development, analysis, and application of deep learning models and methods.



Jens Lehmann received the PhD degree with summa cum laude from the University of Leipzig with visits to the University of Oxford. He leads the Smart Data Analytics research group, University of Bonn and Fraunhofer IAIS with 40 researchers. His research interests include 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. He was the recipient of more than ten international awards for his research work. He is currently the founder, a leader or a contributor of several community research projects, including SANSa, DL-Learner, DBpedia and LinkedGeoData. He studied computer science with the Technical University of Dresden.



Volker Tresp received the diploma degree from the University of Goettingen, Germany, in 1984, and the MSc and PhD 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 with Siemens, Research and Technology, Munich, Germany. He filed more than 70 patent applications and was an inventor of the year of Siemens in 1996. He has authored or coauthored more than 100 scientific articles and administered more than

20 Ph.D. dissertations. The company Panoratio is a spin-off out of his team. His research interests include machine learning in information networks for modeling knowledge graphs, medical decision processes, and sensor networks. He is currently the coordinator of one of the first nationally funded big data projects for the realization of precision medicine. In 2011, he became an honorary professor with the Ludwig Maximilian University of Munich, Germany, where he teaches an annual course on machine learning.

▷ **For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.**

TABLE 4

Investigated interaction models [33] and their required number of parameters. k corresponds to the number of neurons in the hidden layer, n_f to the number of convolutional kernels, k_r and k_c to the height and width of the convolutional kernels.

Model	Parameters
ComplEx ^a	$ \mathcal{E} 2d + \mathcal{R} 2d$
ConvE ^b	$ \mathcal{E} d + \mathcal{R} d + d + n_f k_r k_c + 2 + 2n_f + 2d$ $+ (h - k_r + 1)(w - k_c + 1)n_f d + \mathcal{E} $
ConvKB	$ \mathcal{E} d + \mathcal{R} d + n_f(d + 4) + 1$
DistMult	$ \mathcal{E} d + \mathcal{R} d$
ER-MLP	$ \mathcal{E} d + \mathcal{R} d + k(3d + 2) + 1$
HolE	$ \mathcal{E} d + \mathcal{R} d$
KG2E	$ \mathcal{E} 2d + 2 \mathcal{R} d$
MuRE	$ \mathcal{E} (d + 2) + 3 \mathcal{R} d$
NTN	$ \mathcal{E} d + \mathcal{R} k(d^2 + 2d + 2)$
ProjE	$ \mathcal{E} d + \mathcal{R} d + 3d + 1$
QuatE ^c	$ \mathcal{E} 4d + \mathcal{R} 4d$
RESCAL	$ \mathcal{E} d + \mathcal{R} d^2$
RotatE ^a	$ \mathcal{E} 2d + \mathcal{R} d$
SE	$ \mathcal{E} d + 2 \mathcal{R} d^2$
Simple	$ \mathcal{E} 2d + 2 \mathcal{R} d$
TransE	$ \mathcal{E} d + \mathcal{R} d$
TransH	$ \mathcal{E} d + 2 \mathcal{R} d$
TransR	$ \mathcal{E} d_e + \mathcal{R} d_r + d_e d_r$
UM	$ \mathcal{E} d$
Tucker	$ \mathcal{E} d_e + \mathcal{R} d_r + d_e^2 d_r + 4d_e$

^a $2d$, because of complex valued vectors, i.e. imaginary and real part of a number.

^b w and h correspond to the height and weight of the reshaped input.

^c $4d$, because of hyper-complex valued (quaternion) vectors, i.e. a real part and three imaginary parts of a quaternion.

TABLE 5

Denotes for each proposed model whether results have been reported for FB15K, WN18, or their alterations. Furthermore, it indicates whether an official implementation exists where P corresponds to a PyTorch based implementation, T to a TensorFlow based implementation, and O to other implementations. A green background indicates that the full experimental setup was available. The models highlighted with * where included in the reproducibility study.

Model	Code	FB15K	FB15K-237	WN18	WN18RR
ComplEx*	O	✓		✓	
ConvE*	P	✓	✓	✓	✓
ConvKB*	T		✓		✓
DistMult*	-	✓		✓	
ER-MLP	-				
HolE*	O	✓		✓	
KG2E*	-	✓		✓	
MuRE*	P		✓		✓
NTN	-				
ProjE	T	✓		✓	
QuatE ^a *	P	✓	✓	✓	✓
UM	-				
RESCAL	O				
RotatE*	P	✓	✓	✓	✓
SE	O				
Simple*	T, P	✓		✓	
TransD*	-	✓		✓	
TransE*	O	✓		✓	
TransH*	-	✓		✓	
TransR*	O	✓		✓	
Tucker*	P	✓	✓	✓	✓
UM	-				

^a Code is based on the framework OpenKE <https://github.com/thunlp/OpenKE>

TABLE 6
Hyper-Parameter Ranges for Ablation Experiments

	Hyper-Parameter	Range
Shared	Embedding-Dimension	{64,128,256}
	Initialization	{Xavier}
	Optimizers ^a	{Adam, Adadelta}
	Learning Rate (log scale)	[0.001, 0.1]
	Batch Size ^b	{128, 256, 512}
	Model inverse relations	{Yes, No}
	Epochs	1,000
sLCWA	Loss	{BCEL, MRL, NSSAL, SPL}
	Margin for MRL	{0.5, 1.5, ..., 9.5}
	Margin for NSSAL	{1, 3, 5, ..., 29}
	ADV T for NSSAL	{0.1, 0.2, ..., 1.0}
	Number of Negatives ^c	{1, 2, ..., 100}
LCWA	Loss	{BCEL, CEL, SPL}
	Label Smoothing (log scale)	[0.001, 1.0]

^a For Kinships, we evaluated Adam and Adadelta, and for the remaining datasets we stucked to Adam since it performed almost in every experiment at least equally good as Adadelta and in many experiments significantly better.

^b For YAGO3-10, the batch-size has been sampled from the set {1024, 2048, 2096, 8192}.

^c For YAGO3-10, the number of negative triples per each each positive has been sampled from the set {1, 2, ..., 50}.

TABLE 7
Evaluation statistics

Metric	Value
Datasets	4
Interaction Models	21
Training approaches	2
Loss Functions	5
Negative Samplers	1
Optimizers	2
Ablation Studies	1,207
Number of Experiments	73,683
Compute Time (hours)	24,804

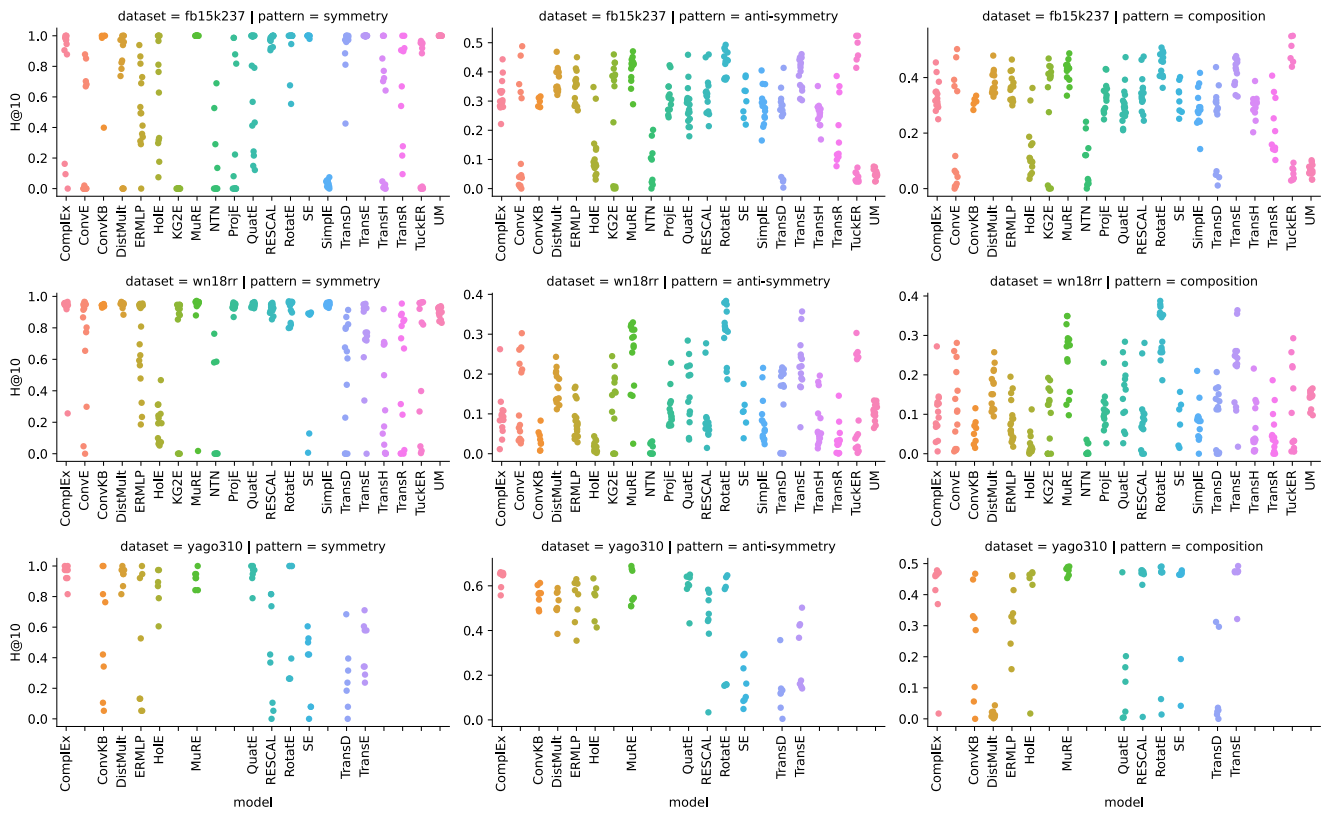


Fig. 16. Performance of best models' for each configuration for each dataset and pattern type, grouped by interaction function.

ADDITIONAL RESULTS FROM REPRODUCIBILITY STUDY

TABLE 8

Reproduction of Studies on FB15K where **pub** refers to published results, **R** to results based on the realistic ranking, **O** to results based on the optimistic ranking, and **P** to results based on the pessimistic ranking. For published results, there are two additional rank types, **U** for undefined due to missing official implementation and **ND** for non-deterministic. We only show the results of the optimistic and pessimistic ranking in case they differ from the realistic ranking.

model		MRR (%)	Hits@1 (%)	Hits@3 (%)	Hits@5 (%)	Hits@10 (%)	MR	AMR (%)
CompLex	pub (O)	69.20	59.90	75.90		84.00		
	R	21.94 ± 0.71	12.73 ± 0.74	24.18 ± 0.65	30.67 ± 0.60	40.61 ± 0.74	170.56 ± 17.18	2.31 ± 0.23
ConvE	pub (ND)	65.70	55.80	72.30		83.10	51.00	
	R	75.45 ± 0.17	68.26 ± 0.27	80.47 ± 0.08	83.94 ± 0.01	87.68 ± 0.04	43.97 ± 0.60	0.60 ± 0.01
DistMult	pub (U)	35.00				57.70		
	R	28.47 ± 0.23	18.59 ± 0.19	31.77 ± 0.29	38.24 ± 0.38	47.81 ± 0.36	127.16 ± 0.85	1.72 ± 0.01
HolE	pub (ND)	52.40	40.20	61.30		73.90		
	R	39.72 ± 0.32	27.15 ± 0.33	46.13 ± 0.40	54.05 ± 0.36	64.02 ± 0.27	186.22 ± 6.21	2.52 ± 0.08
KG2E	pub (U)					71.50	59.00	
	R	0.63 ± 0.08	0.15 ± 0.04	0.41 ± 0.11	0.66 ± 0.17	1.25 ± 0.21	5784.42 ± 22.26	78.31 ± 0.30
QuatE¹	pub (O)	77.00	70.00	82.10		87.80	41.00	
	R	22.19 ± 0.17	14.65 ± 0.16	23.76 ± 0.26	29.37 ± 0.40	37.42 ± 0.39	229.99 ± 1.57	3.11 ± 0.02
RotatE	pub (ND)	79.70	74.60	83.00		88.40	40.00	
	R	64.94 ± 0.03	53.05 ± 0.05	73.31 ± 0.06	78.74 ± 0.06	84.85 ± 0.03	35.66 ± 0.06	0.48 ± 0.00
Simple	pub (O)	72.70	66.00	77.30		83.80		
	R	0.04 ± 0.00	0.01 ± 0.00	0.03 ± 0.01	0.03 ± 0.01	0.05 ± 0.00	7386.02 ± 2.11	99.99 ± 0.03
	O	23.62 ± 12.90	11.67 ± 8.68	24.65 ± 16.33	34.28 ± 20.19	51.91 ± 24.57	148.27 ± 89.28	
	P	0.03 ± 0.00	0.01 ± 0.00	0.03 ± 0.01	0.03 ± 0.01	0.05 ± 0.00	14623.77 ± 91.95	
TransD	pub (U)					77.30	91.00	
	R	37.30 ± 0.05	24.45 ± 0.08	44.22 ± 0.09	51.78 ± 0.09	61.31 ± 0.07	146.55 ± 3.10	1.98 ± 0.04
TransE	pub (U)					47.10	125.00	
	R	29.11 ± 0.20	17.99 ± 0.27	33.53 ± 0.18	40.76 ± 0.21	50.84 ± 0.28	122.01 ± 1.09	1.65 ± 0.01
TransH	pub (U)					64.40	87.00	
	R	2.59 ± 0.27	1.89 ± 0.35	2.87 ± 0.23	3.16 ± 0.11	3.46 ± 0.15	6318.90 ± 18.86	85.54 ± 0.26
TransR	pub (ND)					68.70	77.00	
	R	1.23 ± 0.04	0.38 ± 0.00	1.34 ± 0.10	1.93 ± 0.12	2.79 ± 0.09	6130.41 ± 9.59	82.99 ± 0.13
TuckER	pub (ND)	79.50	74.10	83.30		89.20		
	R	79.02 ± 0.12	73.10 ± 0.11	83.05 ± 0.13	85.93 ± 0.16	89.10 ± 0.10	40.35 ± 0.83	0.55 ± 0.01

TABLE 9

Reproduction of Studies on FB15K-237 where **pub** refers to published results, **R** to results based on the realistic ranking, **O** to results based on the optimistic ranking, and **P** to results based on the pessimistic ranking. For published results, there are two additional rank types, **U** for undefined due to missing official implementation and **ND** for non-deterministic. We only show the results of the optimistic and pessimistic ranking in case they differ from the realistic ranking.

model		MRR (%)	Hits@1 (%)	Hits@3 (%)	Hits@5 (%)	Hits@10 (%)	MR	AMR (%)
ConvE	pub (ND)	32.50	23.70	35.60		50.10	244.00	
	R	29.69 ± 0.19	21.13 ± 0.21	32.32 ± 0.19	38.57 ± 0.12	47.19 ± 0.08	245.83 ± 4.97	3.45 ± 0.07
ConvKB	pub (O)	39.60				51.70	257.00	
	R	4.22 ± 0.18	2.75 ± 0.27	3.65 ± 0.19	4.44 ± 0.19	7.18 ± 0.71	4314.45 ± 27.24	60.46 ± 0.38
MuRE	pub (R)	33.60	24.50	37.00		52.10		
	R	25.16 ± 0.20	16.12 ± 0.30	27.67 ± 0.21	34.21 ± 0.32	43.78 ± 0.13	190.61 ± 0.58	2.67 ± 0.01
QuatE¹	pub (O)	31.10	22.10	34.20		49.50	176.00	
	R	0.26 ± 0.02	0.18 ± 0.03	0.23 ± 0.02	0.25 ± 0.02	0.30 ± 0.01	7119.76 ± 36.06	99.78 ± 0.51
RotatE	pub (ND)	33.80	24.10	37.50		53.30	177.00	
	R	28.79 ± 0.07	19.74 ± 0.08	31.67 ± 0.05	37.89 ± 0.07	47.13 ± 0.07	176.70 ± 0.48	2.48 ± 0.01
TuckER	pub (ND)	35.80	26.60	39.40		54.40		
	R	35.51 ± 0.08	26.20 ± 0.15	39.05 ± 0.10	45.59 ± 0.12	54.11 ± 0.04	152.46 ± 2.32	2.14 ± 0.03

TABLE 10

Reproduction of Studies on WN18 where **pub** refers to published results, **R** to results based on the realistic ranking, **O** to results based on the optimistic ranking, and **P** to results based on the pessimistic ranking. For published results, there are two additional rank types, **U** for undefined due to missing official implementation and **ND** for non-deterministic. We only show the results of the optimistic and pessimistic ranking in case they differ from the realistic ranking.

model		MRR (%)	Hits@1 (%)	Hits@3 (%)	Hits@5 (%)	Hits@10 (%)	MR	AMR (%)
ComplEx	pub (O)	94.10	93.60	94.50		94.70		
	R	18.28 ± 2.10	11.65 ± 1.32	19.04 ± 2.44	23.38 ± 3.06	30.70 ± 3.90	442.51 ± 47.32	2.16 ± 0.23
ConvE	pub (ND)	94.30	93.50	94.60		95.60	374.00	
	R	94.23 ± 0.08	93.54 ± 0.16	94.68 ± 0.04	95.03 ± 0.02	95.39 ± 0.09	462.53 ± 32.15	2.26 ± 0.16
DistMult	pub (U)	83.00				94.20		
	R	82.41 ± 0.24	74.74 ± 0.31	89.09 ± 0.19	91.36 ± 0.22	93.44 ± 0.15	454.41 ± 43.08	2.22 ± 0.21
HolE	pub (ND)	93.80	93.00	94.50		94.90		
	R	73.43 ± 0.40	63.22 ± 0.57	81.80 ± 0.40	85.81 ± 0.20	89.30 ± 0.26	786.05 ± 33.16	3.84 ± 0.16
KG2E	pub (U)					92.80	331.00	
	R	3.73 ± 0.22	1.46 ± 0.19	3.27 ± 0.26	4.77 ± 0.32	7.39 ± 0.33	2732.49 ± 57.69	13.35 ± 0.28
	O	3.74 ± 0.22	1.46 ± 0.19	3.27 ± 0.26	4.77 ± 0.32	7.39 ± 0.33	2732.49 ± 57.69	
QuatE¹	pub (O)	94.90	94.10	95.40		96.00	388.00	
	R	67.28 ± 0.70	58.38 ± 0.88	73.05 ± 0.61	77.86 ± 0.53	83.25 ± 0.38	327.12 ± 12.44	1.60 ± 0.06
RotatE	pub (ND)	94.90	94.40	95.20		95.90	309.00	
	R	93.71 ± 0.03	92.27 ± 0.03	94.87 ± 0.06	95.34 ± 0.04	95.83 ± 0.05	270.22 ± 7.24	1.32 ± 0.04
Simple	pub (O)	94.20	93.90	94.40		94.70		
	R	0.04 ± 0.02	0.01 ± 0.01	0.03 ± 0.02	0.04 ± 0.03	0.06 ± 0.03	20355.98 ± 19.42	99.48 ± 0.09
	O	32.95 ± 8.10	28.19 ± 6.94	33.94 ± 8.84	37.28 ± 9.69	42.40 ± 10.53	469.49 ± 161.36	
	P	0.03 ± 0.01	0.01 ± 0.01	0.03 ± 0.02	0.04 ± 0.03	0.06 ± 0.03	40242.47 ± 195.66	
TransD	pub (U)					92.20	212.00	
	R	37.33 ± 0.52	4.31 ± 0.42	67.90 ± 0.93	81.01 ± 0.30	87.80 ± 0.33	460.00 ± 7.40	2.25 ± 0.04
TransE	pub (U)					89.20	251.00	
	R	37.04 ± 1.37	9.29 ± 1.83	60.28 ± 1.25	72.02 ± 0.75	81.51 ± 0.52	489.84 ± 42.13	2.39 ± 0.21
TransH	pub (U)					82.30	388.00	
	R	0.17 ± 0.17	0.08 ± 0.12	0.17 ± 0.20	0.21 ± 0.24	0.31 ± 0.29	19551.68 ± 166.54	95.55 ± 0.81
TransR	pub (ND)					92.00	225.00	
	R	0.24 ± 0.03	0.00 ± 0.01	0.22 ± 0.05	0.38 ± 0.05	0.63 ± 0.07	18882.20 ± 240.51	92.27 ± 1.18
TuckER	pub (ND)	95.30	94.90	95.50		95.80		
	R	94.89 ± 0.05	94.52 ± 0.05	95.17 ± 0.07	95.30 ± 0.07	95.50 ± 0.06	532.05 ± 45.91	2.60 ± 0.22

TABLE 11

Reproduction of Studies on WN18RR where **pub** refers to published results, **R** to results based on the realistic ranking, **O** to results based on the optimistic ranking, and **P** to results based on the pessimistic ranking. For published results, there are two additional rank types, **U** for undefined due to missing official implementation and **ND** for non-deterministic. We only show the results of the optimistic and pessimistic ranking in case they differ from the realistic ranking.

model		MRR (%)	Hits@1 (%)	Hits@3 (%)	Hits@5 (%)	Hits@10 (%)	MR	AMR (%)
ConvE	pub (ND)	43.00	40.00	44.00		52.00	4187.00	
	R	45.28 ± 0.13	41.93 ± 0.19	46.64 ± 0.25	49.07 ± 0.22	51.98 ± 0.24	5203.77 ± 129.07	25.67 ± 0.64
ConvKB	pub (O)	24.80				52.50	2554.00	
	R	0.34 ± 0.05	0.11 ± 0.04	0.27 ± 0.02	0.43 ± 0.06	0.63 ± 0.08	13905.99 ± 962.71	68.60 ± 4.75
QuatE¹	pub (O)	48.10	43.60	50.00		56.40	3472.00	
	R	0.58 ± 0.05	0.38 ± 0.06	0.56 ± 0.08	0.66 ± 0.06	0.88 ± 0.09	20404.47 ± 196.81	100.65 ± 0.97
RotatE	pub (ND)	47.60	42.80	49.20		57.10	3340.00	
	R	49.39 ± 0.06	45.49 ± 0.12	51.03 ± 0.10	53.36 ± 0.15	57.05 ± 0.14	4046.79 ± 89.15	19.96 ± 0.44
TuckER	pub (ND)	47.00	44.30	48.20		52.60		
	R	47.62 ± 0.58	44.91 ± 0.62	48.81 ± 0.59	50.40 ± 0.58	52.80 ± 0.45	5646.84 ± 146.30	27.85 ± 0.72

^a For MuRE, we obtained non-finite loss values while training on WN18RR with the setting defined in [24]. This might be explained by the fact that the specified learning rate of 50 is comparably large. In our benchmarking study, we show that we can outperform the published results with a different setting (Section 7.2).

TABLE 12
Model sizes in bytes for the best reported configurations studied for the the reproducibility study.

Dataset Model	FB15K	FB15K-237	WN18	WN18RR
ComplEx	26.1 MB	-	49.2 MB	-
ConvE	22.5 MB	20.3 MB	41.2 MB	40.9 MB
ConvKB	-	5.9 MB	-	8.2 MB
DistMult	6.5 MB	-	16.4 MB	-
HolE	9.8 MB	-	24.6 MB	-
KG2E	6.5 MB	-	16.4 MB	-
RotatE	130.4 MB	117.9 MB	163.8 MB	162.3 MB
Simple	26.1 MB	-	65.5 MB	-
TransD	6.5 MB	-	16.4 MB	-
TransE	3.3 MB	-	3.3 MB	-
TransH	7.1 MB	-	8.2 MB	-
TransR	16.7 MB	-	8.4 MB	-
TuckER	46.1 MB	-	37.6 MB	-

ADDITIONAL RESULTS FROM BENCHMARKING STUDY

TABLE 13
Pareto-optimal models for FB15k237 regarding Model Bytes and Hits@10

Model	Loss	Training Approach	Inverse Relations	Model Bytes	Hits@10 (%)
TuckER	BCEL	LCWA	yes	8.0 MiB	52.857
DistMult	CEL	LCWA	yes	3.7 MiB	47.387
TransE	SPL	LCWA	no	3.6 MiB	45.318
UM	MRL	sLCWA	no	3.5 MiB	3.432
UM	MRL	sLCWA	yes	3.5 MiB	3.305

TABLE 14
Pareto-optimal models for Kinships regarding Model Bytes and Hits@10

Model	Loss	Training Approach	Inverse Relations	Model Bytes	Hits@10 (%)
TuckER	SPL	LCWA	yes	1.0 MiB	98.603
RotatE	MRL	sLCWA	yes	154.0 KiB	98.557
RotatE	MRL	sLCWA	no	129.0 KiB	98.324
SimpleE	BCEL	LCWA	yes	77.0 KiB	97.765
ProjE	SPL	sLCWA	yes	39.3 KiB	96.648
ProjE	SPL	sLCWA	no	33.0 KiB	94.600
HolE	CEL	LCWA	no	32.2 KiB	88.873
UM	SPL	LCWA	yes	26.0 KiB	11.313
UM	SPL	sLCWA	yes	26.0 KiB	6.844

TABLE 15
Pareto-optimal models for WN18RR regarding Model Bytes and Hits@10

Model	Loss	Training Approach	Inverse Relations	Model Bytes	Hits@10 (%)
RotatE	BCEL	LCWA	yes	79.3 MiB	60.089
RotatE	SPL	LCWA	yes	19.8 MiB	58.328
TuckER	CEL	LCWA	yes	11.9 MiB	56.088
MuRE	SPL	LCWA	no	10.2 MiB	55.489
TransH	MRL	sLCWA	no	9.9 MiB	48.170
UM	SPL	LCWA	yes	9.9 MiB	44.682
UM	SPL	sLCWA	yes	9.9 MiB	39.022

TABLE 16
Pareto-optimal models for YAGO310 regarding Model Bytes and Hits@10

Model	Loss	Training Approach	Inverse Relations	Model Bytes	Hits@10 (%)
MuRE	SPL	sLCWA	yes	61.1 MiB	66.851
ConvKB	NSSAL	sLCWA	no	30.1 MiB	52.921
DistMult	SPL	sLCWA	yes	30.1 MiB	50.562
TransE	BCEL	sLCWA	no	30.1 MiB	14.663

TABLE 17
Best configuration for each model in FB15k237

Model	Loss	Training Approach	Inverse Relations	Hits@10 (%)
ComplEx	CEL	LCWA	True	44.838
ConvE	BCEL	LCWA	True	49.212
ConvKB	SPL	sLCWA	False	32.261
DistMult	CEL	LCWA	True	47.387
ERMLP	BCEL	LCWA	True	45.100
HolE	CEL	LCWA	True	42.225
KG2E	SPL	LCWA	True	45.501
MuRE	BCEL	LCWA	True	47.199
NTN	SPL	sLCWA	False	20.342
ProjE	BCEL	LCWA	True	41.616
QuatE	CEL	LCWA	True	46.166
RESCAL	CEL	LCWA	True	46.460
RotatE	NSSAL	sLCWA	False	49.750
SE	NSSAL	sLCWA	True	39.427
Simple	CEL	LCWA	True	40.307
TransD	MRL	sLCWA	True	41.856
TransE	MRL	sLCWA	False	46.423
TransH	MRL	sLCWA	False	35.295
TransR	CEL	LCWA	True	39.187
TuckER	BCEL	LCWA	True	52.857
UM	CEL	LCWA	False	8.024

TABLE 18
Best configuration for each model in Kinships

Model	Loss	Training Approach	Inverse Relations	Hits@10 (%)
ComplEx	CEL	LCWA	True	98.371
ConvE	NSSAL	sLCWA	True	98.557
ConvKB	NSSAL	sLCWA	True	97.067
DistMult	CEL	LCWA	True	93.529
ERMLP	SPL	sLCWA	True	97.486
HolE	CEL	LCWA	True	93.715
KG2E	MRL	sLCWA	True	91.853
MuRE	SPL	LCWA	True	95.019
NTN	BCEL	sLCWA	True	93.622
ProjE	SPL	sLCWA	True	96.648
QuatE	CEL	LCWA	True	98.184
RESCAL	SPL	sLCWA	True	97.719
RotatE	NSSAL	sLCWA	False	98.557
SE	NSSAL	sLCWA	True	98.324
Simple	BCEL	sLCWA	False	98.277
TransD	CEL	LCWA	True	45.205
TransE	CEL	LCWA	True	92.877
TransH	CEL	LCWA	True	52.048
TransR	MRL	sLCWA	False	73.324
TuckER	SPL	LCWA	True	98.603
UM	SPL	LCWA	True	11.313

TABLE 19
Best configuration for each model in WN18RR

Model	Loss	Training Approach	Inverse Relations	Hits@10 (%)
ComplEx	CEL	LCWA	False	53.745
ConvE	CEL	LCWA	True	56.327
ConvKB	NSSAL	sLCWA	True	42.083
DistMult	CEL	LCWA	True	52.616
ERMLP	SPL	sLCWA	True	47.657
HolE	CEL	LCWA	False	50.017
KG2E	SPL	LCWA	False	52.035
MuRE	SPL	LCWA	True	57.900
NTN	MRL	sLCWA	False	31.857
ProjE	CEL	LCWA	True	51.727
QuatE	CEL	LCWA	False	55.010
RESCAL	CEL	LCWA	False	53.916
RotatE	BCEL	LCWA	True	60.089
SE	SPL	sLCWA	False	45.486
SimpleE	CEL	LCWA	True	50.889
TransD	MRL	sLCWA	False	46.546
TransE	SPL	LCWA	False	56.977
TransH	MRL	sLCWA	False	48.170
TransR	MRL	sLCWA	False	42.510
TuckER	CEL	LCWA	True	56.088
UM	SPL	LCWA	False	44.887

TABLE 20
Best configuration for each model in YAGO310

Model	Loss	Training Approach	Inverse Relations	Hits@10 (%)
ComplEx	BCEL	sLCWA	True	62.575
ConvKB	SPL	sLCWA	True	58.149
DistMult	BCEL	sLCWA	False	55.580
ERMLP	BCEL	sLCWA	True	58.531
HolE	BCEL	sLCWA	False	60.177
MuRE	SPL	sLCWA	True	66.851
QuatE	SPL	sLCWA	True	60.709
RESCAL	SPL	sLCWA	True	54.045
RotatE	NSSAL	sLCWA	True	63.077
SE	NSSAL	sLCWA	True	29.757
TransD	MRL	sLCWA	False	35.397
TransE	MRL	sLCWA	True	49.217

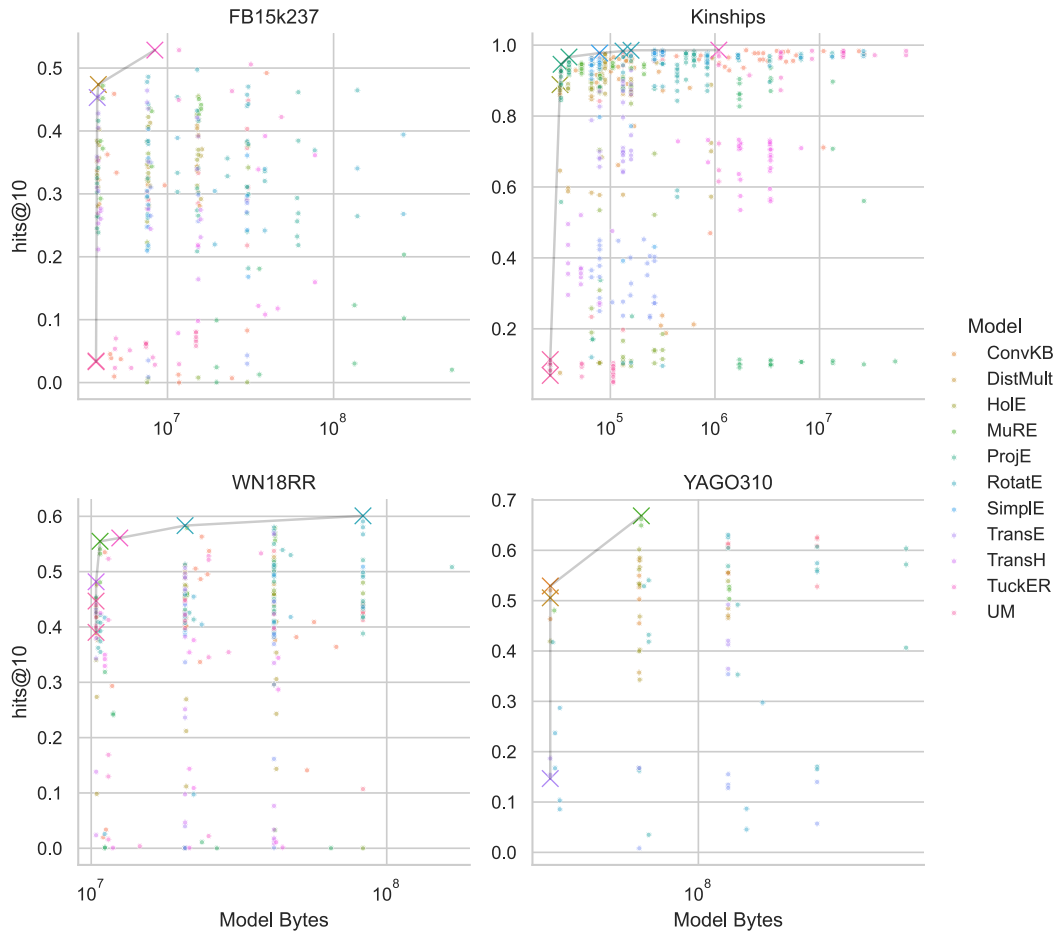


Fig. 17. Scatter plots comparing model size in number of bytes and model performance in terms of Hits@10 for all trained models on each dataset. The color indicates the model type, and the model size is shown on a logarithmic axis. Pareto-optimal models are highlighted by cross symbols. In general we only see a low correlation between model size and performance. A more thorough comparison can be found in Figures [4](#), [7](#), [10](#) and [13](#)

training_approach = LCWA				training_approach = sLCWA					
	BCEL	CEL	SPL	BCEL	MRL	NSSAL	SPL		
inverse_relations = False	ComplEx	23.37%	97.21%	19.74%	ComplEx	98.14%	97.49%	98.18%	97.86%
	ConvE	93.62%	95.34%	93.90%	ConvE	92.88%	95.95%	96.42%	95.95%
	ConvKB	20.90%	66.15%	18.76%	ConvKB	77.14%	92.55%	89.43%	93.62%
	DistMult	7.59%	84.73%	9.87%	DistMult	64.62%	85.94%	58.38%	57.82%
	ERMLP	72.44%	57.26%	68.39%	ERMLP	88.83%	91.06%	91.48%	90.97%
	HoIE	87.76%	89.90%	88.18%	HoIE	87.90%	87.85%	87.80%	87.57%
	KG2E	10.01%	69.41%	25.74%	KG2E	10.29%	91.81%	91.06%	52.14%
	MuRE	91.62%	88.41%	91.39%	MuRE	93.58%	93.39%	94.83%	94.93%
	NTN	10.61%	9.87%	10.01%	NTN	10.66%	9.96%	11.31%	9.03%
	ProjE	16.99%	84.50%	17.04%	ProjE	55.77%	89.29%	91.11%	85.15%
	QuatE	87.66%	96.28%	90.27%	QuatE	97.67%	96.88%	97.30%	97.25%
	RESICAL	59.03%	91.39%	89.76%	RESICAL	91.15%	87.10%	93.20%	94.46%
	RotatE	11.96%	96.65%	10.29%	RotatE	88.45%	97.53%	95.95%	87.29%
	SE	95.22%	95.53%	95.67%	SE	97.37%	97.58%	97.21%	97.00%
	SimplE	39.20%	94.32%	43.11%	SimplE	92.60%	90.46%	95.81%	94.32%
	TransD	27.23%	38.45%	23.65%	TransD	28.68%	35.10%	29.10%	27.23%
	TransE	70.95%	83.85%	64.15%	TransE	69.79%	87.01%	88.92%	69.97%
	TransR	57.45%	62.06%	53.49%	TransR	71.97%	72.44%	65.36%	70.90%
	TuckER	97.16%	97.16%	96.65%	TuckER	96.97%	96.14%	87.29%	97.21%
	UM	8.89%	7.22%	10.01%	UM	10.34%	5.45%	4.84%	8.19%
inverse_relations = True	ComplEx	84.22%	98.04%	91.90%	ComplEx	98.09%	96.93%	98.32%	97.72%
	ConvE	98.09%	98.32%	97.72%	ConvE	97.49%	97.72%	98.56%	97.72%
	ConvKB	23.74%	87.20%	21.23%	ConvKB	92.50%	96.09%	90.41%	87.94%
	DistMult	10.38%	90.46%	10.47%	DistMult	59.64%	87.52%	57.73%	58.75%
	ERMLP	89.48%	90.27%	86.31%	ERMLP	94.13%	95.07%	94.74%	93.44%
	HoIE	87.48%	93.72%	86.92%	HoIE	88.13%	88.69%	86.87%	87.99%
	KG2E	10.89%	86.64%	30.87%	KG2E	9.54%	91.85%	90.69%	83.52%
	MuRE	91.81%	93.81%	91.99%	MuRE	94.41%	93.85%	94.60%	94.55%
	NTN	10.34%	9.64%	10.99%	NTN	11.17%	10.99%	9.96%	9.92%
	ProjE	33.71%	92.13%	23.51%	ProjE	90.50%	91.90%	94.18%	92.32%
	QuatE	95.44%	96.32%	95.72%	QuatE	96.97%	97.44%	97.72%	97.35%
	RESICAL	93.67%	96.14%	95.16%	RESICAL	96.32%	93.72%	96.83%	97.11%
	RotatE	9.54%	97.02%	10.47%	RotatE	88.13%	97.95%	96.88%	88.31%
	SE	96.23%	97.49%	96.32%	SE	96.88%	97.11%	98.32%	96.51%
	SimplE	77.14%	95.72%	79.70%	SimplE	96.23%	93.72%	95.16%	95.07%
	TransD	23.18%	44.93%	25.00%	TransD	29.42%	40.32%	27.47%	27.65%
	TransE	74.58%	92.88%	67.78%	TransE	68.90%	86.13%	88.97%	70.25%
	TransR	56.61%	70.48%	55.96%	TransR	59.45%	70.62%	68.44%	67.46%
	TuckER	97.95%	98.18%	98.37%	TuckER	96.46%	98.04%	98.28%	97.81%
	UM	10.01%	7.36%	11.31%	UM	9.68%	6.47%	5.03%	6.84%

Fig. 18. Results for all configurations on Kinships based on Adadelta. **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

training_approach = LCWA				training_approach = sLCWA					
	BCEL	CEL	SPL	BCEL	MRL	NSSAL	SPL		
inverse_relations = False	ComplEx	94.23%	97.63%	94.88%	ComplEx	98.04%	95.44%	97.30%	97.30%
	ConvE	95.58%	95.58%	95.90%	ConvE	97.11%	96.51%	71.14%	96.42%
	ConvKB	93.90%	89.06%	92.88%	ConvKB	91.62%	96.00%	95.72%	95.53%
	DistMult	86.27%	90.22%	85.57%	DistMult	87.66%	86.55%	88.31%	87.62%
	ERMLP	92.04%	70.07%	90.83%	ERMLP	93.44%	95.72%	95.30%	95.95%
	HolE	87.34%	88.87%	88.31%	HolE	88.59%	88.18%	87.80%	87.24%
	KG2E	11.13%	16.62%	13.64%	KG2E	12.06%	89.34%	18.95%	14.01%
	MuRE	94.79%	88.27%	94.74%	MuRE	94.83%	92.88%	94.74%	94.65%
	NTN	82.73%	89.20%	86.13%	NTN	86.31%	84.78%	56.05%	90.46%
	ProjE	92.64%	92.32%	88.83%	ProjE	94.79%	92.64%	92.74%	94.60%
	QuatE	88.27%	97.11%	90.50%	QuatE	97.72%	95.11%	97.63%	97.67%
	RESCAL	57.22%	95.95%	84.31%	RESCAL	93.48%	92.27%	95.72%	97.44%
	RotatE	98.18%	97.72%	92.27%	RotatE	94.46%	98.32%	98.56%	94.32%
	SE	94.18%	95.11%	94.83%	SE	96.79%	96.46%	96.74%	97.02%
	SimplE	96.42%	97.16%	91.67%	SimplE	98.28%	95.95%	97.77%	97.72%
	TransD	36.50%	40.50%	23.70%	TransD	37.76%	44.83%	39.15%	34.54%
	TransE	64.43%	79.61%	67.23%	TransE	69.65%	78.82%	89.85%	68.95%
	TransH	26.72%	43.67%	27.33%	TransH	38.45%	49.53%	35.52%	29.56%
	TransR	64.57%	66.95%	63.64%	TransR	70.62%	73.32%	69.09%	72.86%
	TuckER	97.81%	97.58%	97.63%	TuckER	94.37%	91.48%	61.55%	97.07%
UM	9.92%	9.26%	10.38%	UM	9.50%	8.38%	9.92%	8.80%	
inverse_relations = True	ComplEx	97.21%	98.37%	96.42%	ComplEx	98.23%	95.81%	97.49%	97.95%
	ConvE	98.37%	98.28%	98.23%	ConvE	46.97%	96.88%	97.07%	94.88%
	ConvKB	96.32%	95.25%	94.37%	ConvKB	96.60%	96.18%	97.07%	96.79%
	DistMult	87.62%	93.53%	86.03%	DistMult	87.71%	87.29%	85.99%	88.18%
	ERMLP	96.14%	95.07%	95.20%	ERMLP	97.49%	96.46%	97.11%	97.49%
	HolE	89.53%	93.30%	88.31%	HolE	89.01%	87.85%	87.90%	87.71%
	KG2E	10.34%	18.53%	14.01%	KG2E	11.45%	91.29%	70.48%	53.45%
	MuRE	94.83%	93.53%	95.02%	MuRE	94.37%	92.74%	93.25%	94.04%
	NTN	87.06%	90.69%	89.71%	NTN	93.62%	89.99%	70.76%	10.80%
	ProjE	95.20%	94.97%	92.97%	ProjE	95.25%	94.41%	95.20%	96.65%
	QuatE	94.83%	98.18%	95.53%	QuatE	97.25%	94.23%	96.60%	97.35%
	RESCAL	87.10%	96.93%	89.62%	RESCAL	88.64%	91.67%	95.44%	97.72%
	RotatE	98.00%	98.18%	93.16%	RotatE	94.74%	98.56%	98.46%	94.69%
	SE	96.60%	97.21%	96.51%	SE	96.97%	96.42%	96.97%	97.25%
	SimplE	97.77%	98.00%	94.46%	SimplE	98.28%	95.39%	97.16%	98.04%
	TransD	43.53%	45.20%	39.80%	TransD	40.08%	42.36%	34.64%	32.77%
	TransE	69.97%	86.96%	65.69%	TransE	68.25%	87.38%	82.68%	67.97%
	TransH	36.87%	52.05%	37.06%	TransH	34.54%	47.58%	32.54%	35.66%
	TransR	71.51%	73.09%	72.63%	TransR	71.93%	72.11%	70.95%	73.23%
	TuckER	98.42%	98.37%	98.60%	TuckER	72.25%	96.83%	64.71%	90.83%
UM	9.31%	9.50%	10.20%	UM	11.17%	6.52%	8.38%	8.75%	

Fig. 19. Results for all configurations on Kinships based on Adam. **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

training_approach = LCWA				training_approach = sLCWA				
	BCEL	CEL	SPL	BCEL	MRL	NSSAL	SPL	
ComplEx	45.57%	53.74%	41.23%	44.27%	42.97%	42.72%	43.72%	inverse_relations = False
ConvE	50.55%	48.65%	49.52%	14.09%	3.37%	29.34%	36.41%	
ConvKB				39.71%	40.22%	38.94%	37.59%	
DistMult	42.17%	48.82%	45.78%	49.38%	50.00%	50.00%	45.43%	
ERMLP	15.51%	35.36%	19.27%	26.93%	30.57%	14.35%	24.30%	
HoIE	47.55%	50.02%	38.87%	45.95%	40.83%	43.57%	45.91%	
KG2E	0.02%	42.72%	52.03%	0.02%	45.14%	45.91%	46.60%	
MuRE	54.05%	48.07%	55.49%	56.43%	45.76%	51.13%	56.89%	
NTN		1.09%		0.10%	31.86%		0.14%	
ProjE	45.08%	48.20%	39.23%	42.46%	40.70%	41.11%	43.16%	
QuatE	44.20%	55.01%		44.68%	51.09%	41.89%	41.72%	
RESICAL	39.43%	53.92%	37.84%	39.21%	36.22%	42.58%	40.87%	
RotatE	59.08%	50.14%	57.99%	51.28%	50.79%	58.02%	51.23%	
SE		9.73%		41.81%			45.49%	
SimplE	45.84%	49.42%	38.20%	42.25%	38.65%	40.71%	42.42%	
TransD	0.10%	16.16%	0.09%	44.66%	46.55%	36.92%	44.25%	
TransE	3.98%	34.30%	56.98%	43.16%	48.91%	49.64%	42.56%	
TransH	2.36%	9.75%	0.85%	25.12%	48.17%	7.66%	38.25%	
TransR	1.09%	14.36%	0.15%	34.97%	42.51%	34.18%	35.41%	
TuckER	53.30%	52.84%	52.14%	34.52%	12.98%	2.21%	35.45%	
UM	44.68%	41.14%	44.89%	40.15%	39.60%	42.31%	39.79%	
	BCEL	CEL	SPL	BCEL	MRL	NSSAL	SPL	
ComplEx	44.03%	10.70%	38.22%	42.53%	42.53%	40.75%	42.24%	inverse_relations = True
ConvE	53.73%	56.33%	53.51%	38.17%	1.98%	40.89%	33.67%	
ConvKB				40.15%	38.24%	42.08%	37.65%	
DistMult	46.03%	52.62%	44.07%	51.16%	46.48%	47.76%	45.13%	
ERMLP	42.68%	46.27%	41.34%	47.35%	41.81%	45.67%	47.66%	
HoIE	44.75%	29.62%	34.05%	45.52%	39.38%	42.77%	45.62%	
KG2E	0.00%	48.02%	50.53%	0.03%	42.65%	43.76%	46.07%	
MuRE	3.28%	53.25%	57.90%	57.83%	46.07%	56.79%	55.63%	
NTN		0.00%		24.38%	24.15%		0.02%	
ProjE	45.33%	51.73%	42.92%	42.61%	42.08%	43.67%	41.64%	
QuatE	49.38%	52.07%	50.82%	38.89%	49.57%	38.82%	43.30%	
RESICAL	35.48%	52.99%	40.77%	39.28%	37.40%	41.35%	42.17%	
RotatE	60.09%	56.74%	58.33%	49.86%	49.13%	57.40%	49.06%	
SE		2.58%		40.44%			41.69%	
SimplE	43.62%	50.89%	39.91%	40.89%	38.56%	39.43%	41.88%	
TransD	0.12%	29.60%	0.02%	44.03%	44.56%	33.62%	38.78%	
TransE	56.74%	46.82%	23.63%	44.39%	48.96%	50.67%	42.37%	
TransH	1.45%	13.83%	1.80%	33.50%	38.01%	4.67%	3.37%	
TransR	0.10%	37.62%	0.09%	32.95%	10.89%	28.69%	34.42%	
TuckER		56.09%	52.33%	16.89%	41.26%	0.38%	1.59%	
UM	43.95%	42.17%	44.68%	39.88%	39.84%	41.76%	39.02%	

Fig. 20. Results for all configurations on WN18RR based on Adam. **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

training_approach = LCWA				training_approach = sLCWA				
	BCEL	CEL	SPL	BCEL	MRL	NSSAL	SPL	
CompEx	27.67%	40.33%	21.80%	37.29%	29.09%	30.27%	33.85%	inverse_relations = False
ConvE	36.26%	31.35%	33.38%	3.89%	8.28%	4.53%	0.96%	
ConvKB				30.32%	28.08%	31.47%	32.26%	
DistMult	34.36%	40.87%	33.69%	33.70%	40.41%	35.24%	34.34%	
ERMLP	28.06%	28.02%	26.76%	36.07%	35.47%	29.18%	34.93%	
HolE	38.45%	32.30%	32.99%	35.42%	31.30%	25.58%	34.70%	
KG2E	0.05%	22.27%	42.37%	0.06%	38.28%	38.19%	42.70%	
MuRE	42.65%	29.25%	41.77%	43.24%	38.41%	45.07%	43.63%	
NTN	10.21%			2.44%	9.89%	0.03%	20.34%	
ProjE	31.65%	29.45%	24.52%	30.87%	32.50%	26.45%	30.24%	
QuatE	31.14%	24.00%	23.23%	27.38%	31.08%	26.92%	29.35%	
RESCAL	26.81%	36.92%	21.33%	26.20%	32.81%	32.46%	35.63%	
RotatE	47.81%	45.76%	43.96%	38.37%	43.22%	49.75%	38.37%	
SE				24.17%	33.96%	38.89%	34.04%	
Simple	24.24%	36.11%	20.97%	29.62%	24.71%	30.48%	29.68%	
TransD	4.31%	26.51%	0.87%	28.96%	35.01%	27.10%	31.02%	
TransE	44.54%	33.94%	45.32%	30.97%	46.42%	42.16%	35.84%	
TransH	16.42%	21.58%	21.15%	27.54%	35.29%	25.31%	26.93%	
TransR	10.81%	21.84%	9.79%		36.14%	7.85%		
TuckER	46.33%	42.22%	44.93%	5.13%	5.37%	3.99%	2.33%	
UM	6.27%	8.02%	6.84%	6.12%	3.43%	5.70%	6.30%	
	BCEL	CEL	SPL	BCEL	MRL	NSSAL	SPL	
CompEx	33.76%	44.84%	29.16%	33.32%	30.99%	28.96%	28.51%	inverse_relations = True
ConvE	49.21%	45.90%		0.00%	1.26%	0.69%	3.73%	
ConvKB						28.35%		
DistMult	40.60%	47.39%	39.34%	32.68%	39.13%	35.31%	34.24%	
ERMLP	45.10%	41.30%	40.27%	35.69%	37.76%	31.23%	37.08%	
HolE	38.15%	42.23%	33.32%	37.00%	31.05%	32.01%	36.22%	
KG2E	0.79%	37.70%	45.50%	0.65%	36.41%	34.19%	40.68%	
MuRE	47.20%	34.37%	45.22%	44.14%	37.21%	43.52%	43.15%	
NTN	18.10%			12.30%	2.05%	1.27%	3.03%	
ProjE	41.62%	41.57%	34.13%	26.34%	35.28%	23.86%	26.52%	
QuatE	30.15%	46.17%	34.08%	26.91%	29.80%	21.85%	29.97%	
RESCAL	45.39%	46.46%	33.34%	30.32%	32.04%	26.45%	33.59%	
RotatE	44.29%	47.04%	44.25%	37.91%	37.94%	48.61%	38.40%	
SE				30.45%	26.80%	39.43%	21.96%	
Simple	35.68%	40.31%	21.81%	27.75%	16.82%	26.54%	28.28%	
TransD	3.00%	28.55%	3.52%	25.65%	41.86%	24.76%	29.87%	
TransE	39.15%	45.51%	42.79%	30.44%	43.53%	40.49%	35.11%	
TransH	27.27%	25.95%	27.67%	24.42%	27.45%	23.11%	26.03%	
TransR	15.95%	39.19%	11.77%		33.89%	12.20%		
TuckER	52.86%	50.58%	52.85%	2.84%	7.00%	2.33%	2.92%	
UM	6.51%	7.98%	7.28%	6.09%	3.31%	5.82%	6.17%	
	BCEL	CEL	SPL	BCEL	MRL	NSSAL	SPL	

Fig. 21. Results for all configurations on FB15K-237 based on Adam. **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

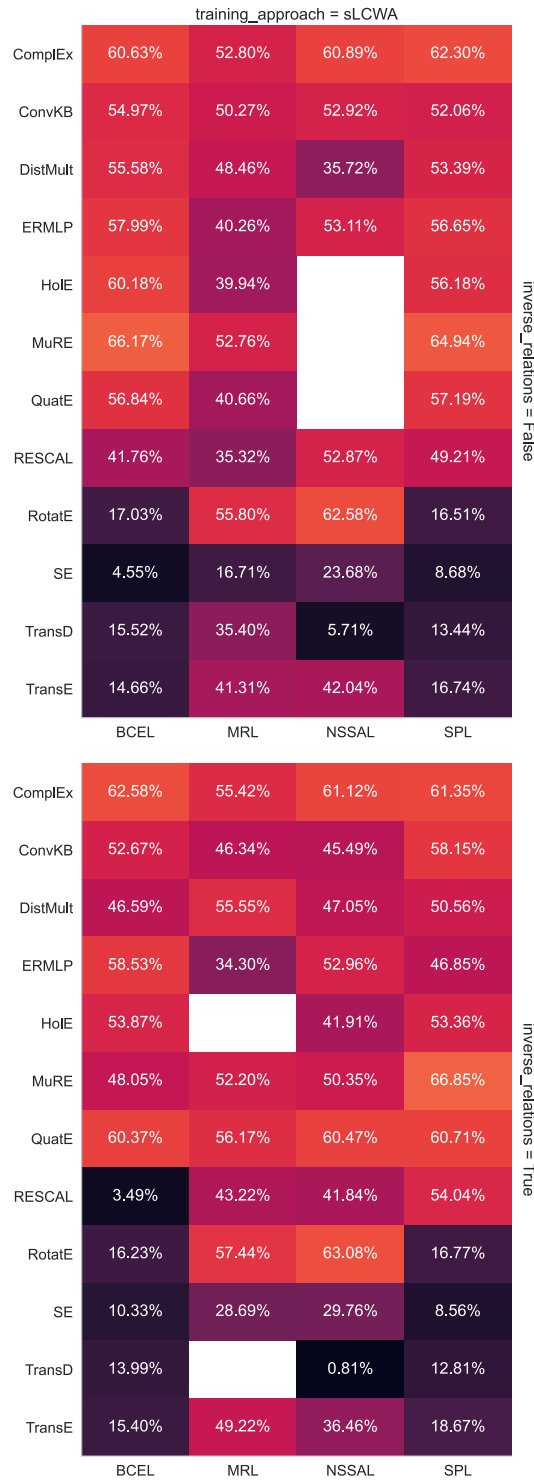


Fig. 22. Results for all configurations on YAGO3-10 based on Adam. **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

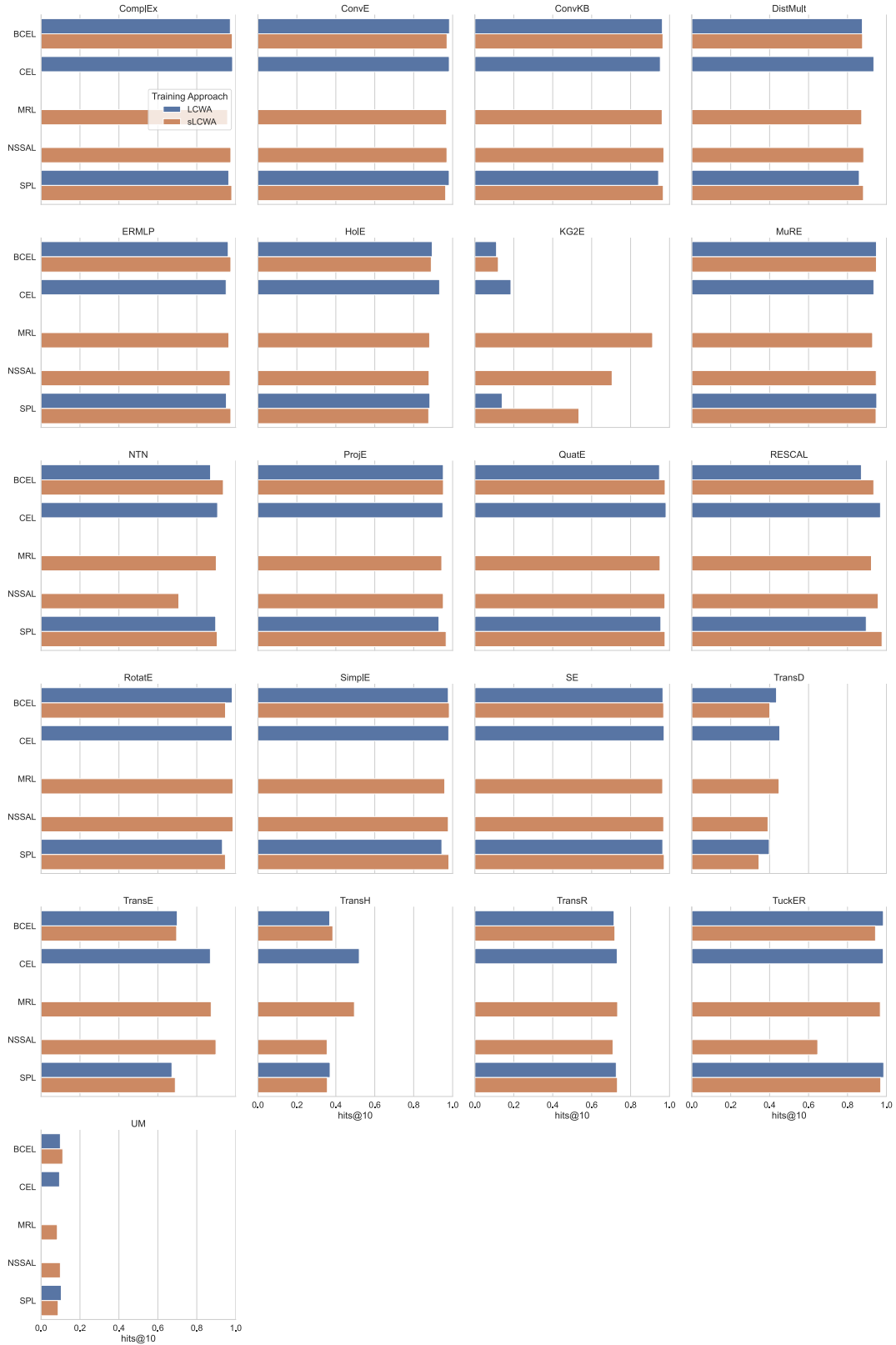


Fig. 23. Impact of the training approach on the performance for a fixed interaction model and loss function for the Kinships dataset (results represent for each setting the best-performing configuration). **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.



Fig. 24. Impact of the training approach on the performance for a fixed interaction model and loss function for the WN18RR dataset (results represent for each setting the best-performing configuration). **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

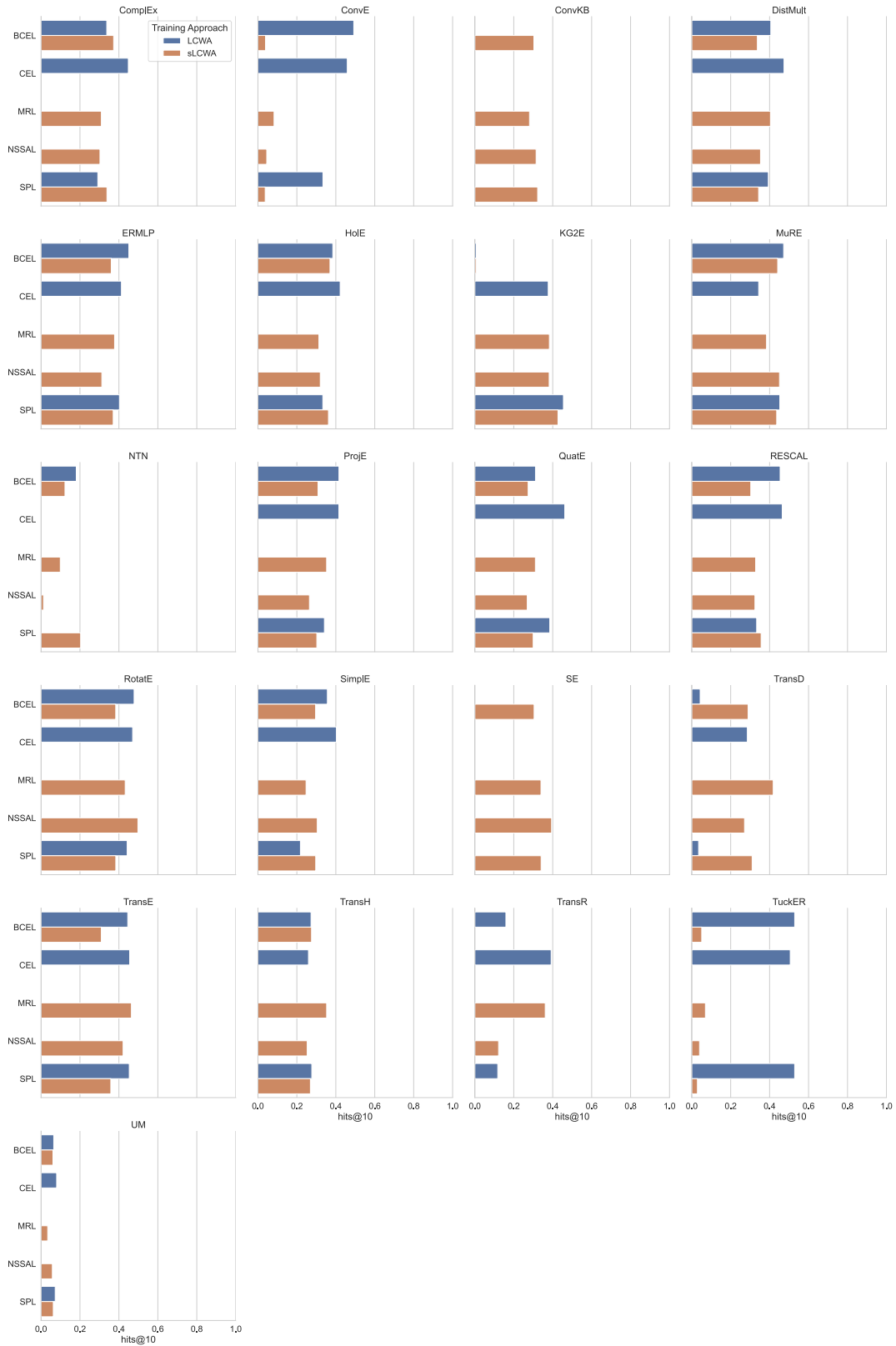


Fig. 25. Impact of the training approach on the performance for a fixed interaction model and loss function for the FB15K-237 dataset (results represent for each setting the best-performing configuration). **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

Appendix D

Improving Inductive Link Prediction Using Hyper-relational Facts



Improving Inductive Link Prediction Using Hyper-relational Facts

Mehdi Ali^{1,2} (✉), Max Berrendorf³, Mikhail Galkin⁴, Veronika Thost⁵,
Tengfei Ma⁵, Volker Tresp^{3,6}, and Jens Lehmann^{1,2}

¹ Smart Data Analytics Group, University of Bonn, Bonn, Germany
{mehdi.ali, jens.lehmann}@cs.uni-bonn.de

² Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS),
Sankt Augustin, Dresden, Germany

{mehdi.ali, jens.lehmann}@iaais.fraunhofer.de

³ Ludwig-Maximilians-Universität München, Munich, Germany
{berrendorf, tresp}@dbs.ifi.lmu.de

⁴ Mila, McGill University, Montreal, Canada
mikhail.galkin@mila.quebec

⁵ IBM Research, MIT-IBM Watson AI Lab, Cambridge, USA
vth@zurich.ibm.com, tengfei.ma1@ibm.com

⁶ Siemens AG, Munich, Germany
volker.tresp@siemens.com

Abstract. For many years, link prediction on knowledge graphs (KGs) has been a purely transductive task, not allowing for reasoning on unseen entities. Recently, increasing efforts are put into exploring semi- and fully inductive scenarios, enabling inference over unseen and emerging entities. Still, all these approaches only consider triple-based KGs, whereas their richer counterparts, hyper-relational KGs (e.g., Wikidata), have not yet been properly studied. In this work, we classify different inductive settings and study the benefits of employing hyper-relational KGs on a wide range of semi- and fully inductive link prediction tasks powered by recent advancements in graph neural networks. Our experiments on a novel set of benchmarks show that qualifiers over typed edges can lead to performance improvements of 6% of absolute gains (for the Hits@10 metric) compared to triple-only baselines. Our code is available at https://github.com/mali-git/hyper_relational_ilp.

1 Introduction

Knowledge graphs are notorious for their sparsity and incompleteness [16], so that predicting missing links has been one of the first applications of machine learning and embedding-based methods over KGs [9, 22]. A flurry [2, 20] of such algorithms has been developed over the years, and most of them share certain commonalities, i.e., they operate over *triple-based* KGs in the *transductive* setup, where all entities are known at training time. Such approaches can neither operate on unseen entities, which might emerge after updating the graph, nor

M. Ali and M. Berrendorf—Equal contribution.

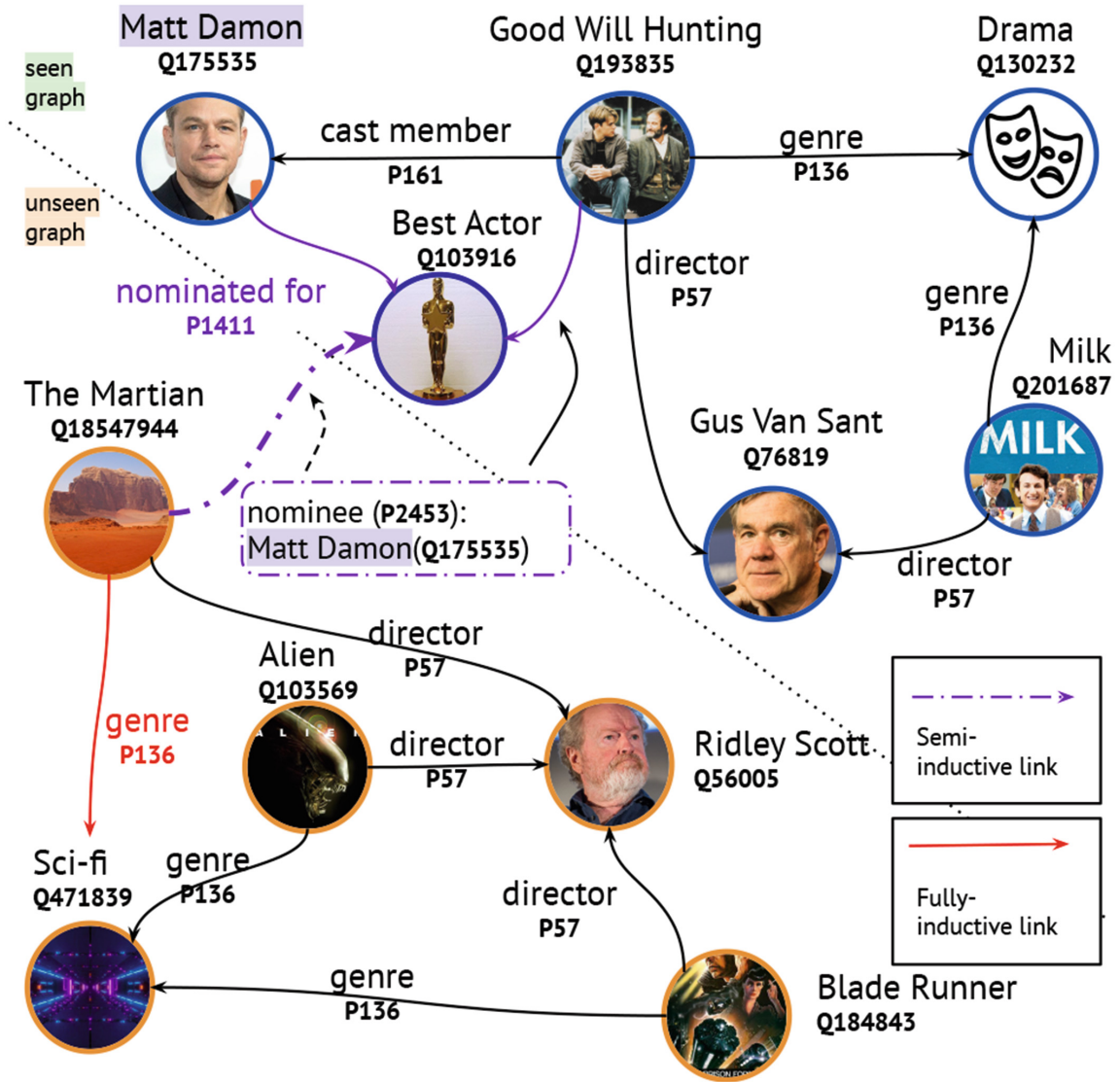


Fig. 1. Different types of inductive LP. Semi-inductive: the link between *The Martian* and Best Actor from the seen graph. Fully-inductive: the *genre* link between unseen entities given a new unseen subgraph *at inference time*. The qualifier (*nominee: Matt Damon*) over the original relation *nominated for* allows to better predict the semi-inductive link.

on new (sub-)graphs comprised of completely new entities. Those scenarios are often unified under the *inductive* link prediction (LP) setup. A variety of NLP tasks building upon KGs have inductive nature, for instance, entity linking or information extraction. Hence, being able to work in inductive settings becomes crucial for KG representation learning algorithms. For instance (cf. Fig. 1), the *director-genre* pattern from the seen graph allows to predict a missing *genre* link for *The Martian* in the unseen subgraph.

Several recent approaches [13, 24] tackle an inductive LP task, but they usually focus on a specific inductive setting. Furthermore, their underlying KG structure is still based on triples. On the other hand, new, more expressive KGs

like Wikidata [26] exhibit a *hyper-relational* nature where each triple (a typed edge in a graph) can be further instantiated with a set of explicit relation-entity pairs, known as *qualifiers* in the Wikidata model. Recently, it was shown [17] that employing hyper-relational KGs yields significant gains in the transductive LP task compared to their triple-only counterparts. But the effect of such KGs on inductive LP is unclear. Intuitively (Fig. 1), the (nominee: Matt Damon) qualifier provides a helpful signal to predict Best Actor as an object of nominated for of The Martian given that Good Will Hunting received such an award with the same nominee.

In this work, we systematically study hyper-relational KGs in different inductive settings:

- We propose a classification of inductive LP scenarios that describes the settings formally and, to the best of our knowledge, integrates all relevant existing works. Specifically, we distinguish *fully-inductive* scenarios, where target links are to be predicted in a new subgraph of unseen entities, and *semi-inductive* ones where unseen nodes have to be connected to a known graph.
- We then adapt two existing baseline models for the two inductive LP tasks probing them in the hyper-relational settings.
- Our experiments suggest that models supporting hyper-relational facts indeed improve link prediction in both inductive settings compared to strong triple-only baselines by more than 6% Hits@10.

2 Background

We assume the reader to be familiar with the standard link prediction setting (e.g. from [22]) and introduce the specifics of the setting with qualifiers.

2.1 Statements: Triples Plus Qualifiers

Let $G = (\mathcal{E}, \mathcal{R}, \mathcal{S})$ be a hyper-relational KG where \mathcal{E} is a set of entities, \mathcal{R} is a set of relations, and \mathcal{S} a set of statements. Each statement can be formalized as a 4-tuple (h, r, t, q) of a head and tail entity¹ $h, t \in \mathcal{E}$, a relation $r \in \mathcal{R}$, and a set of qualifiers, which are relation-entity pairs $q \subseteq \mathfrak{P}(\mathcal{R} \times \mathcal{E})$ where \mathfrak{P} denotes the power set. For example, Fig. 1 contains a statement (Good Will Hunting, nominated for, Best Actor, {(nominee, Matt Damon)}) where (nominee, Matt Damon) is a qualifier pair for the main triple. We define the set of all possible statements as set

$$\mathcal{S}(\mathcal{E}_H, \mathcal{R}, \mathcal{E}_T, \mathcal{E}_Q) = \mathcal{E}_H \times \mathcal{R} \times \mathcal{E}_T \times \mathfrak{P}(\mathcal{R} \times \mathcal{E}_Q)$$

with a set of relations \mathcal{R} , a set of head, tail and qualifier entities $\mathcal{E}_H, \mathcal{E}_T, \mathcal{E}_Q \subseteq \mathcal{E}$. Further, \mathcal{S}_{train} is the set of training statements and \mathcal{S}_{eval} are evaluation statements. We assume that we have a feature vector $\mathbf{x}_e \in \mathbb{R}^d$ associated with

¹ We use *entity* and *node* interchangeably.

each entity $e \in \mathcal{E}$. Such feature vectors can, for instance, be obtained from entity descriptions available in some KGs or represent topological features such as Laplacian eigenvectors [6] or regular graph substructures [10]. In this work, we focus on the setting with one fixed set of known relations. That is, we do not require $\mathbf{x}_r \in \mathbb{R}^d$ features for relations and rather learn relation embeddings during training.

2.2 Expressiveness

Models making use of qualifiers are strictly more expressive than those which do not: Consider the following example with two statements, $s_1 = (h, r, t, q_1)$ and $s_2 = (h, r, t, q_2)$, sharing the same triple components, but differing in their qualifiers, such that $s_1|q_1 = False$ and $s_2|q_2 = True$. For a model f_{NQ} not using qualifiers, i.e., only using the triple component (h, r, t) , we have $f_{NQ}(s_1) = f_{NQ}(s_2)$. In contrast, a model f_Q using qualifiers can predict $f_Q(s_1) \neq f_Q(s_2)$, thus being strictly more expressive.

Table 1. Inductive LP in the literature, a discrepancy in terminology. The approaches differ in the kind of auxiliary statements \mathcal{S}_{inf} used at inference time: in whether they contain entities seen during training E_{tr} and whether new entities E_{inf} are connected to seen ones (k -shot scenario), or (only) amongst each other, in a new graph. Note that the evaluation settings also vary.

Named scenario	\mathcal{S}_{inf}	Unseen \leftrightarrow Unseen	Unseen \leftrightarrow Seen	Scoring against	In our framework
Out-of-sample [1]	k -shot	–	✓	E_{tr}	SI
Unseen entities [12]	k -shot	–	✓	E_{tr}	SI
Inductive [8]	k -shot	–	✓	E_{tr}	SI
Inductive [24]	New graph	✓	–	E_{inf}	FI
Transfer [13]	New graph	✓	–	E_{inf}	FI
Dynamic [13]	k -shot + new graph	✓	✓	$E_{tr} \cup E_{inf}$	FI/SI
Out-of-graph [4]	k -shot + new graph	✓	✓	$E_{tr} \cup E_{inf}$	FI/SI
Inductive [27]	k -shot + new graph	✓	✓	$E_{tr} \cup E_{inf}$	FI/SI

3 Inductive Link Prediction

Recent works (cf. Table 1) have pointed out the practical relevance of different inductive LP scenarios. However, there exists a terminology gap as different authors employ different names for describing conceptually the same task or, conversely, use the same *inductive LP* term for practically different setups. We propose a unified framework that provides an overview of the area and describes the settings formally.

Let \mathcal{E}_\bullet denote the set of entities occurring in the training statements \mathcal{S}_{train} at any position (head, tail, or qualifier), and $\mathcal{E}_\circ \subseteq \mathcal{E} \setminus \mathcal{E}_\bullet$ denote a set of unseen entities. In the *transductive* setting, all entities in the evaluation statements are seen during training, i.e., $\mathcal{S}_{eval} \subseteq \mathbb{S}(\mathcal{E}_\bullet, \mathcal{R}, \mathcal{E}_\bullet, \mathcal{E}_\bullet)$. In contrast, in *inductive* settings, \mathcal{S}_{eval} , used in validation and testing, may contain unseen entities. In

order to be able to learn representations for these entities at inference time, inductive approaches may consider an additional set \mathcal{S}_{inf} of *inference statements* about (un)seen entities; of course $\mathcal{S}_{inf} \cap \mathcal{S}_{eval} = \emptyset$.

The *fully-inductive* setting (FI) is akin to transfer learning where link prediction is performed over a set of entities not seen before, i.e., $\mathcal{S}_{eval} \subseteq \mathbb{S}(\mathcal{E}_o, \mathcal{R}, \mathcal{E}_o, \mathcal{E}_o)$. This is made possible by providing an auxiliary inference graph $\mathcal{S}_{inf} \subseteq \mathbb{S}(\mathcal{E}_o, \mathcal{R}, \mathcal{E}_o, \mathcal{E}_o)$ containing statements about the unseen entities in \mathcal{S}_{eval} . For instance, in Fig. 1, the training graph is comprised of entities **Matt Damon**, **Good Will Hunting**, **Best Actor**, **Gus Van Sant**, **Milk**, **Drama**. The inference graph contains new entities **The Martian**, **Alien**, **Ridley Scott**, **Blade Runner**, **Sci-fi** with one missing link to be predicted. The fully-inductive setting is considered in [13, 24].

In the *semi-inductive* setting (SI), new, unseen entities are to be connected to seen entities, i.e., $\mathcal{S}_{eval} \subseteq \mathbb{S}(\mathcal{E}_\bullet, \mathcal{R}, \mathcal{E}_o, \mathcal{E}_\bullet) \cup \mathbb{S}(\mathcal{E}_o, \mathcal{R}, \mathcal{E}_\bullet, \mathcal{E}_\bullet)$. Illustrating with Fig. 1, **The Martian** as the only unseen entity connecting to the seen graph, the semi-inductive statement connects **The Martian** to the seen **Best Actor**. Note that there are other practically relevant examples beyond KGs, such as predicting interaction links between a new drug and a graph containing existing proteins/drugs [5, 18]. We hypothesize that, in most scenarios, we are not given any additional information about the new entity, and thus have $\mathcal{S}_{inf} = \emptyset$; we will focus on this case in this paper. However, the variation where \mathcal{S}_{inf} may contain k statements connecting the unseen entity to seen ones has been considered too [1, 8, 12] and is known as *k-shot learning* scenario.

A mix of the fully- and semi-inductive settings where evaluation statements may contain two instead of just one unseen entity is studied in [4, 13, 27]. That is, unseen entities might be connected to the seen graph, i.e., \mathcal{S}_{eval} may contain seen entities, and, at the same time, the unseen entities might be connected to each other; i.e., $\mathcal{S}_{inf} \neq \emptyset$.

Our framework is general enough to allow \mathcal{S}_{eval} to contain new, unseen relations r having their features \mathbf{x}_r at hand. Still, to the best of our knowledge, research so far has focused on the setting where all relations are seen in training; we will do so, too.

We hypothesize that qualifiers, being explicit attributes over typed edges, provide a strong inductive bias for LP tasks. In this work, for simplicity, we require both qualifier relations and entities to be seen in the training graph, i.e., $\mathcal{E}_Q \subseteq \mathcal{E}_\bullet$ and $\mathcal{R}_Q \subseteq \mathcal{R}$, although the framework accommodates a more general case of unseen qualifiers given their respective features.

4 Approach

Both semi- and fully-inductive tasks assume node features to be given. Recall that relation embeddings are learned and, often, to reduce the computational complexity, their dimensionality is smaller than that of node features.

4.1 Encoders

In the semi-inductive setting, an unseen entity arrives without any graph structure pointing to existing entities, i.e., $\mathcal{S}_{inf} = \emptyset$. This fact renders message passing approaches [19] less applicable, so we resort to a simple linear layer to project all entity features (including those of qualifiers) into the relation space: $\phi : \mathbb{R}^{d_f} \rightarrow \mathbb{R}^{d_r}$

In the fully inductive setting, we are given a non-empty inference graph $\mathcal{S}_{inf} \neq \emptyset$, and we probe two encoders: (i) the same linear projection of features as in the semi-inductive scenario which does not consider the graph structure; (ii) GNNs which can naturally work in the inductive settings [11]. However, the majority of existing GNN encoders for multi-relational KGs like CompGCN [25] are limited to only triple KG representation. To the best of our knowledge, only the recently proposed STARE [17] encoder supports hyper-relational KGs which we take as a basis for our inductive model. Its aggregation formula is:

$$\mathbf{x}'_v = f \left(\sum_{(u,r) \in \mathcal{N}(v)} \mathbf{W}_{\lambda(r)} \phi_r(\mathbf{x}_u, \gamma(\mathbf{x}_r, \mathbf{x}_q)_{vu}) \right) \quad (1)$$

where γ is a function that infuses the vector of aggregated qualifiers \mathbf{x}_q into the vector of the main relation \mathbf{x}_r . The output of the GNN contains updated node and relation features based on the adjacency matrix A and qualifiers Q :

$$\mathbf{X}', \mathbf{R}' = \text{STARE}(A, \mathbf{X}, \mathbf{R}, Q)$$

Finally, in both inductive settings, we linearize an input statement in a sequence using a padding index where necessary: $[\mathbf{x}'_h, \mathbf{x}'_r, \mathbf{x}'_{q_1^r}, \mathbf{x}'_{q_1^e}, [\text{PAD}], \dots]$. Note that statements can greatly vary in length depending on the amount of qualifier pairs, and padding mitigates this issue.

Table 2. Semi-inductive (SI) and fully-inductive (FI) datasets. $S_{ds}(Q\%)$ denotes the number of statements with the qualifiers ratio in train ($ds = tr$), validation ($ds = vl$), test ($ds = ts$), and inductive inference ($ds = inf$) splits. E_{ds} is the number of distinct entities. R_{ds} is the number of distinct relations. \mathcal{S}_{inf} is a basic graph for vl and ts in the FI scenario.

Type	Name	Train			Validation			Test			Inference		
		$S_{tr}(Q\%)$	E_{tr}	R_{tr}	$S_{vl}(Q\%)$	E_{vl}	R_{vl}	$S_{ts}(Q\%)$	E_{ts}	R_{ts}	$S_{inf}(Q\%)$	E_{inf}	R_{inf}
SI	WD20K (25)	39,819 (30%)	17,014	362	4,252 (25%)	3544	194	3,453 (22%)	3028	198	–	–	–
SI	WD20K (33)	25,862 (37%)	9251	230	2,423 (31%)	1951	88	2,164 (28%)	1653	87	–	–	–
FI	WD20K (66) V1	9,020 (85%)	6522	179	910 (45%)	1516	111	1,113 (50%)	1796	110	6,949 (49%)	8313	152
FI	WD20K (66) V2	4,553 (65%)	4269	148	1,480 (66%)	2322	79	1,840 (65%)	2700	89	8,922 (58%)	9895	120
FI	WD20K (100) V1	7,785 (100%)	5783	92	295 (100%)	643	43	364 (100%)	775	43	2,667 (100%)	4218	75
FI	WD20K (100) V2	4,146 (100%)	3227	57	538 (100%)	973	43	678 (100%)	1212	42	4,274 (100%)	5573	54

4.2 Decoder

Given an encoded sequence, we use the same Transformer-based decoder for all settings:

$$f(h, r, t, q) = g(\mathbf{x}'_h, \mathbf{x}'_r, \mathbf{x}'_{q_1}, \mathbf{x}'_{q_e}, \dots)^T \mathbf{x}'_t \text{ with}$$

$$g(\mathbf{x}'_1, \dots, \mathbf{x}'_k) = \text{Agg}(\text{Transformer}([\mathbf{x}'_1, \dots, \mathbf{x}'_k]))$$

In this work, we evaluated several aggregation strategies and found a simple mean pooling over all non-padded sequence elements to be preferable. Interaction functions of the form $f(h, r, t, q) = f_1(h, r, q)^T f_2(t)$ are particularly well-suited for fast 1-N scoring for tail entities, since the first part only needs to be computed only once.

Here and below, we denote the linear encoder + Transformer decoder model as QBLP (that is, Qualifier-aware BLP, an extension of BLP [13]), and the STARE encoder + Transformer decoder, as STARE.

4.3 Training

In order to compare results with triple-only approaches, we train the models, as usual, on the subject and object prediction tasks. We use stochastic local closed world assumption (sLCWA) and the local closed world assumption (LCWA) commonly used in the KG embedding literature [2]. Particular details on sLCWA and LCWA are presented in Appendix A. Importantly, in the semi-inductive setting, the models score against all entities in the training graph E_{tr} in both training and inference stages. In the fully-inductive scenario, as we are predicting links over an unseen graph, the models score against all entities in E_{tr} during training and against unseen entities in the inference graph E_{inf} during inference.

5 Datasets

We take the original transductive splits of the WD50K [17] family of hyper-relational datasets as a leakage-free basis for sampling our semi- and fully-inductive datasets which we denote by WD20K.

5.1 Fully-Inductive Setting

We start with extracting *statement entities* \mathcal{E}' , and sample n entities and their k -hop neighbourhood to form the statements (h, r, t, q) of the transductive train graph S_{train} . From the remaining $\mathcal{E}' \setminus \mathcal{E}_{train}$ and $S \setminus S_{train}$ sets we sample m entities with their l -hop neighbourhood to form the statements S_{ind} of the inductive graph. The entities of S_{ind} are disjoint with those of the transductive train

graph. Further, we filter out all statements in S_{ind} whose relations (main or qualifier) were not seen in S_{train} . Then, we randomly split S_{ind} with the ratio about 55%/20%/25% into inductive inference, validation, and test statements, respectively. The evaluated models are trained on the transductive train graph S_{train} . During inference, the models receive an unseen inductive inference graph from which they have to predict validation and test statements. Varying k and l , we sample two different splits: V1 has a larger training graph with more seen entities whereas V2 has a bigger inductive inference graph.

5.2 Semi-inductive Setting

Starting from all statements, we extract all entities occurring as head or tail entity in any statement, denoted by \mathcal{E}' and named *statement entities*. Next, we split the set of statement entities into a train, validation and test set: $\mathcal{E}_{train}, \mathcal{E}_{validation}, \mathcal{E}_{test}$. We then proceed to extract statements $(h, r, t, q) \in S$ with one entity (h/t) in \mathcal{E}_{train} and the other entity in the corresponding statement entity split. We furthermore filter the qualifiers to contain only pairs where the entity is in a set of allowed entities, formed by $\mathcal{A}_{split} = \mathcal{E}_{train} \cup \mathcal{E}_{split}$, with split being train/validation/test. Finally, since we do not assume relations to have any features, we do not allow unseen relations. We thus filter out relations which do not occur in the training statements.

5.3 Overview

To measure the effect of hyper-relational facts on both inductive LP tasks, we sample several datasets varying the ratio of statements with and without qualifiers. In order to obtain the initial node features we mine their English surface forms and descriptions available in Wikidata as `rdfs:label` and `schema:description` values. The surface forms and descriptions are concatenated into one string and passed through the Sentence BERT [23] encoder based on RoBERTa [21] to get 1024-dimensional vectors. The overall datasets statistics is presented in Table 2.

6 Experiments

We design our experiments to investigate whether the incorporation of qualifiers improves inductive link prediction. In particular, we investigate the fully-inductive setting (Sect. 6.2) and the semi-inductive setting (Sect. 6.3). We analyze the impact of the qualifier ratio (i.e., the number of statements with qualifiers) and the dataset’s size on a model’s performance.

Table 3. Results on FI WD20K (100) V1 & V2. #QP denotes the number of qualifier pairs used in each statement (including padded pairs). Best results **in bold**, second best underlined.

Model	#QP	WD20K (100) V1					WD20K (100) V2				
		AMR(%)	MRR(%)	H@1(%)	H@5(%)	H@10(%)	AMR(%)	MRR(%)	H@1(%)	H@5(%)	H@10(%)
BLP	0	22.78	5.73	1.92	8.22	12.33	36.71	3.99	1.47	4.87	9.22
CompGCN	0	37.02	10.42	<u>5.75</u>	15.07	18.36	74.00	2.55	0.74	3.39	5.31
QBLP	0	28.91	5.52	1.51	8.08	12.60	35.38	4.94	2.58	5.46	9.66
StarE	2	41.89	9.68	3.73	16.57	20.99	40.60	2.43	0.45	3.86	6.17
StarE	4	35.33	10.41	4.82	15.84	21.76	37.16	5.12	1.41	7.93	12.89
StarE	6	34.86	11.27	6.18	15.93	21.29	47.35	4.99	1.92	6.71	11.06
QBLP	2	18.91	10.45	3.73	16.02	<u>22.65</u>	28.03	6.69	3.49	<u>8.47</u>	12.04
QBLP	4	<u>20.19</u>	<u>10.70</u>	3.99	<u>16.12</u>	24.52	<u>31.30</u>	5.87	2.37	7.85	13.93
QBLP	6	23.65	7.87	2.75	10.44	17.86	34.35	<u>6.53</u>	<u>2.95</u>	9.29	<u>13.13</u>

Table 4. Results on the FI WD20K (66) V1 & V2. #QP denotes the number of qualifier pairs used in each statement (including padded pairs). Best results **in bold**, second best underlined.

Model	#QP	WD20K (66) V1					WD20K (66) V2				
		AMR(%)	MRR(%)	H@1(%)	H@5(%)	H@10(%)	AMR(%)	MRR(%)	H@1(%)	H@5(%)	H@10(%)
BLP	0	34.96	2.10	0.45	2.29	4.44	45.29	1.56	0.27	1.88	3.35
CompGCN	0	35.99	5.80	2.38	8.93	12.79	47.24	<u>2.56</u>	<u>1.17</u>	3.07	4.46
QBLP	0	35.30	3.69	1.30	4.85	7.14	42.48	0.94	0.08	0.79	1.82
StarE	2	37.72	6.84	3.24	9.71	<u>13.44</u>	52.78	2.62	0.74	<u>3.55</u>	<u>5.78</u>
StarE	4	38.91	<u>6.40</u>	<u>2.83</u>	<u>8.94</u>	13.39	51.93	5.06	2.09	7.34	9.82
StarE	6	38.20	6.87	3.46	8.98	13.57	47.01	4.42	2.04	5.73	8.97
QBLP	2	<u>30.37</u>	3.70	1.26	4.90	8.14	53.67	1.39	0.41	1.66	2.59
QBLP	4	30.84	3.20	0.90	4.00	7.14	37.10	2.08	0.38	2.20	4.92
QBLP	6	26.34	4.34	1.66	5.53	9.25	<u>39.12</u>	1.95	0.41	2.15	4.10

6.1 Experimental Setup

We implemented all approaches in Python building upon the open-source library `pykeen` [3] and make the code publicly available.² For each setting (i.e., dataset + number of qualifier pairs per triple), we performed a hyperparameter search using early stopping on the validation set and evaluated the final model on the test set. We used AMR, MRR, and Hits@k as evaluation metrics, where the Adjusted Mean Rank (AMR) [7] is a recently proposed metric which sets the mean rank into relation with the expected mean rank of a random scoring model. Its value ranges from 0%–200%, and a lower value corresponds to better model performance. Each model was trained at most 1000 epochs in the fully inductive setting, at most 600 epochs in the semi-inductive setting, and evaluated based on the early-stopping criterion with a frequency of 1, a patience of 200 epochs (in the semi-inductive setting, we performed all HPOs with a patience of 100 and 200 epochs), and a minimal improvement $\delta > 0.3\%$ optimizing the *hits@10* metric. For both inductive settings, we evaluated the effect of incorporating 0, 2, 4, and 6 qualifier pairs per triple.

² https://github.com/mali-git/hyper_relational_ilp.

6.2 Fully-Inductive Setting

In the full inductive setting, we analyzed the effect of qualifiers for four different datasets (i.e., WD20K (100) V1 & V2 and WD20K (66) V1 & V2, which have different ratios of qualifying statements and are of different sizes (see Sect. 5). As triple-only baselines, we evaluated CompGCN [25] and BLP [13]. To evaluate the effect of qualifiers on the fully-inductive LP task, we evaluated StarE [17] and QBLP. It should be noted that StarE without the use of qualifiers is equivalent to CompGCN.

General Overview. Tables 3 and 4 show the results obtained for the four datasets. The main findings are that (i) for all datasets, the use of qualifiers leads to increased performance, and (ii) the ratio of statements with qualifiers and the size of the dataset has a major impact on the performance. CompGCN and StarE apply message-passing to obtain enriched entity representations while BLP and QBLP only apply a linear transformation. Consequently, CompGCN and StarE require \mathcal{S}_{inf} to contain useful information in order to obtain the entity representations while BLP and QBLP are independent of \mathcal{S}_{inf} . In the following, we discuss the results for each dataset in detail.

Results on WD20K (100) FI V1 & V2. It can be observed that the performance gap between BLP/QBLP (0) and QBLP (2,4,6) is considerably larger than the gap between CompGCN and StarE. This might be explained by the fact that QBLP does not take into account the graph structure provided by \mathcal{S}_{inf} , therefore is heavily dependent on additional information, i.e. the qualifiers compensate for the missing graph information. The overall performance decrease observable between V1 and V2 could be explained by the datasets' composition (Table 2), in particular, in the composition of the training and inference graphs: \mathcal{S}_{inf} of V2 comprises more entities than V1, so that each test triple is ranked against more entities, i.e., the ranking becomes more difficult. At the same time, the training graph of V1 is larger than that of V2, i.e., during training more entities (along their textual features) are seen which may improve generalization.

Results on WD20K (66) FI V1 & V2. Comparing StarE (2,4) to CompGCN (0), there is only a small improvement on this dataset. Also, the improvement of QBLP (2,4,6) compared to BLP and QBLP (0) is smaller than on the previous datasets. This can be connected to the decreased ratio of statements with qualifiers. Besides, the training graph also has fewer qualifier pairs, \mathcal{S}_{inf} which is used by CompGCN and StarE for message passing consists of only 49% of statements with at least one qualifier pair, and only 50% of test statements have at least one qualifier pair which has an influence on all models. This observation supports why StarE outperforms QBLP as the amount of provided qualifier statements cannot compensate for the graph structure in \mathcal{S}_{inf} .

6.3 Semi-inductive Setting

In the semi-inductive setting, we evaluated BLP as a triple-only baseline and QBLP as a statement baseline (i.e., involving qualifiers) on the WD20K SI datasets. We did not evaluate CompGCN and StarE since message-passing-based approaches are not directly applicable in the absence of \mathcal{S}_{inf} . The results highlight that aggregating qualifier information improves the prediction of semi-inductive links despite the fact that the ratio of statements with qualifiers is not very large (37% for SI WD20K (33), and 30% for SI WD20K (25)). In the case of SI WD20K (33), the baselines are outperformed even by a large margin. Overall, the results might indicate that in semi-inductive settings, performance improvements can already be obtained with a decent amount of statements with qualifiers.

Table 5. Results on the WD20K SI datasets. #QP denotes the number of qualifier pairs used in each statement (including padded pairs). Best results **in bold**, second best underlined.

Model	#QP	WD20K (33) SI					WD20K (25) SI				
		AMR(%)	MRR(%)	H@1(%)	H@5(%)	H@10(%)	AMR(%)	MRR(%)	H@1(%)	H@5(%)	H@10(%)
BLP	0	4.76	13.95	7.37	17.28	24.65	6.01	12.45	5.98	17.29	23.43
QBLP	0	7.04	28.35	14.44	28.58	36.32	6.75	17.02	8.82	22.10	29.50
QBLP	2	11.51	35.95	20.70	34.98	41.82	<u>5.99</u>	<u>20.36</u>	<u>11.77</u>	24.86	32.26
QBLP	4	11.38	<u>34.35</u>	<u>19.41</u>	<u>33.90</u>	<u>40.20</u>	12.18	21.05	12.32	24.07	30.09
QBLP	6	<u>4.98</u>	25.94	15.20	30.06	38.70	5.73	19.50	11.14	<u>24.73</u>	<u>31.60</u>

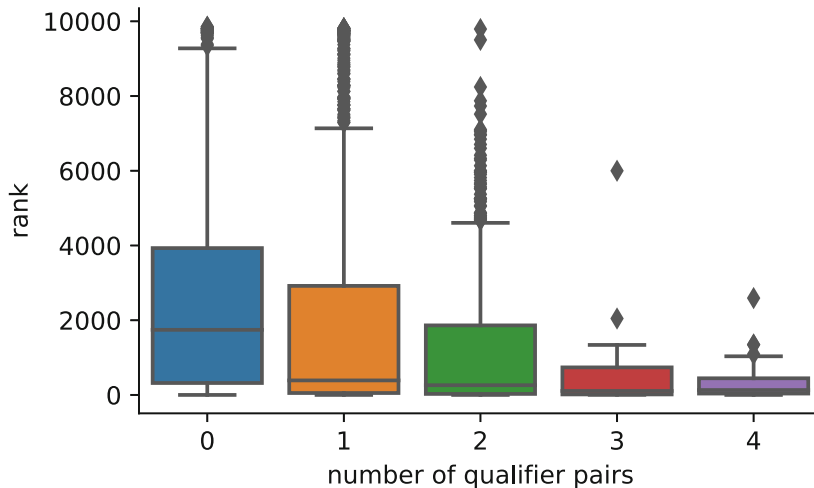


Fig. 2. Distribution of individual ranks for head/tail prediction with StarE on WD20K (66) V2. The statements are grouped by the number of qualifier pairs.

6.4 Qualitative Analysis

We obtain deeper insights on the impact of qualifiers by analyzing the StarE model on the fully-inductive WD20K (66) V2 dataset. In particular, we study individual ranks for head/tail prediction of statements with and without qualifiers (cf. Fig. 2) varying the model from zero to four pairs. First, we group the test statements by the number of available qualifier pairs. We observe generally smaller ranks which, in turn, correspond to better predictions when more qualifier pairs are available. In particular, just one qualifier pair is enough to significantly reduce the individual ranks. Note that we have less statements with many qualifiers, cf. Appendix D.

We then study how particular qualifiers affect ranking and predictions. For that, we measure ranks of predictions for distinct statements in the *test set* with and without masking the qualifier relation from the inference graph \mathcal{S}_{inf} . We then compute ΔMR and group them by used qualifier relations (Fig. 3). Interestingly, certain qualifiers, e.g., **convicted of** or **including**, deteriorate the performance which we attribute to the usage of rare, qualifier-only entities. Conversely, having qualifiers like **replaces** reduces the rank by about 4000 which greatly improves prediction accuracy. We hypothesize it is an effect of qualifier entities: helpful qualifiers employ well-connected nodes in the graph which benefit from message passing.

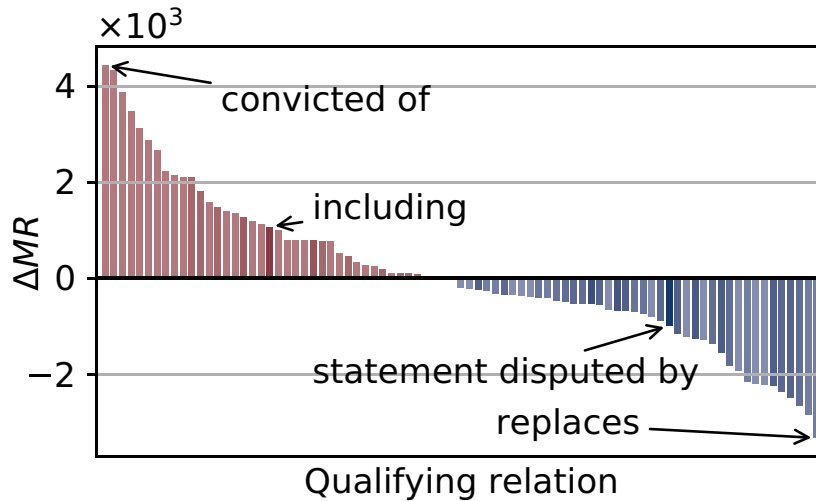


Fig. 3. Rank deviation when masking qualifier pairs containing a certain relation. Transparency is proportional to the occurrence frequency, bar height/color indicates difference in MR for evaluation statements using this qualifying relation if the pair is masked. More negative deltas correspond to better predictions.

Table 6. Top 3 worst and best qualifier relations affecting the overall mean rank (the last column). Negative ΔMR with larger absolute value correspond to better predictions.

WD20K (100) V1 FI		
Wikidata ID	relation name	ΔMR
P2868	subject has role	0.12
P463	member of	-0.04
P1552	has quality	-0.34
WD20K (66) V2 FI		
P805	statement is subject of	13.11
P1012	including	5.95
P812	academic major	5.07
P17	country	-19.96
P1310	statement disputed by	-20.92
P1686	for work	-56.87

Finally, we study the average impact of qualifiers on the whole graph, i.e., we take the whole *inference graph* and mask out all qualifier pairs containing one relation and compare the overall evaluation result on the test set (in contrast to Fig. 3, we count ranks of all test statements, not only those which have that particular qualifier) against the non-masked version of the same graph. We then sort relations by ΔMR and find top 3 most confusing and most helpful relations across two datasets (cf. Table 6). On the smaller WD20K (100) V1 where all statements have at least one qualifier pair, most relations tend to improve MR. For instance, qualifiers with the `distributed by` relations reduce MR by about 29 points. On the larger WD20K (66) V2 some qualifier relations, e.g., `statement is subject of`, tend to introduce more noise and worsen MR which we attribute to the increased sparsity of the graph given an already rare qualifier entity. That is, such rare entities might not benefit enough from message passing.

7 Related Work

We focus on semi- and fully inductive link prediction approaches and disregard classical approaches that are fully transductive, which have been extensively studied in the literature [2, 20].

In the domain of triple-only KGs, both settings have recently received a certain traction. One of the main challenges for realistic KG embedding is the impossibility of learning representations of unseen entities since they are not present in the train set.

In the semi-inductive setting, several methods alleviating the issue were proposed. When a new node arrives with a certain set of edges to known nodes, [1] enhanced the training procedure such that an embedding of an unseen node is a linear aggregation of neighbouring nodes. If there is no connection to the seen nodes, [27] propose to *densify* the graph with additional edges obtained from pairwise similarities of node features. Another approach applies a special meta-learning framework [4] when during training a meta-model has to learn representations decoupled from concrete training entities but transferable to unseen entities. Finally, reinforcement learning methods [8] were employed to learn relation paths between seen and unseen entities.

In the fully inductive setup, the evaluation graph is a separate subgraph disjoint with the training one, which makes trained entity embeddings even less useful. In such cases, the majority of existing methods [12, 13, 28, 29] resort to pre-trained language models (LMs) (e.g., BERT [15]) as *universal featurizers*. That is, textual entity descriptions (often available in KGs at least in English) are passed through an LM to obtain initial semantic node features. Nevertheless, mining and employing structural graph features, e.g., shortest paths within sampled subgraphs, has been shown [24] to be beneficial as well. This work is independent from the origin of node features and is able to leverage both, although the new datasets employ Sentence BERT [23] for featurizing.

All the described approaches operate on triple-based KGs whereas our work studies inductive LP problems on enriched, hyper-relational KGs where we show that incorporating such hyper-relational information indeed leads to better performance.

8 Conclusion

In this work, we presented a study of the inductive link prediction problem over hyper-relational KGs. In particular, we proposed a theoretical framework to categorize various LP tasks to alleviate an existing terminology discrepancy pivoting on two settings, namely, semi- and fully-inductive LP. Then, we designed WD20K, a collection of hyper-relational benchmarks based on Wikidata for inductive LP with a diverse set of parameters and complexity. Probing statement-aware models against triple-only baselines, we demonstrated that hyper-relational facts indeed improve LP performance in both inductive settings by a considerable margin. Moreover, our qualitative analysis showed that the achieved gains are consistent across different setups and still interpretable.

Our findings open up interesting prospects for employing inductive LP and hyper-relational KGs along several axes, e.g., large-scale KGs of billions statements, new application domains including life sciences, drug discovery, and KG-based NLP applications like question answering or entity linking.

In the future, we plan to extend inductive LP to consider unseen relations and qualifiers; tackle the problem of suggesting best qualifiers for a statement; and provide more solid theoretical foundations of representation learning over hyper-relational KGs.

Acknowledgements. 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 authors of this work take full responsibilities for its content.

A Training

In the sLCWA, negative training examples are created for each true fact $(h, r, t) \in KG$ by corrupting the head or tail entity resulting in the triples $(h', r, t)/(h, r, t')$. In the LCWA, for each triple $(h, r, t) \in KG$ all triples $(h, r, t') \notin KG$ are considered as non-existing, i.e., as negative examples.

Under the sLCWA, we trained the models using the margin ranking loss [9]:

$$L(f(t_i^+), f(t_i^-)) = \max(0, \lambda + f(t_i^-) - f(t_i^+)) \quad , \quad (2)$$

where $f(t_i^+)$ denotes the model’s score for a positive training example and $f(t_i^-)$ for a negative one.

For training under the LCWA, we used the binary cross entropy loss [14]:

$$\begin{aligned} L(f(t_i), l_i) = & - (l_i \cdot \log(\sigma(f(t_i))) \\ & + (1 - l_i) \cdot \log(1 - \sigma(f(t_i)))) \end{aligned} \quad (3)$$

where l_i corresponds to the label of the triple t_i .

B Hyperparameter Ranges

The following tables summarizes the hyper-parameter ranges explored during hyper-parameter optimization. The best hyper-parameters for each of our 46 ablation studies will be available online upon publishing.

C Infrastructure and Parameters

We train each model on machines running Ubuntu 18.04 equipped with a GeForce RTX 2080 Ti with 12 GB RAM. In total, we performed 46 individual hyperparameter optimizations (one for each dataset/model/number-of-qualifier combination). Depending on the exact configuration, the individual models have between 500k and 5M parameters and take up to 2 h for training.

D Qualifier Ratio

Figure 4 shows the ratio of statements with a given number of available qualifier pairs for all datasets and splits. We generally observe that there are only few statements with a large number of qualifier pairs, while most of them have zero to two qualifier pairs.

Table 7. Hyperparameter ranges explored during hyper-parameter optimization. FI denotes the fully-inductive setting and SI the semi-inductive setting. For the sLCWA training approach, we trained the models with the margin ranking loss (MRL), and with the LCWA we used the BCEL (Binary Cross Entropy loss)

Hyper-parameter	Value
GCN layers	{2,3}
Embedding dim.	{32, 64, ... , 256 }
Transformer hid. dim.	{512, 576, ... , 1024 }
Num. attention heads	{2, 4}
Num. transformer heads	{2, 4}
Num. transformer layers	{2, 3, 4}
Qualifier aggr.	{sum, attention}
Qualifier weight	0.8
Dropout	{0.1, 0.2, ... , 0.5 }
Attention slope	{0.1, 0.2, 0.3, 0.4 }
Training approaches	{sLCWA, LCWA}
Loss fcts.	{MRL, BCEL}
Learning rate (log scale)	[0.0001, 1.0)
Label smoothing	{0.1, 0.15}
Batch size	{128, 192, ... , 1024}
Max Epochs FI setting	1000
Max Epochs SI setting	600

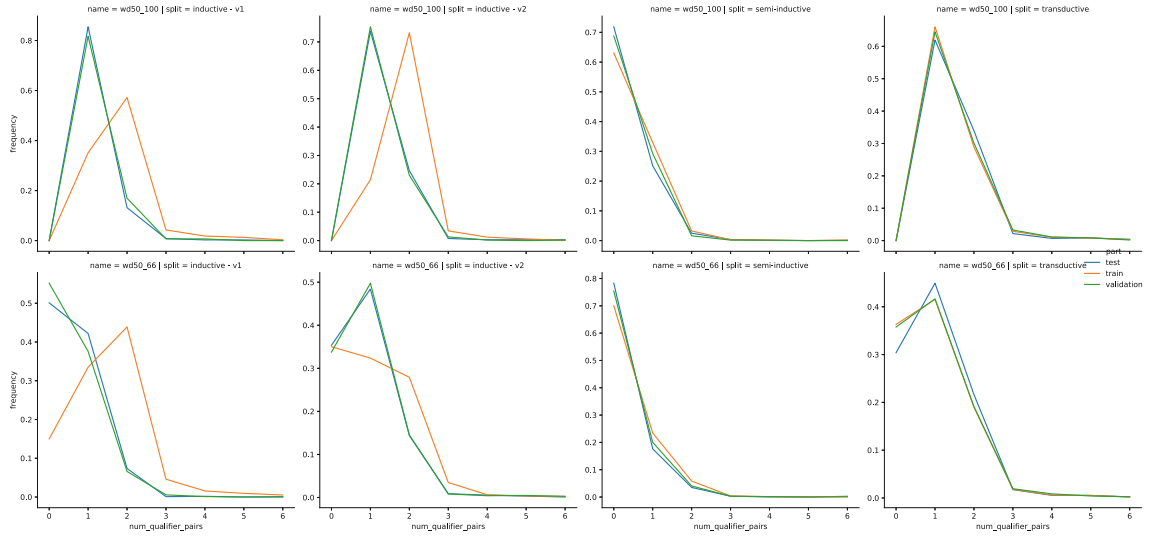


Fig. 4. Percentage of statements with the given number of available qualifier pairs for all datasets and splits.

References

1. Albooyeh, M., Goel, R., Kazemi, S.M.: Out-of-sample representation learning for knowledge graphs. In: Cohn, T., He, Y., Liu, Y. (eds.) *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16–20 November 2020*, pp. 2657–2666. Association for Computational Linguistics (2020)
2. Ali, M., et al.: Bringing light into the dark: a large-scale evaluation of knowledge graph embedding models under a unified framework. CoRR [arXiv:2006.13365](https://arxiv.org/abs/2006.13365) (2020)
3. Ali, M., et al.: PyKEEN 1.0: a python library for training and evaluating knowledge graph embeddings. *J. Mach. Learn. Res.* **22**(82), 1–6 (2021). <http://jmlr.org/papers/v22/20-825.html>
4. Baek, J., Lee, D.B., Hwang, S.J.: Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, 6–12 December 2020* (2020). Virtual
5. Bagherian, M., Sabeti, E., Wang, K., Sartor, M.A., Nikolovska-Coleska, Z., Najarian, K.: Machine learning approaches and databases for prediction of drug-target interaction: a survey paper. *Brief. Bioinform.* **22**(1), 247–269 (2020). <https://doi.org/10.1093/bib/bbz157>
6. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, Vancouver, British Columbia, Canada, 3–8 December 2001]*, pp. 585–591. MIT Press (2001)
7. Berrendorf, M., Faerman, E., Vermue, L., Tresp, V.: Interpretable and fair comparison of link prediction or entity alignment methods with adjusted mean rank. In: *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2020)*. IEEE (2020)
8. Bhowmik, R., de Melo, G.: Explainable link prediction for emerging entities in knowledge graphs. In: Pan, J.Z., et al. (eds.) *ISWC 2020. LNCS*, vol. 12506, pp. 39–55. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62419-4_3
9. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a Meeting Held 5–8 December 2013, Lake Tahoe, Nevada, United States*, pp. 2787–2795 (2013)
10. Bouritsas, G., Frasca, F., Zafeiriou, S., Bronstein, M.M.: Improving graph neural network expressivity via subgraph isomorphism counting. CoRR [arXiv:2006.09252](https://arxiv.org/abs/2006.09252) (2020)
11. Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., Murphy, K.: Machine learning on graphs: a model and comprehensive taxonomy. CoRR [arXiv:2005.03675](https://arxiv.org/abs/2005.03675) (2020)
12. Clouatre, L., Trempe, P., Zouaq, A., Chandar, S.: MLMLM: link prediction with mean likelihood masked language model (2020)

13. Daza, D., Cochez, M., Groth, P.: Inductive entity representations from text via link prediction (2020)
14. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: AAAI, pp. 1811–1818. AAAI Press (2018)
15. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, 2–7 June 2019, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics (2019)
16. Dong, X., et al.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: Macskassy, S.A., Perlich, C., Leskovec, J., Wang, W., Ghani, R. (eds.) The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, New York, NY, USA, 24–27 August, 2014, pp. 601–610. ACM (2014)
17. Galkin, M., Trivedi, P., Maheshwari, G., Usbeck, R., Lehmann, J.: Message passing for hyper-relational knowledge graphs. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, 16–20 November 2020, pp. 7346–7359. Association for Computational Linguistics (2020)
18. Gaudelet, T., et al.: Utilising graph machine learning within drug discovery and development. CoRR [arXiv:2012.05716](https://arxiv.org/abs/2012.05716) (2020)
19. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272. PMLR (2017)
20. Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P.S.: A survey on knowledge graphs: representation, acquisition and applications. CoRR [arXiv:2002.00388](https://arxiv.org/abs/2002.00388) (2020)
21. Liu, Y., et al.: RoBERTa: a robustly optimized BERT pretraining approach. CoRR [arXiv:1907.11692](https://arxiv.org/abs/1907.11692) (2019)
22. Nickel, M., Tresp, V., Kriegel, H.: A three-way model for collective learning on multi-relational data. In: Getoor, L., Scheffer, T. (eds.) Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, 28 June–2 July 2011, pp. 809–816. Omnipress (2011)
23. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2019). <https://arxiv.org/abs/1908.10084>
24. Teru, K., Denis, E., Hamilton, W.: Inductive relation prediction by subgraph reasoning. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13–18 July 2020, Virtual Event. Proceedings of Machine Learning Research, vol. 119, pp. 9448–9457. PMLR (2020)
25. Vashishth, S., Sanyal, S., Nitin, V., Talukdar, P.P.: Composition-based multi-relational graph convolutional networks. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. OpenReview.net (2020). https://openreview.net/forum?id=BylA_C4tPr
26. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
27. Wang, B., Wang, G., Huang, J., You, J., Leskovec, J., Kuo, C.J.: Inductive learning on commonsense knowledge graph completion. CoRR [arXiv:2009.09263](https://arxiv.org/abs/2009.09263) (2020)

28. Yao, L., Mao, C., Luo, Y.: KG-BERT: BERT for knowledge graph completion (2019)
29. Zhang, Z., Liu, X., Zhang, Y., Su, Q., Sun, X., He, B.: Pretrain-KGE: learning knowledge representation from pretrained language models. In: Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16–20 November 2020, pp. 259–266. Association for Computational Linguistics (2020)

Appendix E

BioKEEN: a library for learning and evaluating biological knowledge graph embeddings

Data and text mining

BioKEEN: a library for learning and evaluating biological knowledge graph embeddings

Mehdi Ali^{1,*}, Charles Tapley Hoyt^{2,3}, Daniel Domingo-Fernández^{2,3},
Jens Lehmann^{1,4} and Hajira Jabeen¹

¹Department of Computer Science, Rheinische Friedrich-Wilhelms-Universität Bonn, Bonn 53113, Germany, ²Department of Life Science Informatics, Bonn-Aachen International Center for IT, Rheinische Friedrich-Wilhelms-Universität Bonn, Bonn 53113, Germany, ³Department of Bioinformatics, Fraunhofer Institute for Algorithms and Scientific Computing (SCAI), Sankt Augustin 53754, Germany and ⁴Department of Enterprise Information Systems, Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), Sankt Augustin 53754, Germany

*To whom correspondence should be addressed.

Associate Editor: Jonathan Wren

Received on November 20, 2018; revised on January 31, 2019; editorial decision on February 10, 2019; accepted on February 13, 2019

Abstract

Summary: Knowledge graph embeddings (KGEs) have received significant attention in other domains due to their ability to predict links and create dense representations for graphs' nodes and edges. However, the software ecosystem for their application to bioinformatics remains limited and inaccessible for users without expertise in programming and machine learning. Therefore, we developed BioKEEN (Biological KnowlEdge EmbeddiNgs) and PyKEEN (Python KnowlEdge EmbeddiNgs) to facilitate their easy use through an interactive command line interface. Finally, we present a case study in which we used a novel biological pathway mapping resource to predict links that represent pathway crosstalks and hierarchies.

Availability and implementation: BioKEEN and PyKEEN are open source Python packages publicly available under the MIT License at <https://github.com/SmartDataAnalytics/BioKEEN> and <https://github.com/SmartDataAnalytics/PyKEEN>

Contact: mehdi.ali@cs.uni-bonn.de

Supplementary information: [Supplementary data](#) are available at *Bioinformatics* online.

1 Introduction

Knowledge graphs (KGs) are multi-relational, directed graphs in which nodes represent entities and edges represent their relations (Bordes *et al.* 2013). While they have been successfully applied for question answering, information extraction and named entity disambiguation outside of the biomedical domain, their usage in biomedical applications remains limited (Su *et al.*, 2018).

Because KGs are inherently incomplete and noisy, several methods have been developed for deriving or predicting missing edges (Nickel *et al.*, 2016). One method is to apply reasoning based on formal logic to derive missing edges, but it usually requires a large set of user-defined formulas to achieve generalization. Another method is to train KG embeddings (KGEs; low-dimensional vector/matrix representations of entities and relations whose elements

correspond to latent features of the KG) that best preserve the structural characteristics of the KG and then predict new edges using their respective KGE models (Wang *et al.*, 2017).

In a biological setting, relation prediction not only enables researchers to expand their KGs, but also to generate new hypotheses that can be tested experimentally.

Here, we present BioKEEN (Biological KnowlEdge EmbeddiNgs): a Python package for training and evaluating KGEs on biological KGs that is accessible and facile for bioinformaticians without expert knowledge in machine learning through an interactive command line interface (CLI). Through the integration of the Bio2BEL software (<https://github.com/bio2bel>) within BioKEEN, numerous biomedical databases containing structured knowledge are directly accessible. Additionally, we have externalized BioKEEN's core machine learning components for training and evaluating KGE models in an

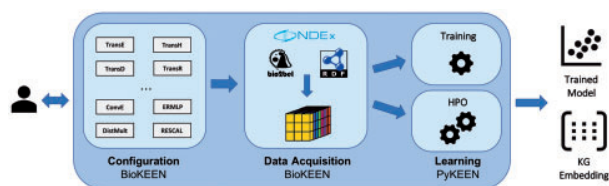


Fig. 1. Software architecture of BioKEEN (i) **Configuration:** Users define experiments through the CLI. (ii) **Data Acquisition:** Dataset(s) are (down-) loaded and transformed into a tensor. (iii) **Learning:** The KGE model is trained with user-defined hyper-parameters or a hyper-parameter search is applied to find the best set of hyper-parameter values. The functionality of this layer is externalized in the Python KnowlEdge EmbeddINgs package

independent Python package, PyKEEN, such that they can be reused in other domains (see Fig. 1).

Although there exists other toolkits like OpenKE (Han *et al.*, 2018) and scikit-kge (<https://github.com/mnick/scikit-kge>), they are not specialized for bioinformatics applications and require more expertise in programming and in KGEs. To the best of our knowledge, BioKEEN is the first framework specifically designed to facilitate the use of KGE models for users in the bioinformatics community.

2 Software architecture

The BioKEEN software package consists of three layers: (i) the model configuration layer, (ii) the data acquisition and transformation layer and (iii) the learning layer (see Fig. 1).

2.1 Configuration layer

Because every KGE model has its own set of hyper-parameters, the configuration of an experiment for a non-expert can be very complicated and discouraging. This possible obstacle is addressed in the configuration layer through an interactive CLI that assists users in setting up their experiments (i.e. defining the datasets, the model, and its parameters). Based on the configuration, BioKEEN builds a machine learning pipeline containing the appropriate components (e.g. data acquisition, training, evaluation and prediction).

Currently, we provide implementations of 10 embedding models (e.g. TransE, TransH, ConvE etc.; Dettmers *et al.*, 2017; Wang *et al.*, 2017). A full list can be found in Supplementary Table S1. Moreover, BioKEEN can be executed in training and hyper-parameter optimization (HPO) mode.

2.2 Data acquisition layer

Because extracting and preparing training data can be a time-consuming process, BioKEEN integrates the Bio2BEL software to download and parse numerous biomedical databases (Supplementary Table S2). This allows users to focus on the experiments, to automatically incorporate the latest database versions, and to have access to new datasets as they are incorporated into Bio2BEL. In addition, users can provide their own datasets as tab-separated values, RDF or from NDEX (Pratt *et al.*, 2015). BioKEEN processes the selected and provided datasets then transforms them into a tensor (i.e. a multi-dimensional matrix) for further processing.

2.3 Learning layer

Determining the appropriate values for the hyper-parameters of a KGE model requires both machine learning and domain specific knowledge. If the user specifies hyper-parameters, BioKEEN can be run directly in *training mode*. Otherwise, it first runs in *HPO mode*, where *random search* is applied to find suitable hyper-parameters values from (user

predefined sets. We implemented *random search* instead of the widely applied *grid search* because it converges faster to appropriate hyper-parameter values (Goodfellow *et al.* 2016). Finally, the user can run BioKEEN in *training mode* with the resulting hyper-parameter values.

To train the models, negative training examples are generated based on the algorithm described in Bordes *et al.* To evaluate the trained models, BioKEEN computes two common evaluation metrics for KGE models: mean rank and hits@k.

3 Application

We used BioKEEN to train and evaluate several KGE models on the pathway mappings from ComPath (Domingo-Fernández *et al.*, 2019), the first manually curated intra- and inter-database pathway mapping resource that bridges the representations of similar biological pathways in different databases. Then, we used the best model to predict new relations representing pathway crosstalks and hierarchies. After removing reflexive triplets, we found that the highest ranked novel equivalence between TGF-beta Receptor Signaling (wikipathways: WP560) and TGF-beta signaling pathway (kegg: hsa04350), as well as the highest ranked hierarchical link that Lipoic acid (kegg: hsa00785) is a part of Lipid metabolism (reactome: R-HSA-556833) both represented novel pathway crosstalks. Upon manual evaluation, each fulfilled the ComPath curation criteria and can be added to the resource.

We performed HPO for five different models to illustrate the need for choosing the appropriate hyper-parameter values. For the TransE model, comparing the hyper-parameters similar to those reported by Bordes *et al.* with the hyper-parameters from HPO showed an improvement in the hits@10 metric from 19.10 to 63.20%.

Moreover, the nature of the model strongly influences the results. We found that the simpler models (e.g. TransE, UM and DistMult) performed similar or even better than the more complex ones (e.g. TransH and TransR). This might be explained by the fact that the more expressive models overfit since ComPath is a not a large dataset. Ultimately, this case scenario illustrates the ability of BioKEEN to assist users in finding reasonable combinations of models and their hyper-parameter values to predict novel links.

4 Discussion and future work

Although BioKEEN already includes several models and components to build machine learning pipelines, it has limitations that could benefit from several additions and improvements.

Modeling multiscale biology (i.e. the -omics, pathway, phenotype and population levels) results in KGs with a variety of compositions, structural features, and topologies for which different KGE models that have not yet been included in BioKEEN may be more appropriate. Further, because of the heterogeneity and lack of structure in most biological and clinical data, we plan to implement additional KGE models that incorporate text, logical rules, and images in addition to the triples in KGs (Hamilton *et al.* 2018; Wang *et al.*, 2017).

The negative sampling approach described by Bordes *et al.* included in BioKEEN is prone to false negatives. We plan to mitigate them by incorporating prior biological knowledge and constraints to generate triples guaranteed to be true negatives such as: (i) type constraints for predicates (e.g. the relation *transcribed* is only valid from gene to protein), (ii) valid attribute range for predicates (e.g. protein weight is below 1000kDa) and (iii) functional

constraints such as mutual exclusion (e.g. a protein is coded by one gene) (Nickel et al., 2016).

Although BioKEEN assists in HPO, it does not provide assistance in selecting a particular KGE model, which is an obscure process even for machine learning experts. We plan to address this by implementing KG analyses with rule-based suggestions (e.g. DistMult performs poorly for KGs with antisymmetric relations).

Finally, we plan to present this software as a web application to enable a wider audience of researchers who may not be comfortable with scripting or CLIs to train and evaluate KGE models.

Acknowledgements

We thank our partners from the Bio2Vec, MLwin and SimpleML projects for their assistance.

Funding

This research was supported by Bio2Vec project (<http://bio2vec.net/>, CRG6) grant [3454] with funding from King Abdullah University of Science and Technology (KAUST).

Conflict of Interest: none declared.

References

- Bordes, A. et al. (2013) *Translating embeddings for modeling multi-relational data*. In: Burges, C.J.C. et al. (eds) *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pp. 2787–2795, <http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf>.
- Dettmers, T. et al. (2017) Convolutional 2d knowledge graph embeddings. *arXiv Preprint arXiv*, 1707, 01476.
- Domingo-Fernández, D. et al. (2019) ComPath: an ecosystem for exploring, analyzing, and curating pathway databases. *NPJ Syst. Biol. Appl.*, 5, 3.
- Goodfellow, I. et al. (2016) *Deep Learning*. Vol. 1. MIT Press, <http://www.deeplearningbook.org>.
- Hamilton, W. et al. (2018) Embedding logical queries on knowledge graphs. *arXiv Preprint arXiv*, 1806, 01445.
- Han, X. et al. (2018) OpenKE: an open toolkit for knowledge embedding. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. pp. 139–144.
- Nickel, M. et al. (2016) A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104.1, 11–33.
- Pratt, D. et al. (2015) NDEx, the network data exchange. *Cell Syst.*, 1, 302–305.
- Su, C. et al. (2018) Network embedding in biomedical data science. *Brief. Bioinformatics*, bby117.
- Wang, Q. et al. (2017) Knowledge graph embedding: a survey of approaches and applications. *IEEE Trans. Knowledge Data Eng.*, 29, 2724–2743.

Appendix F

CLEP: a hybrid data- and knowledge-driven framework for generating patient representations

Systems biology

CLEP: a hybrid data- and knowledge-driven framework for generating patient representations

Vinay Srinivas Bharadhwaj^{1,2}, Mehdi Ali ^{3,4}, Colin Birkenbihl^{1,2}, Sarah Mubeen^{1,2,5}, Jens Lehmann^{3,4}, Martin Hofmann-Apitius^{1,2}, Charles Tapley Hoyt ^{1,3,5} and Daniel Domingo-Fernández ^{1,3,5,*}

¹Department of Bioinformatics, Fraunhofer Institute for Algorithms and Scientific Computing, 53757 Sankt Augustin, Germany, ²Bonn-Aachen International Center for Information Technology (B-IT), University of Bonn, 53115 Bonn, Germany, ³Rheinische Friedrich-Wilhelms-Universität Bonn, 53113 Bonn, Germany, ⁴Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), Dresden and Sankt Augustin, Germany and ⁵Fraunhofer Center for Machine Learning, Bonn, Germany

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on November 5, 2020; revised on March 29, 2021; editorial decision on April 29, 2021; accepted on May 3, 2021

Abstract

Summary: As machine learning and artificial intelligence increasingly attain a larger number of applications in the biomedical domain, at their core, their utility depends on the data used to train them. Due to the complexity and high dimensionality of biomedical data, there is a need for approaches that combine prior knowledge around known biological interactions with patient data. Here, we present CLinical Embedding of Patients (CLEP), a novel approach that generates new patient representations by leveraging both prior knowledge and patient-level data. First, given a patient-level dataset and a knowledge graph containing relations across features that can be mapped to the dataset, CLEP incorporates patients into the knowledge graph as new nodes connected to their most characteristic features. Next, CLEP employs knowledge graph embedding models to generate new patient representations that can ultimately be used for a variety of downstream tasks, ranging from clustering to classification. We demonstrate how using new patient representations generated by CLEP significantly improves performance in classifying between patients and healthy controls for a variety of machine learning models, as compared to the use of the original transcriptomics data. Furthermore, we also show how incorporating patients into a knowledge graph can foster the interpretation and identification of biological features characteristic of a specific disease or patient subgroup. Finally, we released CLEP as an open source Python package together with examples and documentation.

Availability and implementation: CLEP is available to the bioinformatics community as an open source Python package at <https://github.com/hybrid-kg/clep> under the Apache 2.0 License.

Contact: daniel.domingo.fernandez@scai.fraunhofer.de

Supplementary information: [Supplementary data](#) are available at *Bioinformatics* online.

1 Introduction

Recent advancements in machine learning (ML) and artificial intelligence (AI) methodologies have initiated a paradigm shift in bioinformatics. As new technologies have steadily generated large volumes of -omics data, AI methods have garnered great insights into human health and biology. With the availability of large-scale biological datasets, ML/AI are becoming highly relevant for biomedical applications, such as predictive modeling, patient stratification and simulation (Fröhlich *et al.*, 2018). However, despite the successful application of ML/AI in the biomedical domain, the datasets underlying the generation of models can play a far more crucial role in a

given application than the complexity of the model itself. For example, in some cases, if data is predictive enough, simpler methods can outperform state-of-the-art ML/AI methods in prediction tasks (Lynam *et al.*, 2020; Smith *et al.*, 2020).

The development of novel high-throughput experimental techniques has led to a broad availability of biological data from multiple entity types (Zitnik *et al.*, 2019). In practice, integrating multiple data types can be advantageous, particularly in the context of complex diseases, where no single type of data can effectively explain the cause of dysfunction. However, biological datasets tend to be both inherently complex and noisy (Fan *et al.*, 2014), making their integration challenging. Furthermore, biological data typically

contain a far greater number of features than samples due to several factors, including a lack of available resources and obstacles in sample collection (Xu and Jackson, 2019). In failing to address these challenges and generating comprehensive representations of biological data, novel techniques can suffer in a range of analytic tasks.

One approach adopted by the systems biology community is in representing biological data in the form of networks. By doing so, multiple scales of biology can be represented, as can the relationships within and across these scales. These networks are also advantaged by their ability to integrate heterogeneous biological data types (Hu et al., 2017). Generally, one can classify biological networks into two categories depending on the source of information used to generate them (Yu et al., 2013).

The first of these two classes of networks are constructed from biological data by using a variety of methodologies. For instance, co-expression networks can be generated from transcriptomics data to represent pairwise correlations between genes (Langfelder and Horvath, 2008). Another example is Bayesian networks which can be used to model conditional interdependencies across heterogeneous biological entities (Khanna et al., 2018). While the majority of these methods transform biological data into networks comprising relations between the biological entities under study, other methods directly translate patient-level data into networks through correlation techniques (e.g. Pearson correlation) as an indicator of similarity between patients. These patient similarity networks closely link patients that are more similar to each other while less similar patients contain fewer close connections. Patient similarity networks have been successfully used to represent multimodal patient level-data for clustering tasks (Cavalli et al., 2017; Pai et al., 2019; Pai and Bader, 2018; Raphael et al., 2017; Wang et al., 2014a). While so far, these methods have been mostly employed for clustering tasks based on patient similarity networks, (Pai et al., 2019) recently expanded the concept to incorporate gene-level measurements into pathways (via gene sets) in order to reduce dimensionality and ultimately use pathway features for patient classification.

A second class of networks can be generated from prior knowledge of known interactions between biological entities. When sets of these interactions are assembled, they can be used to represent discrete biological networks. These networks can be referred to as knowledge graphs (KGs) when they comprise entities from various biological modalities and the complex interactions between them. However, despite their advantages, networks cannot be directly represented in vector space; thus, impeding their direct use by ML/AI techniques. This impediment has led to the development of methodologies (i.e. Knowledge Graph Embedding Models; KGEMs) designed to encode entities and relations in a KG into a latent feature space while preserving its structure. While these new representations can be used for entity disambiguation, clustering and several downstream ML/AI tasks, they have been primarily used in the biomedical domain for link prediction tasks, including the prediction of side effects (Zitnik et al., 2018), disease-gene associations (Himmelstein and Baranzini, 2015) and novel therapeutic targets (Muslu et al., 2020). Recently, the notion of adding patients into KGs has been proposed from Electronic Health Records (EHRs) (i.e. MIMIC-III dataset) with the ultimate goal of generating patient embeddings through KGEMs that can be used for clustering (Gong et al., 2021) and medicine recommendation tasks (Lin et al., 2020). However, until now, there have yet to be integrative approaches which incorporate both patient-similarity networks and KGs from non-textual patient-level data, such as multi-omics, and prior knowledge.

In this work, we introduce CLinical Embedding of Patients (CLEP), a hybrid data- and knowledge-driven framework that exploits -omics patient-level data and incorporates this information into a KG. Once the KG has been generated by CLEP, the framework then drives the generation of novel patient representations through various KGEMs. In building upon previous data-driven approaches (Pai et al., 2019; Wang et al., 2014a) which have demonstrated the advantages of representing patients as nodes in a network, we show that additionally integrating prior knowledge can make for more robust analyses. We showcase our approach by

generating new patient representations derived from two different transcriptomics datasets and a KG comprising heterogeneous protein-protein interactions from multiple biological databases. Using the new patient representations, we find that performance on a panel of ML models trained to classify between patients and controls is significantly improved with respect to the use of original data. Furthermore, the flexibility of our approach makes it applicable to heterogeneous multimodal biological datasets, enabling researchers to generate new patient representations by combining patient-level data with the prior knowledge contained in a KG. Finally, we have made CLEP available to the bioinformatics community as an open source Python package (<https://github.com/hybrid-kg/clep>) under the Apache 2.0 License.

2 Materials and methods

2.1 Framework description

Figure 1 illustrates each step of the presented framework. The methodology requires a patient-level dataset and a KG. Patients are incorporated into the KG as new nodes and connected to features that most closely characterize a given patient. Once patients are embedded into the KG, KGEMs are then used to generate new patient representations, learnt from both prior knowledge underlying the KG and the patient-feature connections that have been generated. Finally, these novel patient representations can subsequently be employed for a variety of downstream applications, as we demonstrate in the case scenarios section.

2.1.1 Input data

Our framework requires two inputs: a patient-level dataset and a KG (Fig. 1a). It can be applied to any dataset and KG so long as the dataset features can be mapped to nodes in the KG. In other words, if we intend to use CLEP on a transcriptomics dataset such as RNA-Seq, the KG must contain relationships between genes/proteins that are mappable to the transcripts that have been measured. We would like to note that although the framework does not require that all features in the dataset are also present in the KG, it is recommended to maximize this overlap.

2.1.2 Incorporating patients into the knowledge graph

The first step of the methodology consists of incorporating patients as nodes in the KG in order to subsequently generate novel individualized feature representations for each patient (i.e. embeddings) through the use of KGEMs (Fig. 1b). These models generate the embeddings by exploiting the topology of the KG. Generating comprehensive embeddings requires an approach that can position a patient node in the KG according to that patient's most informative features (i.e. features that distinguish them from patients who possess clinically different features). This way, patient nodes with similar features would be close together in a network and would thus have similar embeddings, while patients with dissimilar features would be farther away (see example in Fig. 4b). On the other hand, connecting every patient to a large number of nodes in the KG instead of a smaller node set that represents a patient's most relevant features, would result in poor patient representations due to large overlaps of irrelevant features. The positioning of the patient nodes in the KG can be seen as a feature selection technique in which our method identifies the most characteristic features of each patient in order to generate edges between the patient and these features, thus, embedding the patient as a node in the KG. The rationale behind this method is that by connecting patients with their most characteristic features, we are able to generate a KG that integrates patient-specific information (i.e. relations between patients and features) together with prior knowledge (i.e. relations among these features).

To identify the most characteristic features and incorporate patients into the KG, our methodology leverages the empirical cumulative distribution function (eCDF) based on the quantitative measurements for each feature in the dataset which can be mapped to the KG. Using all samples in the dataset or exclusively the healthy

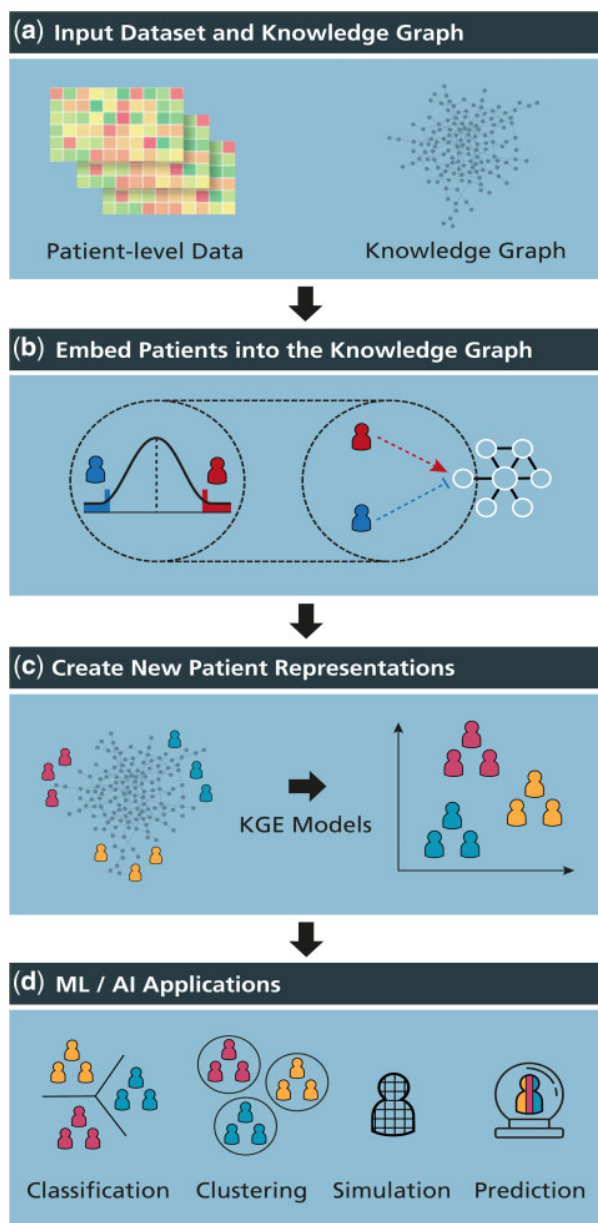


Fig. 1. Schematic illustration of the framework. (a) CLEP requires two inputs: (i) a patient-level dataset such as multi-omics, and (ii) a KG comprising relations between features measured in the previously mentioned dataset. (b) Using one of the proposed methods, CLEP incorporates patients into the KG by connecting them to their most distinctive features in the dataset. (c) KGEMs are then used to generate new patient representations based on both data- and knowledge-driven features. (d) These patient representations can subsequently be used for several downstream tasks, such as patient classification and stratification. A high quality version of this figure is available at <https://doi.org/10.6084/m9.figshare.12834605>

controls data, we generate a reference distribution with data points that represent the measurements of each feature. We then identify patients that fall at the extreme ends of this distribution based on a predetermined threshold (e.g. 5% of the eCDF) (Fig. 1b). Finally, these patients are connected to the node in the KG that represents that particular feature with an edge (i.e. -1 or +1) depending on the extreme in which the patient falls. The sign of the edge indicates whether a particular patient has a higher or lower measurement for that given feature to differentiate the patient from other patients in the distribution.

In the [Supplementary Text](#), we present alternative methodologies that can similarly be used to incorporate patients to the KG. However, we have focused here on one specific method as it can be applied to any type of patient-level data and it does not make any assumptions on the feature distribution as opposed to other methods which either require pathway information or assume that the features are normally distributed.

2.1.3 Generating new patient representations

Once patients have been incorporated into the KG, the next step involves generating new patient representations through KGEMs (Fig. 1c). We restricted ourselves to KGEMs since they consider both directionality and edge types, and our KG contains several directed edge types. Incorporating edge types during the learning process can help the model to differentiate between node types (e.g. patients and biological entities) and node sub-types (e.g. diseased patients, and controls). CLEP has adopted PyKEEN (Ali *et al.*, 2021) as the KGEM-software due to its wide range of functionalities (e.g. a large number of KGEMs, hyperparameter optimization functionalities).

2.1.4 Applications of patient representations

Once novel patient representations have been generated with CLEP, they can be used for a vast number of applications including classification and clustering tasks (Fig. 1d). Because our approach leverages prior knowledge of known interactions, we hypothesize that the use of these representations can yield superior performances on these tasks with respect to the use of the original patient-level datasets from which they were generated. Furthermore, our methodology facilitates biologically meaningful interpretations of patient-level data by positioning patients into different KG neighborhoods which may potentially correspond to biological processes that are characteristic for specific patient subgroups.

2.2 Software implementation

CLEP is implemented as a Python package to facilitate its usage within the scientific community. It contains several workflows corresponding to each of the steps presented in the methods from generating new patient representations to conducting downstream applications presented in the case scenario. Each workflow is both accessible through a command line interface (CLI) as well as programmatically, allowing users to input their own patient-level datasets and custom KGs. In total, CLEP offers three different methods for incorporating patients into the KG, all KGEMs available through PyKEEN (Ali *et al.*, 2021), and five ML classifiers. Furthermore, thanks to its flexible implementation, users can independently use each of its modules as well as incorporate classifiers tasks into the framework (Supplementary Fig. S2). Finally, the source code of the CLEP Python package is available at <https://github.com/hybrid-kg/clep> under the Apache 2.0 License, its latest documentation can be found at <https://clep.readthedocs.io>, and it is distributed via PyPI at <https://pypi.org/project/clep>.

2.3 Case scenarios

2.3.1 Patient-level data

The first dataset, the Alzheimer's Disease Neuroimaging Initiative (ADNI) (Mueller *et al.*, 2005), is one of the world's largest dementia cohorts and, with more than 1300 citations, the most referential resource for data-driven dementia research. In this work, we used the blood plasma transcriptomic data collected in the study [we refer to Saykin *et al.* (2015) and <http://adni.loni.usc.edu> for details]. The dataset is already preprocessed and contains a total of 260 cognitively healthy control participants, 215 patients with early mild cognitive impairment, 225 patients with late mild cognitive impairment and 44 patients with Alzheimer's disease. To conduct the binary classification task, the latter three (i.e. all cognitively impaired patients) were grouped together into a single class ($n=494$). Preprocessed gene expression data [Robust Multichip Average (RMA) normalized data] was directly used as a baseline for the

benchmarking of CLEP. To incorporate the patients into the KG, the expression of genes whose transcripts appear multiple times were considered individually.

The second dataset is a transcriptomics dataset containing samples from three psychiatric disorders (i.e. major depressive disorder, schizophrenia and bipolar disorder) and healthy controls (Hagenauer et al., 2018). In total, this dataset contains 172 samples (41 major depressive disorder, 22 schizophrenia, 26 bipolar disorder and 83 control samples). Similar to the ADNI dataset, the preprocessed gene expression data (RMA normalized data) from the second dataset was directly used as a baseline for the validation of CLEP. Similar to the previous dataset, we grouped the three psychiatric indications together to conduct a binary classification task.

2.3.2 Knowledge graph

For the case scenario, we employ a KG referred to as PPI-KG, which comprises protein–protein interactions from six resources: PathMe (Domingo-Fernández et al., 2019) [which includes KEGG (Kanehisa et al., 2017) Reactome (Jassal et al., 2020) and WikiPathways (Slenter et al., 2018)], BioGrid (Oughtred et al., 2019), IntAct (Orchard et al., 2014) and Pathway Commons (Rodchenkov et al., 2020) (Supplementary Text).

2.3.3 Generating representations of patients

We used different thresholds (i.e. the 1%, 1.5%, 2.5%, 5% 10% and 20% quantiles of the eCDF) to define the tail of the distribution (i.e. extreme) which is required to incorporate ADNI patients and the patients of the GSE92538 dataset to the PPI-KG using the eCDF-based method described in Section 4.1. By applying this method on each threshold, we generated different KGs (i.e. one for each threshold) by connecting the patients that fall in the extremes of the reference distributions (Supplementary Table S4). Reference distributions were generated based on the expression values of healthy controls for each gene that was present in both the transcriptomics data and the PPI-KG (Supplementary Fig. S3). The vast majority of nodes in the PPI-KG were among the proteins measured in the transcriptomics dataset, resulting in 8085 mapped proteins in total.

We would like to note that the resulting KG (i.e. PPI-KG after incorporating the ADNI patients) must be split into three triple sets (i.e. training, validation and testing set) in order to train a KGEM. Thus, we developed an algorithm that splits a given KG in a way that relations and nodes are balanced across the three different splits. The pseudocode for this algorithm can be found in the Supplementary Figure S1 and is implemented in the CLEP Python package.

2.3.4 Selected knowledge graph embedding models

We selected RotatE (Sun et al., 2019), TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016) and HolE (Nickel et al., 2016) to learn the patient embeddings, because they have been shown to be effective in a large-scale benchmarking study for KGEMs (Ali et al., 2020). To train the final models, we used the best hyperparameters obtained by the hyperparameter optimization (Supplementary Table S3).

2.3.5 Classifying between cognitively impaired and healthy controls

The new representations generated from KGEMs were used to classify between normal and cognitively impaired patients (i.e. AD and MCI) using five different statistical modeling and ML methods (Table 1).

Prediction performance was evaluated via 5 times repeated 5-fold cross-validation in which the hyperparameters of the model were tuned within the cross-validation loop via a grid search (Fig. 2). The cross-validation prediction performance for the binary classification task was evaluated using the area under the receiver operating characteristic curve (AUC-ROC) or the area under the precision–recall curve (AUC-PR) as a metric.

We then conducted three experiments to evaluate the robustness of our results. First, we trained the ML models using the new representations generated from RotatE with the 5% threshold while permuting patient labels (i.e. y-scrambling). Second, we trained the ML models on patient representations again generated using RotatE with a 5% threshold but using instead a permuted version of the KG generated using the XSwap algorithm (Hanhijärvi et al., 2009) in order to investigate if a KG's topology makes meaningful improvements to performance. By using this algorithm, we ensured that the permuted versions preserved the original structure of the original network (i.e. each node has the same number of in- and out-edges) but edges were randomly generated. Third, we trained the ML models using a subset of the PPI-KG corresponding to a single database (i.e. either WikiPathways or KEGG) in order to investigate the importance of KG size and completeness.

2.3.6 Classifying between psychiatric disorders and healthy controls

Using the same setting used in the previous dataset (Fig. 2), we trained the same five ML models to classify between normal samples and patients with a psychiatric disorder (i.e. bipolar disorder, major depressive disorder and schizophrenia) using both the raw data and the new representations of the GSE92538 data generated by CLEP. Similar to the previous dataset, we used RotatE KGEM and a threshold value of 2.5% on either side of the control distribution for generating the new representations.

3 Results

3.1 CLEP's representations outperform raw data in classifying cognitively impaired patients and healthy controls

Here, we present the results of our methodology and demonstrate how CLEP can improve the performance of ML models on patient classification tasks within the context of Alzheimer's disease (AD). We incorporated ADNI patients (Mueller et al., 2005) into a protein–protein interaction KG (i.e. PPI-KG) in order to generate novel patient representations using various KGEMs (see Section 2). We then compared the performance of several ML models in distinguishing between cognitively impaired patients from controls depending on whether the input data was the original transcriptomics data or novel patient representations. We summarize the performance of each of the five ML models in Figure 3a and b.

Our results show that using original transcriptomics data as input for the binary classifier leads to relatively low prediction power (Fig. 3b), as opposed to the new representations generated by CLEP which substantially increase prediction performance (Fig. 3a). As an illustration, the SVM model, which was the highest performing, yielded area under the receiver operator characteristic curve (AUC-ROC) values ranging from 0.48 to 0.64 when trained on the transcriptomics data. In comparison, the same model, when trained on the new patient representations, yielded AUC-ROC values between 0.83 and 0.91. This difference was consistent across each of the five ML models, demonstrating how CLEP generates patient representations that significantly outperform the original data for this particular binary classification task. However, it is worth noting that the performance of the two tree-based methods (i.e. random forest and XGBoost) is significantly worse than the remaining models, yet they still outperformed their corresponding counterpart models trained on the raw data. Furthermore, to validate that the results are not an effect of the class imbalance between cognitively impaired patients and healthy controls (Saito and Rehmsmeier, 2015), we reevaluated the ML models using the area under the precision–recall curve (AUC-PR) as a metric, which resulted in similar results (Supplementary Fig. S6). Finally, we would like to note that through the usage of the KGEMs, the dimension of the patient representations could significantly be compressed since the input dimension of the raw transcriptomics was larger than 40 000 features, while the representations generated by the KGEMs had a dimension of 256.

Table 1. List of statistical modeling and machine learning methods available at CLEP for conducting classification tasks

Model	Reference	Implementation
Logistic regression with L2 regularization	–	scikit-learn (Pedregosa <i>et al.</i> , 2011)
Logistic regression with elastic net regularization	Zou and Hastie (2005)	scikit-learn (Pedregosa <i>et al.</i> , 2011)
Support Vector Machines	Cortes and Vapnik (1995)	scikit-learn (Pedregosa <i>et al.</i> , 2011)
Random Forest	Ho (1995)	scikit-learn (Pedregosa <i>et al.</i> , 2011)
XGBoost	Chen and Guestrin (2016)	XGBoost (Chen and Guestrin, 2016)

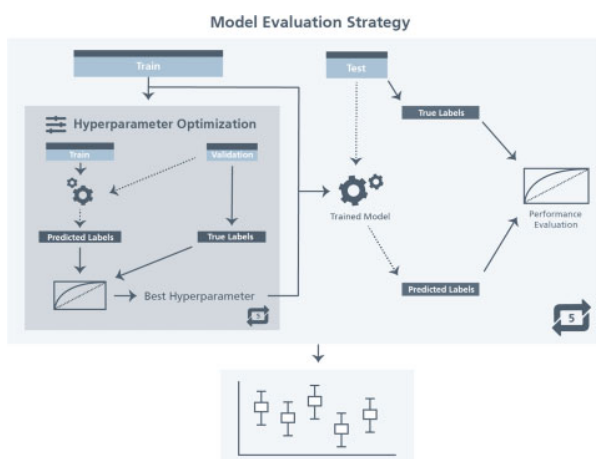


Fig. 2. Schematic representation of the model evaluation strategy. The first step is to split the data into training (80%) and hold-out test set (20%). Next, we performed 5-fold stratified cross-validation by further repeatedly splitting the training data into 80% training and 20% validation in order to identify the best hyperparameter settings. Once these five cross-validation rounds were performed, the best hyperparameters were used to train the model on the full training data and the trained model was evaluated on the held-out test set. This entire strategy was repeated 5 times to generate 5 AUC-ROC scores that were used to determine the performance of each of the five models (Table 1). We would like to note that the first split (training-test) ensured maintaining the overall class distribution and classes have been shuffled beforehand to avoid having the same data used in training/test. Furthermore, we chose AUC-ROC scores as evaluation metrics for both best hyperparameter selection and model evaluation. A high quality version of this figure is available at <https://doi.org/10.6084/m9.figshare.12834608>

To investigate the robustness of our approach, we conducted two independent experiments. The first experiment consisted of training each of the five classifiers using new representations with randomized patient labels (Fig. 3c) in order to confirm that the models did not fit to arbitrary artifacts in the data. In contrast, in the second experiment, patient representations were generated on a permuted version of the original KG (see Section 2 for details) to ensure that new representations reflect the information encoded in the KG by generating patient representations on a permuted version of the original KG (Fig. 3d). The results of these experiments yielded models with a performance equivalent to a random classifier (i.e. AUC-ROC values ~ 0.5); thus, confirming (i) the robustness of our model evaluation strategy and (ii) that the new representations are driven by information in the KG.

While we were able to show how CLEP successfully generated novel representations that yield superior prediction performance, the process of generating these representations is non-trivial. As an initial step, a threshold that determines which patient-measurable edges will be incorporated into the KG must first be selected as this parameter influences the KG generated (see Section 4.1). We demonstrate the effect of the threshold on the previous binary classification task by training a single classifier using KGs derived from different thresholds (Supplementary Fig. S4). Unsurprisingly, while lower threshold settings (i.e. from 1% to 5%) resulted in increased predictive power, higher thresholds (i.e. 10% and 20%) penalized the performance of the model. This can be attributed to the fact that a low

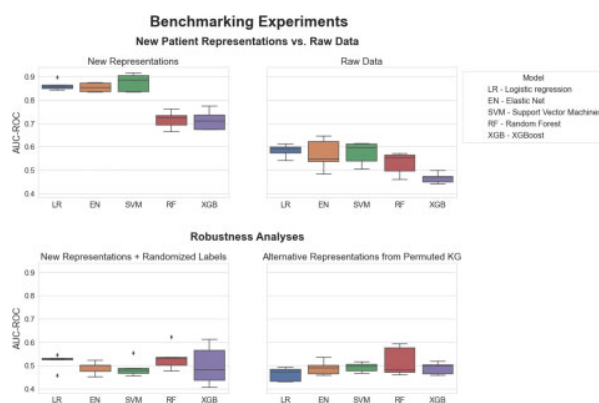


Fig. 3. Benchmarking of five ML models trained to classify between cognitively impaired patients and healthy controls. Each boxplot shows the distribution of the AUC-ROC values over 5 repeats of the 5-fold nested cross-validation procedure. Statistical modeling and ML methods are listed in Table 1. The new patient representations were generated by incorporating ADNI patients into the PPI-KG using a threshold of 2.5% on the eCDF of the control distribution for each mapped feature. The RotatE KGEM was trained on the KG using PyKEEN. The patient representations were used to train the five ML models with the original patient labels (a) and compared against the raw transcriptomics data (b). To investigate the robustness of our results, we randomized patient labels and trained the five ML models using the new representations (c). Furthermore, we trained the five ML models using the new representations generated from a permuted version of the original KG (d). Both robustness analyses yielded AUC-ROC values on all machine learning models equivalent to that of a random classifier (i.e. AUC-ROC values ~ 0.5)

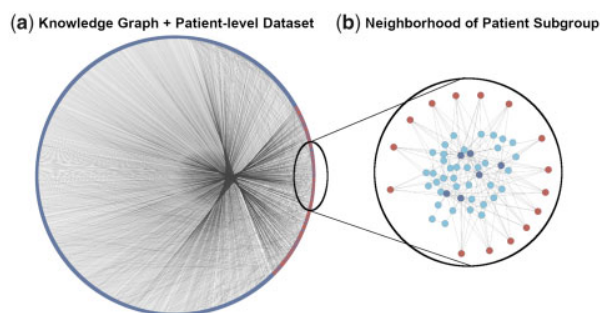


Fig. 4. Incorporating patients into a KG fosters biological interpretation and identification of patient subgroups. (a) PPI-KG subgraph after incorporating ADNI participants visualized using a circular layout. Red nodes represent ADNI patients and blue nodes represent proteins present in the PPI-KG. The majority of connections are between ADNI patients and their most characteristic proteins, which is represented by the large number of outgoing edges between the right part of the circle (where patients are located) and the rest of the KG. (b) Local neighborhood around the subgroup of cognitively impaired patients investigated in the case scenario. Red nodes represent patients, dark blue the six proteins linked to the patient subgroup and light blue other related proteins present in the neighborhood

threshold exclusively creates connections between KG nodes and the patients falling at the extreme ends of a feature distribution, thus capturing patients that can best characterize a particular feature. On the other hand, higher thresholds generate a larger number of edges

between patients and features, resulting both in a loss of specificity and the ability to distinguish between different patient groups.

The right choice of the KGEM is essential for generating patient representations that are useful for the classifiers. While the results of the case scenario are based on RotatE, we also generated patient representations based on further KGEMs (Supplementary Fig. S5). We can observe that patient representations generated by RotatE outperforms the remaining KGEMs by a large margin. It is known that specific relational patterns that can be modeled by RotatE cannot be modeled by TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016) and HolE (Nickel et al., 2016). However, to obtain a clearer picture, relational patterns around patient nodes could be investigated. Finally, we also investigated the effect the size of a KG can have on results by evaluating the performance of ML models on smaller versions of the PPI-KG. As expected, smaller PPI-KGs resulted in lower performance for each of the ML models (Supplementary Fig. S7).

3.2 Biological interpretation and patient subgroup identification through CLEP

Though the advent of machine learning methods have led to a new range of applications in the biomedical field, these methods inherently come with a tradeoff with regards to interpretability. The so-called black-box models lack transparency, providing no explanation as to what accounts for the predictions they generate. In the biomedical field in particular, this can often translate to a lack of success in clinical practice (Fröhlich et al., 2018); while discrete patterns that arise from ML techniques can be discerned, without an understanding of the fundamental cause and the characteristic features of a disease, clinicians may not have an adequate level of knowledge to make a confident diagnosis. In this section, we demonstrate how the hybrid KG generated by CLEP can drive the identification of patient-specific mechanisms and pathways made possible by the incorporation of patients into a KG comprising mechanistic knowledge (Fig. 4a).

In the previous section, we demonstrated how novel representations generated on the ADNI dataset outperformed raw data in a diagnosis prediction task. However, these representations cannot be directly interpreted as they are low-dimensional vector representations embedded in a latent space. Nonetheless, because the original KG comprises biological knowledge, features and/or mechanisms that are connected to a given embedded patient or patient subgroup can be easily pinpointed by studying their local neighborhood in the KG. Thus, using the KG derived from the ADNI dataset, we identified sets of genes with connections to cognitively impaired patients (groups) but without any connections to control participants. Of these gene sets, we focused on a particular set of genes (HS2ST1, ESR2, IKBKG, UBE2D3, PCGF5 and NFIC), all of which were connected to each other, and were also identified in a subgroup of cognitively impaired patients ($n=15$). We then investigated the interplay of genes in the local neighborhood of the KG around this subset of patients in order to identify and deconvolute common pathways that could be responsible for their phenotypic make-up (Fig. 4b).

To identify the biological pathways these genes participate in, pathway enrichment analysis was run on this gene set (Supplementary Table S2). Of the nine enriched pathways we identified (q -value < 0.05), six were related to Toll-like receptor signaling (specifically, Toll-like receptor 4 signaling) due to the involvement of IKBKG and UBE2D3 in this pathway. Interestingly, this inflammatory pathway is often noted for its association to AD (Tahara et al., 2006; Walter et al., 2007), while UBE2D3 has been proposed as a potential biomarker for Alzheimer's disease (Wu et al., 2019). Furthermore, HS2ST1 is a member of the heparan sulfate biosynthetic enzyme family and responsible for the synthesis of heparan sulfate (HS), the latter of which is known to cause protein aggregation and lead to neurodegenerative disease (Maïza et al., 2018). Additionally, ESR2 polymorphisms have been linked to cognitive impairment and an increased risk for AD, predominantly in women (Ulhaq and Garcia, 2020; Zhao et al., 2015). It is worth noting that

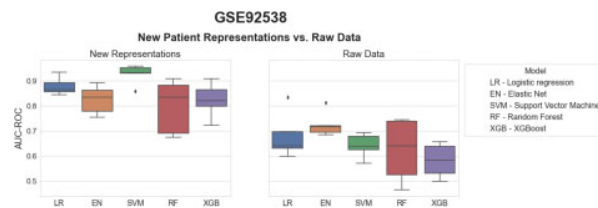


Fig. 5. Benchmarking of five ML models trained to classify between psychiatric patients and healthy controls. Each boxplot shows the distribution of the AUC-ROC values over 5 repeats of the 5-fold nested cross-validation procedure. Statistical modeling and ML methods are listed in Table 1. The new patient representations were generated by incorporating the patient data from GSE92538 into the PPI-KG using a threshold of 2.5% on the eCDF of the control distribution for each mapped feature. The RotatE KGEM was trained on the KG using PyKEEN. The patient representations were used to train the five ML models with the original patient labels (a) and compared against the raw transcriptomics data (b)

by investigating patient subgroups characterized by particular features, we can also assess whether they share common alleles. Finally, the remaining two genes, HFIC and PCGF5, are both present in the enriched 'Gene expression (transcription)' pathway, and are associated with the regulation of transcriptional factors.

3.3 CLEP's representations outperform raw data in classifying psychiatric conditions and healthy controls

We also reproduced our methodology on an additional dataset containing transcriptomics experiments on psychiatric conditions and healthy controls (see Section 2). The results for this dataset resembled the ones observed in the previous case scenario (Fig. 5). The patient representations generated by CLEP outperformed the original data in the binary classification task by a large margin. By investigating the individual ML models, we observed that the difference in performance between the two tree-based methods (i.e. random forest and XGBoost) and the rest was smaller for this dataset. Similarly to the previous case scenario, SVM yielded the highest performance. However, the variability in performance for the elastic net, random forest, and XGBoost was significantly larger compared to the ADNI dataset.

4 Discussion

In this work, we have presented CLEP, a novel hybrid data- and knowledge-driven framework which leverages patient-level data and KGs for generating personalized patient representations. In the case scenarios, we demonstrated the utility of our framework on two independent datasets by employing transcriptomics data and an integrative KG containing knowledge from several protein-protein interaction databases. When compared to the raw transcriptomics data, we have shown how these representations yield superior performance in a binary classification task using a broad panel of machine-learning models. Furthermore, we have illustrated how incorporating knowledge from the KG for the generation of patient representations not only improves performance in classification tasks, but also facilitates the interpretation of the biological mechanisms that uniquely characterize patients and/or patient subgroups. In summary, we have shown the utility of a hybrid approach for the generation of new patient representations that can then be used in a broad range of applications, ranging from predictive modeling to personalized medicine. We have also made CLEP available as an extensible and reproducible software package, enabling researchers to conduct these experiments on various kinds of datasets and networks.

Our framework shares several limitations inherent to many ML methods. The first is the computational cost and time associated with training and optimizing KGEMs, which could be improved with the help of libraries focused on using multiple GPUs and distributing computation across a cluster. The second is that our

methodology does not compensate for a lack of training data, as is often the case with clinical datasets, nor for poor quality data. The third is that this approach relies on the ability to meaningfully integrate two or more datasets with a KG. If there are too few mappable features between the clinical data and the KG, then the resulting patient representations may only have limited quality and utility, and some patients may be excluded entirely. Fourthly, the methodology to incorporate patients into the KG has been exclusively devised for continuous features. Lastly, despite our framework being generalizable to any dataset, there may be cases where CLEP fails to improve the performance as the original dataset possesses sufficient informative power for a particular application. In these cases, however, one could always employ the KG to interpret and identify biological mechanisms (e.g. pathways in the KG) associated with an individual or group of patients, as demonstrated in our case scenario.

While we generated two types of edges (i.e. -1 or +1) between ADNI patients and proteins in the PPI-KG, we also implicitly generated negative edges between each ADNI patient and all other proteins to which neither a -1 nor +1 edge was inferred. Unfortunately, we were unable to explicitly incorporate them in the training algorithm because PyKEEN, as well as most KGEM packages, generate negative edges through uniform negative sampling or Bernoulli negative sampling (Wang *et al.*, 2014b) with respect to the given positive KG. Thus, we are interested to make improvements to the upstream package itself to enable these kinds of explicit inclusions, as there are a growing number of negative edges available in various biological knowledge sources.

There are a number of applications in precision medicine for the representations generated by CLEP. For instance, in the AD area, patient representations of cognitively impaired patients could be systematically used to stratify patients in order to identify shared mechanisms that explain their observed phenotype. Furthermore, our approach can be generalized such that different types of patient-level data can be integrated and mapped to heterogeneous biological KGs, thus enabling scientists to combine their datasets with context-specific knowledge. In the future, we plan to extend this work as well as to adapt it to well-known network representation learning methods such as node2vec (Grover and Leskovec, 2016) and LINE (Tang *et al.*, 2015). Finally, we ambition that our framework could serve as an integration platform for multimodal datasets including clinical, imaging and -omics data which could lead to more comprehensive patient representations.

Acknowledgements

The authors thank André Gemünd for his technical assistance running the experiments. Data used in the preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf. Data collection and sharing for this project was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG024904) and DOD ADNI (Department of Defense award number W81XWH-12-2-0012). ADNI is funded by the National Institute on Aging, the National Institute of Biomedical Imaging and Bioengineering and through generous contributions from the following: AbbVie, Alzheimer's Association; Alzheimer's Drug Discovery Foundation; Araclon Biotech; BioClinica, Inc.; Biogen; Bristol-Myers Squibb Company; CereSpir, Inc.; Cogstate; Eisai Inc.; Elan Pharmaceuticals, Inc.; Eli Lilly and Company; EuroImmun; F. Hoffmann-La Roche Ltd and its affiliated company Genentech, Inc.; Fujirebio; GE Healthcare; IXICO Ltd.; Janssen Alzheimer Immunotherapy Research & Development, LLC.; Johnson & Johnson Pharmaceutical Research & Development LLC.; Lumosity; Lundbeck; Merck & Co., Inc.; Meso Scale Diagnostics, LLC.; NeuroRx Research; Neurotrack Technologies; Novartis Pharmaceuticals Corporation; Pfizer Inc.; Piramal Imaging; Servier; Takeda Pharmaceutical Company; and Transition Therapeutics. The Canadian Institutes of Health Research is providing funds to support ADNI clinical sites in Canada. Private sector contributions are facilitated by the Foundation for the National Institutes of Health (www.fnih.org). The grantee organization is

the Northern California Institute for Research and Education, and the study is coordinated by the Alzheimer's Therapeutic Research Institute at the University of Southern California. ADNI data are disseminated by the Laboratory for Neuro Imaging at the University of Southern California.

Author Contributions

D.D.-F. and C.T.H. conceived and designed the study. V.S.B. implemented CLEP and ran the experiments with supervision and support from D.D.-F. M.A. guided and supported the implementation of the knowledge graph embedding generation and classification tasks. C.B. assisted with data and method development. S.M. processed the knowledge graph. M.H.-A. and J.L. acquired the funding. All the authors contributed to the writing of the manuscript. All authors have read and approved the final manuscript.

Funding

This work was funded by the Fraunhofer Cluster of Excellence 'Cognitive Internet Technologies' as well as by the German Federal Ministry of Education and Research (BMBF) [01IS18050D] (project 'MLWin').

Conflict of Interest: none declared.

References

- Ali, M. *et al.* (2021) PyKEEN 1.0: a Python library for training and evaluating knowledge graph embeddings. *J. Mach. Learn. Res.*, **22**, 1–6.
- Ali, M. *et al.* (2020) Bringing light into the dark: a large-scale evaluation of knowledge graph embedding models under a unified framework. *arXiv preprint arXiv:2006.13365*.
- Bordes, A. *et al.* (2013) Translating embeddings for modeling multi-relational data. *Adv. Neural Inf. Process. Syst.*, **27**, 2787–2795.
- Cavalli, F.M. *et al.* (2017) Intertumoral heterogeneity within medulloblastoma subgroups. *Cancer Cell*, **31**, 737–754.
- Chen, T. and Guestrin, C. (2016) XGBoost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794.
- Cortes, C. and Vapnik, V. (1995) Support-vector networks. *Mach. Learn.*, **20**, 273–297.
- Domingo-Fernández, D. *et al.* (2019) PathMe: merging and exploring mechanistic pathway knowledge. *BMC Bioinformatics*, **20**, 243.
- Fan, J. *et al.* (2014) Challenges of big data analysis. *Natl. Sci. Rev.*, **1**, 293–314.
- Fröhlich, H. *et al.* (2018) From hype to reality: data science enabling personalized medicine. *BMC Medicine*, **16**, 150.
- Gong, F. *et al.* (2021) SMR: medical knowledge graph embedding for safe medicine recommendation. *Big Data Res.*, **23**, 100174.
- Grover, A. and Leskovec, J. (2016) Node2Vec: scalable feature learning for networks. In *KDD: Proceedings. International Conference on Knowledge Discovery and Data Mining, 2016*, pp. 855–864.
- Hagenauer, M.H. *et al.* (2018) Inference of cell type content from human brain transcriptomic datasets illuminates the effects of age, manner of death, dissection, and psychiatric diagnosis. *PLoS One*, **13**, e0200003.
- Hanhijärvi, S. *et al.* (2009) Randomization techniques for graphs. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pp. 780–791.
- Himmelstein, D.S. and Baranzini, S.E. (2015) Heterogeneous network edge prediction: a data integration approach to prioritize disease-associated genes. *PLoS Comput. Biol.*, **11**, e1004259.
- Ho, T.K. (1995) Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, pp. 278–282.
- Hu, W. *et al.* (2017) BioSearch: a semantic search engine for Bio2RDF. *Database*, **2017**, bax059.
- Jassal, B. *et al.* (2020) The reactome pathway knowledgebase. *Nucleic Acids Res.*, **48**, D498–D503.
- Kanehisa, M. *et al.* (2017) KEGG: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Res.*, **45**, D353–D361.
- Khanna, S. *et al.* (2018) Using multi-scale genetic, neuroimaging and clinical data for predicting Alzheimer's disease and reconstruction of relevant biological mechanisms. *Sci. Rep.*, **8**, 1–13.
- Langfelder, P. and Horvath, S. (2008) WGCNA: an R package for weighted correlation network analysis. *BMC Bioinformatics*, **9**, 559.

- Lin,Z. *et al.* (2020) Patient similarity via joint embeddings of medical knowledge graph and medical entity descriptions. *IEEE Access*, **8**, 156663–156676.
- Lynam,A.L. *et al.* (2020) Logistic regression has similar performance to optimised machine learning algorithms in a clinical setting: application to the discrimination between type 1 and type 2 diabetes in young adults. *Diagn. Prognostic Res.*, **4**, 1–10.
- Maiza,A. *et al.* (2018) The role of heparan sulfates in protein aggregation and their potential impact on neurodegeneration. *FEBS Lett.*, **592**, 3806–3818.
- Mueller,S. *et al.* (2005) Ways toward an early diagnosis in Alzheimer's disease: the Alzheimer's Disease Neuroimaging Initiative (ADNI). *Alzheimer's Dementia*, **1**, 55–66.
- Muslu,O. *et al.* (2020) GuiltyTargets: prioritization of novel therapeutic targets with deep network representation learning. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, doi: 10.1109/TCBB.2020.3003830.
- Nickel,M. *et al.* (2016) Holographic embeddings of knowledge graphs. In: *Thirtieth AAAI Conference on Artificial Intelligence*.
- Orchard,S. *et al.* (2014) The MIntAct project—IntAct as a common curation platform for 11 molecular interaction databases. *Nucleic Acids Res.*, **42**, D358–D363.
- Oughtred,R. *et al.* (2019) The BioGRID interaction database: 2019 update. *Nucleic Acids Res.*, **47**, D529–D541.
- Pai,S. and Bader,G.D. (2018) Patient similarity networks for precision medicine. *J. Mol. Biol.*, **430**, 2924–2938.
- Pai,S. *et al.* (2019) netDx: interpretable patient classification using integrated patient similarity networks. *Mol. Syst. Biol.*, **15**, e8497.
- Pedregosa,F. *et al.* (2011) Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
- Raphael,B.J. *et al.* (2017) Integrated genomic characterization of pancreatic ductal adenocarcinoma. *Cancer Cell*, **32**, 185–203.
- Rodchenkov,I. *et al.* (2020) Pathway Commons 2019 Update: integration, analysis and exploration of pathway data. *Nucleic Acids Res.*, **48**, D489–D497.
- Saito,T. and Rehmsmeier,M. (2015) The precision–recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS One*, **10**, e0118432.
- Saykin,A.J. *et al.*; Alzheimer's Disease Neuroimaging Initiative. (2015) Genetic studies of quantitative MCI and AD phenotypes in ADNI: progress, opportunities, and plans. *Alzheimer's Dementia*, **11**, 792–814.
- Slenter,D.N. *et al.* (2018) WikiPathways: a multifaceted pathway database bridging metabolomics to other omics research. *Nucleic Acids Res.*, **46**, D661–D667.
- Smith,A.M. *et al.* (2020) Standard machine learning approaches outperform deep representation learning on phenotype prediction from transcriptomics data. *BMC Bioinformatics*, **21**, 1–18.
- Sun,Z. *et al.* (2019) Rotate: knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.
- Tahara,K. *et al.* (2006) Role of toll-like receptor signalling in A β uptake and clearance. *Brain*, **129**, 3006–3019.
- Tang,J. *et al.* (2015) Line: large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077.
- Trouillon,T. *et al.* (2016) Complex embeddings for simple link prediction. In: *International Conference on Machine Learning*, pp. 2071–2080.
- Ulhaq,Z.S. and Garcia,C.P. (2020) Estrogen receptor beta (ESR2) gene polymorphism and susceptibility to dementia. *Acta Neurol. Belgica*, doi: 10.1007/s13760-020-01360-z.
- Walter,S. *et al.* (2007) Role of the toll-like receptor 4 in neuroinflammation in Alzheimer's disease. *Cell Physiol. Biochem.*, **20**, 947–956.
- Wang,B. *et al.* (2014a) Similarity network fusion for aggregating data types on a genomic scale. *Nat. Methods*, **11**, 333–337.
- Wang,Z. *et al.* (2014b) Knowledge graph embedding by translating on hyperplanes. *AAAI*, **14**, 1112–1119.
- Wu,M. *et al.* (2019) Identification of key genes and pathways for Alzheimer's disease via combined analysis of genome-wide expression profiling in the hippocampus. *Biophys. Rep.*, **5**, 98–109.
- Xu,C. and Jackson,S.A. (2019) Machine learning and complex biological data. *Genome Biol.*, **20**, 76.
- Yu,D. *et al.* (2013) Review of biological network data and its applications. *Genomics Inf.*, **11**, 200–210.
- Zhao,L. *et al.* (2015) Estrogen receptor β in Alzheimer's disease: from mechanisms to therapeutics. *Ageing Res. Rev.*, **24**, 178–190.
- Zitnik,M. *et al.* (2018) Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, **34**, i457–i466.
- Zitnik,M. *et al.* (2019) Machine learning for integrating data in biology and medicine: principles, practice, and opportunities. *Inf. Fusion*, **50**, 71–91.
- Zou,H. and Hastie,T. (2005) Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)*, **67**, 301–320.