Machine Learning for Managing Structured and Semi-Structured Data

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

eingereicht von
Max Berrendorf
am 15. Oktober 2021

1. Gutachter:	Prof. Dr. Volker Tresp
2. Gutachter:	Prof. Dr. Axel-Cyrille Ngonga Ngomo
3. Gutachter:	Prof. Dr. Achim Rettinger
Tag der mündlichen Prüfung:	18.01.2022

Eidesstattliche Versicherung

Hiermit erkläre ich, Max Berrendorf, an Eides statt, dass die vorliegende Dissertation ohne unerlaubte Hilfe gemäß Promotionsordnung vom 12.07.2011, §8, Abs. 2 Pkt. 5, angefertigt worden ist.

München, 15. Oktober 2021 _____

Max Berrendorf

Contents

Abstract

As the digitalization of private, commercial, and public sectors advances rapidly, an increasing amount of data is becoming available. In order to gain insights or knowledge from these enormous amounts of raw data, a deep analysis is essential. The immense volume requires highly automated processes with minimal manual interaction. In recent years, machine learning methods have taken on a central role in this task. In addition to the individual data points, their interrelationships often play a decisive role, e.g. whether two patients are related to each other or whether they are treated by the same physician. Hence, relational learning is an important branch of research, which studies how to harness this explicitly available structural information between different data points. Recently, graph neural networks have gained importance. These can be considered an extension of convolutional neural networks from regular grids to general (irregular) graphs.

Knowledge graphs play an essential role in representing facts about entities in a machine-readable way. While great efforts are made to store as many facts as possible in these graphs, they often remain incomplete, i.e., true facts are missing. Manual verification and expansion of the graphs is becoming increasingly difficult due to the large volume of data and must therefore be assisted or substituted by automated procedures which predict missing facts. The field of knowledge graph completion can be roughly divided into two categories: Link Prediction and Entity Alignment. In Link Prediction, machine learning models are trained to predict unknown facts between entities based on the known facts. Entity Alignment aims at identifying shared entities between graphs in order to link several such knowledge graphs based on some provided seed alignment pairs.

In this thesis, we present important advances in the field of knowledge graph completion. For Entity Alignment, we show how to reduce the number of required seed alignments while maintaining performance by novel active learning techniques. We also discuss the power of textual features and show that graph-neural-network-based methods have difficulties with noisy alignment data. For Link Prediction, we demonstrate how to improve the prediction for unknown entities at training time by exploiting additional metadata on individual statements, often available in modern graphs. Supported with results from a large-scale experimental study, we present an analysis of the effect of individual components of machine learning models, e.g., the interaction function or loss criterion, on the task of link prediction. We also introduce a software library that simplifies the implementation and study of such components and makes them accessible to a wide research community, ranging from relational learning researchers to applied fields, such as life sciences. Finally, we propose a novel metric for evaluating ranking results, as used for both completion tasks. It allows for easier interpretation and comparison, especially in cases with different numbers of ranking candidates, as encountered in the de-facto standard evaluation protocols for both tasks.

Zusammenfassung

Mit der rasant fortschreitenden Digitalisierung des privaten, kommerziellen und öffentlichen Sektors werden immer größere Datenmengen verfügbar. Um aus diesen enormen Mengen an Rohdaten Erkenntnisse oder Wissen zu gewinnen, ist eine tiefgehende Analyse unerlässlich. Das immense Volumen erfordert hochautomatisierte Prozesse mit minimaler manueller Interaktion. In den letzten Jahren haben Methoden des maschinellen Lernens eine zentrale Rolle bei dieser Aufgabe eingenommen. Neben den einzelnen Datenpunkten spielen oft auch deren Zusammenhänge eine entscheidende Rolle, z.B. ob zwei Patienten miteinander verwandt sind oder ob sie vom selben Arzt behandelt werden. Daher ist das relationale Lernen ein wichtiger Forschungszweig, der untersucht, wie diese explizit verfügbaren strukturellen Informationen zwischen verschiedenen Datenpunkten nutzbar gemacht werden können. In letzter Zeit haben Graph Neural Networks an Bedeutung gewonnen. Diese können als eine Erweiterung von CNNs von regelmäßigen Gittern auf allgemeine (unregelmäßige) Graphen betrachtet werden.

Wissensgraphen spielen eine wesentliche Rolle bei der Darstellung von Fakten über Entitäten in maschinenlesbaren Form. Obwohl große Anstrengungen unternommen werden, so viele Fakten wie möglich in diesen Graphen zu speichern, bleiben sie oft unvollständig, d. h. es fehlen Fakten. Die manuelle Überprüfung und Erweiterung der Graphen wird aufgrund der großen Datenmengen immer schwieriger und muss daher durch automatisierte Verfahren unterstützt oder ersetzt werden, die fehlende Fakten vorhersagen. Das Gebiet der Wissensgraphenvervollständigung lässt sich grob in zwei Kategorien einteilen: Link Prediction und Entity Alignment. Bei der Link Prediction werden maschinelle Lernmodelle trainiert, um unbekannte Fakten zwischen Entitäten auf der Grundlage der bekannten Fakten vorherzusagen. Entity Alignment zielt darauf ab, gemeinsame Entitäten zwischen Graphen zu identifizieren, um mehrere solcher Wissensgraphen auf der Grundlage einiger vorgegebener Paare zu verknüpfen.

In dieser Arbeit stellen wir wichtige Fortschritte auf dem Gebiet der Vervollständigung von Wissensgraphen vor. Für das Entity Alignment zeigen wir, wie die Anzahl der benötigten Paare reduziert werden kann, während die Leistung durch neuartige aktive Lerntechniken erhalten bleibt. Wir erörtern auch die Leistungsfähigkeit von Textmerkmalen und zeigen, dass auf Graph-Neural-Networks basierende Methoden Schwierigkeiten mit verrauschten Paar-Daten haben. Für die Link Prediction demonstrieren wir, wie die Vorhersage für unbekannte Entitäten zur Trainingszeit verbessert werden kann, indem zusätzliche Metadaten zu einzelnen Aussagen genutzt werden, die oft in modernen Graphen verfügbar sind. Gestützt auf Ergebnisse einer groß angelegten experimentellen Studie präsentieren wir eine Analyse der Auswirkungen einzelner Komponenten von Modellen des maschinellen Lernens, z. B. der Interaktionsfunktion oder des Verlustkriteriums, auf die Aufgabe der Link Prediction. Außerdem stellen wir eine Softwarebibliothek vor, die die

Implementierung und Untersuchung solcher Komponenten vereinfacht und sie einer breiten Forschungsgemeinschaft zugänglich macht, die von Forschern im Bereich des relationalen Lernens bis hin zu angewandten Bereichen wie den Biowissenschaften reicht. Schließlich schlagen wir eine neuartige Metrik für die Bewertung von Ranking-Ergebnissen vor, wie sie für beide Aufgaben verwendet wird. Sie ermöglicht eine einfachere Interpretation und einen leichteren Vergleich, insbesondere in Fällen mit einer unterschiedlichen Anzahl von Kandidaten, wie sie in den de-facto Standardbewertungsprotokollen für beide Aufgaben vorkommen.

List of Publications and Declaration of Authorship

- **Max Berrendorf***, Evgeniy Faerman*, and Volker Tresp. „Active Learning for Entity Alignment.“ In: *Advances in Information Retrieval*. Ed. by Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani. * equal contribution. Cham: Springer International Publishing, 2021, pp. 48–62. ISBN: 978-3-030-72113-8. DOI: 10.1007/978-3-030-72113-8_4

  The original research contributions were developed and conceptualized by Max Berrendorf and Evgeniy Faerman and discussed with Volker Tresp. Max Berrendorf did the main part of the implementation, with smaller contributions by Evgeniy Faerman. Max Berrendorf designed and conducted the experiments and analyzed their results. The findings were discussed with Evgeniy Faerman. Max Berrendorf and Evgeniy Faerman wrote and revised the manuscript.

  This publication serves as Chapter 4 of this thesis.

- Mehdi Ali*, **Max Berrendorf***, Mikhail Galkin, Veronika Thost, Tengfei Ma, Volker Tresp, and Jens Lehmann. „Improving Inductive Link Prediction Using Hyper-relational Facts.“ In: *The Semantic Web – ISWC 2021* (2021). * equal contribution, awarded with Best Research Paper Award, pp. 74–92. DOI: 10.1007/978-3-030-88361-4_5

  The research idea was developed and conceptualized by Max Berrendorf, Mehdi Ali and Mikhail Galkin. Max Berrendorf and Mehdi Ali did the main part of the implementation, with Max Berrendorf focussing on the models, training loop and evaluation, and Mehdi Ali taking care of the data splits and hyperparameter optimization. Mehdi Ali conducted the experiments, and Max Berrendorf additionally analyzed their results. Max Berrendorf, Mehdi Ali and Mikhail Galkin jointly wrote manuscript, and all authors revised it.

  This publication serves as Chapter 5 of this thesis.

- 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

  The idea was developed and discussed by Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt, Sahand Sharifzadeh and Laurent Vermue after each of the authors independently discovered (different) issues with existing works. Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt, and Laurent Vermue analyzed existing implementations, developed the library's architecture and implemented it. Mehdi Ali made a first draft of the manuscript and Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt, Sahand Sharifzadeh, and Laurent Vermue developed and wrote it. All authors revised the manuscript.

  This publication serves as Chapter 6 of this thesis.

- 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: *CoRR* abs/2006.13365 (2020). * equal contribution. arXiv: 2006.13365

  The idea was developed and discussed by Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt, Sahand Sharifzadeh and Laurent Vermue in close interaction with the development of the PyKEEN library. Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt and Laurent Vermue did the implementation. Mehdi Ali, Laurent Vermue and Mikhail Galkin conducted the experiments and Max Berrendorf coordinated the collection of all results. Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt and Laurent Vermue analyzed the results. Max Berrendorf additionally implemented and conducted the relation-specific analysis, and evaluated its results. Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt and Laurent Vermue wrote the manuscript, and all authors revised the manuscript.

  This publication serves as Chapter 7 of this thesis.

- **Max Berrendorf**, Ludwig Wacker, and Evgeniy Faerman. „A Critical Assessment of State-of-the-Art in Entity Alignment.“ In: *Advances in Information Retrieval*. Ed. by Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani. Cham: Springer International Publishing, 2021, pp. 18–32. ISBN: 978-3-030-72240-1. DOI: 10.1007/978-3-030-72240-1_2

  Max Berrendorf identified the research gap, and conducted initial experiments. The idea was then further developed and conceptualized by Max Berrendorf and Evgeniy Faerman, and later on discussed with Ludwig Wacker. Max Berrendorf and Ludwig Wacker did the implementation. Max Berrendorf and Evgeniy Faerman designed the experiments, and Ludwig Wacker conducted most of them. Max Berrendorf and Ludwig Wacker analyzed the results, and discussed the results with Evgeniy Faerman. Max Berrendorf and Evgeniy Faerman wrote the manuscript.

  This publication serves as Chapter 8 of this thesis.

- **Max Berrendorf**, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. „Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods.“ In: *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE, Dec. 2020. DOI: 10.1109/wiiat50758.2020.00053

  The research idea was proposed by Max Berrendorf and discussed with Laurent Vermue. Max Berrendorf and Evgeniy Faerman developed and conceptualized it further, and discussed with all co-authors. Max Berrendorf did the implementation, designed and conducted the experiments and evaluated their results. Max Berrendorf and Evgeniy Faerman wrote the manuscript.

  This publication serves as Chapter 9 of this thesis.

1 Introduction

While the ever-growing volumes of available data are a valuable resource in themselves, there is considerable added value in connecting them. As an example, consider the prediction of side effects of drugs for a patient. A simple model here can utilize, for instance, patient data and a textual description of the drug. The prediction can presumably be improved by including additional relationships. For example, information about other drugs used at the same time can be included, e.g., because certain effects are only produced by the combination of several drugs. Connections to existing drug incompatibilities, information on their ingredients and mechanisms of action, and their connections to the drug currently under consideration are also likely to improve prediction. Further links of the patient, such as to similar patients, e.g., from the family, represent other valuable pieces of information.

As seen from the example, real-world connections can easily become complex and irregular, and thus, there is a need for flexible yet expressive modeling schemes which can capture these relationships in machine-readable format. A prominent example are graphs, where connections are explicitly given as edges between the nodes of related entities. Exemplary applications include social networks, road or communication networks, biomedical interaction graphs, or configurations of, e.g., production sites. Besides information about single entities, there is often additional information attached to the relationship between them. This information can describe, e.g., the provenance of the relationship information, e.g., from which source this information stems, or detail the type of relation, e.g., whether a drug inhibits or upregulates a biomolecular process. In general, this information can be modeled via edge features or more complex, hyper-relational modeling schemes.

In particular, when dealing with facts about entities, Knowledge Graphs (KGs) are an intriguing variant of graphs. KGs are multi-relational graphs, where individual connections between entities are of certain relation types. Hence, they can naturally model a statement *subject predicate object* as a connection of the type *predicate* between the nodes *subject* and *object*. Conversely, it is trivial to enumerate statements about a specific entity from a given KG, which makes them a highly interpretable data format.

In the recent decades we have witnessed the emergence of several large-scale KGs [106]. There are general purpose ones, such as Cyc [151], YAGO [203], Freebase [34], DBpedia [131], Wikidata [219], or NELL [154], as well as domain-specific open KGs, encompassing, e.g., the legal domain [156, 79, 120], healthcare [187, 61, 189], bio-medical [103, 41, 220, 261, 111], chemistry [157], material science [158], geo-science [50], academia [223, 74, 12], or fact checking [209, 147].

Since Google's announcement about using KGs to enrich their search results [199], a multitude of other companies also announced their usage of own KGs, e.g., Accenture [171],

AirBnB [48], Amazon [127], Apple [148], AstraZeneca [21], Bing [198], Bloomberg [152], Bosch [100], eBay [178], IBM [66], LinkedIn [98], Maana [59], Pinterest [88], Siemens [110], Thomson Reuters [211], or Uber [94].

Given their ubiquity and vast extent, KGs and machine learning thereupon have been successfully utilized to improve several downstream tasks, such as semantic search [16, 68], question answering [257], or recommender systems [93, 139, 102]. Moreover, they are applied to the fields of, e.g., drug repurposing [146, 241, 254], manifacturing [185, 124, 262], finance [55, 80], or culture [46].

A common problem across most KGs is their incompleteness [226]. While manually curated rules and KG interlinks have been successfully applied for a longer time, machine learning approaches offer an appealing alternative with the potential to overcome the inherent scalability issues of handcrafted rules. Thus, they have become a decisive technique in this active research area. Most approaches for KG completion or enrichment fall into two broad categories. First, Link Prediction (LP) methods directly work on the KGs, and aim at predicting missing links given the observed ones. For this setting, we can distinguish the transductive setting, where all entities are known at training time, from inductive settings, where novel entities can be encountered at inference time [95]. Second, Entity Alignment (EA) approaches address incompleteness by establishing links between two KGs, thus enabling to harness the combined knowledge.

Since machine learning approaches for both tasks operate on the same underlying data structure, there is considerable overlap in the employed approaches. Thus, we contribute in both directions in this thesis. We are the first to show how we can minimize required labels for the EA task by interleaving the labeling process with model training, commonly referred to as active learning. Moreover, we propose to leverage hyper-relational information present in modern KGs in practically highly relevant inductive LP settings, where we may encounter unseen entities at inference time. We then elucidate the contribution of individual components of LP and EA models and their training, supported by experimental results from large-scale studies, as well as an open-source library, PyKEEN. We conclude by presenting a novel rank-based evaluation metric, which is more interpretable than existing metrics and allows direct comparison in practically relevant settings with variable number of ranking candidates.

2 Overview of Contributions

This chapter provides an overview of the thesis and positions the included publications within the research area.

- Chapter 3 gives an overview of the broader research area. It reviews and orders existing work and introduces concepts and formalisms commonly employed in the subsequent chapters. It provides a detailed overview about the general process, from formalizing the data structure and tasks thereupon, over ways of learning individual representations, to structural enrichment thereof via neural message passing mechanisms. While until then, the representation learning techniques have been the same for both Knowledge Graph (KG) enrichment tasks; subsequently, we discuss the different "decoders" and training approaches for the two tasks, Link Prediction and Entity Alignment. Finally, we revisit the rank-based evaluation protocol applied for both tasks.

- Chapter 4 addresses the problem of obtaining labels for Entity Alignment (EA). In practice, current approaches usually use 30% of all shared entities for training. Obtaining this high number of labels can be costly for large KG pairs or domain-specific KGs where skilled annotators are required. Thus, we are the first to propose to use active learning techniques for EA. We formalize the labeling framework and show that existing state-of-the-art heuristics from classification tasks based on uncertainty or embedding space coverage are insufficient to obtain satisfying performance. We propose several solid passive learning strategies based on graph centrality and a novel active learning heuristic. We show that we outperform several baselines, particularly for few label regimes, which are more relevant in practice.

- Chapter 5 studies inductive Link Prediction (LP). While in transductive LP we can assume that all entities encountered during inference have already been seen in training, in the inductive setting, we have to deal with unseen entities. Our first contribution is to categorize different inductive settings, depending on the availability of additional inference facts and whether unseen entities have to be interconnected or only linked to known entities. Moreover, modern KGs, such as Wikidata [219], often contain hyper-relational facts, i.e., there is additional context given to individual facts. We are the first to study their use in semi- and fully-inductive settings, and to this end also propose a novel set of benchmark datasets. Our experiments demonstrate performance improvements of up to 6% points absolute improvement in the Hits@10 metric compared to triple-only baselines.

- Chapter 6 introduces the open-source library PyKEEN[1] for KG embedding models. This library was specifically developed to encode our understanding of important components for LP on KG, e.g., the interaction function or training approach, as outlined in Chapter 3. It enables studying the reproducibility of proposed approaches and offers excellent flexibility in combining individual components reaching far beyond the space explored in existing publications. Chapter 7 then presents an analysis of the impact of the individual components of KG embedding models enabled by our library. It is supported by the results of a large-scale experimental study and thoroughly studies the reasons behind performance differences. To this end, we compare and analyze the performance over multiple benchmark datasets and numerous configurations, isolating individual components' contributions, particularly of the interaction model, the loss function, the training approach, and the explicit use of inverse relations. Our results include several configurations competitive to state-of-the-art and often improve upon published results for a particular interaction function. Moreover, we analyze differences between relational patterns, make several recommendations to practitioners, and conduct a large reproducibility study that quantifies the commonly denounced reproducibility crisis. Our library, PyKEEN, serves as a step towards more reproducible research by providing a simple and re-usable framework for implementing and studying new methods.

- Chapter 8 analyzes the impact of textual features on the performance of state-of-the-art EA methods. In contrast to existing work, we ensure a consistent and fair setting for all methods and reveal the surprisingly good performance of the features in a zero-shot setting, for noisy datasets nearly fully explaining the observed performance of several graph-neural-network-based methods. In other, less noisy settings, we can confirm that neighborhood aggregation indeed is beneficial for the alignment task.

- Chapter 9 introduces a novel rank-based metric based upon the mean rank. Existing metrics automatically become better if the number of ranking candidates decreases. Hence, an interpretation thereof requires consideration of the number of candidates. In contrast, our proposed metric is adjusted for chance. Thereby, it is also implicitly normalized by the number of candidates and thus allows comparison of results when this number varies. Such normalization is helpful for the standard EA evaluation protocol, where only entities participating in at least one test pair serve as candidates, but also for the filtered evaluation setting, which is the de-facto standard in LP.

In summary, we believe that this work significantly advances the field of machine learning on KGs. As a result of this work, we are now able to apply EA approaches in an economically viable way in settings where few labels are available, and utilize the valuable context of qualifier pairs present in modern KGs also in the practically more relevant *inductive* LP setting with unseen entities. Moreover, we elucidated the effect of individual components of LP models, as well as the strength of lingual features in EA, and showed

[1] `https://github.com/pykeen/pykeen`

that current Graph Neural Network (GNN)-based EA approaches cannot always benefit from the additional structural signal given by the graph structure. Finally, we provide a novel rank-based evaluation metric which is more intuitive and allows easy comparison of LP and EA models in the practically highly relevant rank-based evaluation settings with variable number of candidates.

Since reproducible science is of utmost importance to us, we provide openly accessible implementations for all presented works and explicitly refer to them in this thesis in addition to the respective publications. By this mean we hope to contribute towards resolving the reproducibility crisis as well as providing usable research artifacts to the community.

3 Background

This chapter introduces the central concepts of the thesis in greater detail than a single paper allows. It also reviews existing works to contextualize the thesis' contributions better. We begin by introducing general notation in Section 3.1. Since the definition of a KG is not consistent across different works [106], Section 3.2 formalizes the concept used within this thesis. Section 3.3 introduces the two tasks of LP and EA. Section 3.4 presents machine learning techniques for learning (entity) representations in KGs applicable to all enrichment methods. Section 3.5 then presents score functions ("decoders"), training approaches, and losses for LP. Section 3.6 introduces EA-specific components, particularly similarity normalizations, matching techniques, losses and training methods. Section 3.7 finally discusses rank-based evaluation commonly applied to both tasks.

3.1 Notation

In this section, we introduce the notation used throughout the remainder of this chapter, which is also mostly consistent with the individual publications in the other chapters. In general, we use lower-case Greek letters, e.g., α, to denote scalar values, lower-case bold-font letters, e.g., $\mathbf{x}$, to denote vectors, upper-case bold-font, e.g., $\mathbf{X}$, to denote matrices or higher-order tensors. By $\mathbb{R}$ and $\mathbb{C}$ we denote real and complex numbers, respectively. If not noted otherwise, we assume each variable to be real. For a complex number α, we use $\Re(\alpha)$ to denote its real part, and $\overline{\alpha}$ for its complex conjugate. With log we refer to the natural logarithm (ln), if not stated otherwise. By $\langle \mathbf{x}, \mathbf{y} \rangle$ we denote the inner product between $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. If not noted otherwise, we use the standard inner product $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$. By $\|\mathbf{x}\|$ we denote the norm of $\mathbf{x}$, and by $\|\mathbf{x}\|_p = \left(\sum_{i=1}^{d} |\mathbf{x}_i|^p\right)^{1/p}$ the p norm specifically. By $\odot$ we denote the Hadamard product, i.e., the element-wise multiplication, $(\mathbf{x} \odot \mathbf{y})_i := \mathbf{x}_i \cdot \mathbf{y}_i$. We use $[\mathbf{x}; \mathbf{y}] \in \mathbb{R}^{d+d'}$ to denote the concatenation of vectors $\mathbf{x} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^{d'}$. For a matrix $\mathbf{W} \in \mathbb{R}^{d \times d'}$, we use $\text{vec}(\mathbf{W}) \in \mathbb{R}^{d \cdot d'}$ to denote the flattened matrix with $(\text{vec}(\mathbf{W}))_i := (\mathbf{W})_{\lfloor i/d \rfloor, (i \mod d)}$. Moreover, we use $\text{stack}(\mathbf{x}_1, \ldots, \mathbf{x}_k) \in \mathbb{R}^{d \times k}$ to denote the combination of multiple vectors $\mathbf{x}_i \in \mathbb{R}^d$ to a matrix, with $(\text{stack}(\mathbf{x}_1, \ldots, \mathbf{x}_k))_{i,j} := (\mathbf{x}_j)_i$.

We commonly use the following activation functions. When applied to vectors, matrices, or tensors, we understand them as elementwise operations.

- The (logistic) sigmoid function

$$\sigma(\alpha) = \frac{1}{1 + \exp(-\alpha)}$$

- The hyperbolic tangent

$$\tanh(\alpha) = \frac{\exp(\alpha) - \exp(-\alpha)}{\exp(\alpha) + \exp(-\alpha)}$$

- The Rectified Linear Unit (ReLU)

$$\mathrm{ReLU}(\alpha) = \max\{0, \alpha\}$$

- The Leaky ReLU [144]

$$\mathrm{LeakyReLU}(\alpha) = \begin{cases} \alpha & \text{if } \alpha > 0 \\ \beta \cdot \alpha & \text{otherwise} \end{cases}$$

  where $\beta = 10^{-2}$ if not specified differently.

- The softplus activation

$$\mathrm{softplus}(\alpha) = \log(1 + \exp(\alpha))$$

- The softmax

$$(\mathrm{softmax}(\mathbf{x}))_i = \frac{\exp(\mathbf{x}_i)}{\sum_j \exp(\mathbf{x}_j)}$$

  With slight abuse of notation, we use the same function symbol with an additional argument to denote the vectorized softmax operation applied to the rows/columns of a matrix. The second argument denotes the axis along which the normalization is applied, e.g., $\mathrm{softmax}(\mathbf{X}, 1)$ denotes the row-wise softmax.

For some interaction functions, cf. Section 3.3.1, we further require basic notions of hyperbolic geometry, in particular the Poincaré ball model, $\{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x}\|_2^2 < \gamma^{-1}\}$ for $\gamma > 0$, which is a hyperbolic geometry with curvature $-\gamma$. In [84], the vector addition $\oplus^\gamma$, linear transformation $\otimes^\gamma$, and distance d^γ are defined for this space using the exponential and logarithmic map:

$$\exp_{\mathbf{0}}^\gamma(\mathbf{x}) = \frac{\tanh(\sqrt{\gamma}\|\mathbf{x}\|)}{\sqrt{\gamma}\|\mathbf{x}\|}\mathbf{x} \tag{3.1}$$

$$\log_{\mathbf{0}}^\gamma(\mathbf{x}) = \frac{\operatorname{arctanh}(\sqrt{\gamma}\|\mathbf{x}\|)}{\sqrt{\gamma}\|\mathbf{x}\|}\mathbf{x} \tag{3.2}$$

$$\mathbf{x} \oplus^\gamma \mathbf{y} = \frac{(1 + 2\gamma\mathbf{x}^T\mathbf{y} + \gamma\|\mathbf{y}\|^2)\mathbf{x} + (1 - \gamma\|\mathbf{x}\|^2)\mathbf{y}}{1 + 2\gamma\mathbf{x}^T\mathbf{y} + \gamma^2\|\mathbf{x}\|^2\|\mathbf{y}\|^2} \tag{3.3}$$

$$\mathbf{W} \otimes^\gamma \mathbf{y} = \exp_{\mathbf{0}}^\gamma(\mathbf{W}\log_{\mathbf{0}}^\gamma(\mathbf{x})) \tag{3.4}$$

$$d^\gamma(\mathbf{x}, \mathbf{y}) = \frac{2}{\sqrt{\gamma}}\operatorname{arctanh}(\sqrt{\gamma}\|-\mathbf{x} \oplus^\gamma \mathbf{y}\|) \tag{3.5}$$

We also consider the following abbreviation for the Hadamard product: $\mathbf{x} \odot^\gamma \mathbf{y} = \mathrm{diag}(\mathbf{x}) \otimes^\gamma \mathbf{y}$.

3.2 Knowledge Graphs

KGs are a flexible and versatile data structure to store knowledge as facts. In this work, we focus on KGs in the following definition:

Definition 3.2.1 (Knowledge Graph)**.** *A Knowledge Graph (KG) is a 3-tuple* $\mathcal{K} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ *with a set of entities* $\mathcal{E}$*, a set of relations* $\mathcal{R}$*, and a set of triples* $\mathcal{T} \subseteq (\mathcal{E} \times \mathcal{R} \times \mathcal{E})$*.*

We call $(h, r, t) \in \mathcal{T}$ a *triple* or fact, with the *head* entity h, the *relation* r, and the *tail* entity t. For example, we can encode the fact that Alan Turing was educated at Princeton University as a triple (`Alan Turing`, `educated at`, `Princeton University`) with the head entity `Alan Turing`, the relation `educated at`, and the tail entity `Princeton University`. An alternative view on KGs is as a directed multi-relational graph, where the entities are nodes, and each triple describes a directed edge from the head to the tail entity, with the corresponding relation type. Notice that this graph may contain self-loops and multiple edges (of a different type) between two entities. As an example, Fig. 3.1 shows an excerpt of Wikidata [219] including the aforementioned fact.

When describing approaches working on KG, we often make the simplifying assumption that we use numeric entity and relations, i.e., $\mathcal{E} = \{1, \dots, |\mathcal{E}|\}$, or $\mathcal{R} = \{1, \dots, |\mathcal{R}|\}$, and thus can use entities and relations to index vectors and matrices. In their implementations, this is often done in a preprocessing step. After that, the actual training and evaluation parts can work purely index-based, allowing vectorized operations directly supported by utilized libraries and hardware.

For $h, t \in \mathcal{E}, r \in \mathcal{R}$, we use

$$\mathcal{T}_{(h,r,\cdot)} = \{(h', r', t') \in \mathcal{T} \mid h' = h \wedge r' = r\}$$

to denote the set of triples where the head and relation match h and r, and $\cdot$ can be arbitrary. For instance, in the example KG given in Fig. 3.1, we would have

$$\begin{aligned}\mathcal{T}_{(\texttt{Alan Turing},\texttt{educated at},\cdot)} = \{&(\texttt{Alan Turing}, \texttt{educated at}, \texttt{Princeton Unversity}),\\ &(\texttt{Alan Turing}, \texttt{educated at}, \texttt{Cambridge Unversity}),\\ &(\texttt{Alan Turing}, \texttt{educated at}, \texttt{King's College})\}\end{aligned}$$

Analogously, we use $\mathcal{T}_{(h,\cdot,t)}$, $\mathcal{T}_{(\cdot,r,t)}$, and also $\mathcal{T}_{(\cdot,r,\cdot)}$, where the latter denotes the set of all triples with the given relation. Moreover, for a set of triples $\mathcal{T}$, we use $\mathcal{E}_\mathcal{T} \subseteq \mathcal{E}$ and $\mathcal{R}_\mathcal{T} \subseteq \mathcal{R}$ to denote the set of occurring entities or relations. If we further want to restrict the position at which an entity occurs, we can use $\mathcal{E}^h_\mathcal{T}$ and $\mathcal{E}^t_\mathcal{T}$ to denote the set of head and tail entities. By combination it is easy to denote, e.g., the set of head entities co-occurring with a certain relation as r as $\mathcal{E}^h_{\mathcal{T}_{(\cdot,r,\cdot)}}$.

A common practice in machine learning for KGs is to add *inverse triples* [65, 216, 239, 149]. For each triple $(h, r, t) \in \mathcal{T}$, we add another triple (t, r^{-1}, h), where r^{-1} is a new relation denoting the *inverse relation* to r. By adding inverse triples, we double the number of relations and triples. While this only adds redundant facts and may appear as a pure data augmentation technique, it can have a beneficial impact, e.g., by allowing

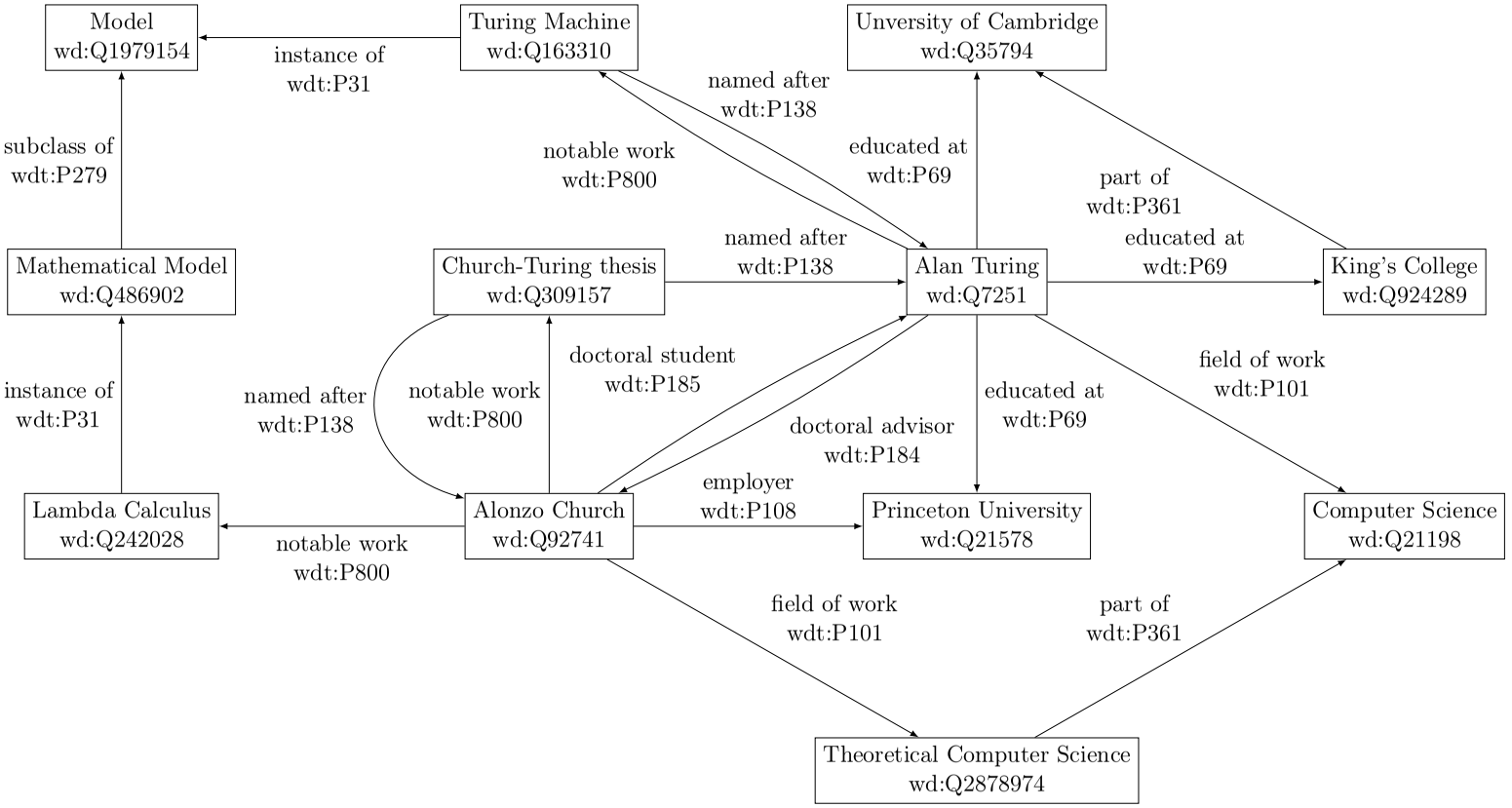


Figure 3.1: An excerpt of the Wikidata [219] KG showing several selected entities and a selection of relationships between them. The entity / relation identifiers are abbreviated with the commonly used `wd`/`wdt` prefixes, and English labels are added for readability.

to learn two separate representations for each original relation. Its impact is thoroughly discussed in Chapter 7. For GNN-based methods, *self-loops* are sometimes introduced: For each entity $e \in \mathcal{E}$, the triple $(e, r_{\circlearrowleft}, e)$ is added, where $r_{\circlearrowleft}$ is a new relation symbol. We use $\text{dir}(r) \in \{\rightarrow, \leftarrow, \circlearrowleft\}$ to denote whether a relation is a normal relation ($\rightarrow$), or an inverse relation ($\leftarrow$), or the self-loop relation ($\circlearrowleft$).

An extension to KGs are hyper-relational KG. These allow to extend triples (h, r, t) to *statements* $(h, r, t, \mathcal{Q})$, where $\mathcal{Q} \subseteq \mathcal{R} \times \mathcal{E}$ is a set of *qualifier pairs* which provides additional context to the main triple (h, r, t). For instance, we can provide additional context to the statements about Alan Turing's education by attaching the degree, e.g., (`Alan Turing`, `educated at`, `Cambridge`, {(`academic degree`, `Master of Arts`)}). Notice that we can have multiple different statements with the same base triple, but different qualifier pairs. For instance, we could also formulate another true statement about Alan Turing's education as (`Alan Turing`, `educated at`, `Cambridge`, {(`academic degree`, `Bachelor of Arts`)}).

3.3 Knowledge Graph Enrichment

Real-world KGs tend to be reliable [170, 2, 85], i.e., the vast majority of contained facts is true, but they are known to be incomplete [226], i.e., they do not contain all true facts but only a (small) subset thereof. Therefore, machine learning on KGs usually operate under the *open-world assumption*, i.e., they assume that present facts are valid; however, the absence of a fact does imply falseness but rather only means that this fact is unknown.

To address the incompleteness of KGs, we distinguish two families of approaches: LP and EA. In LP, also referred to as KG completion, the enrichment is done within a single KG. In contrast, in EA, knowledge is enriched by taking several KGs into account and identifying shared entities.

3.3.1 Link Prediction

In the task of Link Prediction (LP), we aim to materialize "latent knowledge" by predicting missing links based upon the observed ones. To this end, we are provided with a set of triples $\mathcal{T}_{train}$ from the incomplete KG and are to infer missing ones.

In contrast to the related task of *triple classification*, where each triple is assigned with a probability score, the LP task is commonly framed as a ranking task: In tail prediction, given a head entity $h \in \mathcal{E}$ and a relation $r \in \mathcal{R}$, the task is to rank all possible entities $t \in \mathcal{E}$ such that true triples receive a larger score than others. The head prediction is defined analogously, as scoring possible head entities $h \in \mathcal{E}$ for a given (r, t) pair. As an example, we could consider the finding notable works of Alan Turing, i.e., the task (`Alan Turing`, `notable work`, `?`), cf., Fig. 3.2. We would expect from a link prediction model to score the entity `Turing Machine` higher than other entities such as, e.g., `Lambda Calculus`, or `Alonzo Church`. Notice that a link prediction might have picked up correlations such as if something is named after someone, it may be a notable work. Thus, we might also see high scores for `Church-Turing thesis`, which is a plausible fact, yet not contained in

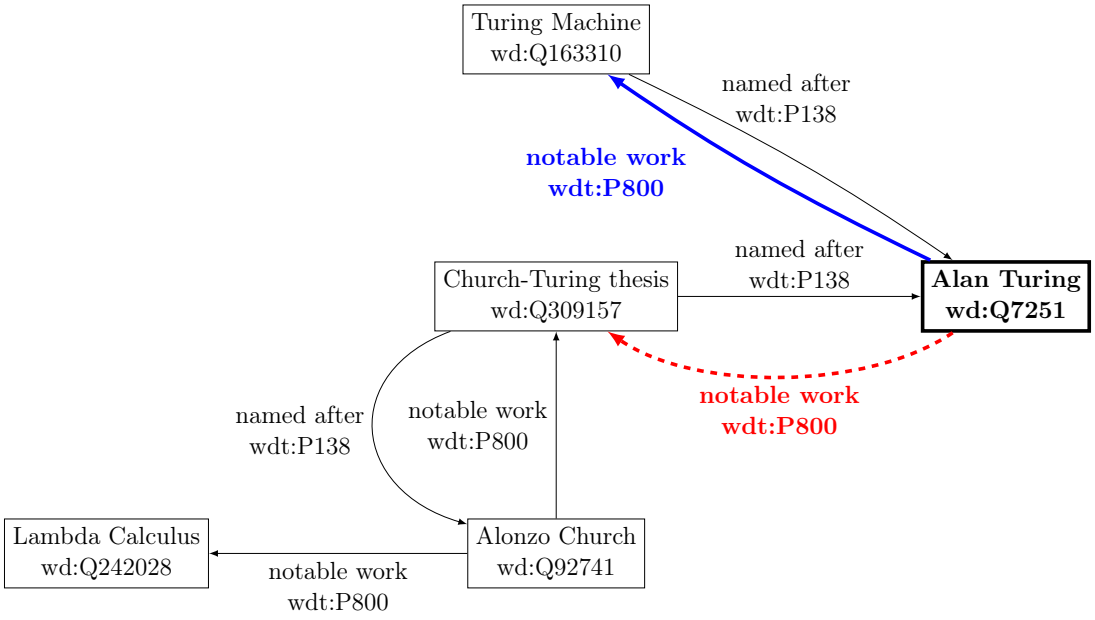


Figure 3.2: A subset of the triples from Fig. 3.1 showcasing the task of link prediction. Assume we are looking for notable works of Alan Turing, i.e., entities $e \in \mathcal{E}$ for which the triple (`Alan Turing`, `notable work`, e) holds. Besides recovering true entities such as `Turing Machine`, we may also want to have generalization capabilities, e.g., that the model picks up correlations between the `named after` and `notable work` relations, and also yields a high score for `Church-Turing thesis`.

Wikidata (as of the time of writing).

For LP, we can distinguish the *transductive* setting and different inductive settings [95, 6]: In the transductive setting, all entities are known at training time, i.e., during inference, we are only asked to predict links between known entities. In contrast, inductive settings allow unknown entities. We can further distinguish different inductive settings, cf. Chapter 5: In the fully inductive setting, we predict links between unknown entities, while the semi-inductive setting requires connecting unknown entities to known ones. Furthermore, we may be presented with additional inference triples $\mathcal{T}_{\text{inf}}$, which can be utilized, e.g., by GNN-based models, to improve their prediction.

3.3.2 Entity Alignment

In contrast to LP, in Entity Alignment (EA), the goal is to enrich the information present in a single KG by linking between (some of) its entities and entities from another KG. Thus, two KGs, $\mathcal{K}^L = (\mathcal{E}^L, \mathcal{R}^L, \mathcal{T}^L)$ and $\mathcal{K}^R = (\mathcal{E}^R, \mathcal{R}^R, \mathcal{T}^R)$, are given where some entities are shared among the two graphs, or considered to be equivalent. We denote by $\mathcal{A} \subseteq (\mathcal{E}^L \times \mathcal{E}^R)$ a set of corresponding or matching entity pairs, also called

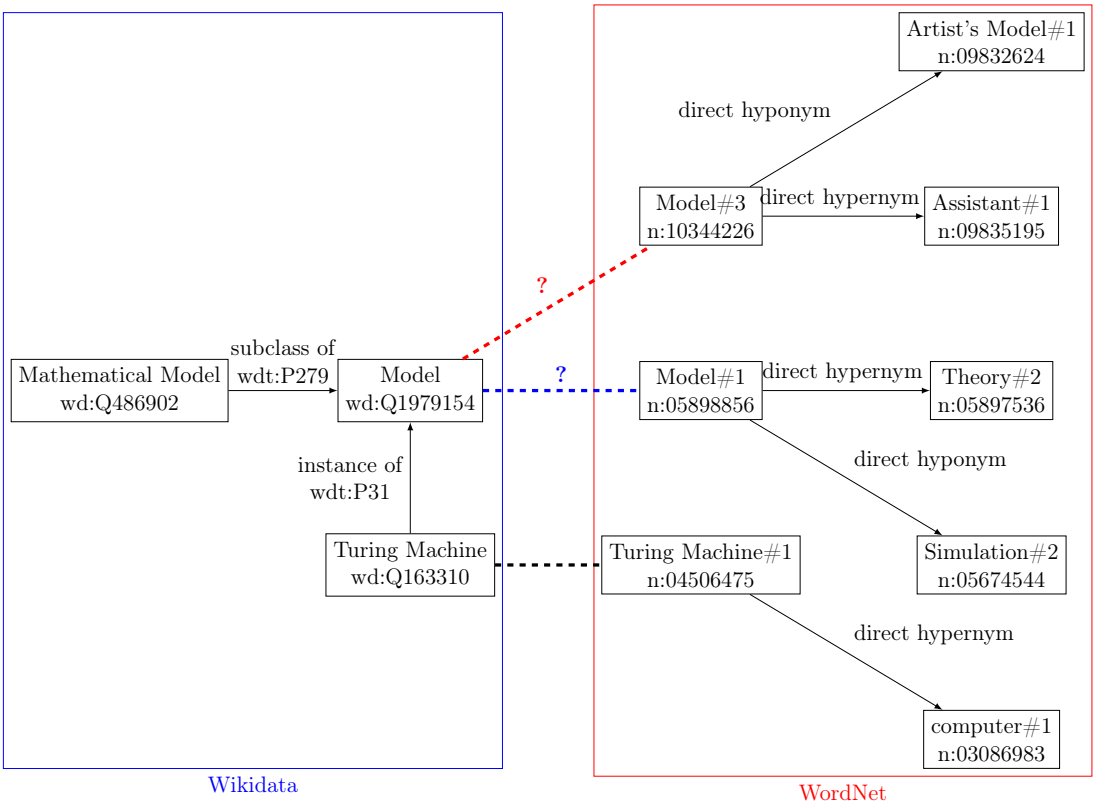


Figure 3.3: Example for an EA between Wikidata [219] and WordNet [153]. The task is to identify shared entities between the two KGs. Often, the task is posed in a supervised setting, where some entity pairs are given, e.g., (`Turing Machine`, `Turing Machine#1`), and the remainder is to be inferred, e.g., (`Model`, `Model#1`).

alignments, and formulate this task in a supervised manner, where a subset of alignments $\mathcal{A}_{train} \subset \mathcal{A}$ is given, and the remainder is to be inferred. Similarly to LP, EA is also often seen as a ranking task, where candidate entities from $e^R \in \mathcal{E}^R$ are to be ranked for a given entity $e^L \in \mathcal{E}^L$. In Fig. 3.3, we show a (toy) example for an EA between Wikidata [219] and WordNet [153]. While some entities may be easily matched using, e.g., string similarity of entity names, e.g., `Turing Machine`, others, such as `Model` require taking the neighborhood into account. In the absence of node or edge features, the task becomes similar to subgraph matching. However, there are differences: The EA community usually assumes a supervised setting where some alignment pairs are given, and only the remainder has to be inferred. Moreover, both KGs are likely incomplete, or may use different relations or modelling schemes.

Related Tasks

The task of EA is related to the broader tasks of Link Discovery (LD) and Ontology Matching (OM). LD in its general form does not require any graph structure to be present, and defines the task as identification of all pairs of elements from two sets, whose distance is smaller than a threshold [162]. OM on the other hand requires finding a matching between multiple ontologies *on both* the schema and instance level. EA typically does not make use of any schema and thus is closer to instance matching, while requiring the graph structure, i.e., being more specific than link discovery. OM approaches [57, 75, 179, 19] often operate in the unsupervised setting, using a pre-defined similarity based on attributes or features, e.g., making use of string similarities. As with LD, the main objective is optimizing the matching process itself, e.g., by pruning the candidate space [162, 113]. In this aspect, it is closely related to filter-refinement approaches from kNN query processing [193, 18]. There are also approaches focussing on learning a linear [163], or tree-based [164] combination of such atomic distance functions based on user feedback. There are also recent approaches [101] which aim to apply supervised machine learning techniques, as well as EA methods which start to add schema matching as an additional supervised signal into EA approaches [230].

3.4 Knowledge Graph Representation Learning

Machine learning approaches for KG enrichment can be seen within a representation learning framework: The approaches learn to produce entity and relation representations and utilize a score function to score triples or alignment candidates based upon these representations. In the following, we discuss how to obtain such representations, before reviewing specific components for LP in Section 3.5, and EA in Section 3.6, respectively.

A representation of an entity or relation is a numeric representation, usually a real vector, which is adapted during training such that it can be used to predict observed behaviour with an appropriate score function. While frequently representations are real vectors, e.g., [37, 78, 129, 225], some approaches also use complex vectors [214, 207], vectors of quaternions [255], dual quaternions [45], real matrices [168, 38, 138], or higher-order tensors [201]. There are also approaches utilizing representations in hyperbolic spaces [13, 47], or Lie algebras [71].

Some approaches also use multiple representations for single entities or relations. For instance, SimplE [121] uses two separate vectors for a single entity modeling its properties as a head entity and as a tail entity separately. Moreover, it is a common practice to train KG embedding models using inverse relations [121, 65, 129], where each relation has two separate representations for predicting head entities and tail entities, respectively. Other models use multiple representations for a single entity or relation, e.g., to model mean and variance of a normal distribution [99], or transformation weights of a neural network [201], or the normal vector of a hyperplane and a translation therein [224].

In the following, we discuss several ways of obtaining representations, starting from *embeddings* or *features*, which are optionally enriched by GNN layers (cf. Section 3.4.4).

Since our primary focus lies in supervised tasks for KG enrichment, we leave out unsupervised representation learning approaches, such as DeepWalk [176], node2vec [90], metapath2vec [70], or RDF2Vec [186].

If not otherwise noted, we restrict the description to vectors in $\mathbb{R}^d$, and denote with $\mathbf{X}_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ the matrix of all entity representations, where $\mathbf{X}_{\mathcal{E}}[e] \in \mathbb{R}^{d_e}$ is the representation for entity $e \in \mathcal{E}$[1]. Analogously, we use $\mathbf{X}_{\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}| \times d_r}$ to denote the matrix of all relation representations.

3.4.1 Embeddings

An embedding is a simple form of representation, where the representation itself is a directly trainable parameter. While they are universally applicable since they do not require any features, they are also inherently *transductive* [95], i.e., they can only yield representations for entities that have been known while creating the embedding matrix.[2] Note that this does not imply that embeddings cannot be applied in any inductive approach when combined with other components, e.g., message passing [95].

3.4.2 Features

Since entities often refer to real-world entities, additional information about them may be available, which can be utilized to obtain "richer" representations. They can serve as a replacement or addition to transductive embeddings.

For instance, entity labels or textual descriptions in natural language are often available, e.g., for Wikidata [219]. A popular method to integrate this information [227, 78, 29] is to utilize pre-trained word or sentence embeddings, such as Glove vectors [175], or aggregated contextualized word embeddings from BERT [67, 182], e.g., in HMAN [243]. These featurization approaches come with some preprocessing and hyperparameter choices, such as, e.g., extracting entity labels from their URIs, tokenization methods, or choosing pre-trained model weights. For BERT, other choices include the layer from which the latent representations are chosen. Also pooling strategies are needed in case there are varying number of tokens per label.

In other cases, such as for scene graphs, which are dynamic knowledge graphs describing entities and their relations within an image, visual information is available, e.g., in the form of images [249], or depth maps [195]. Feature vectors from these kinds of inputs can be created by extracting feature maps from pre-trained CNN models, e.g., ResNet [97].

Even if there are no additional data sources, graph-based features can be extracted from the relational information, such as Laplacian eigenvectors [20], subgraph counts [40], or personalized page rank vectors [39].

[1] Here, and in the following, we assume that the entities have already been converted to integer IDs.

[2] Surprisingly, some benchmark datasets for (transductive) LP on KG do have entities in the validation/test split, which did not occur in the training set. Due to the dependency of commonly used evaluation metrics on the number of candidates, cf. [25], this can lead to performance differences between different implementations, cf. [64].

3.4.3 Relation Representations from Entity Representations

Some approaches from the field of EA create relation representations from entity representations.

- MRAEA [149] and RDGCN [227] create relation representations by concatenation of the average head and tail entity representation of triples containing the relation:

$$\mathbf{X}_{\mathcal{R}}[r] = \left[\frac{1}{\left|\mathcal{E}^h_{\mathcal{T}_{(\cdot,r,\cdot)}}\right|} \sum_{h \in \mathcal{E}^h_{\mathcal{T}_{(\cdot,r,\cdot)}}} \mathbf{X}_{\mathcal{E}}[h]; \frac{1}{\left|\mathcal{E}^t_{\mathcal{T}_{(\cdot,r,\cdot)}}\right|} \sum_{t \in \mathcal{E}^h_{\mathcal{T}_{(\cdot,r,\cdot)}}} \mathbf{X}_{\mathcal{E}}[t] \right]$$

- AliNet [206] follows the idea of TransE [37] and obtains relation representations as the average difference between head and tail entity representation for triples containing the relation:

$$\mathbf{X}_{\mathcal{R}}[r] = \frac{1}{|\mathcal{T}_{(\cdot,r,\cdot)}|} \sum_{(h,r,t) \in \mathcal{T}_{(\cdot,r,\cdot)}} \mathbf{X}_{\mathcal{E}}[h] - \mathbf{X}_{\mathcal{E}}[t]$$

3.4.4 Graph Neural Networks

In contrast to traditional neural networks, where often independent and identically distributed (i.i.d.) samples are assumed and thus processed independently, Graph Neural Networks (GNNs) can utilize relations between samples given as explicit links. Recent years have witnessed a flurry of works of representation learning for graphs, with the majority of works focussing on uni-relational graphs. A general framework for GNNs is the message passing formulation [87, 17, 237].[3] Here, we present a slight adaption explicitly coined on KGs and the terminology we introduced before. We denote with $\mathbf{X}_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ and $\mathbf{X}_{\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}| \times d_r}$ the input entity and relation representations, and consider a sequence of message passing layer, each comprising three steps:

$$\mathbf{M}^{k+1}[(h,r,t)] = \text{message}(\mathbf{X}^k_{\mathcal{E}}[h], \mathbf{X}^k_{\mathcal{R}}[r], \mathbf{X}^k_{\mathcal{E}}[t]) \qquad \forall (h,r,t) \in \mathcal{T} \quad (3.6)$$

$$\mathbf{Y}^{k+1}_E[e] = \text{aggregate}(\{\mathbf{M}^{k+1}[(h,r,t)] \mid (h,r,t) \in \mathcal{T}, t = e\}) \qquad \forall e \in \mathcal{E} \quad (3.7)$$

$$\mathbf{X}^{k+1}_E[e] = \text{update}(\mathbf{X}^k_E[e], \mathbf{Y}^{k+1}_E[e]) \qquad \forall e \in \mathcal{E} \quad (3.8)$$

In the first step, (3.6), the message function, message, uses the representations of the head and tail entity, and the relation representation, and composes messages $\mathbf{M}^{k+1}[(h,r,t)] \in \mathbb{R}^{d_m}$ for each triple $(h,r,t) \in \mathcal{T}$. In practice, this operation can be efficiently implemented by an index-based lookup in the matrices $\mathbf{X}_{\mathcal{E}}$ and $\mathbf{X}_{\mathcal{R}}$, and a vectorized application of the message function. The result is a matrix of messages $\mathbf{M} \in \mathbb{R}^{|\mathcal{T}| \times d_m}$, which is also called the edge-parallel representation [77]. Next, in (3.7), for each entity $e \in \mathcal{E}$, we aggregate the messages with target e into a single vector representation, resulting in

[3] There are also other views on GNNs, e.g., as a generalization of convolutions from grids over manifolds to graphs [42].

an proposed update $\mathbf{Y}_{\mathcal{E}}^{k+1}[e] \in \mathbb{R}^{d_u}$ for each entity. An essential requirement for the aggregation function is that it receives a *set* of vectors and aggregates it to a single, fixed-size representation. Finally, in (3.8), an update method computes the updated entity representation $\mathbf{X}_{\mathcal{E}}^{k}[e]$ based on the old representation $\mathbf{X}_{\mathcal{E}}^{k}[e]$, and the proposed update $\mathbf{Y}_{\mathcal{E}}^{k+1}[e]$. Some approaches [190, 136] apply the same weights in each message passing round, resembling a recurrent neural network. Thereby, they also enable a different number of iterations for each sample or batch. Most recent variants do not use this variant of weight sharing though. In the following paragraphs, we review message, aggregation, and update mechanisms used in the field of LP and EA for KGs.

Message

The message function

$$\text{message} : \mathbb{R}^{d_e} \times \mathbb{R}^{d_r} \times \mathbb{R}^{d_e} \to \mathbb{R}^{d_m}, (\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) \mapsto \mathbf{m}_{(h,r,t)}$$

composes a message vector $\mathbf{m} \in \mathbb{R}^{d_m}$ from the representations of the head entity $\mathbf{x}_h \in \mathbb{R}^{d_e}$, the relation $\mathbf{x}_r \in \mathbb{R}^{d_r}$, and the tail entity $\mathbf{x}_t \in \mathbb{R}^{d_e}$. While non-parametric message functions are possible, in the context of representation learning for (knowledge) graphs, the message function usually has internal trainable parameters updated during training. One notable exception is GCNAlign [225], which discusses a transformation weight in the paper, but does not employ this weight in their experiments, cf. [24].

The following message functions are encountered in the literature. Most of them are linear transformations, where the variants differ in the transformation weight and the participating representations.

- GCNAlign [225], RDGCN [227], KECG [132], HGCN [228], HMAN [243], AliNet [206], and RE-GCN [244] directly apply the message function of the uni-relational GCN [122], a learned linear transformation of the head representation, ignoring the multi-relational nature of KGs (at least in this stage of the message passing mechanism). The message function is given as

  $$\text{message}(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \mathbf{W}\mathbf{x}_h$$

  where $\mathbf{W} \in \mathbb{R}^{d_m \times d_e}$ is a trainable weight. RDGCN [227], and KECG [132] additionally constrain $\mathbf{W}$ to be a diagonal matrix.

- GMN [239] and DGMC [78] use a direction-dependent linear head transformation, e.g.,

  $$\text{message}(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \mathbf{W}_{\text{dir}(r)}\mathbf{x}_h$$

  The former message function can thus be seen as a special case of this function, where the weights are shared for all directions.

- SACN [194] employs a relation-specific scalar scaling of a relation-independent weight matrix,

  $$\text{message}(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \alpha_r \mathbf{W}\mathbf{x}_h$$

where $\alpha_r \in \mathbb{R}$ is a trainable, relation-specific scalar weight, and $\mathbf{W} \in \mathbb{R}^{d_m \times d_e}$ is a trainable relation-independent weight matrix.

- RGCN [191] uses a relation-specific linear transformation of the head representation,

$$\text{message}(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \mathbf{W}_r \mathbf{x}_h$$

with a trainable relation-specific weight matrix $\mathbf{W}_r \in \mathbb{R}^{d_m \times d_e}$. Since the weight is relation-dependent, we can also regard it as a part of the relation representation, and $\mathbf{W}_r = \textit{reshape}(\mathbf{x}_r)$ with $d_r = d_e \cdot d_m$. RGCN [191] further proposes two factorization variants, basis decomposition and block decomposition, to reduce the number of trainable weights.

- A2N [15] uses a linear transformation of the concatenation of relation and head representation:

$$\text{message}(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \mathbf{W}[\mathbf{x}_r; \mathbf{x}_h]$$

where $\mathbf{W} \in \mathbb{R}^{d_m \times (d_r + d_e)}$ is a trainable parameter, and $[\cdot;\cdot]$ denotes the concatenation of vectors.

- Graph2Seq [134], or KBGAT [160][4] use a linear transformation of concatenation of head, relation and tail representation, :

$$\text{message}(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \mathbf{W}[\mathbf{x}_h; \mathbf{x}_r; \mathbf{x}_t]$$

where $[\cdot;\cdot]$ denotes the concatenation of vectors, and $\mathbf{W} \in \mathbb{R}^{d_m \times (2 \cdot d_e + d_r)}$ is a trainable parameter.

- VR-GCN [247], TransGCN [43], and RAGAT [143] use a linear transformation of the *composition* of head and relation representation

$$\text{message}(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \mathbf{W}\, \phi(\mathbf{x}_h, \mathbf{x}_r)$$

where $\mathbf{W} \in \mathbb{R}^{d_m \times d_h}$ is a trainable weight, and $\phi : \mathbb{R}^{d_e} \times \mathbb{R}^{d_r} \to \mathbb{R}^{d_h}$ is a *composition function*. The composition functions are often inspired by interaction functions of KG LP models, e.g., subtraction in VR-GCN [247], Hadamard product in REA [172], addition or Hadamard product in TransGCN [43]. RAGAT [143] extends the aforementioned composition functions by also considering concatenation, rotation, and cross-correlation.

- CompGCN [216] uses composition functions, but also employs direction-specific transformation weight $\mathbf{W}_{\text{dir}(r)} \in \mathbb{R}^{d_m \times d_h}$

$$\text{message}(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \mathbf{W}_{\text{dir}(r)}\, \phi(\mathbf{x}_h, \mathbf{x}_r)$$

[4] KBGAT [160] uses a different order in the concatenation: (h, t, r) instead of (h, r, t). Since the weight matrix $\mathbf{W}$ is learned, this is an equivalent formulation.

- GMN [239] use an Multi-Layer Perceptron (MLP) to compute the messages based solely on the head representation

$$\text{message}(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = f(\mathbf{x}_h)$$

  where f denotes an MLP with trainable weights.

- RREA [150] uses a reflection matrix constructed from $\mathbf{x}_r$ to ensure an isometric transformation with $\|\mathbf{x}_r\| = 1$

$$\text{message}(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = (\mathbf{I} - \mathbf{x}_r\mathbf{x}_r^T)\mathbf{x}_h = \mathbf{x}_h - \mathbf{x}_r\mathbf{x}_r^T\mathbf{x}_h$$

Aggregation

The aggregation function aggregate is used to combine the variable number of incoming messages for a single entity $e \in \mathcal{E}$ into a single fixed-size representation $\mathbf{y}_e \in \mathbb{R}^{d_u}$. While aggregation functions such as element-wise maximum, used, e.g, in GraphSAGE [95], or more general approaches such as DeepSets [248], or GIN [237], are possible, the majority of methods can be seen as weighted sum

$$\text{aggregate}(\mathcal{M}) = \sum_{\mathbf{m}_{(h,r,t)} \in \mathcal{M}} \alpha_{(h,r,t)} \mathbf{m}_{(h,r,t)} \tag{3.9}$$

where we can distinguish between *static* weights, which are solely based on, e.g., the graph structure, and do not change during training, and dynamic weights, e.g., from attention mechanisms, which depend on the entities' or relations' representations.

Static Weights Static message weights can be efficiently precomputed and often serve to stabilize the message passing process, e.g., by limiting the influence of high-degree nodes or avoiding that the vector length of representations grows proportional to their degree. For uni-relational graphs, they also have a close connection to matrix normalization and the spectrum of a graph's Laplacian, cf., e.g., [122].

- The simplest static weight method is to use uniform weights for all messages, i.e.,

$$\alpha_{(h,r,t)} = 1$$

  While this aggregation does not solve the problems with high-degree nodes, it allows the message passing scheme, e.g., to count [237], which is not possible with some of the later aggregations.

- Mean aggregation (or target normalization) normalizes the message weights such that the weights of all incoming messages for a single entity sum up to one.

$$\alpha_{(h,r,t)} = \deg^{in}(t)^{-1}$$

where $\deg^{in}(e)$ denotes the *in-degree* of an entity $e \in \mathcal{E}$, i.e., the number of triples where the tail equals e:

$$\deg^{in}(e) = \left|\mathcal{T}_{(\cdot,\cdot,e)}\right| = \left|\{(h,r,t) \in \mathcal{T} \mid t = e\}\right|$$

It is used, e.g., by DGMC [78], and GMN [239].

- For relation-specific mean aggregation as used by, e.g., RGCN [191], the in-degree is computed for each relation independently

$$\deg_r^{in}(e) = \left|\mathcal{T}_{(\cdot,r,e)}\right|,$$

and thus the weight becomes

$$\alpha_{(h,r,t)} = \deg_r^{in}(t)^{-1}$$

- GCN [122] employs a symmetric normalization scheme, where the message weight is the inverse of the geometric mean of the head and tail entities' undirected degree,

$$\alpha_{(h,r,t)} = \frac{1}{\sqrt{\deg(h) \cdot \deg(t)}}$$

where $\deg(e)$ denotes the undirected degree[5] of an entity $e \in \mathcal{E}$, i.e.

$$\deg(e) = \left|\mathcal{T}_{(e,\cdot,\cdot)} \cup \mathcal{T}_{(\cdot,\cdot,e)}\right|$$

The same message weighting is also used by RDGCN [227] (in the second message passing phase), and HMAN [243].

- GCN-Align [225] defines relation-specific message weights

$$\alpha_{(h,r,t)} = \mathit{func}(r) + \mathit{ifunc}(r)$$

where *func* and *ifunc* are used to denote the *functionality* and *inverse functionality* of a relation, defined as the number of different head / tail entities divided by the number of triples in which this relation occurs, i.e.,

$$\mathit{func}(r) = \frac{\left|\mathcal{E}^h_{\mathcal{T}_{(\cdot,r,\cdot)}}\right|}{\left|\mathcal{T}_{(\cdot,r,\cdot)}\right|} \qquad \mathit{ifunc}(r) = \frac{\left|\mathcal{E}^t_{\mathcal{T}_{(\cdot,r,\cdot)}}\right|}{\left|\mathcal{T}_{(\cdot,r,\cdot)}\right|}$$

Dynamic Weights Dynamic weights are computed based on learned representations. Thus, they change during training and can amplify or attenuate messages based on the involved entities and relations. Often, dynamic weights employ an attention-mechanism [217], where the weights of messages with the same target entity are normalized using the softmax operation. In the following, $\alpha_{(h,r,t)}$ denotes scores which are *not* normalized afterwards, while $\alpha'_{(h,r,t)}$ denotes pre-softmax scores, i.e., the final weight are given as $\alpha_{(h,r,t)} = \mathrm{softmax}_{\mathcal{T}_{(h,r,\cdot)}}(\alpha'_{(h,r,t)})$ with the softmax being evaluated over the scores for all triples sharing the same head and relation.

[5] Notice that GCN [122] also adds self-loops beforehand.

- A2N [15] computes message weights via a LP interaction function, DistMult, cf., Section 3.3.1, which combines head and relation representation by Hadamard product, denoted as $\odot$, before computing the inner product of the result with the tail representation.

$$\alpha_{(h,r,t)} = \mathit{DistMult}(\mathbf{X}_{\mathcal{E}}[h], \mathbf{X}_{\mathcal{R}}[r]\mathbf{X}_{\mathcal{E}}[t]) = (\mathbf{X}_{\mathcal{E}}[h] \odot \mathbf{X}_{\mathcal{R}}[r])^T(\mathbf{X}_{\mathcal{E}}[t])$$

- SACN/WGCN [194] uses a relation-specific, directly trainable weight $\alpha_r \in \mathbb{R}$:

$$\alpha_{(h,r,t)} = \alpha_r$$

- RDGCN [227] uses (in the first part of the architecture, the *primal-dual attention*) relation-specific message weights, which are predicted from the relation representations $\mathbf{X}_{\mathcal{R}}$ by a trainable linear projection weight $\mathbf{w} \in \mathbb{R}^{d_r}$:

$$\alpha'_{(h,r,t)} = \sigma(\mathbf{w}^T\mathbf{X}_{\mathcal{R}}[r])$$

- GAT [218], and KECG [132] compute attention-scores by first transforming head and tail entity representations by a shared weight matrix $\mathbf{W} \in \mathbb{R}^{d_h \times d_e}$, then concatenating them into a single vector, and computing the dot product with another learned weight $\mathbf{a} \in \mathbb{R}^{2d_h}$. Finally, a LeakyReLU [144] activation is applied.

$$\alpha'_{(h,r,t)} = \text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{X}_{\mathcal{E}}[h]; \mathbf{W}\mathbf{X}_{\mathcal{E}}[t]])$$

- AliNet [206] employs a different variant, where two learned matrices $\mathbf{W}_h, \mathbf{W}_t \in \mathbb{R}^{d_h \times d_e}$ are used to linearly transform the head and tail entity representation, before computing their dot product, and applying a LeakyReLU activation.

$$\alpha'_{(h,r,t)} = \text{LeakyReLU}((\mathbf{W}_h\mathbf{X}_{\mathcal{E}}[h])^T(\mathbf{W}_t\mathbf{X}_{\mathcal{E}}[t]))$$

- G2SKGE [134] and RAGAT [143] obtain message weights from the respective messages by inner product with a learned linear weight $\mathbf{w} \in \mathbb{R}^{d_m}$ and application of a LeakyReLU activation:

$$\alpha'_{(h,r,t)} = \text{LeakyReLU}(\mathbf{w}^T\mathbf{m}_{(h,r,t)})$$

  RAGAT additionally makes use of *multi-head attention*, where multiple independent copies of the attention mechanism are used, and their aggregation results are combined via sum.

- MRAEA [149] uses multi-head *relation-aware self-attention*, where for a single head, the unnormalized scores are given by

$$\alpha'_{(h,r,t)} = \text{LeakyReLU}\left(\mathbf{w}^T\left[\mathbf{X}_{\mathcal{E}}[h]; \mathbf{X}_{\mathcal{E}}[t]; \frac{1}{\left|\mathcal{R}_{\mathcal{T}_{(h,\cdot,t)}}\right|}\sum_{r' \in \mathcal{R}_{\mathcal{T}_{(h,\cdot,t)}}} \mathbf{X}_{\mathcal{R}}[r']\right]\right)$$

  with $\mathcal{R}_{\mathcal{T}_{(h,\cdot,t)}}$ denoting the set of relations connecting the head and tail entity, and a trainable weight $\mathbf{w} \in \mathbb{R}^{2d_e+d_r}$.

Update

The *update* function[6] is used to combine the current entity representation $\mathbf{x} \in \mathbb{R}^{d_e}$ with the aggregated messages $\mathbf{y} \in \mathbb{R}^{d_u}$ to a new entity representation in $\mathbb{R}^{d'_e}$, where, depending on the update function, $d_e = d_u = d'_e$ is often required. Thus, its general form is given by

$$\text{update} : \mathbb{R}^{d_e} \times \mathbb{R}^{d_u} \to \mathbb{R}^{d'_e}, (\mathbf{x}, \mathbf{y}) \mapsto \mathbf{z} \tag{3.10}$$

The following choices are common in literature. Some of them, particularly those resembling skip connections, have been discussed as a solution to the over-smoothing problem [133, 141].

- Only keep the aggregated messages and ignore the previous representation. [225, 191, 132][7]
$$\text{update}(\mathbf{x}, \mathbf{y}) = \mathbf{y}$$
- Addition of the old state and the proposed update,
$$\text{update}(\mathbf{x}, \mathbf{y}) = \mathbf{x} + \mathbf{y}$$
which is equivalent to residual connections [97], and used, e.g., in TransGCN [43].
- Weighted addition with fixed weights $\alpha, \beta \in \mathbb{R}$
$$\text{update}(\mathbf{x}, \mathbf{y}) = \alpha\mathbf{x} + \beta\mathbf{y}$$
RDGCN [227] sets $\alpha = 1$ and $\beta \in \{0.3, 0.1\}$ depending on the layer. Notably, it also adds skip connections to the first layer, which contains entity features before message passing.
- Highway Layer [202]: a convex combination of $\mathbf{x}$ and $\mathbf{y}$, where the weight is obtained from a (trainable) gate function g applied to $\mathbf{x}$ followed by a sigmoid activation σ:
$$\text{update}(\mathbf{x}, \mathbf{y}) = \sigma(g(\mathbf{x}))\mathbf{x} + (1 - \sigma(g(\mathbf{x})))\mathbf{y}$$
A common choice for the gate is a single linear layer:
$$g(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}.$$
Highway layers are, for instance, used by AliNet [206].
- KBAT [160] and SACN [194] both apply another learned transformation $\mathbf{W} \in \mathbb{R}^{d_u \times d_e}$ to the old representation before combination by sum.
$$\text{update}(\mathbf{x}, \mathbf{y}) = \mathbf{W}\mathbf{x} + \mathbf{y}$$
This allows different dimensions $d_e \neq d_u$, and resembles what is used in residual blocks of CNNs [97].

[6] Also called *combine*, cf. [237].
[7] The previous representation can still be considered, e.g., since self-loops have been introduced.

- A2N [15] uses a learned linear transformation $\mathbf{W} \in \mathbf{R}^{d_e \times (d_e + d_u)}$ of the concatenation of the old entity representation and proposed update

$$\text{update}(\mathbf{x}, \mathbf{y}) = \mathbf{W}[\mathbf{x}; \mathbf{y}]$$

- Gated Graph Neural Network (GGNN) [136] use an update mechanism resembling an LSTM [105] layer:

$$\begin{aligned} \mathbf{z} &= \sigma\left(\mathbf{W}^z \mathbf{y} + \mathbf{U}^z \mathbf{x}\right) \\ \mathbf{r} &= \sigma\left(\mathbf{W}^r \mathbf{y} + \mathbf{U}^r \mathbf{x}\right) \\ \tilde{\mathbf{y}} &= \tanh\left(\mathbf{W}\mathbf{y} + \mathbf{U}\left(\mathbf{r} \odot \mathbf{x}\right)\right) \\ \text{update}(\mathbf{x}, \mathbf{y}) &= \left(1 - \mathbf{z}\right) \odot \mathbf{x} + \mathbf{z} \odot \tilde{\mathbf{y}} \end{aligned}$$

  with trainable weights $\mathbf{U}, \mathbf{U}^z, \mathbf{U}^r, \mathbf{W}, \mathbf{W}^z, \mathbf{W}^r \in \mathbb{R}^{d_e \times d_e}$.

- GMN [239] and GIN [237] also use a concatenation of the old representation and the proposed update, but instead of only applying a linear transformation, they utilize a MLP f:

$$\text{update}(\mathbf{x}, \mathbf{y}) = f([\mathbf{x}; \mathbf{y}])$$

Some methods, e.g., AliNet [206], also combine the outputs of several layers to obtain multi-scale representations.

3.5 Link Prediction

Until now, the representation learning part has not depended on the KG enrichment task. However, in order to train the representations and extract scores, both tasks require task-specific decoders. In this section, we discuss specifics of LP approaches.

3.5.1 Interaction Functions

Decoder for LP, also called interaction functions [5], combine head entity, relation and tail entity representations to a scalar score representing a measure of plausibility[8] of a triple. If not specified otherwise, all involved (bold-font) symbols are vectors from $\mathbb{R}^d$. We first present a multitude of interaction functions before discussing them subsequently.

- DistMA [197]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \mathbf{x}_h^T \mathbf{x}_r + \mathbf{x}_r^T \mathbf{x}_t + \mathbf{x}_h^T \mathbf{x}_t$$

- DistMult [242]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \langle \mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t \rangle$$

  where $\langle \cdot, \cdot, \cdot \rangle$ denotes the tri-linear dot product.

[8] not necessarily interpretable as probability

- ComplEx [214],
$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \Re\left(\langle \mathbf{x}_h, \mathbf{x}_r, \overline{\mathbf{x}_t}\rangle\right)$$
where $\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t \in \mathbb{C}^d$, $\Re$ denotes the operation which retrieves the real part of a complex number, and $\overline{\ }$ the complex conjugate.

- RESCAL [168], LFM [115], ANALOGY [140], and DihedralE [236] use the same interaction function
$$f(\mathbf{x}_h, \mathbf{X}_r, \mathbf{x}_t) = \mathbf{x}_h^T \mathbf{X}_r \mathbf{x}_t$$
with $\mathbf{X}_r \in \mathbb{R}^{d \times d}$. ANALOGY additionally enforces normality and commutativity constraints by restricting the relation matrices $\mathbf{X}_r$ to be almost diagonal, i.e., block-diagonal where each block has at most two rows. DihedralE restricts the matrix to be block diagonal, where the blocks comprise elements of the dihedral group. The selection of individual group members is made differentiable via the Gumbel-softmax trick [114].

- TATEC [86]
$$f((\mathbf{x}_h, \mathbf{y}_h), (\mathbf{X}_r, \mathbf{y}_r, \mathbf{z}_r), (\mathbf{x}_t, \mathbf{y}_t)) = \mathbf{x}_h^T \mathbf{X}_r \mathbf{x}_t + \mathbf{y}_r^T \mathbf{y}_h + \mathbf{z}_r^T \mathbf{y}_t + \mathbf{y}_h^T \mathbf{W} \mathbf{y}_t$$
where $\mathbf{X}_r \in \mathbb{R}^{d_e \times d_e}$, and $\mathbf{y}_h, \mathbf{y}_r, \mathbf{z}_r, \mathbf{y}_t \in \mathbb{R}^{d'_e}$.

- TuckER [14]
$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \mathbf{W} \times_1 \mathbf{x}_h \times_2 \mathbf{x}_r \times_3 \mathbf{x}_t$$
where the *core tensor* $\mathbf{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is a global trainable weight, and $\times_i$ denotes the tensor-vector product along the i-th mode. In their implementation, they also introduce several dropout [104] and batch normalization [112] layers.

- SimplE [121]
$$f((\mathbf{x}_h, \mathbf{y}_h), (\mathbf{x}_r, \mathbf{y}_r), (\mathbf{x}_h, \mathbf{y}_t)) = \frac{1}{2}\left(\langle \mathbf{x}_h, \mathbf{x}_r, \mathbf{y}_h\rangle + \langle \mathbf{x}_h, \mathbf{y}_r, \mathbf{y}_h\rangle\right)$$

- TransA [231]
$$f(\mathbf{x}_h, (\mathbf{x}_r, \mathbf{W}_r), \mathbf{x}_t) = (\mathbf{x}_h + \mathbf{x}_r - \mathbf{x}_t)^T \mathbf{W}_r (\mathbf{x}_h + \mathbf{x}_r - \mathbf{x}_t)$$

- TransF [76]
$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = (\mathbf{x}_h + \mathbf{x}_r)^T \mathbf{x}_t + \mathbf{x}_h^T (\mathbf{x}_t - \mathbf{x}_r)$$

- HolE [167]
$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \sigma((\mathbf{x}_h \star \mathbf{x}_t)^T \mathbf{x}_r)$$
where $\star$ denotes the circular correlation.

- HolEx [240]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \sum_{k=1}^{K} ((\mathbf{c}_k \odot \mathbf{x}_h) \star \mathbf{x}_r)^T \mathbf{x}_t$$

  where $\mathbf{c}_k \in \mathbb{R}^{d_e}$ are fixed vectors, either chosen as low-frequency Haar vectors, or random 0/1 vectors, and $\star$ denotes the circular correlation.

- Unstructured Model (UM) / Semantic Matching Energy (SME) [36]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = -\|\mathbf{x}_h - \mathbf{x}_t\|_2^2$$

- Structured Embedding (SE) [38]

$$f(\mathbf{x}_h, (\mathbf{X}_r^h, \mathbf{X}_r^t), \mathbf{x}_t) = -\|\mathbf{X}_r^h \mathbf{x}_h - \mathbf{X}_r^t \mathbf{x}_t\|$$

  where $\mathbf{X}_r^h, \mathbf{X}_r^t \in \mathbb{R}^{d \times d}$.

- TransD [116]

$$f((\mathbf{x}_h, \mathbf{p}_h), (\mathbf{x}_r, \mathbf{p}_r), (\mathbf{x}_t, \mathbf{p}_t)) = -\|(\mathbf{p}_r \mathbf{p}_h^T + \tilde{\mathbf{I}})\mathbf{x}_h + \mathbf{x}_r - (\mathbf{p}_r \mathbf{p}_t^T + \tilde{\mathbf{I}})\mathbf{x}_t\|_2^2$$

  where $\mathbf{x}_r, \mathbf{p}_r \in \mathbb{R}^{d_r}$ and $\tilde{\mathbf{I}} \in \mathbb{R}^{d_r \times d_e}$ is a rectangular "identity matrix", i.e., a rectangular matrix filled by zeros except on the diagonal where the entries are one.

- TransE [37]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = -\|\mathbf{x}_h + \mathbf{x}_r - \mathbf{x}_t\|$$

- TransM [73]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = -\alpha_r \|\mathbf{x}_h + \mathbf{x}_r - \mathbf{x}_t\|_p$$

  where $\alpha_r = \log\left(hpt_r + tph_r\right)^{-1}$ with $hpt_r = \frac{|\{(h,r',t) \in \mathcal{T} | r' = r\}|}{|\{t \in \mathcal{E} | \exists h \in \mathcal{E} : (h,r,t) \in \mathcal{T}\}|}$ and $tph_r = \frac{|\{(h,r',t) \in \mathcal{T} | r' = r\}|}{|\{h \in \mathcal{E} | \exists t \in \mathcal{E} : (h,r,t) \in \mathcal{T}\}|}$.

- TransH [224]

$$f(\mathbf{x}_h, (\mathbf{x}_r, \mathbf{p}_r), \mathbf{x}_t) = -\|(\mathbf{x}_h - \mathbf{p}_r^T \mathbf{x}_h \mathbf{p}_r) + \mathbf{x}_r - (\mathbf{x}_t - \mathbf{p}_r^T \mathbf{x}_t \mathbf{p}_r)\|_2^2$$

  where $\mathbf{x}_r, \mathbf{p}_r \in \mathbb{R}^{d_r}$.

- TransR [138] and TranSparse [117]

$$f(\mathbf{x}_h, (\mathbf{x}_r, \mathbf{W}_r), \mathbf{x}_t) = -\|\mathbf{W}_r \mathbf{x}_h + \mathbf{x}_r - \mathbf{W}_r \mathbf{x}_t\|_2^2$$

  with $\mathbf{x}_r \in \mathbb{R}^{d_r}$ and $\mathbf{W}_r \in \mathbb{R}^{d_r \times d_e}$. TranSparse additionally restricts the matrices $\mathbf{W}_r, \mathbf{W}_r$ to a fixed sparsity pattern. The sparsity pattern is either band-diagonal, or random.

- ITransF [234]

$$f(\mathbf{x}_h, (\mathbf{a}_r^h, \mathbf{a}_r^t, \mathbf{x}_r), \mathbf{x}_t) = -\|(\text{softmax}(\mathbf{a}_r^h) \times_1 \mathbf{W})\mathbf{x}_h + \mathbf{x}_r - (\text{softmax}(\mathbf{a}_r^t) \times_1 \mathbf{W})\mathbf{x}_t\|$$

where $\mathbf{W} \in \mathbf{R}^{d_h \times d_r \times d_e}$ is a shared weight, $\times_1$ denotes the multiplication along the first mode of the tensor, and $\mathbf{a}_r^h, \mathbf{a}_r^t \in \mathbb{R}^{d_h}$.

- TransAt [180]

$$f(\mathbf{x}_h, (\mathbf{x}_r, \mathbf{x}_r^h, \mathbf{x}_r^t), \mathbf{x}_t) = -\|\mathbf{a}_r^h \odot \sigma(\mathbf{x}_r^h) \odot \mathbf{x}_h + \mathbf{x}_r - \mathbf{a}_r^t \odot \sigma(\mathbf{x}_r^t) \odot \mathbf{x}_t\|$$

where σ is the (element-wise) sigmoid function, $\mathbf{a}_r^h, \mathbf{a}_r^t \in \{0, 1\}^e$ is a non-trainable weight, whether the variance in dimension i is at least half the maximum variance, evaluated only on the set of head/tail candidates. These sets are given as the union of k-means clusters in the embedding space, containing at least one training head/tail entity of the given relation. $\mathbf{a}$ is updated every 100 epochs.

- TransMS [246]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \| - \tanh(\mathbf{x}_t \odot \mathbf{x}_r) \odot \mathbf{x}_h + \mathbf{x}_r + \alpha \cdot (\mathbf{x}_h \odot \mathbf{x}_t) - \tanh(\mathbf{x}_h \odot \mathbf{x}_r) \odot \mathbf{x}_t\|_p$$

where $p \in \{1, 2\}$ and $\alpha \in \mathbb{R}$ is a trainable parameter.

- MuRE [13, 9]

$$f((\mathbf{x}_h, \beta_h), (\mathbf{x}_r^\odot, \mathbf{x}_r^+), (\mathbf{x}_t, \beta_t)) = -\|\mathbf{x}_r^\odot \odot \mathbf{x}_h + \mathbf{x}_r^+ - \mathbf{x}_t\|_2^2 + \beta_h + \beta_t$$

with additional trainable scalar head and tail offsets $\beta_h, \beta_t \in \mathbb{R}$.

- RotatE [207] and ModE [258] use the same interaction function:

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = -\|\mathbf{x}_h \odot \mathbf{x}_r - \mathbf{x}_t\|_2$$

For RotatE, $\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t \in \mathbb{C}^d$, the relation representation being element-wise normalized to unit length $|(\mathbf{x}_r)_i| = 1$. ModE uses real vectors instead.

- pRotatE [207]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = -\|\sin((\mathbf{x}_h + \mathbf{x}_r - \mathbf{x}_t)/2)\|_1$$

- PairRE [49]

$$f(\mathbf{x}_h, (\mathbf{x}_r^h, \mathbf{x}_r^t), \mathbf{x}_t) = -\|\mathbf{x}_h \odot \mathbf{x}_r^h - \mathbf{x}_t \odot \mathbf{x}_r^t\|$$

- HAKE [258]

$$f((\mathbf{x}_h^m, \mathbf{x}_h^p), (\mathbf{x}_r^m, \mathbf{x}_r^p), (\mathbf{x}_t^m, \mathbf{x}_t^p)) = -\|\mathbf{x}_h^m \odot \mathbf{x}_r^m - \mathbf{x}_t^m\|_2 - \lambda\|\sin((\mathbf{x}_h^p + \mathbf{x}_r^p - \mathbf{x}_t^p)/2)\|_1$$

- ManifoldE [232]

$$f(\mathbf{x}_h, (\mathbf{x}_r, \gamma_r), \mathbf{x}_t) = -\|g(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) - \gamma_r\|^2$$

where g is a manifold function (essentially equivalent to an interaction function). The paper investigates spheres, where g is the TransE interaction function, and hyperplanes, where

$$g(\mathbf{x}_h, (\mathbf{x}_r^h, \mathbf{x}_r^t), \mathbf{x}_t) = K(\mathbf{x}_r + \mathbf{x}_r^h, \mathbf{x}_t + \mathbf{x}_r^t)$$

for K being a (linear) kernel.

- TorusE [71]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = -\|\psi(\mathbf{x}_h + \mathbf{x}_r - \mathbf{x}_t)\|_p^p$$

where $\psi(x) = \min\{x - \lfloor x \rfloor, \lceil x \rceil - x\}$ is applied element-wise.

- QuatE [255], QctionE [255] and DualE [45] use the same interaction function

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \langle \mathbf{x}_h \odot \tilde{\mathbf{x}}_r, \mathbf{x}_t \rangle$$

with $\odot$ denoting the Hadamard product and $\tilde{}$ the dimension-wise normalization to unit length. They differ in the representation space: QuatE uses vectors of (Hamilton) quaternations, OctionE vectors of octonions, and DualE vectors of dual quaternions.

- 5* [161]

$$f(\mathbf{x}_h, (\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r, \mathbf{d}_r), \mathbf{x}_t) = \Re\left(\left\langle \frac{\mathbf{a}_r \odot \mathbf{x}_h + \mathbf{b}_r}{\mathbf{c}_r \odot \mathbf{x}_h + \mathbf{d}_r}, \overline{\mathbf{x}_t} \right\rangle\right)$$

where $\mathbf{x}_h, \mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r, \mathbf{d}_r, \mathbf{x}_t \in \mathbb{C}^d$, $\odot$ denoting the Hadamard product, the fraction is to be understood element-wise, $\overline{\ }$ denoting the complex conjugate, and $\Re$ denoting the operator which retrieves the real part of a complex number.

- MuRP [13]

$$f((\mathbf{x}_h, \beta_h), (\mathbf{x}_r^{\odot}, \mathbf{x}_r^{+}), (\mathbf{x}_t, \beta_t)) = -d^{\gamma}(\mathbf{x}_r^{\odot} \odot^{\gamma} \mathbf{x}_h, \mathbf{x}_t \oplus^{\gamma} \mathbf{x}_r^{+})^2 + \beta_h + \beta_t$$

with trainable scalar head and tail biases $\beta_h, \beta_t \in \mathbb{R}$, $\odot^{\gamma}$ denoting the Möbius element-wise multiplication, $\oplus^{\gamma}$ Möbius addition, and d^{γ} the hyperbolic distance. In their experiments, they always set the curvature $\gamma = 1$.

- RefH [47]

$$f((\mathbf{x}_h, \beta_h), (\mathbf{x}_r, \Psi_r, \gamma_r), (\mathbf{x}_t, \beta_t)) = -d^{\gamma_r}((\mathit{Ref}(\Psi_r) \otimes^{\gamma_r} \mathbf{x}_h \oplus^{\gamma_r} \mathbf{x}_r), \mathbf{x}_t)^2 + \beta_h + \beta_t$$

where $\oplus^{\gamma}, d^{\gamma}$ denotes the Möbius addition, and hyperbolic distance in hyperbolic space of curvature $\gamma < 0$, and *Ref* is a relation-specific reflection matrix, which is a block-diagonal matrix with 2×2 blocks given as

$$\begin{pmatrix} \cos\psi_i & -\sin\psi_i \\ \sin\psi_i & \cos\psi_i \end{pmatrix}$$

from the vector of angles $\Psi_r \in \mathbb{R}^{d/2}$. $\beta_h, \beta_t \in \mathbb{R}$ are again entity-specific biases.

- RotH [47]

$$f((\mathbf{x}_h, \beta_h), (\mathbf{x}_r, \Phi_r, \gamma_r), (\mathbf{x}_t, \beta_t)) = -d^{\gamma_r}((Rot(\Psi_r) \otimes^{\gamma_r} \mathbf{x}_h \oplus^{\gamma_r} \mathbf{x}_r), \mathbf{x}_t)^2 + \beta_h + \beta_t$$

where *Rot* is the rotation matrix, a block-diagonal matrix, whose 2×2 blocks are given as

$$\begin{pmatrix} \cos\phi_i & \sin\phi_i \\ \sin\phi_i & -\cos\phi_i \end{pmatrix}$$

from a vector of angles $\Phi \in \mathbb{R}^{d/2}$. $\beta_h, \beta_t \in \mathbb{R}$ are again entity-specific biases.

- AttH [47] combines RefH and RotH

$$f((\mathbf{x}_h, \beta_h), (\mathbf{x}_r, \Phi_r, \Psi_r, \mathbf{a}_r, \gamma_r), (\mathbf{x}_t, \beta_t)) = -d^{\gamma_r}((ARR(\mathbf{x}_h, \Phi_r, \Psi_r, \mathbf{a}_r, \gamma_r) \oplus^{\gamma_r} \mathbf{x}_r), \mathbf{x}_t)^2 + \beta_h + \beta_t$$

where $ARR(\mathbf{x}_h, \Phi_r, \Psi_r, \mathbf{a}_r, \gamma_r) = Att(Rot(\Phi_r) \otimes^{\gamma_r} \mathbf{x}_h, Ref(\Psi_r) \otimes^{\gamma_r} \mathbf{x}_h, \mathbf{a}_r)$ is the attention selection from rotation and reflection, with $Att(\mathbf{x}, \mathbf{y}, \mathbf{a})$ computed as

$$\begin{aligned} \mathbf{x}', \mathbf{y}' &= \log_0^{\gamma_r}(\mathbf{x}), \log_0^{\gamma_r}(\mathbf{y}) \\ \alpha_x, \alpha_y &= \text{softmax}(\mathbf{a}^T\mathbf{x}', \mathbf{a}^T\mathbf{y}') \\ Att(\mathbf{x}, \mathbf{y}, \mathbf{a}) &= \exp_0^{\gamma_r}(\alpha_x\mathbf{x}' + \alpha_y\mathbf{y}') \end{aligned}$$

and $\Phi, \Psi \in \mathbb{R}^{d/2}$.

- TransG [233]

$$f((\mathbf{x}_h, \sigma_h), \mathbf{x}_r, (\mathbf{x}_t, \sigma_t)) = \sum_k^{K_r} \pi_{r,k} \exp\left(-\frac{\|\mathbf{x}_h - \mathbf{x}_{r,k} - \mathbf{x}_t\|_2^2}{\sigma_h^2 + \sigma_t^2}\right)$$

where the relation-specific number of components K_r is learned via the Chinese Restaurant Process (CRP).

- KG2E [99]

$$f((\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h), (\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r), (\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)) = sim(\mathcal{N}(\boldsymbol{\mu}_h - \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_h + \boldsymbol{\Sigma}_t), \mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r))$$

with Σ being a diagonal d-dimensional covariance matrix. where *sim* is a similarity between distributions. In the paper, they explore Kullback-Leibler divergence, Jenson-Shannon divergence, or expected likelihood. Since the distributions are Normal distributions with known parameters, the similarities can be computed in closed form.

- NTN [201]

$$f(\mathbf{x}_h, (\mathbf{u}_r, \mathbf{V}_r, \mathbf{W}_r^{[1:k]}, \mathbf{b}_r), \mathbf{x}_t) = \mathbf{u}_r^T \tanh(\mathbf{x}_h^T\mathbf{W}_r^{[1:k]}\mathbf{x}_t + \mathbf{V}_r[\mathbf{x}_h; \mathbf{x}_t] + \mathbf{b}_r)$$

with a relation representation comprising four components: $\mathbf{u}_r, \mathbf{u}_r \in \mathbb{R}^{d_h}$, $\mathbf{V}_r \in \mathbf{R}^{d_h \times 2d_e}$, and $\mathbf{W}_r^{[1:k]} \in \mathbb{R}^{d_e \times d_e \times d_h}$.

- ProjE [196]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \sigma(\mathbf{x}_t^T \tanh(\mathbf{D}^e \mathbf{x}_h + \mathbf{D}^r \mathbf{x}_r + \mathbf{b}) + \beta)$$

with trainable weights $\mathbf{D}^e, \mathbf{D}^r \in \mathbb{R}^{d \times d}$, $\mathbf{b} \in \mathbb{R}^d$, and $\beta \in \mathbb{R}$.

- CrossE [256]

$$f(\mathbf{x}_h, (\mathbf{x}_r, \mathbf{c}_r), \mathbf{x}_t) = \sigma(\tanh((\mathbf{c}_r \odot \mathbf{x}_h + \mathbf{c}_r \odot \mathbf{x}_h \odot \mathbf{x}_r + \mathbf{b})^T \mathbf{x}_t))$$

- ConvE [65]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \sigma(flatten(\psi(\mathbf{x}_h, \mathbf{x}_r) * \Omega)\mathbf{W})^T \mathbf{x}_t$$

- InteractE [215]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \sigma(flatten(\psi(\mathbf{x}_h, \mathbf{x}_r) \bar{\star} \Omega)\mathbf{W})^T \mathbf{x}_t$$

where ψ denotes a chequering function, which interleaves the vectors in a two-dimensional matrix, and $\bar{\star}$ denotes a depth-wise circular correlation operation.

- ConvKB [165]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \mathit{flatten}(\mathit{stack}(\mathbf{x}_h; \mathbf{x}_r; \mathbf{x}_t) * \boldsymbol{\Omega})^T \mathbf{w}$$

where *stack* combines vectors into a matrix with the vectors are columns, and flatten converts a matrix to a long vector, and $*$ denotes the 2D convolution. $\Omega \in \mathbb{R}^{3 \times d_h}$ denotes d_h many 3×1 convolution filters, and $\mathbf{w} \in \mathbb{R}^{d_e \cdot d_e}$ a trainable weight.

- ConEx [63]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = ReLU(\mathit{flatten}(ReLU([\mathbf{x}_h; \mathbf{x}_r] * \Omega)\mathbf{W} + \mathbf{b}))^T \Re\left(\mathbf{x}_h, \mathbf{x}_r, \overline{\mathbf{x}_t}\right)$$

- ER-MLP [69]

$$f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t) = \text{MLP}([\mathbf{x}_h; \mathbf{x}_r; \mathbf{x}_t])$$

where MLP is a trained 2-layer MLP.

- SHALLOM [62]

$$f(\mathbf{x}_h, (\mathbf{x}_r, \beta_r), \mathbf{x}_t) = \sigma(\mathbf{x}_r^T ReLU(\mathbf{W}[\mathbf{x}_h; \mathbf{x}_t] + \mathbf{b}) + \beta_r)$$

where $\mathbf{W} \in \mathbb{R}^{d_r \times 2d_e}$ and $\mathbf{b} \in \mathbb{R}^{d_r}$ are trainable global parameters, and $\beta_r \in \mathbb{R}, \mathbf{x}_r \in \mathbb{R}^{d_h}$.

Discussion Interaction functions are often categorized into distance-based and semantic matching, where the latter uses a similarity instead of distance function [222, 118].

Besides the aforementioned categorization, we can also distinguish between functional, or stateless, interaction functions, which have a fixed form, and parametric, or stateful interaction functions, which have internal, global parameters not associated with a specific entity or relation, but also trained end-to-end. Examples for stateless interaction functions are, e.g., ComplEx [214], or RESCAL [168], while parametric interaction functions encompass, e.g., ConvE [65], or TuckER [14]. Stateful interaction functions can also be seen as whole parametric families of interaction functions, from which a suitable interaction is chosen end-to-end.

Approaches using GNN-based representations tend to use simple, stateless interaction functions, e.g. R-GCN [191], or CompGCN [216] both using DistMult, instead of more complex stateful decoders. One reason may be that the GNN part is expressive enough to capture the complex interactions, and thus a simple, and fast, interaction function is sufficient.

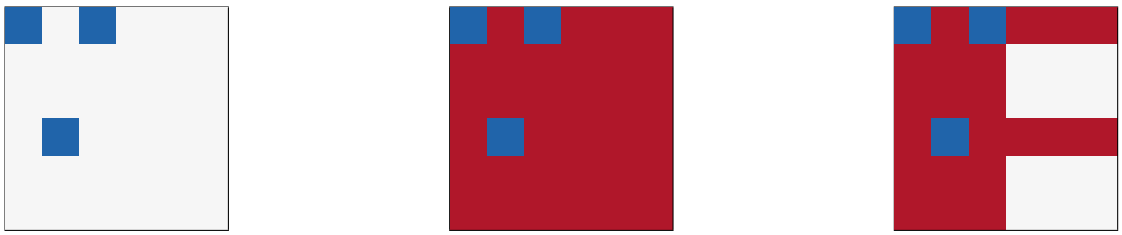
Interaction functions are often motivated and analyzed regarding their modeling capabilities of different *relational patterns* and *cardinality types*. The principal relational patterns discussed in the literature are *symmetry*, *anti-symmetry*, *inversion*, and *composition* [212, 214, 207]. For a symmetric relation, $(h, r, t) \in \mathcal{T} \implies (t, r, h) \in \mathcal{T}$, while for an anti-symmetric relation, $(h, r, t) \in \mathcal{T} \implies (t, r, h) \notin \mathcal{T}$. A relation r follows the inversion pattern, if there is another relation $r' \in \mathcal{R}$ such that $(h, r', t) \in \mathcal{T} \implies (t, r, h) \in \mathcal{T}$. Finally, a r is a composition, if there are two relations $r_1, r_2 \in \mathcal{R}$ such that $(h, r_1, e), (e, r_2, t) \in \mathcal{T} \implies (h, r, t) \in \mathcal{T}$. The authors of [224] introduce¨ the cardinality types[9]: one-to-one $(1 : 1)$, one-to-many $(1 : n)$, many-to-one $(m : 1)$, and many-to-many $(m : n)$. These depend on the number of different tail/head entities a head/tail can have for the relation of interest. Some of these relation types have motivated extensions to existing interaction functions which lacked support thereof. Notice that independently some of the problems could also have been mitigated by use of inverse relations: e.g., DistMult is symmetric due to the symmetry of the tri-linear product. However, when by the virtue of explicit inverse relations a different relation representation is used to predict the head for a given relation and tail, it can also model anti-symmetric relations.

For the plethora of different interaction functions proposed over time, the new interaction function has often been the primary focus of a publication. However, as shown by recent works [5, 188] this exclusive performance attribution to interaction functions is not always correct, but other components such as training approaches, loss functions, or the use of inverse relations can also play a decisive role. This is discussed in greater detail in Chapter 7.

3.5.2 Training

KGs usually only contain positive information, i.e., triples corresponding to true facts, but do not explicitly encode negative information in the form of false triples [166]. Thus,

[9] originally called *mapping property* of a relation



(a) OWA (b) CWA (c) LCWA

Figure 3.4: Visualization of different training assumptions for a toy example of a KG with one relation, six entities, and three known triples $\{(0,0,0),(0,0,2),(2,0,1)\}$. Blue color indicates triples that are assumed to be true; red is used for triples assumed to be false; white corresponds to triples where neither of the two assumptions is made.

instead of operating under the Closed World Assumption (CWA) where non-observed triples are assumed to be false, it is more appropriate to make the Open World Assumption (OWA), where non-observed triples are assumed to be unknown rather than false. However, only having access to examples for positive triples aggravates a naïve application of machine learning methods: the lack of examples for the negative class can lead a model collapse where the model always predicts the positive class [166]. A compromise between the two assumptions is the Local Closed World Assumption (LCWA), where an unknown triple (h,r,t) is only assumed to be false, if (h',r,t), or (h,r,t') has been observed for $h' \neq h, t' \neq t$, cf. Fig. 3.4.[10]

Applying the LCWA allows to efficiently generate negative examples from positive triples by *corruption*, i.e., replacing a single component by a (randomly sampled) different entity/relation. A common strategy is to generate a fixed number of negative examples for each positive example in one mini-batch [201]. For suitable interaction functions which are *decomposable*[11] as $f(\mathbf{x}_h,\mathbf{x}_r,\mathbf{x}_t) = g_2(g_1(\mathbf{x}_h,\mathbf{x}_r),\mathbf{x}_t)$ this permits efficiently computing the scores for one positive example and many corrupted ones by computing $g_1(\mathbf{x}_h,\mathbf{x}_r)$ only once. Similar optimizations are possible for head corruption if the interaction decomposes as $f(\mathbf{x}_h,\mathbf{x}_r,\mathbf{x}_t) = g_2(\mathbf{x}_h,g_1(\mathbf{x}_r,\mathbf{x}_t))$. If only one of the directions decomposes efficiently, inverse relations can be employed to predict (t,r^{-1},h) instead of (h,r,t), as done, e.g., for ConvE [65].

Random corruption can lead to too trivial negative examples. Hence, more advanced corruption strategies have been proposed to generate more realistic negative examples, e.g., Bernoulli negative sampling [224], typed negative sampling [125], or hard negative mining [125] where replacements are obtained by nearest neighbor search in the representation space.

[10] In some works, e.g. [166], this assumptions is only made for the tail entity.

[11] [188] defines decomposable interactions as $f(\mathbf{x}_h,\mathbf{x}_r,\mathbf{x}_t) = g_4(\sum_z g_3([g_1(\mathbf{x}_h,\mathbf{x}_r)\circ g_1(\mathbf{x}_r,\mathbf{x}_t)]_z))$ for scalar functions g_3, g_4, and an element-wise combination operation $\circ$. In contrast to our definition, this formulation does not permit the same optimized evaluation for $1:k$ scoring *per se*.

3.5.3 Losses

We can distinguish three general types of LP losses [155, 5]: *pointwise*, *pairwise* and *setwise*.

Pointwise Losses

These consider the score for a single triple in isolation. Therefore, the scores are encouraged to fulfill *global* properties, e.g., a global decision threshold. This interpretation is closer to modeling link prediction as a classification problem, i.e., triple classification, rather than a ranking task. With denoting the predicted score as $\hat{y}$, the label by y, and the triple loss as $l(\hat{y}, y)$, we can consider the following examples:

- For $y \in \{0, 1\}$, the *squared error* loss, e.g., [168], is given by:

$$l(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2.$$

- For $y \in \{-1, 1\}$, the *pointwise hinge* loss, e.g., [167], is given as

$$l(\hat{y}, y) = \text{ReLU}(\lambda - y \cdot \hat{y}).$$

- For $y \in \{-1, 1\}$, the *pointwise logistic* loss, e.g., [214, 165, 121], is given as

$$l(\hat{y}, y) = \text{softplus}(-y \cdot \hat{y}).$$

- For $y \in \{0, 1\}$, the *binary cross entropy* (BCE) loss, e.g., [65, 14], is given as

$$l(\hat{y}, y) = -y \log \sigma(\hat{y}) - (1 - y) \cdot \log(1 - \sigma(\hat{y}))$$

  The BCE loss is equivalent to the pointwise logistic loss, since

$$\begin{aligned} l(\hat{y}, y) &= -y \log \sigma(\hat{y}) - (1 - y) \cdot \log(1 - \sigma(\hat{y})) \\ &= -y \log \sigma(\hat{y}) - (1 - y) \cdot \log \sigma(-\hat{y}) \\ &= -y \log((1 + \exp(-\hat{y}))^{-1}) - (1 - y) \cdot \log((1 + \exp(\hat{y}))^{-1}) \\ &= y \log(1 + \exp(-\hat{y})) + (1 - y) \cdot \log(1 + \exp(\hat{y})) \\ &= y \, \text{softplus}(-\hat{y}) + (1 - y) \, \text{softplus}(\hat{y}) \\ &= \text{softplus}(-y' \cdot \hat{y}) \end{aligned}$$

  where the last equality uses $y' = -1$ if $y = 0$ else $y' = 1$.

- The *pointwise square* loss [155] is given as

$$l(\hat{y}, y) = \frac{1}{2}(\text{ReLU}(\lambda - y \cdot \hat{y}))^2.$$

Pairwise Losses

These losses consider the difference between a pair of scores for a positive and a negative triple, $\hat{y}^+$ and $\hat{y}^+$. They generally follow the form

$$l(\hat{y}^+, \hat{y}^-) = m(\lambda + \hat{y}^- - \hat{y}^+),$$

where m is either ReLU resulting in the *pairwise hinge* or *margin ranking* loss, e.g. [37, 242], or softplus, resulting in the *pairwise logistic* loss [155], which is a soft-margin formulation. In contrast to pointwise losses, for pairwise losses, only the *difference* in scores if relevant, while the absolute numbers do not matter. The authors of [263] propose a combination of margin ranking loss, with an additional *limit-based* loss, which encourages a minimum score $\lambda^+ \in \mathbb{R}$ for positive triples, with a balance hyperparameter $0 < \mu \in \mathbb{R}$: ´

$$l(\hat{y}^+, \hat{y}^-) = \mathrm{ReLU}(\lambda + \hat{y}^- - \hat{y}^+) + \mu\,\mathrm{ReLU}(\hat{y}^+ + \lambda^+)$$

Setwise Losses

These losses consider a larger number of scores at once.

- The *cross-entropy* loss is given as

  $$-\sum_t p(t \mid h, r) \log q(t \mid h, r)$$

  where $q(t \mid h, r)$ is the predicted probability distribution over all tail entities given head and relation. This distribution is usually obtained via softmax normalization and for numerical reasons fused with the subsequent application of the logarithm. The true probability distribution $p(t \mid h, r)$ is extracted from the training triples using the relative frequencies [65]. As the training triples are grouped by (head, relation) pairs, multiple triples $\{(h, r, t_1), \ldots, (h, r, t_k)\}$ are combined to a single training instance $(h, r, \{t_1, \ldots, t_k\})$. Hence, this conversion also represents an (implicit) form of "class" balancing. Although the single-label assumption is usually wrong, this formulation has empirically shown strong performance [119, 145]. The reason is not yet known [188].

- The *negative sampling self-adversarial* loss (NSSA) [207] has been proposed as a computationally appealing variant to hard negative mining. It weights negative samples according to their predicted scores, where higher scores lead to larger weights. With $\hat{y}^+$ denoting the score of a positive example, and $\{\hat{y}_i^-\}$ a set of corresponding negative examples, the loss is given as

  $$-\log \sigma(\gamma + \hat{y}^+) - \log \sum p(\hat{y}_i) \log \sigma(-\gamma - \hat{y}_i^-)$$

  where $p(\hat{y}_i)$ is obtained by softmax $p(\hat{y}_i) = (\mathrm{softmax}(\alpha\hat{y}_1, \ldots, \alpha\hat{y}_k))_i$, where $\alpha \in \mathbb{R}$ is a temperature parameter.

3.6 Entity Alignment

For EA, we can generally distinguish two families of approaches based upon whether they embed the entities of both graphs into a shared representation space, e.g., GCN-Align [225], or RDGCN [227], or use different representation spaces with a learned transformation between them, e.g., MTransE [53], SEA [173], or AKE [137]. When GNNs are employed, Siamese architectures are used, which apply the same message passing mechanism and weights independently on the two KGs.

3.6.1 Similarities

After the entity representations are mapped into a common representation space, a simple vector similarity[12] measure is used to compute matching scores, followed by optional normalization steps. With denoting by $sim(\mathbf{x}_i, \mathbf{x}_j)$ the similarity between representations $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$, the following similarity functions are often encountered:

- negative L_p distances, in particular for $p = 1$ [225, 243, 174, 229, 235, 150, 149, 267, 268, 51], and $p = 2$ [52, 137, 132, 44, 265, 54, 169, 245, 204, 51]:

$$sim(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_p$$

- the dot product [78], and

$$sim(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

- the cosine similarity [205, 253, 213, 252, 250], equivalent to the dot product similarity between unit-length normalized vectors,

$$sim(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \cdot \|\mathbf{x}_j\|_2}$$

While there is a body of work discussing the suitability of different distance/similarity measures for high-dimensional spaces [3, 126], for EA, the similarity is usually just treated as a hyperparameter without further discussing its implications.

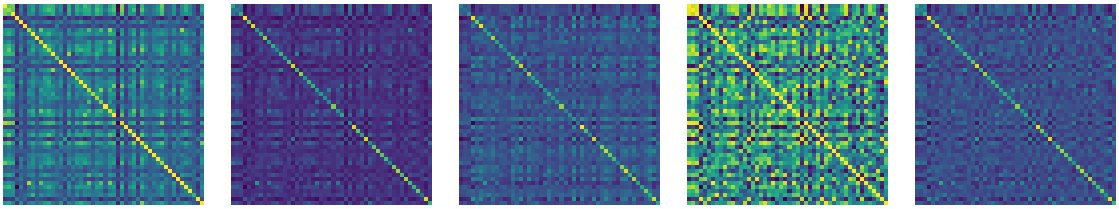
Normalization

Sometimes, the similarity matrices are further normalized, e.g., to reduce the problem of hubness [181], or overcome the issue of different scales of similarity values.

- RAGA [267] modifies the similarity matrix by using the sum of the row- and column-wise softmax of similarity scores. Therefore, the absolute similarity values are irrelevant, but only the row- and column-wise relative similarities.

$$\mathbf{S}' = \text{softmax}(\mathbf{S}, 1) + \text{softmax}(\mathbf{S}, 2)$$

[12] Some approaches use a distance measure instead. For all studied works, we can rewrite them into a similarity formulation by using the negative distance. Different distance-to-similarity transformations are also possible [72].



(a) raw (b) softmax (c) CSLS, k=10 (d) rank (e) Sinkhorn

Figure 3.5: Visualization of different similarity matrix normalization methods. The color scale is only consistent inside a single image, but not across them due to the different value ranges and the otherwise resulting loss of intra-image contrast. (a) shows the original similarity matrix, which has been obtained as the cosine similarity between the initialization from GMN [239] for the first 100 entities (with the matching ones on the diagonal), (b) the sum of row- and column-wise softmax, (c) the CSLS normalization for $k = 10$, (d) the sum of row- and column-wise ranks, and (e) the Sinkhorn normalization.

- RREA [150] and KEnS [54] use Cross-domain Similarity Local Scaling (CSLS) [130], an approach which is used to overcome the problem of hubs and has been applied in learning multi-lingual word embeddings. It adjusts the similarity by subtracting the average similarity to the k nearest neighbors of the source and target. Therefore, the similarity from and to elements with high similarity to many other elements is reduced,
$$\mathbf{S}'_{ij} = 2\mathbf{S}_{ij} - \mathit{avg}(\mathit{top}(\mathbf{S}_{i,:}, k)) - \mathit{avg}(\mathit{top}(\mathbf{S}_{:,j}, k))$$
where *top* denotes the operation which selects the largest k entries. CSLS closely resembles local centering [96], although being more generally applicable to non-Euclidean distances.

- [197] use the sum of the row- and column-wise rank matrices. Similar to RAGA's approach, this ignores absolute similarity value scales and only considers the relative order per row/column. Since the operation is non-differentiable, this technique is only applied at inference time and not during training:
$$\mathbf{S}' = \mathit{rank}(\mathbf{S}, 1) + \mathit{rank}(\mathbf{S}, 2)$$
where *rank* is used to denote the operation of computing the rank of the entries in the sorted order along the specified dimension.

- DGMC [78] uses the Sinkhorn-Knopp algorithm [200, 123, 58] to make the matching matrix approximately doubly-stochastic. The Sinkhorn-Knopp algorithm alternating ensures that each row/column sums up to 1. Therefore, they can interpret each row/column corresponding to a (candidate) entity in one graph as a probability distribution over all (candidate) entities in the other graph. By unrolling the iterative procedure, the method is differentiable. Furthermore, during training, only a moderate number of iterations are applied.

3.6.2 Matching

Although some frequently used benchmark datasets do not exhibit solely 1:1 matches [29], some approaches employ different kinds of matching algorithms. These encompass:

- JEA [238] applies the Hungarian method / Kuhn-Munkres algorithm [128, 159] to solve the resulting linear assignment problem. Values below a certain similarity threshold are masked by setting infinite costs for the matching algorithm to improve run-time.
- CEA and RAGA [251, 267] take the high worst-case runtime of the Hungarian method as a reason to solve the stable matching problem instead. To this end, the deferred acceptance algorithm (DAA)/Gale-Shapley algorithm [82] is applied.
- DGMC [78] proposes a trainable correspondence refinement mechanism. The normalized (and sparsified) similarity matrix is used as a bijection to transfer a node coloring given by random vectors between the two graphs. The transferred coloring is subsequently dispersed by a GNN layer. An MLP then uses the updated node colorings to predict updates for the similarity matrix. The process is repeated multiple times.

3.6.3 Training

Just like in the task of LP, EA usually does not require explicit negative labels but instead generates negative examples by random corruption of aligned entity pairs. There are some additional, EA-specific techniques:

Bootstrapping

Since most EA methods' performance strongly depends on the number of available training alignments, some approaches adopt a bootstrapping scheme, where the model is used to obtain additional (pseudo-) labels. There are several different methods to obtain such additional alignment pairs:

- MRAEA [149], KEnS [54], and UEA [250] use mutual nearest neighbors to generate high quality candidates. UEA further employs a threshold, which is linearly decreased over iterations.
- BootEA [205] and COTSAE [245] first use a similarity threshold and then create candidate pairs based on kNN. These matching candidate pairs form a bipartite graph, for which the maximum weighted matching problem is solved via the `igraph` library, which in turn uses the push-relabel algorithm [56]. BootEA further mentions [1] as a faster matching method, although not making use of it in their public code.

- RMN [268] only search uni-directional neighbors, i.e., for each entity in the left graph, the nearest neighbor in the right graph is determined. Conflicts are resolved greedily. This approach also applies the same mechanism to generate relation alignments.
- EHD [238] uses an absolute similarity threshold for generating candidate pairs. The threshold is changed throughout the training.
- DAT [252] uses the difference between the largest and second-largest similarity in conjunction with a threshold to generate candidates.

Additional Tricks

In [44], the rule miner AMIE [81] is used to complete the KG. Moreover, some approaches do not only use the (training) alignments for supervision through the loss function, but also other tricks such as:

- sharing the parameters between aligned entities [174, 197, 264],
- randomly swapping representations of aligned entities [266, 253, 205], or
- transferring triples between aligned entities [206, 252, 91, 92].

3.6.4 Losses

Losses used for EA are quite similar to those encountered for LP. Let $s^+ \in \mathbb{R}$ denote the similarity between the representations of a positive pair, i.e., a pair of aligned entities, and $s^- \in \mathbb{R}$ the score for a negative pair. The following losses are being used:

- The *margin* loss
$$\mathrm{ReLU}(\lambda + s^- - s^+)$$
with a margin parameter $\lambda \in \mathbb{R}$.
- The *contrastive* loss [206]
$$-s^+ + \mu\,\mathrm{ReLU}(\lambda + s^-)$$
with a balance parameter $0 < \mu \in \mathbb{R}$, and a margin $\lambda \in \mathbb{R}$.
- The *double margin* loss [205] is given as
$$\mathrm{ReLU}(s^+ - \lambda^+) + \mu \cdot \mathrm{ReLU}(s^- - \lambda^-)$$
with two separate margin parameters $\lambda^+, \lambda^- \in \mathbb{R}$, and a balance parameter $\mu \in \mathbb{R}$.

Moreover, *group-wise* losses do not only consider the similarity between a pair of representations but additionally require that the distribution of these representations become alike. To this end, a GAN-like approach is used, where an adversarial discriminator is trained to distinguish between the entity representation spaces of the left and right KG. AKE [137] uses vanilla GAN [89], while OTEA [173] utilizes Wasserstein GAN [11] instead.

head	relation	tail candidate	score
Alan Turing	educated at	King's College	5
		Cambridge University	4
		Princeton University	4
		Alonzo Church	1
		Alan Turing	0

Figure 3.6: Example for rank-based evaluation for the tail entity ranking task (Alan Turing, educated at, ?) on a subset of the KG from Fig. 3.1. Assume (Alan Turing, educated at, Cambridge University) to be a training triple, (Alan Turing, educated at, Princeton University) to be the currently considered evaluation triple, and (Alan Turing, educated at, King's College) to be true yet neither known in training nor evaluation set. The *score* column shows the predicted score, where larger score means greater plausibility.

3.7 Rank-Based Evaluation

Both tasks, EA and LP, are commonly evaluated using rank-based metrics. The general concept is as follows: For each individual ranking task, e.g., predicting a tail entity t for a given head-relation pair (h, r), the scores for all candidates are computed, and subsequently used to order to candidates in descending order. Then, the position of the "true" answer, i.e., the ground truth candidate, is taken as the *rank* of the answer. Consider the example shown in Fig. 3.6, where we want to evaluate the tail prediction for an evaluation triple (Alan Turing, educated at, Princeton University). To this end, we used a LP model to score all tail entity candidates[13], where larger scores indicate higher plausibility. In this example, Princeton University would be at rank 3.

Notice that the intuitive definition of a rank as the position within a sorted list does not specify what happens if there are multiple candidates with exactly scores, as shown for the entities Princeton University and Cambridge University. However, such cases do occur in real-world applications [208] on a scale which can strongly distort results depending on the exact implementation. In [27], we provide an extensive overview and categorization of the used variants of numerous LP and EA codebases.

For LP, the *unfiltered* and *filtered* evaluation setting are distinguished [37], where the latter is the default for recent publications. In the unfiltered setting, all entities are candidates. Since there may be more than one tail entity for a given head-relation combination, the candidates may contain other known true triples. In the unfiltered setting, a model is penalized with a larger rank if other true triples are ranked higher. The filtered setting mitigates this by excluding other *known* true triples from the candidates, i.e., in the example in Fig. 3.6 Cambridge University would be excluded from the candidates for the evaluation triple (Alan Turing, educated at, Princeton University). Notice,

[13]For readability, we assume in this example that there are only five entities.

however, that the remaining triples may still contain unknown true triples, and hence the evaluation results may be pessimistic, e.g., `King's College` in the example of Fig. 3.6.

Let $\mathcal{I}$ denote the ranks obtained for each individual ranking task. To obtain single-figure measures, several different aggregation metrics are used, such as the Mean Rank (MR), the Mean Reciprocal Rank (MRR), or Hits@k (H@k). They are given as

$$\begin{aligned} \mathrm{MR}(\mathcal{I}) &= \frac{1}{|\mathcal{I}|} \sum_{i \in |\mathcal{I}|} i \\ \mathrm{MRR}(\mathcal{I}) &= \frac{1}{|\mathcal{I}|} \sum_{i \in |\mathcal{I}|} \frac{1}{i} \\ \mathrm{H@k}(\mathcal{I}) &= \frac{1}{|\mathcal{I}|} \sum_{i \in |\mathcal{I}|} \mathbb{I}[i \leq k] \end{aligned}$$

where $\mathbb{I}$ denotes the indicator function, which is one for a true condition and zero otherwise.

Since the rank is the *absolute* position within the sorted candidates, its value range depends on the number of considered candidates: a lower number of candidates reduces the worst possible rank and thus also improves the performance of a random ranking model. This ostensible improvement may be undesirable since it complicates the performance comparison, e.g., for different filtering settings. In Chapter 9, we discuss an extension of the MR which adjusts it for chance, and thus implicitly also normalizes by the number of candidates. There have been further extensions [210] of this work applying a similar normalization.

4 Active Learning for Entity Alignment

This chapter comprises the publication

> **Max Berrendorf***, Evgeniy Faerman*, and Volker Tresp. „Active Learning for Entity Alignment.“ In: *Advances in Information Retrieval*. Ed. by Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani. * equal contribution. Cham: Springer International Publishing, 2021, pp. 48–62. ISBN: 978-3-030-72113-8. DOI: 10.1007/978-3-030-72113-8_4

In addition, an extended abstract has been published as

> **Max Berrendorf***, Evgeniy Faerman*, and Volker Tresp. „Active Learning for Entity Alignment.“ In: *The 5th International Workshop on Deep Learning for Graphs (DL4G@WWW2020)* (2020). * equal contribution. arXiv: 2001.08943

and the code is available at

> **Max Berrendorf** and Evgeniy Faerman. *mberr/ea-active-learning: Zenodo*. Version 1.0.1. Dec. 2020. DOI: 10.5281/zenodo.4588896

Declaration of Authorship The original research contributions were developed and conceptualized by Max Berrendorf and Evgeniy Faerman and discussed with Volker Tresp. Max Berrendorf did the main part of the implementation, with smaller contributions by Evgeniy Faerman. Max Berrendorf designed and conducted the experiments and analyzed their results. The findings were discussed with Evgeniy Faerman. Max Berrendorf and Evgeniy Faerman wrote and revised the manuscript.

Active Learning for Entity Alignment

Max Berrendorf[1(✉)], Evgeniy Faerman[1], and Volker Tresp[1,2]

[1] Ludwig-Maximilians-Universität München, Munich, Germany
{berrendorf,faerman}@dbs.ifi.lmu.de
[2] Siemens AG, Munich, Germany
volker.tresp@siemens.com

Abstract. In this work, we propose a novel framework for labeling entity alignments in knowledge graph datasets. Different strategies to select informative instances for the human labeler build the core of our framework. We illustrate how the labeling of entity alignments is different from assigning class labels to single instances and how these differences affect the labeling efficiency. Based on these considerations, we propose and evaluate different active and passive learning strategies. One of our main findings is that passive learning approaches, which can be efficiently precomputed, and deployed more easily, achieve performance comparable to the active learning strategies. In the spirit of reproducible research, we make our code available at https://github.com/mberr/ea_active_learning.

Keywords: Entity alignment · Active learning · Knowledge graphs

1 Introduction

A knowledge graph (KG) is a way to store information (semi-)structurally to enable automatic data processing and data interpretation. KGs are utilized in various Information Retrieval related applications requiring semantic search of information [1,11]. While there exist various large open-source KGs, such as YAGO-3 [25], Wikidata [38], or ConceptNet [33], they often contain orthogonal information, and have their respective strength and weaknesses. Hence, being able to combine information from different knowledge graphs is required in many applications. An important subtask is identifying matching entities across several graphs, called *entity alignment* (EA). Recent years witnessed substantial advances regarding the methodology, in particular involving graph neural networks (GNNs) [6,7,19,28,34–37,40,42,44,46]. Common among these approaches is that they use a set of given seed alignments and infer the remaining ones. While several benchmark datasets are equipped with alignments, acquiring them in practice is cumbersome and expensive, often requiring human annotators. To address this problem, we propose to use *active learning* for entity alignment. In summary, our contributions are as follows:

M. Berrendorf and E. Faerman—equal contribution.

D. Hiemstra et al. (Eds.): ECIR 2021, LNCS 12656, pp. 48–62, 2021.
https://doi.org/10.1007/978-3-030-72113-8_4

- To the best of our knowledge, we are the first to propose using active learning for entity alignment in knowledge graphs. We investigate and formalize the problem, identify critical aspects, and highlight differences to the classical active learning setting for classification.
- A specialty of entity alignment is that learning is focused on information about aligned nodes. We show how to additionally utilize information about exclusive nodes in an active learning setting, which leads to significant improvements.
- We propose several different heuristics, based upon node centrality, graph and embedding coverage, Bayesian model uncertainty, and certainty matching.
- We thoroughly evaluate and discuss the heuristics' empirical performance on a well-established benchmark dataset using a recent GNN-based model. Thereby, we show that state-of-the-art heuristics for classification tasks perform poorly compared to surprisingly simple node centrality based approaches.

2 Problem Setting

We study the problem of entity alignment for knowledge graphs (EA). A knowledge graph can be represented by the triple $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ a set of relations, and $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ a set of triples. The alignment problem now considers two such graphs $\mathcal{G}^L, \mathcal{G}^R$ and seeks to identify entities common to both, together with their mapping. The mapping can be defined by the set of matching entity pairs $\mathcal{A} = \{(e, e') \mid e \in \mathcal{E}^L, e' \in \mathcal{E}^R, e \equiv e'\}$, where $\equiv$ denotes the matching relation. While some works are using additional information such as attributes or entity labels, we solely consider the graph structure's relational information. Thus, a subset of alignments $\mathcal{A}_{train} \subseteq \mathcal{A}$ is provided, and the task is to infer the remaining alignments $\mathcal{A}_{test} := \mathcal{A} \setminus \mathcal{A}_{train}$. With $\mathcal{A}^L := \{e \in \mathcal{E}^L \mid \exists e' \in \mathcal{E}^R : (e, e') \in \mathcal{A}\}$ we denote the set of entities from $\mathcal{G}^L$ which do have a match in $\mathcal{A}$, and $\mathcal{A}^R$ analogously. With $\mathcal{X}^L = \mathcal{E}^{\mathcal{L}} \setminus \mathcal{A}^L$ we denote the set of exclusive entities in the graph $\mathcal{G}^L$ which occur neither in train nor test alignment, and $\mathcal{X}^R$ analogously.

In practice, obtaining high-quality training alignments means employing a human annotator. As knowledge graphs can become large, annotating a sufficient number of alignment pairs may require significant labeling efforts and might be costly. Thus, we study strategies to select the most informative alignment labels to achieve higher performance with fewer labels, commonly referred to as active learning. The following section surveys existing literature about active learning with a particular focus on graphs and reveals differences in our setting.

3 Related Work

Classical active learning approaches [31] often do not perform well in batch settings with neural network architectures. Therefore, developing active learning heuristics for neural networks is an active research area. New approaches were

proposed for image [2,16,18,30,39,43], text [32,45] and relational [5,17,23,27,41] data. Active learning algorithms aim to select the most informative training instances. For instance, the intuition behind uncertainty sampling [22] is that instances about which the model is unconfident comprise new or not yet explored information. However, the estimation of neural networks' uncertainty is not a trivial task since neural networks are often overconfident about their predictions [15]. One approach to tackle this problem is to use Monte-Carlo dropout to estimate the uncertainty for active learning heuristics [16,27,32]. Alternatively, [2] demonstrated that ensembles of different models lead to better uncertainty estimation and consequently better instance selection. The method described in [23] adopts a different approach and queries labels for instances for which it is the most certain that they are unlabeled. For this assessment, the authors propose an adversarial framework, where the discriminator differentiates between labeled and unlabeled data.

Geometric or density-based approaches [5,17,18,30,41,43], on the other hand, aim to select the most representative instances. Therefore, unlabeled instances are selected for labeling, such that labeled instances cover unlabeled data in the embedding space. Other approaches to estimate the informativeness of unlabeled samples use, e.g., the expected length of gradient [45].

Active learning approaches with neural networks on relational data were so far applied to the classification of nodes in homogeneous graphs [5,17,23,41] and link prediction in knowledge graphs [27]. In [8,9,26] authors propose active learning approaches for the graph matching problem, where the matching costs are known *in advance*, and the goal is to minimize assignment costs. Note that this is different from our task, where the goal is to learn meaningful representations of the entities.

4 Methodology

In this section, we introduce our proposed labeling setting and describe data post-processing to leverage exclusive nodes. Moreover, we propose numerous new labeling strategies: Some strategies take inspiration from existing state-of-the-art heuristics for classification. Others are developed entirely new based on our intuitions. Finally, we present our evaluation framework for the evaluation of different heuristics.

4.1 Labeling Setting

Since we are dealing with matching KGs, where entities have meaningful labels, we assume that human annotators use these entity names for matching. Therefore, we see two different possibilities to formulate the labeling task:

1. The system presents annotators with possible matching pairs, and they label it as `True` or `False`
2. The system presents annotators a node from one of the two KGs, and the task is to find all matching nodes in the other KG.

It is easier to label a single instance in the first scenario, as it is a yes/no question. However, since each node can have more than one matching node in the other KG, $|\mathcal{E}^L| \times |\mathcal{E}^R|$ queries are necessary to label the whole dataset. In contrast, in the second scenario, human annotators need a similar qualification but the time spent per labeled instance increases because they have to search for possible matchings. However, there are the following advantages of the second scenario:

First, there are only $|\mathcal{E}^L| + |\mathcal{E}^R|$ possible queries. Second, in both scenarios, the learning algorithm needs positive matchings to start training. Assuming $|\mathcal{A}^L| \approx |\mathcal{A}^R| \approx |\mathcal{A}|$ and $|\mathcal{E}^R| \approx |\mathcal{E}^L| \approx |\mathcal{E}|$, the probability to select a match with a random query is in the first scenario $|\mathcal{A}|/|\mathcal{E}|^2$, whereas for the second scenario it is $|\mathcal{A}|/|\mathcal{E}|$. Additionally, in the second scenario, it is possible to start with some simple graph-based heuristics, e.g., based on a graph centrality score like degree or betweenness. For many KGs, it is a valid assumption that the probability of having a match is higher for more central nodes. Cold-start labeling performance is especially relevant when the labeling budget is restricted. Third, in the classical active learning scenario, there is the assumption that each query returns a valid label. However, for EA, the information that two nodes do not match is limited since negative examples can also be obtained by negative sampling. In contrast, in the second scenario, we can use information about missing matchings to adapt the dataset, see Sect. 4.2.

In this paper, we focus on the second scenario. However, heuristics relying on information from the matching model described in Sect. 4.3 can also be applied in the first scenario.

4.2 Dataset Adjustment

The EA task's main motivation is either the fusion of knowledge into a single database or exchanging information between different databases. In both cases, the primary assumption is that there is information in one KG, which is not available in the other. This information comes in relations between aligned entities, relations with exclusive entities, or relations between exclusive entities. While larger differences between the KGs increase their fusion value, they also increase the difficulty of matching processes. One possibility to partially mitigate this problem is to enrich both KGs independently using link prediction and transfer links between aligned entities in the training set [6,23]. As this methodology does only deal with missing relations between shared entities, in this work, we go a step further: Since we control the labeling process, we naturally learn about exclusive nodes from the annotators. Therefore, we propose to remove the exclusive nodes from the KGs for the matching step. After the matching is finished, the exclusive nodes can be re-introduced. In the classical EA setting, where the KGs and partial alignments are already given, and there is no control over dataset creation, the analogous removal of exclusive nodes is not possible: To determine whether a node is exclusive or just not contained in the training alignment requires access to the test alignments, hence representing a form of test leakage.

4.3 Active Learning Heuristics

The main goal of active learning approaches is to select the most informative set of examples. In our setting, each query either results in matches or verified exclusiveness, both providing new information. Nodes with an aligned node in the other KG contribute to the signal for the supervised training. State-of-the-art GNN models for EA learn by aggregating the k-hop neighborhood of a node. Two matching nodes in training become similar when their aggregated neighborhood is similar. Therefore, the centrality of identified alignments or their coverage is vital for the performance. On the other hand, exclusive nodes improve training by making both KGs more similar. Since it is not clear from the outset, what affects the final performance most, we analyze heuristics with different inductive biases.

Node Centrality – Selecting nodes with high centrality in the graph has the following effects: (a) a higher probability of a match in the opposite graph, and (b) updates for a larger number of neighbors if a match or significant graph changes when being exclusive. Although there is a large variety of different centrality measures in graphs [10], we observed in initial experiments that they perform similarly. Therefore, in this work, we evaluate two heuristics based on the nodes' role in the graph. The first, *degree* heuristic (denoted as *deg*), orders nodes by their degree, and the nodes with a higher degree are selected first. The second, *betweenness* heuristic (*betw*), works similarly and relies on the betweenness centrality measure.

Graph Coverage – Real-World graphs tend to have densely connected components [12]. In this case, if nodes for labeling are selected according to some centrality measure, there may be a significant overlap of neighborhoods. At the same time, large portions of the graph do receive no or infrequent updates. Therefore, we propose a heuristic, seeking to distribute labels across the graph. We adopt an *approximate vertex cover* algorithm [29] to define an active learning heuristic for entity alignment. Each node is initialized with a weight equal to its degree. Subsequently, we select the node from both graphs with the largest weight, remove it from the candidate list, and decrease all its neighbors' weight by one. We denote this heuristic as *avc*.

Embedding Space Coverage – The goal of embedding space coverage approaches is to cover the parts of the embedding space containing data as well as possible. Here we adapt the state-of-the art method *coreset* [30] (denoted as *cs*) for the EA task. Thereby, we aim to represent each graph's embedding space by nodes with *positive* matchings. We adopt a greedy approach from [30], which in each step selects the object with the largest distance to the nearest neighbor among already chosen items. Its performance was similar to the mixed-integer program algorithm while being significantly faster. In the process of node selection, it is not known whether nodes in the same batch have matchings or are exclusive. Thereby, in each step, each candidate node is associated with a score according to its distance to the nearest positive matching or the nodes already

selected as potential positives in the same batch. The node with the largest distance to the closest positive point is added to the batch.

Embedding Space Coverage by Central Nodes – The possible disadvantage of *coreset* heuristic in the context of entity alignment is that selected nodes may have low centrality and therefore affect only a small portion of the graph. Intuitively, it is possible because each next candidate is maximally distant from all nodes with positive matchings, which are expected to be more or less central. In this heuristic, we try to remedy this effect and sample nodes with high centrality in different parts of embedding space. Therefore, in each step, we perform clustering of node representations from both graphs in the joint space, c.f. Fig. 1. We count already labeled nodes in each cluster and determine the number of candidates selected from this specific cluster. This number is inversely proportional to the number of already labeled nodes in the cluster. We then use a node centrality based heuristic to select the chosen number of candidates per cluster. We denote this heuristic by *esccn*.

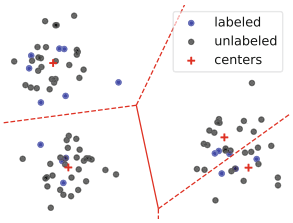


Fig. 1. Schematic visualization of the *esccn* heuristic. The labeled nodes per cluster are counted and used to derive how many samples to draw from this cluster. Another heuristic is then used to select the specific number from the given clusters, e.g., a graph-based *degree* heuristic.

Uncertainty Matching – Uncertainty-based approaches are motivated by the idea that the most informative nodes are those for which the model is most uncertain about the final prediction. We reformulate EA as a classification problem: The number of classes corresponds to the number of matching candidates, and we normalize the vector of similarities to the matching candidates with the softmax operation. A typical uncertainty metric for classification is *Shannon entropy* computed over the class probability distribution, where large entropy corresponds to high uncertainty. We can employ *Monte-Carlo Dropout* to compute a Bayesian approximation of the softmax for the entropy similarly to [17]. However, the repeatable high entropy across multiple dropout masks indicates the *prediction uncertainty*, where the model is *certain* that a right prediction is impossible. In the context of entity alignment, we expect high prediction uncertainty for the exclusive nodes since a model may be *certain* about lacking good matchings. Therefore we opt for model uncertainty for the entity alignment. The model uncertainty is high if the model makes different (certain) decisions

for the same instances in multiple runs [14]. We employ *BALD* [21] with Monte-Carlo Dropout [17]. The heuristic computes the expected difference between the entropy of single model prediction and expected entropy. Note that numerous classes may lead to similar entropy and BALD values for the whole dataset. To mitigate this effect, we employ softmax temperature [20].

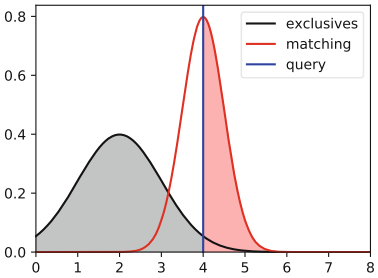


Fig. 2. Visualization of scoring method of the *prexp* heuristic. We fit two normal distributions for matching and exclusive nodes. Each distribution models the maximum similarity these nodes have to any node in the other graph ($s_{max}(q)$). To assess the quality of a query q, we get its maximum similarity, and evaluate $P_{match}(s_{max}(e) \leq s_{max}(q)) - P_{excl}(s_{max}(e) \geq s_{max}(q))$, i.e. the black area minus the red one. (Color figure online)

Certainty Matching – A distinctive property of EA is that the supervised learning signal is provided only by the part of the labeled nodes that have a matching partner in the other graph. Therefore, we propose a heuristic that prefers nodes having matches in the opposite graph, named *previous-experience-based (prexp)*. As the model is trained to have high similarities between matching nodes, the node with maximum similarity is the most likely matching partner for a given node. Moreover, we expect that higher similarity values indicate a better match, such that we can utilise this maximum similarity as a matching score: $s_{max}(e) = \max_{e' \in \mathcal{E}^R} similarity(e, e')$ for $e \in \mathcal{E}^L$. Thus, we hypothesize that the distribution of maximum similarity values between exclusive nodes and those having a matching partner differ and can be used to distinguish those categories. However, we note that the similarity distribution for already labeled nodes may differ from those that are not labeled, as the labeled nodes directly receive updates by a supervised loss signal. Hence, we use *historical* similarity values acquired when we selected unlabeled nodes for labeling, and the ground truth information about them having a match received after the labeling. Based on these, we fit two normal distributions for maximum similarities: The first distribution with the probability function P_{match} describes the distribution of maximal similarity score of nodes with matchings. Similarly, the function P_{excl} computes the probability that the maximal similarity score belongs to an exclusive node. For each entity in question e, we take its maximal similarity score to the candidate in other graph and compute a difference between two probabilities $P_{match}(s_{max}(e) \leq x) - P_{excl}(s_{max}(e) \geq x)$ as heuristic score, c.f. Fig. 2. This

score is large if the maximal similarity of exclusive nodes is smaller than that of nodes with matchings. We keep only entities with the score greater than threshold t, where t is a hyperparameter. This way, we make sure that the score is used only if matching and exclusive nodes are distinguishable. If there are not enough entities that fulfill this requirement, we use some simple fallback heuristic, e.g., degree, for the remaining nodes.

5 Evaluation Framework

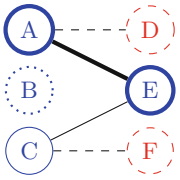


Fig. 3. Visualization of node categorisation for $\mathcal{E}^L = \{A, B, C\}$, and $\mathcal{E}^R = \{D, E, F\}$. Solid lines represent training alignments, whereas dashed ones denote test alignments. Node B is the only exclusive node. All blue nodes are in the initial pool $\mathcal{P}_0$. The red dashed nodes D and F may not be requested for labeling as they neither are exclusive nor participate in a training alignment. When node A is requested, only the alignment (A, E) is returned, and A, as well as E, become unavailable. The second training alignment (C, E) can still be obtained by requesting C. (Color figure online)

To evaluate active learning heuristics in-vitro, an alignment dataset comprising two graphs and labeled alignments is used. These alignments are split into training alignments $\mathcal{A}_{train}$ and test alignments $\mathcal{A}_{test}$. We employ an incremental batch-wise pool-based framework. At step i, there is a pool of potential queries $\mathcal{P}_i \subseteq \left(\mathcal{E}^L \cup \mathcal{E}^R\right)$, from which a heuristic selects a fixed number of elements $\mathcal{Q}_i \subseteq \mathcal{P}_i$, where $b = |\mathcal{Q}_i|$ is often called the budget. These queries are then passed to an alignment oracle $\mathfrak{O}$ simulating the labeling process and returning $\mathfrak{O}(\mathcal{Q}_i) = (\mathcal{A}_i, \mathcal{X}_i^L, \mathcal{X}_i^R)$, where the first component comprises the discovered alignments $\mathcal{A}_i = \{(a, a') \in \mathcal{A}_{train} \mid \{a, a'\} \cap \mathcal{Q}_i \neq \emptyset\}$, and the last components the exclusive nodes $\mathcal{X}_i^L = \mathcal{X}^L \cap \mathcal{Q}_i$, and $\mathcal{X}_i^R$ analogously. Afterward, the labeled nodes are removed from the pool, i.e. $\mathcal{P}_{i+1} = \mathcal{P}_i \setminus \left(\mathcal{A}_i^L \cup \mathcal{A}_i^R \cup \mathcal{X}_i^L \cup \mathcal{X}_i^R\right)$. Note that when dealing with 1:n matchings, we remove all matches from the set of available nodes, despite some of them having additional alignment partners. As each alignment edge can be retrieved using any of its endpoints, this does not pose a problem. Now, the model is trained with all already found alignments, denoted by $\mathcal{A}_{\leq i}$, and without all exclusive nodes discovered so far, denoted by $\mathcal{X}_{\leq i}^L, \mathcal{X}_{\leq i}^R$, given as

$$\mathcal{A}_{\leq i} = \bigcup_{j \leq i} \mathcal{A}_j, \quad \mathcal{X}_{\leq i}^L = \bigcup_{j \leq i} \mathcal{X}_j^L, \quad \mathcal{X}_{\leq i}^R = \bigcup_{j \leq i} \mathcal{X}_j^R.$$

Following [27,32], we do not reset the parameters but warm-start the model with the previous iteration's parameters. The pool is initialized with $\mathcal{P}_0 := \mathcal{A}^L_{train} \cup \mathcal{A}^R_{train} \cup \mathcal{X}^L \cup \mathcal{X}^R$. We exclude nodes that are not contained in the training alignment, but in the test alignments, as in this case, either a test alignment has to be revealed, or a node has to be unfaithfully classified as exclusive. An illustration of the pool construction and an example query of size one is given in Fig. 3.

6 Experiments

6.1 Setup

For evaluation, we use both subsets of the WK3l-15k dataset [7][1]. Similarly to [28] we extract additional entity alignments from the triple alignments. Besides using the official train-test split, we perform an additional 80-20 train-validation split shared across all runs. We additionally evaluate the transferability of the hyperparameter settings. One of the challenges in active learning is that hyperparameter search for a new dataset is not possible because of the lack of labeled data at the beginning. Therefore, for the evaluation of the second subset `en-fr`, we use the best hyperparameter settings which we obtained using `en-de` and compare how consistent are results for both subsets.

We employ a GNN-based model, GCN-Align [40]. We use the best settings as found in [3]. To allow for Monte-Carlo Dropout estimation for the Bayesian heuristics, we additionally add a dropout layer between the embeddings and the GCN and vary the dropout rate. We employ a margin-based matching loss, and we exclude so far identified exclusive nodes from the pool of negative samples. Following [2], we use 25 runs with different dropout masks for Bayesian approaches. As evaluation protocol, we always retrieve 200 queries from the heuristic, update the exclusives and alignments using the oracle, and train the model for up to 4k epochs with early stopping on validation mean reciprocal rank (MRR) evaluated every 20 epochs, with a patience value of 200 epochs. There are different scores for the evaluation of entity alignment, which evaluate different performance aspects [4]. In this work, we report Hits@1 (H@1) on the test alignments since this metric is most relevant for the applications. We selected the heuristics' hyperparameters according to the AUC of the step vs. validation H@1 score. Using the best hyperparameter configuration, we re-ran the experiments five times and report the mean and the standard deviation of the results on the test set.

6.2 Results

Removal of Exclusives – Figure 4 shows the test performance of the random selection baseline heuristic compared to the number of queries, with the

[1] Note that the frequently used DBP15k dataset is not suitable for our experiments due to its construction. Exclusive nodes in DBP15K are exactly those having a degree of one and are therefore trivial to identify.

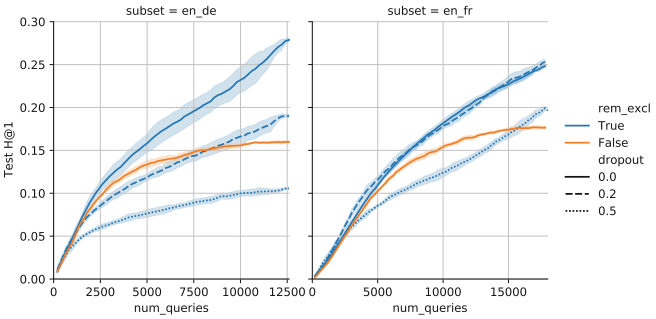


Fig. 4. Performance vs number of queries for random baseline with different levels of dropout, and when removing exclusive nodes from message passing. Removing exclusives significantly improves the final performance.

standard deviation across five runs shown as shaded areas. As can be seen by comparing the two solid lines, removing exclusives is advantageous, in particular, when many queries are performed, i.e., many exclusives are removed. Therefore, we focus the subsequent analysis only on the case, when found exclusives are removed from the graph. Moreover, we can see that using a high dropout value of 0.5 is disadvantageous on both datasets. While a dropout value of 0.2 also hurts performance for the `en-de` subset, it does not have a negative influence on `en-fr`.

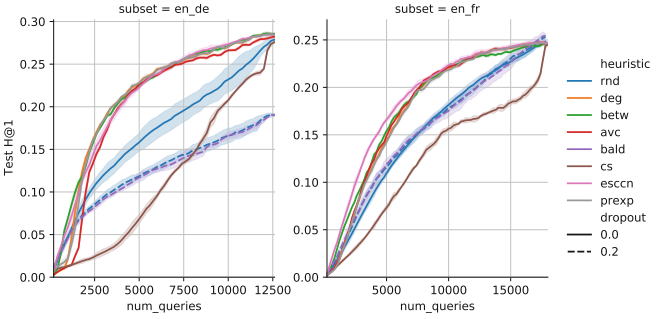


Fig. 5. Performance on test alignments vs. number of queries for different heuristics.

Comparison of Different Heuristics – Figure 5 compares the performance of different heuristics through all steps. Since there is a large overlap across different heuristics, we additionally compute AUC for each heuristic and report it in Table 1. From the results, we observe that our expectations about the performance of different heuristics are mostly confirmed. Most of the heuristics perform significantly better than random sampling. Our intuitions about possible problems with *coreset* in the context of entity alignment are also verified:

Table 1. Mean and standard deviation of AUC number of queries vs. test hits @ 1 aggregated from five different runs for each heuristic and subset. The * symbol indicates significant results compared to the `rnd` baseline according to unequal variances t-test (Welch's t-test) with $p < 0.01$.

Subset	en-de	en-fr
avc	0.2020 ± 0.0005*	0.1748 ± 0.0005*
bald	0.1222 ± 0.0039*	0.1514 ± 0.0013
betw	0.2134 ± 0.0005*	0.1773 ± 0.0004*
cs	0.1117 ± 0.0011*	0.1185 ± 0.0016*
deg	0.2105 ± 0.0005*	0.1741 ± 0.0005*
esccn	0.2114 ± 0.0006*	0.1828 ± 0.0021*
prexp	0.2103 ± 0.0009*	0.1733 ± 0.0009*
rnd	0.1605 ± 0.0040	0.1510 ± 0.0019

The heuristic performs consistently worse than the random sampling baseline. On the other hand, our new *esccn* heuristic, which also tries to cover embedding space, but uses most central nodes instead, is one of the best performing heuristics. We also observe an inferior performance of the uncertainty-based heuristic, which performance is comparable with the random heuristic. Note, that we also evaluated softmax entropy and maximal variation ratio heuristics from [17] and their performance was similar. Overall, we see similar patterns for both subsets: There is a set of good performing heuristics and their performance is very similar.

Performance in Earlier Stages – In many real-life applications, the labeling budget is limited; therefore, the model performance in the first steps is of higher relevance. Therefore, in Fig. 6, we analyze the model performance in the first

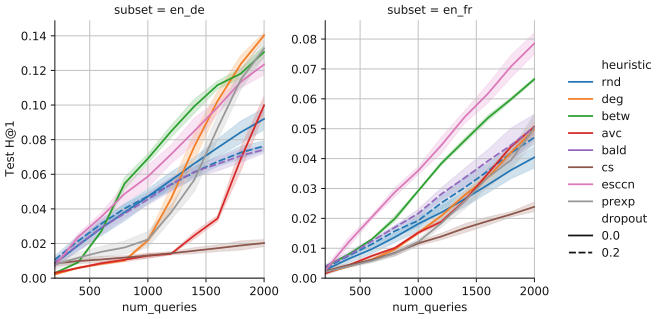


Fig. 6. Performance on test alignments vs. number of queries for different heuristics. This figure shows only queries up to 2,000, i.e., the region where not many alignments have been found so far.

2,000 iterations. We observe that the *escnn* and *betw* heuristics compete for first prize and that towards the end, they are superseded by other heuristics.

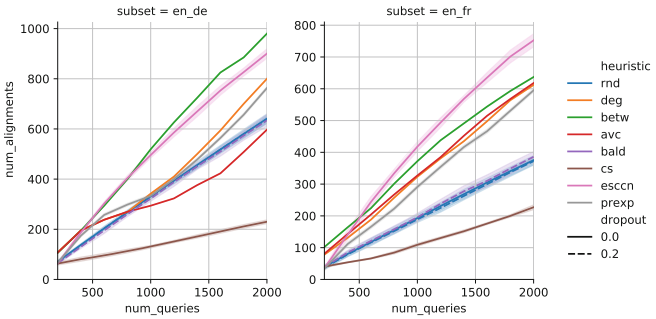


Fig. 7. Number of found training alignments vs. number of queries for different heuristics. This figure shows only queries up to 2,000, i.e., the region where not many alignments have been found so far.

Influence of Positive Matchings – In Fig. 7, we show the number of alignment pairs identified by each heuristic in the first 2,000 steps. For most heuristics, the plots look very similar to the plots in Fig. 6 above with the performance on the y axis. In Fig. 4, we also saw that the removal of exclusive nodes affects the performance only at later iterations. Therefore, we can conclude that finding nodes with matches is especially important in the early training stages.

On the whole, we can conclude that node centrality based heuristics like *betw* are the right choice for active learning for entity alignment. It achieves performance comparable with model-based approaches and does not require access to model predictions during the labeling process. The labeling ordering can be precomputed and does not change, also facilitating to parallelize the labeling process for a fixed budget to multiple annotators, e.g., using systems such as Amazon Mechanical Turk.

7 Conclusion

In this paper, we introduced the novel task of active learning for entity alignment and discussed its differences to the classical active learning setting. Moreover, we proposed several different heuristics, both, adaptions of existing heuristics used for classification, as well as heuristics specifically designed for this particular task. In a thorough empirical analysis, we showed strong performance of simple centrality and graph cover heuristics, while adaptations of state-of-the-art heuristics for classification showed inferior performance. For future work, we envision transferring our approaches to other graph matching problems, such as matching road networks [13] or approximating graph edit distance [24]. Moreover, we aim to study the generalization of our findings to other datasets and models.

Acknowledgement. This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

References

1. Bast, H., Björn, B., Haussmann, E.: Semantic search on text and knowledge bases. Found. Trends Inf. Retrieval **10**(2–3), 119–271 (2016)
2. Beluch, W.H., Genewein, T., Nürnberger, A., Köhler, J.M.: The power of ensembles for active learning in image classification. In: CVPR, pp. 9368–9377. IEEE Computer Society (2018)
3. Berrendorf, M., Faerman, E., Melnychuk, V., Tresp, V., Seidl, T.: Knowledge graph entity alignment with graph convolutional networks: lessons learned. arXiv preprint arXiv:1911.08342 (2019)
4. Berrendorf, M., Faerman, E., Vermue, L., Tresp, V.: Interpretable and fair comparison of link prediction or entity alignment methods with adjusted mean rank. arXiv preprint arXiv:2002.06914 (2020)
5. Cai, H., Zheng, V.W., Chang, K.C.C.: Active learning for graph embedding. arXiv preprint arXiv:1705.05085 (2017)
6. Cao, Y., Liu, Z., Li, C., Liu, Z., Li, J., Chua, T.: Multi-channel graph neural network for entity alignment. In: ACL, vol. 1, pp. 1452–1461. ACL (2019)
7. Chen, M., Tian, Y., Yang, M., Zaniolo, C.: Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In: IJCAI, pp. 1511–1517. ijcai.org (2017)
8. Cortés, X., Serratosa, F.: Active-learning query strategies applied to select a graph node given a graph labelling. In: Kropatsch, W.G., Artner, N.M., Haxhimusa, Y., Jiang, X. (eds.) GbRPR 2013. LNCS, vol. 7877, pp. 61–70. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38221-5_7
9. Cortés, X., Serratosa, F., Solé-Ribalta, A.: Active graph matching based on pairwise probabilities between nodes. In: Gimel'farb, G., et al. (eds.) SSPR /SPR 2012. LNCS, vol. 7626, pp. 98–106. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34166-3_11
10. Das, K., Samanta, S., Pal, M.: Study on centrality measures in social networks: a survey. Soc. Netw. Anal. Min. **8**(1), 1–11 (2018). https://doi.org/10.1007/s13278-018-0493-2
11. Dietz, L., Kotov, A., Meij, E.: Utilizing knowledge graphs for text-centric information retrieval. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, pp. 1387–1390. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3209978.3210187
12. Faerman, E., Borutta, F., Fountoulakis, K., Mahoney, M.W.: LASAGNE: locality and structure aware graph node embedding. In: WI, pp. 246–253. IEEE Computer Society (2018)
13. Faerman, E., Voggenreiter, O., Borutta, F., Emrich, T., Berrendorf, M., Schubert, M.: Graph alignment networks with node matching scores. In: Graph Representation Learning NeurIPS 2019 Workshop (2019)
14. Gal, Y.: Uncertainty in deep learning. Ph.D. thesis, University of Cambridge (2016)
15. Gal, Y., Ghahramani, Z.: Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: ICML JMLR Workshop and Conference Proceedings, vol. 48, pp. 1050–1059. JMLR.org (2016)

16. Gal, Y., Islam, R., Ghahramani, Z.: Deep Bayesian active learning with image data. In: Proceedings of Machine Learning Research (ICML), vol. 70, pp. 1183–1192. PMLR (2017)
17. Gao, L., Yang, H., Zhou, C., Wu, J., Pan, S., Hu, Y.: Active discriminative network representation learning. In: IJCAI, pp. 2142–2148. ijcai.org (2018)
18. Geifman, Y., El-Yaniv, R.: Deep active learning over the long tail. arXiv preprint arXiv:1711.00941 (2017)
19. Guo, L., Sun, Z., Hu, W.: Learning to exploit long-term relational dependencies in knowledge graphs. In: Proceedings of Machine Learning Research (ICML), vol. 97, pp. 2505–2514. PMLR (2019)
20. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
21. Houlsby, N., Huszár, F., Ghahramani, Z., Lengyel, M.: Bayesian active learning for classification and preference learning. arXiv preprint arXiv:1112.5745 (2011)
22. Lewis, D.D., Catlett, J.: Heterogeneous uncertainty sampling for supervised learning. In: ICML, pp. 148–156. Morgan Kaufmann (1994)
23. Li, C., Cao, Y., Hou, L., Shi, J., Li, J., Chua, T.: Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model. In: EMNLP/IJCNLP, vol. 1, pp. 2723–2732. ACL (2019)
24. Li, Y., Gu, C., Dullien, T., Vinyals, O., Kohli, P.: Graph matching networks for learning the similarity of graph structured objects. In: Proceedings of Machine Learning Research (ICML), vol. 97, pp. 3835–3845. PMLR (2019)
25. Mahdisoltani, F., Biega, J., Suchanek, F.M.: YAGO3: a knowledge base from multilingual wikipedias. In: CIDR (2015). www.cidrdb.org
26. Malmi, E., Gionis, A., Terzi, E.: Active network alignment: a matching-based approach. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1687–1696 (2017)
27. Ostapuk, N., Yang, J., Cudré-Mauroux, P.: ActiveLink: deep active learning for link prediction in knowledge graphs. In: WWW, pp. 1398–1408. ACM (2019)
28. Pei, S., Yu, L., Hoehndorf, R., Zhang, X.: Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference. In: WWW, pp. 3130–3136. ACM (2019)
29. Puthal, D., Nepal, S., Paris, C., Ranjan, R., Chen, J.: Efficient algorithms for social network coverage and reach. In: BigData Congress, pp. 467–474. IEEE Computer Society (2015)
30. Sener, O., Savarese, S.: Active learning for convolutional neural networks: a core-set approach. In: ICLR (Poster). OpenReview.net (2018)
31. Settles, B.: Active learning literature survey. University of Wisconsin-Madison Department of Computer Sciences, Technical report (2009)
32. Shen, Y., Yun, H., Lipton, Z.C., Kronrod, Y., Anandkumar, A.: Deep active learning for named entity recognition. In: ICLR (Poster). OpenReview.net (2018)
33. Speer, R., Chin, J., Havasi, C.: Conceptnet 5.5: an open multilingual graph of general knowledge. In: AAAI, pp. 4444–4451. AAAI Press (2017)
34. Sun, Z., Hu, W., Li, C.: Cross-lingual entity alignment via joint attribute-preserving embedding. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10587, pp. 628–644. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68288-4_37
35. Sun, Z., Hu, W., Zhang, Q., Qu, Y.: Bootstrapping entity alignment with knowledge graph embedding. In: IJCAI, pp. 4396–4402 (2018)
36. Sun, Z., et al.: Knowledge graph alignment network with gated multi-hop neighborhood aggregation. arXiv preprint arXiv:1911.08936 (2019)

37. Trisedya, B.D., Qi, J., Zhang, R.: Entity alignment between knowledge graphs using attribute embeddings. In: AAAI, pp. 297–304. AAAI Press (2019)
38. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
39. Wang, K., Zhang, D., Li, Y., Zhang, R., Lin, L.: Cost-effective active learning for deep image classification. IEEE Trans. Circ. Syst. Video Techn. **27**(12), 2591–2600 (2017)
40. Wang, Z., Lv, Q., Lan, X., Zhang, Y.: Cross-lingual knowledge graph alignment via graph convolutional networks. In: EMNLP, pp. 349–357. ACL (2018)
41. Wu, Y., Xu, Y., Singh, A., Yang, Y., Dubrawski, A.: Active learning for graph neural networks via node feature propagation. arXiv preprint arXiv:1910.07567 (2019)
42. Xu, K., et al.: Cross-lingual knowledge graph alignment via graph matching neural network. In: ACL, vol. 1, pp. 3156–3161. ACL (2019)
43. Yang, L., Zhang, Y., Chen, J., Zhang, S., Chen, D.Z.: Suggestive annotation: a deep active learning framework for biomedical image segmentation. In: Descoteaux, M., et al. (eds.) MICCAI 2017. LNCS, vol. 10435, pp. 399–407. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66179-7_46
44. Zhang, Q., Sun, Z., Hu, W., Chen, M., Guo, L., Qu, Y.: Multi-view knowledge graph embedding for entity alignment. In: IJCAI, pp. 5429–5435. ijcai.org (2019)
45. Zhang, Y., Lease, M., Wallace, B.C.: Active discriminative text representation learning. In: AAAI, pp. 3386–3392. AAAI Press (2017)
46. Zhu, Q., Zhou, X., Wu, J., Tan, J., Guo, L.: Neighborhood-aware attentional representation for multilingual knowledge graphs. In: IJCAI, pp. 1943–1949. ijcai.org (2019)

5 Improving Inductive Link Prediction Using Hyper-Relational Facts

This chapter comprises the publication

> Mehdi Ali*, **Max Berrendorf***, Mikhail Galkin, Veronika Thost, Tengfei Ma, Volker Tresp, and Jens Lehmann. „Improving Inductive Link Prediction Using Hyper-relational Facts.“ In: *The Semantic Web – ISWC 2021* (2021). * equal contribution, awarded with Best Research Paper Award, pp. 74–92. DOI: 10.1007/978-3-030-88361-4_5

and the code is available at

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

Declaration of Authorship The research idea was developed and conceptualized by Max Berrendorf, Mehdi Ali and Mikhail Galkin. Max Berrendorf and Mehdi Ali did the main part of the implementation, with Max Berrendorf focussing on the models, training loop and evaluation, and Mehdi Ali taking care of the data splits and hyperparameter optimization. Mehdi Ali conducted the experiments, and Max Berrendorf additionally analyzed their results. Max Berrendorf, Mehdi Ali and Mikhail Galkin jointly wrote manuscript, and all authors revised it.

Improving Inductive Link Prediction Using Hyper-relational Facts

Mehdi Ali[1,2(✉)], Max Berrendorf[3], Mikhail Galkin[4], Veronika Thost[5], Tengfei Ma[5], 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}@iais.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.

A. Hotho et al. (Eds.): ISWC 2021, LNCS 12922, pp. 74–92, 2021.
https://doi.org/10.1007/978-3-030-88361-4_5

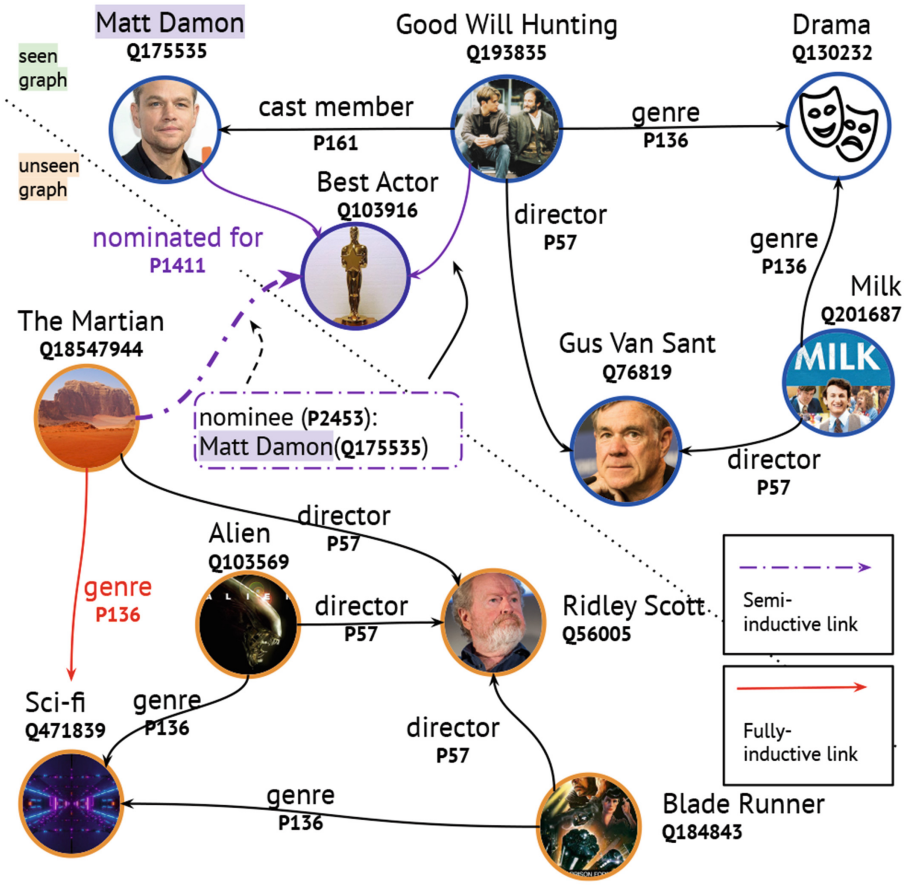


Fig. 1. Different types of inductive LP. Semi-inductive: the link between `The Martian` and `Best Actor` from the seen graph. Fully-inductive: the `genre` link between unseen entities given a new unseen subgraph *at inference time*. The qualifier `(nominee: Matt Damon)` over the original relation `nominated 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 entity[1] $h, t \in \mathcal{E}$, 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

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

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 ↔ Unseen	Unseen ↔ 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., $\mathcal{S}_{eval} \subseteq \mathbb{S}(\mathcal{E}_\circ, \mathcal{R}, \mathcal{E}_\circ, \mathcal{E}_\circ)$. This is made possible by providing an auxiliary inference graph $\mathcal{S}_{inf} \subseteq \mathbb{S}(\mathcal{E}_\circ, \mathcal{R}, \mathcal{E}_\circ, \mathcal{E}_\circ)$ 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., $\mathcal{S}_{eval} \subseteq \mathbb{S}(\mathcal{E}_\bullet, \mathcal{R}, \mathcal{E}_\circ, \mathcal{E}_\bullet) \cup \mathbb{S}(\mathcal{E}_\circ, \mathcal{R}, \mathcal{E}_\bullet, \mathcal{E}_\bullet)$. 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:

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

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: $[\mathbf{x}'_h, \mathbf{x}'_r, \mathbf{x}'_{q_1^r}, \mathbf{x}'_{q_1^e}, [\text{PAD}], \ldots]$. 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. 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:

$$f(h,r,t,q) = g(\mathbf{x}'_h, \mathbf{x}'_r, \mathbf{x}'_{q_1^r}, \mathbf{x}'_{q_1^e}, \ldots)^T \mathbf{x}'_t \quad \text{with}$$
$$g(\mathbf{x}'_1, \ldots, \mathbf{x}_k) = \text{Agg}(\text{Transformer}([\mathbf{x}'_1, \ldots, \mathbf{x}'_k]))$$

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	5.75	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	22.65	**28.03**	**6.69**	**3.49**	8.47	12.04
QBLP	4	20.19	10.70	3.99	16.12	**24.52**	31.30	5.87	2.37	7.85	**13.93**
QBLP	6	23.65	7.87	2.75	10.44	17.86	34.35	6.53	2.95	**9.29**	13.13

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	2.56	1.17	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**	13.44	52.78	2.62	0.74	3.55	5.78
StarE	4	38.91	6.40	2.83	8.94	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	30.37	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	39.12	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**	5.99	20.36	11.77	**24.86**	**32.26**
QBLP	4	11.38	34.35	19.41	33.90	40.20	12.18	**21.05**	**12.32**	24.07	30.09
QBLP	6	4.98	25.94	15.20	30.06	38.70	**5.73**	19.50	11.14	24.73	31.60

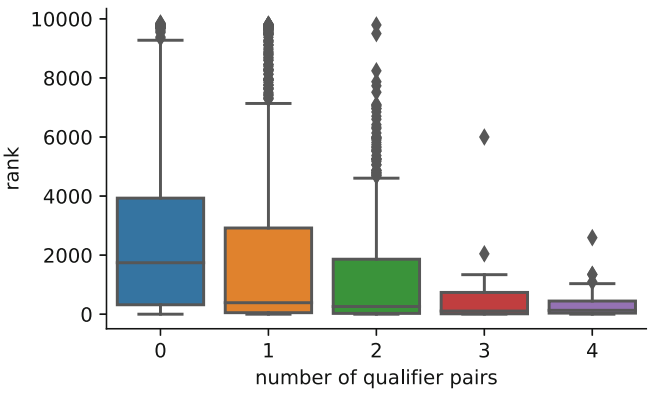


Fig. 2. Distribution of individual ranks for head/tail prediction with StarE on WD20K (66) V2. The statements are grouped by the number of qualifier pairs.

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.

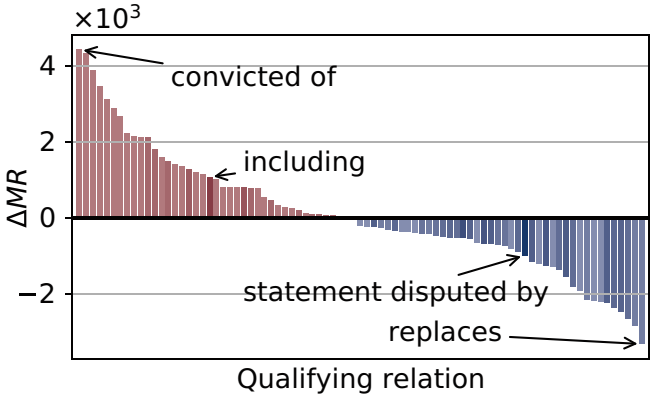


Fig. 3. Rank deviation when masking qualifier pairs containing a certain relation. Transparency is proportional to the occurrence frequency, bar height/color indicates difference in MR *for evaluation statements using this qualifying relation* if the pair is masked. More negative deltas correspond to 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
P2241	reason for deprecation	-26.44
P47	shares border with	-28.91
P750	distributed by	-29.12
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]:

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

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]:

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

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

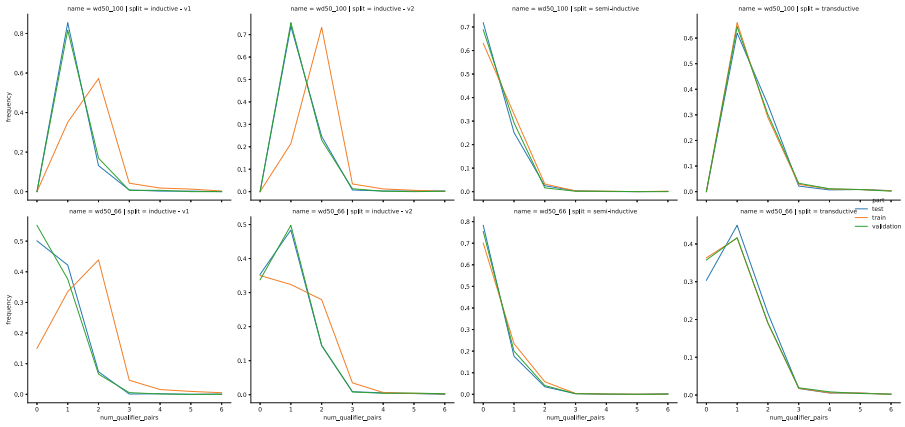


Fig. 4. Percentage of statements with the given number of available qualifier pairs for all datasets and splits.

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 (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 (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 (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 (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 (2020)
21. Liu, Y., et al.: RoBERTa: a robustly optimized BERT pretraining approach. CoRR arXiv: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. Vrandecic, 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 (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)

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

This chapter includes the following publication:

> 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

Declaration of Authorship The idea was developed and discussed by Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt, Sahand Sharifzadeh and Laurent Vermue after each of the authors independently discovered (different) issues with existing works. Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt, and Laurent Vermue analyzed existing implementations, developed the library's architecture and implemented it. Mehdi Ali made a first draft of the manuscript and Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt, Sahand Sharifzadeh, and Laurent Vermue developed and wrote it. All authors revised the manuscript.

Journal of Machine Learning Research 22 (2021) 1-6 Submitted 7/20; Revised 2/21; Published 3/21

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

Mehdi Ali* MEHDI.ALI@CS.UNI-BONN.DE
Smart Data Analytics Group, University of Bonn & Fraunhofer IAIS

Max Berrendorf* BERRENDORF@DBS.IFI.LMU.DE
Ludwig-Maximilians-Universität München

Charles Tapley Hoyt* CHARLES.HOYT@ENVEDATX.COM
Enveda Biosciences

Laurent Vermue* LAUVE@DTU.DK
Technical University of Denmark

Sahand Sharifzadeh SHARIFZADEH@DBS.IFI.LMU.DE
Ludwig-Maximilians-Universität München

Volker Tresp VOLKER.TRESP@SIEMENS.COM
Ludwig-Maximilians-Universität München & Siemens AG

Jens Lehmann JENS.LEHMANN@CS.UNI-BONN.DE
Smart Data Analytics Group, University of Bonn & Fraunhofer IAIS

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 KnowlEdge 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 refereed 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`.

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 Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 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.

7 Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework

This chapter includes the following publication:

> 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: *CoRR* abs/2006.13365 (2020). * equal contribution. arXiv: 2006.13365

All experimental artifacts are publicly available at

> Mehdi Ali, **Max Berrendorf***, Charles Tapley Hoyt*, Laurent Vermue*, and Mikhail Galkin. *pykeen/benchmarking: Accompanying arXiv announcement*. Version v1.0. * equal contribution. June 2020. DOI: 10.5281/zenodo.3907252. URL: https://doi.org/10.5281/zenodo.3907252

Declaration of Authorship The idea was developed and discussed by Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt, Sahand Sharifzadeh and Laurent Vermue in close interaction with the development of the PyKEEN library. Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt and Laurent Vermue did the implementation. Mehdi Ali, Laurent Vermue and Mikhail Galkin conducted the experiments and Max Berrendorf coordinated the collection of all results. Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt and Laurent Vermue analyzed the results. Max Berrendorf additionally implemented and conducted the relation-specific analysis, and evaluated its results. Max Berrendorf, Mehdi Ali, Charles Tapley Hoyt and Laurent Vermue wrote the manuscript, and all authors revised the manuscript.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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

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

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

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

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

1 Introduction

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

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

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

This paper makes two major contributions towards addressing these challenges:

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

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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

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

2 Knowledge Graphs

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

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

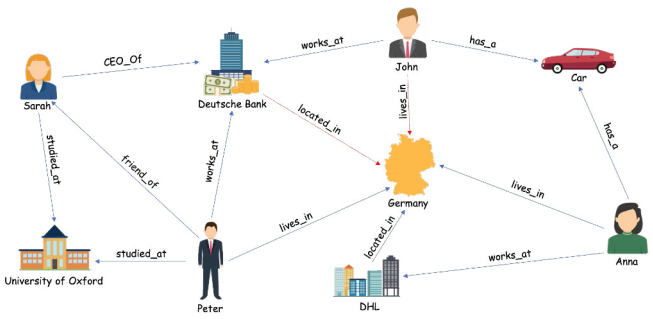


Fig. 1. Exemplary KG: nodes represent entities and edges their respective relations.

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

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

3 Knowledge Graph Embedding Models

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

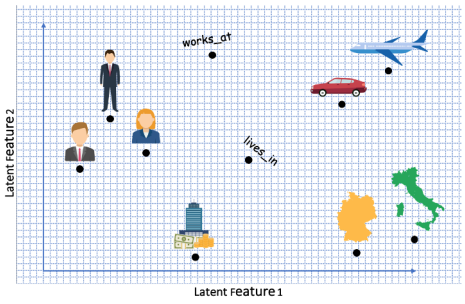


Fig. 2. An example embedding of the entities and relations from the knowledge graph portrayed by Figure 2

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

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

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

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

3.1 Interaction Models

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

3.1.1 Translational Distance Interaction Models

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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:

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

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

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

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:

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

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

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

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

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

3.1.2 Semantic Matching Interaction Models

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2 Training Approaches

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

3.2.1 Local closed world assumption

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

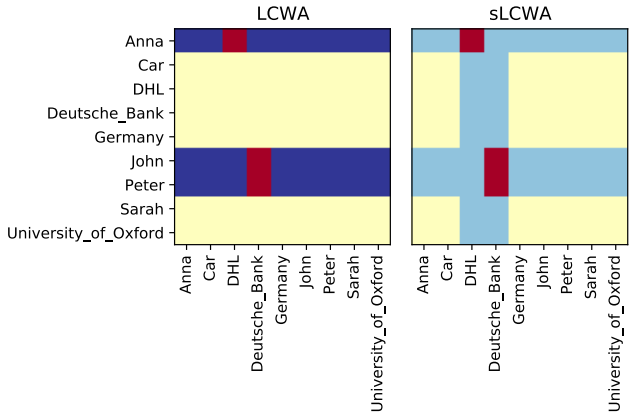


Fig. 3. Visualization of different training approaches for the relation `works_at` in the KG in Figure 1. Red color indicates positive examples, i.e. true triples present in the KG. Dark blue color denotes triples used as negative examples in LCWA. Light blue color sampling candidates for negative examples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

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

3.2.2 Stochastic local closed world assumption

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

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

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

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

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

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

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

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]:

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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

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

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

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

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

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

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

3.3.2 *Pairwise Loss Functions*

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

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

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

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

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

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

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

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

3.3.3 *Setwise Loss Functions*

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

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

$$p((h_i', r, t_i')|(h_i, r_i, t_i)) = \frac{\exp(\alpha f(h_i', r, t_i'))}{\sum_{j=1}^{n} \exp(\alpha f(h_j', r, t_j'))} \ , \tag{35}$$

where $(h_i, r_i, t_i) \in \mathcal{K}$ denotes a true triple, $\{(h_i', r, t_i')\}_{i=1}^{K}$ it's set of negative samples generated, and $\alpha \in \mathbb{R}$ a temperature parameter. Because sampling from this distribution may be computationally expensive, the probabilities obtained by Equation 35 are used to weight the generated negative examples in the loss function [23].

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

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

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

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

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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

3.4 Explicitly Modeling Inverse Relations

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

4 Evaluation Metrics for KGEMs

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

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

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

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

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

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

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

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

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

Smaller values indicate better performance.

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

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

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

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

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

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

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

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

Larger values indicate better performance.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

5 Existing Benchmark Datasets

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

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

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

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

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

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

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

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

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

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

TABLE 1
Existing Benchmark Datasets.

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

6 Reproducibility Studies

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

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

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

6.1 Reproductions Requiring Alternate Hyper-Parameters

One of the observations we made is that for some experiments, results could only be reproduced with a different set of hyper-parameter values. For instance, the results for TransE could only be reproduced by adapting the batch

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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

6.2 Unreported Hyper-parameters Impedes Reproduction

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

6.3 Two Perspectives: Publication versus Implementation

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

6.4 Lack of Official Implementations Impedes Reproduction

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

6.5 Reproducibility is Dependent on The Ranking Approach

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

7 Benchmarking

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

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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

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

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

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

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

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

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

7.1 Results on the Kinships Dataset

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

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

Impact of the Loss Function

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

Impact of Explicitly Modeling Inverse Relations

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

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

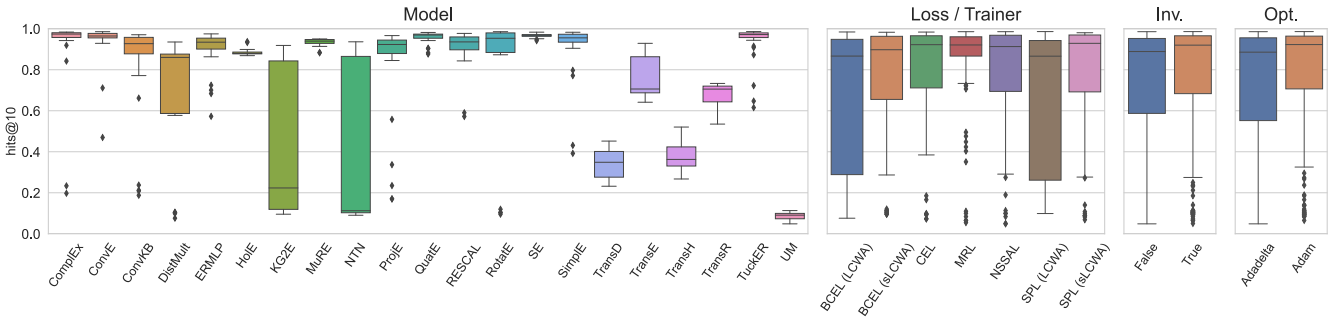


Fig. 4. Overall hits@10 results for Kinships where box-plots summarize the best results across different configurations, i.e., combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

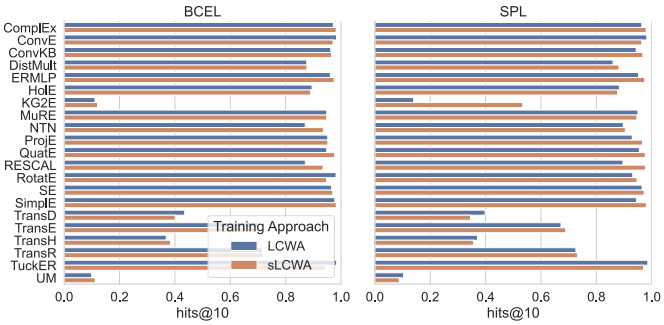


Fig. 5. Impact of training approach on the performance for a fixed interaction model and loss function for the Kinships dataset based on Adam.

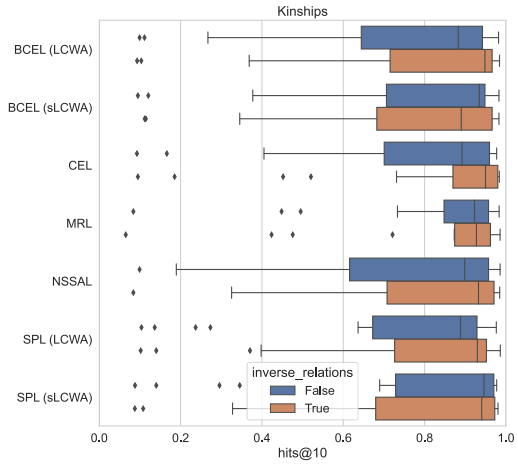


Fig. 6. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the Kinships dataset.

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

7.2 Results on the WN18RR Dataset

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

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

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

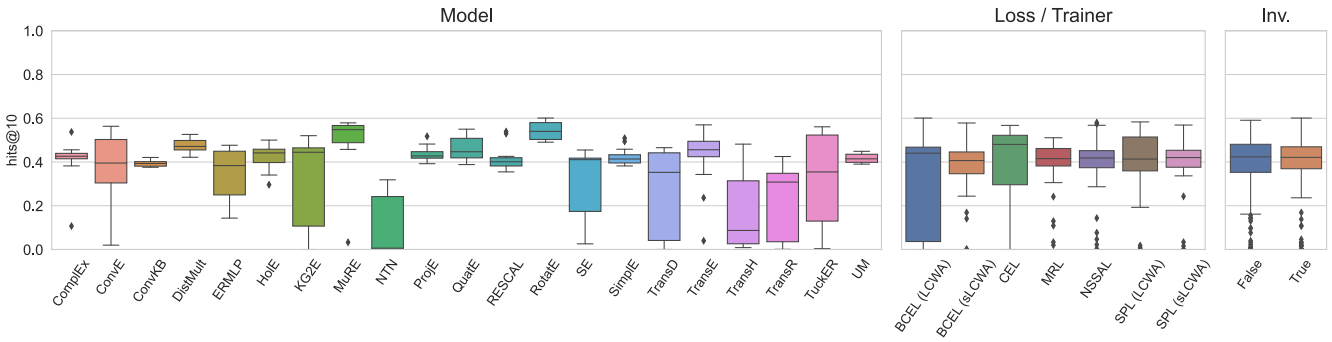


Fig. 7. Overall hits@10 results for WN18RR where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

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

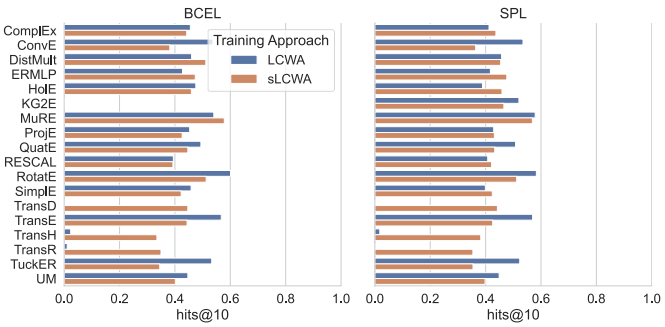


Fig. 8. Impact of training approach on the performance for a fixed interaction model and loss function for the WN18RR dataset.

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

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

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

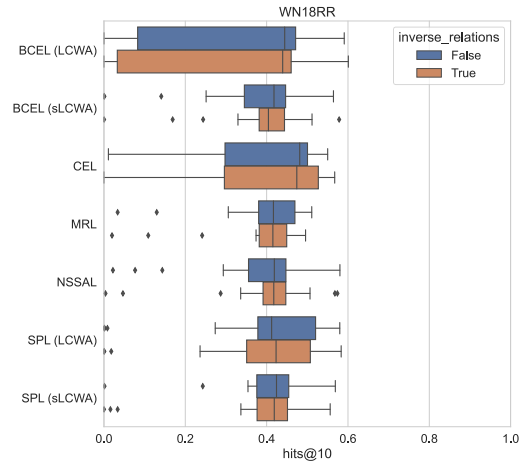


Fig. 9. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the WN18RR dataset.

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

7.3 Results on the FB15K-237 Dataset

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

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

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

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

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

7.4 Results on the YAGO3-10 Dataset

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

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

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

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

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

8 Relational Pattern Analysis

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

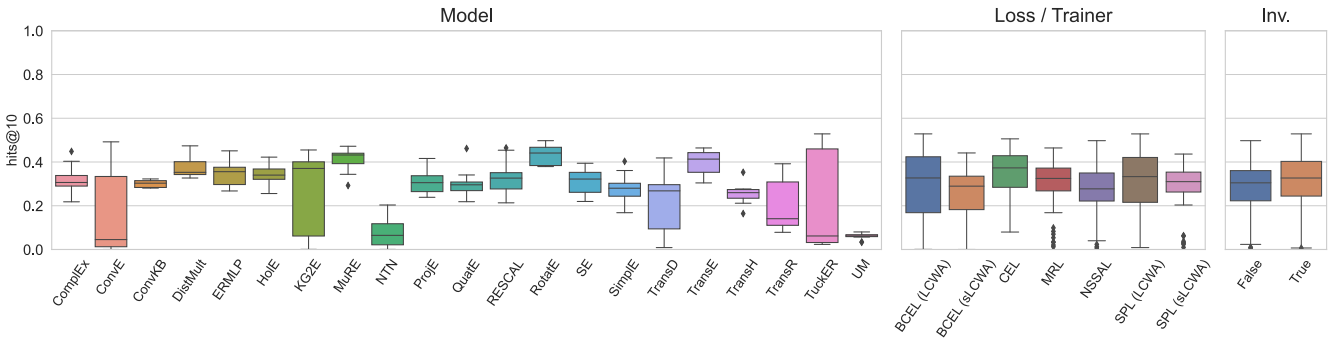


Fig. 10. Overall hits@10 results for FB15K-237 where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

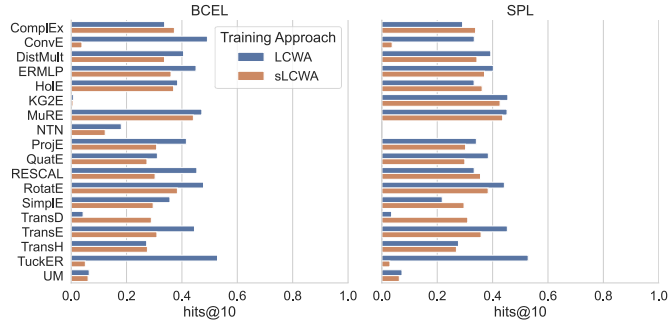


Fig. 11. Impact of training approach on the performance for a fixed interaction model and loss function for the FB15K-237 dataset.

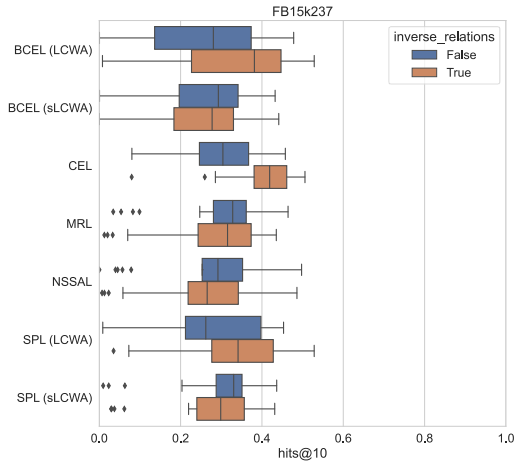


Fig. 12. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the FB15K-237 dataset.

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

8.1 Relational Patterns and their Detection

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

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

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

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

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

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

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

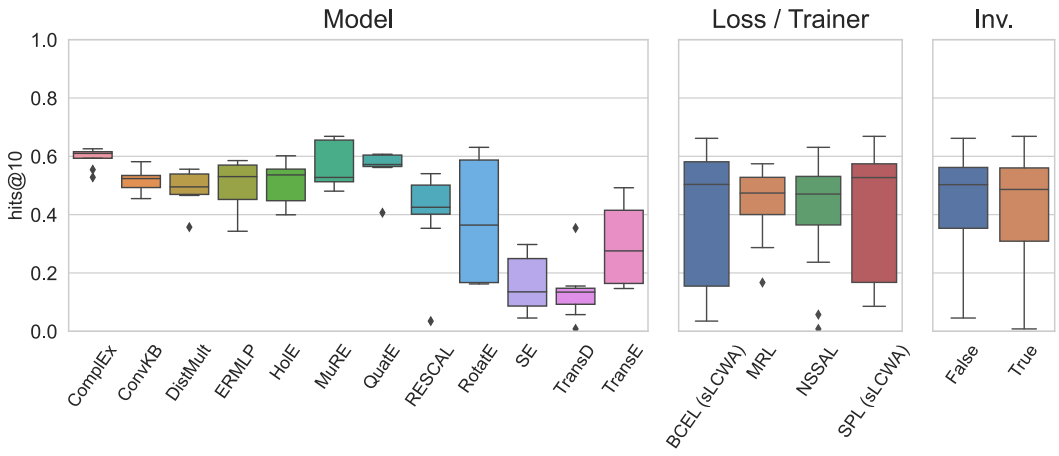


Fig. 13. Overall hits@10 results for YAGO3-10 where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations. In contrast, to the previous datasets, the models have only been trained based on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

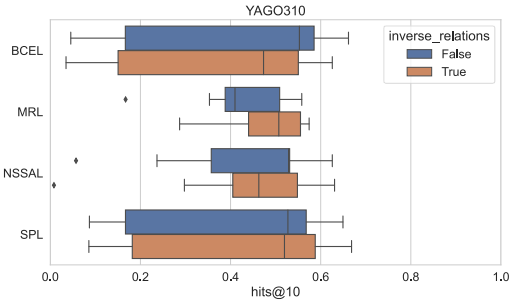


Fig. 14. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the YAGO3-10 dataset.

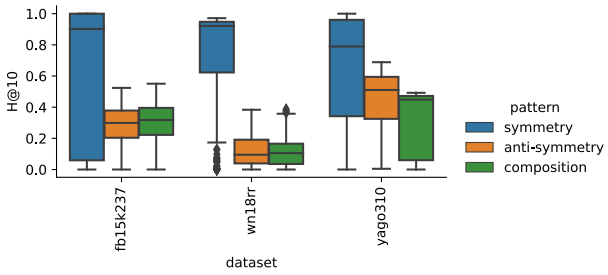


Fig. 15. Performance Distribution of all best models per configuration in H@10.

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

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

8.2 Relation Patterns in Benchmark Datasets

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

8.3 Experimental Setup

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

8.4 Results

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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

9 Discussion & Future Work

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

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

- For a set of experiments, the results can sometimes only be reproduced with a different set of 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 KGEM should be used with caution and should be differentiated from the actual interaction model since our results highlight that the specific *combination* of the interaction model, training approach, loss function, and the usage of explicit inverse relations is often fundamental for the performance.

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

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

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

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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

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

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

ACKNOWLEDGMENT

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

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[43] K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference," in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, 2015, pp. 57–66.

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

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

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

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

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3124805, IEEE Transactions on Pattern Analysis and Machine Intelligence

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

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

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

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

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

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

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

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

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

8 A Critical Assessment of State-of-the-Art in Entity Alignment

This chapter includes the following publication:

> **Max Berrendorf**, Ludwig Wacker, and Evgeniy Faerman. „A Critical Assessment of State-of-the-Art in Entity Alignment.“ In: *Advances in Information Retrieval*. Ed. by Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani. Cham: Springer International Publishing, 2021, pp. 18–32. ISBN: 978-3-030-72240-1. DOI: 10.1007/978-3-030-72240-1_2

In addition, a preprint is available as

> **Max Berrendorf**, Ludwig Wacker, and Evgeniy Faerman. „A Critical Assessment of State-of-the-Art in Entity Alignment.“ In: *CoRR* abs/2010.16314 (2020). arXiv: 2010.16314

and the code is made public as

> **Max Berrendorf**, Ludwig Wacker, and Evgeniy Faerman. *mberr/ea-sota-comparison: Zenodo*. Version v1.1.1. Dec. 2020. DOI: 10.5281/zenodo.4588894

Declaration of Authorship Max Berrendorf identified the research gap, and conducted initial experiments. The idea was then further developed and conceptualized by Max Berrendorf and Evgeniy Faerman, and later on discussed with Ludwig Wacker. Max Berrendorf and Ludwig Wacker did the implementation. Max Berrendorf and Evgeniy Faerman designed the experiments, and Ludwig Wacker conducted most of them. Max Berrendorf and Ludwig Wacker analyzed the results, and discussed the results with Evgeniy Faerman. Max Berrendorf and Evgeniy Faerman wrote the manuscript.

A Critical Assessment of State-of-the-Art in Entity Alignment

Max Berrendorf(✉), Ludwig Wacker, and Evgeniy Faerman

Ludwig-Maximilians-Universität München, Munich, Germany
{berrendorf,faerman}@dbs.ifi.lmu.de, l.wacker@campus.lmu.de

Abstract. In this work, we perform an extensive investigation of two state-of-the-art (SotA) methods for the task of Entity Alignment in Knowledge Graphs. Therefore, we first carefully examine the benchmarking process and identify several shortcomings, making the results reported in the original works not always comparable. Furthermore, we suspect that it is a common practice in the community to make the hyperparameter optimization directly on a test set, reducing the informative value of reported performance. Thus, we select a representative sample of benchmarking datasets and describe their properties. We also examine different initializations for entity representations since they are a decisive factor for model performance. Furthermore, we use a shared train/validation/test split for an appropriate evaluation setting to evaluate all methods on all datasets. In our evaluation, we make several interesting findings. While we observe that most of the time SotA approaches perform better than baselines, they have difficulties when the dataset contains noise, which is the case in most real-life applications. Moreover, in our ablation study, we find out that often different features of SotA method are crucial for good performance than previously assumed. The code is available at https://github.com/mberr/ea-sota-comparison.

Keywords: Knowledge Graph · Entity Alignment · Word embeddings

1 Introduction

The quality of information retrieval crucially depends on the accessible storage of information. Knowledge Graphs (KGs) often serve as such data structure [6]. Moreover, to satisfy diverse information needs, a combination of multiple data sources is often inevitable. Entity Alignment (EA) [2] is the discipline of aligning entities from different KGs. Once aligned, these entities facilitate information transfer between knowledge bases, or even fusing multiple KGs to a single knowledge base.

In this work, our goal is to analyze a SotA approach for the task of EA and identify which factors are essential for its performance. Although papers often use the same dataset in the evaluation and report the same evaluation metrics, the selection of SotA is not a trivial task: as we found out in our analysis, the usage

D. Hiemstra et al. (Eds.): ECIR 2021, LNCS 12657, pp. 18–32, 2021.
https://doi.org/10.1007/978-3-030-72240-1_2

of different types of external information for the initialization or train/test splits of different sizes[1] makes the results in different works incomparable. Therefore, while still guided by the reported evaluation metrics, we identified these common factors among strongly performing methods in multiple works:

- They are based on Graph Neural Networks (GNNs). GNNs build the basis of the most recent works [4,7,9,10,12,14,16–23,25].
- They utilize entity names in the model. Supported by recent advances in word embeddings, these attributes provide distinctive features.
- They consider different types of relations existing in KGs. Most GNNs ignore different relationship types and aggregate them in the preprocessing step.

Given these criteria, we selected Relation-aware Dual-Graph Convolutional Network (RDGCN) [17], as it also has demonstrated impressive performance in recent benchmarking studies [15,24]. Additionally, we include the recently published Deep Graph Matching Consensus (DGMC) [7] method in our analysis for two reasons: the studies mentioned above did not include it, and the authors reported surprisingly good performance, considering that this method does not make use of relation type information.

We start our study by reviewing the used datasets and discussing the initializations based on entity names. Although both methods utilize entity names, the actual usage differs. For comparison, we thus evaluate both methods on all datasets with all available initializations. We also report the zero-shot performance, i.e., when only using initial representations alone, as well as a simple GNN model baseline. Furthermore, we address the problem of hyperparameter optimization. Related works often do not discuss how they chose hyperparameters and, e.g., rarely report validation splits. So far, this problem was not addressed in the community. In the recent comprehensive survey [15], the authors use cross-validation for the estimation of the test performance. The models are either evaluated with hyperparameters recommended for other datasets or selected by not reported procedure. Also, in the published code of the investigated approaches, we could not find any trace of train-validation splits, raising questions about reproducibility and fairness of their comparisons. We thus create a shared split with a test, train, and validation part and extensively tune the model's hyperparameters for each of the dataset/initialization combinations to ensure that they are sufficiently optimized. Finally, we provide an ablation study for many of the parameters of a SotA approach (RDGCN), giving insight into the individual components' contributions to the final performance.

2 Datasets and Initialization

Table 1 provides a summary of a representative sample of datasets used for benchmarking of EA approaches. In the following, we first discuss each dataset's properties and, in the second part, the initialization of entity name attributes.

[1] Commonly used evaluation metrics in EA automatically become better with a smaller size of test set [3].

Table 1. Summary of the used EA datasets. We denote the entity set as $\mathcal{E}$, the relation set as $\mathcal{R}$, the triple set as $\mathcal{T}$, the aligned entities as $\mathcal{A}$ and the exclusive entities as $\mathcal{X}$.

Dataset	Subset	Graph	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{X}\rvert$
DBP15k	zh-en	zh	19,388	1,701	70,414	15,000	4,388
		en	19,572	1,323	95,142	15,000	4,572
	ja-en	ja	19,814	1,299	77,214	15,000	4,814
		en	19,780	1,153	93,484	15,000	4,780
	fr-en	fr	19,661	903	105,998	15,000	4,661
		en	19,993	1,208	115,722	15,000	4,993
WK3l15k	en-de	en	15,126	1,841	209,041	9,783	5,343
		de	14,603	596	144,244	10,021	4,582
	en-fr	en	15,169	2,228	203,356	7,375	7,794
		fr	15,393	2,422	169,329	7,284	8,109
OpenEA	en-de	en	15,000	169	84,867	15,000	0
		de	15,000	96	92,632	15,000	0
	en-fr	en	15,000	193	96,318	15,000	0
		fr	15,000	166	80,112	15,000	0
	d-y	d	15,000	72	68,063	15,000	0
		y	15,000	21	60,970	15,000	0
	d-w	d	15,000	167	73,983	15,000	0
		w	15,000	121	83,365	15,000	0

2.1 Datasets

DBP15k. The DBP15k dataset is the most popular dataset for the evaluation of EA approaches. It has three subsets, all of which base upon DBpedia. Each subset comprises a pair of graphs from different languages. As noted by [2], there exist multiple variations of the dataset, sharing the same entity alignment but differing in the number of exclusive entities in each graph. The alignments in the datasets are always 1:1 alignments, and due to the construction method for the datasets, exclusive entities do not have relations between them, but only to shared entities. Exclusive entities complicate the matching process, and in real-life applications, they are not easy to identify. Therefore, we believe that this dataset describes a realistic use-case only to a certain extent. We found another different variant of DBP15k as part of the PyTorch Geometric repository[2], having a different set of aligned entities. This is likely due to extraction of alignments from data provided by [20] via Google Drive[3] as described in their

[2] https://github.com/rusty1s/pytorch_geometric/blob/d42a690fba68005f5738008a04f375ffd39bbb76/torch_geometric/datasets/dbp15k.py.

[3] https://drive.google.com/open?id=1dYJtj1_J4nYJdrDY95ucGLCuZXDXI7PL.

GitHub repository.[4] As a result, the evaluation results published in [7] are not directly comparable to other published results. In our experiments, we use the (smaller) JAPE variant with approximately 19–20k entities in each graph since it is the predominantly used variant.

OpenEA. The OpenEA datasets published by [15] comprise graph pairs from DBPedia, YAGO, and Wikidata obtained by iterative degree-based sampling to match the degree distribution between the source KG and the extracted subset. The alignments are exclusively 1:1 matchings, and there are no exclusive entities, i.e., every entity occurs in both graphs. We believe that this is a relatively unrealistic scenario. In our experiments, we use all graph pairs with 15k entities (`15K`) in the dense variant (`V2`), i.e., `en-de-15k-v2`, `en-fr-15k-v2`, `d-y-15k-v2`, `d-w-15k-v2`.

WK3l15k. The Wk3l datasets are multi-lingual KG pairs extracted from Wikipedia. As in [2], we extract additional entity alignments from the triple alignments. The graphs contain additional exclusive entities, and there are m:n matchings. We only use the 15k variants, where each graph has approximately 15k entities. There are two graph pairs, `en-de` and `en-fr`. Moreover, the alignments in the dataset are relatively noisy: for example, `en-de` contains besides valid alignments such as ("trieste", "triest"), or ("frederick i, holy roman emperor", "friedrich i. (hrr)"), also ambiguous ones such as ("1", "1. fc saarbrücken"), ("1", "1. fc schweinfurt 05"), and errors such as ("1", "157"), and ("101", "100"). While the noise aggravates alignment, it also reflects a realistic setting.

2.2 Label-Based Initializations

Prepared Translations (DBP15k). For DBP15k, we investigate label-based initializations based on prepared translations to English from [17] and [7] (which, in turn, originate from [20]). Afterwards, they use Glove [11] embeddings to obtain an entity representation. While [17] only provides the final entity representation vectors without further describing the aggregation, [7] splits the label into words (by white-space) and uses the sum over the words' embeddings as entity representation. [17] additionally normalizes the norm of the representations to unit length.

Prepared RDGCN Embeddings (OpenEA). OpenEA [15] benchmarks a large variety of contemporary entity alignment methods in a unified setting, also including RDGCN [17]. Since the graphs DBPedia and YAGO collect data from similar sources, the labels are usually equal. For those graph pairs, the authors propose to delete the labels. However, RDGCN requires a label based initialization. Thus, the authors obtain labels via attribute triples of a pre-defined set of

[4] https://github.com/syxu828/Crosslingula-KG-Matching/blob/56710f8131ae072f00de97eb737315e4ac9510f2/README.md#how-to-run-the-codes.

Table 2. The statistics about label-based initialization in the OpenEA codebase: *attribute* denotes initialization via attribute values for a predefined set of "name attributes". *id* denotes initialization with the last part of the entity URI. For `d-y` this basically leaks ground truth, whereas, for Wikidata, the URI contains only a numeric identifier, thus rendering the initialization "label" useless.

Subset	Side	via attribute	via id	via id (%)
d-w	d	0	15,000	100.00%
	w	8,391	7,301	48.67%
d-y	d	2,883	12,122	80.81%
	y	15,000	0	0.00%

"name-attributes"[5]: `skos:prefLabel`, http://dbpedia.org/ontology/birthName for DBPedia-YAGO, and http://www.wikidata.org/entity/P373, http://www.wikidata.org/entity/P1476 for DBPedia-Wikidata.

However, when investigating the published code, we noticed that if the label is not found via attribute, the last part of the entity URI is used instead. For DBPedia/YAGO, this effectively leaks ground truth since they share the same label. For DBPedia/Wikidata, this results in useless labels for the Wikidata side since their labels are the Wikidata IDs, e.g., `Q3391163`. Table 2 summarizes the frequency of both cases. For `d-w`, DPBedia entities always use the ground truth label. For 49% of the Wikidata entities, useless labels are used for initialization. For `d-y`, YAGO entity representations are always initialized via an attribute triple. For DBPedia, in 81% of all cases, the ground truth label is used. We store these initial entity representations produced by the OpenEA codebase into a file and refer in the following to them as *Sun* initialization (since they are taken from the implementation of [15]).

Multi-lingual BERT (WK3l15k). Since we did not find related work with entity embedding initialization from labels on WK3l15k, we generated those using a pre-trained multi-lingual BERT model [5], `BERT-Base, Multilingual Cased`[6]. Following [5], we use the sum of the last four layers as token representation since it has comparable performance to the concatenation at a quarter of its size. To summarize the token representations of a single entity label, we explore sum, mean, and max aggregation as hyperparameters.

[5] https://github.com/nju-websoft/OpenEA/tree/2a6e0b03ec8cdcad4920704d1c38547a3ad72abe.

[6] https://github.com/google-research/bert/blob/cc7051dc592802f501e8a6f71f8fb3cf9de95dc9/multilingual.md.

3 Methods

We evaluate two SotA EA methods, RDGCN [17] which we reimplemented and DGMC [7] for which we used the original method implementation with adapted evaluation. In the following, we revisit their architectures and highlight differences between the architecture described in the paper and what we found in the published code.

Similarly to all GNN-based approaches, both models employ a Siamese architecture. Therefore, the same model with the same weights is applied to both graphs yielding representations of entities from both KGs. Given these entity representations, the EA approaches compute an affinity matrix that describes the similarity of entity representations from both graphs. Since the main difference between methods is the GNN model in the Siamese architecture, for brevity we only describe how it is applied on a single KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$.

3.1 Relation-Aware Dual-Graph Convolutional Network (RDGCN)

Architecture. The RDGCN [17] model comprises two parts performing message-passing processes applied sequentially. The message passing process performed by the first part can be seen as *relation-aware*. The model tries to learn the importance of relations and weights the messages from the entities connected by these relations correspondingly. The message passing performed by the second component utilizes a simple adjacency matrix indicating the existence of any relations between entities, which we call *standard message passing*. Both components employ a form of skip connections: (weighted) residual connections [8] in the first part and highway layers [13] in the second part.

Relation-Aware Message Passing. The entity embeddings from the first component are computed by several *interaction rounds* comprising four steps

$$\mathbf{X_c} = RC(\mathbf{X_e}), \mathbf{X_c} \in \mathbb{R}^{|\mathcal{R}| \times 2d} \tag{1}$$

$$\mathbf{X_r} = DA(\mathbf{X_r}, \mathbf{X_c}), \mathbf{X_r} \in \mathbb{R}^{|\mathcal{R}| \times 2d} \tag{2}$$

$$\mathbf{X_e} = PA(\mathbf{X_e}, \mathbf{X_r}) \tag{3}$$

$$\mathbf{X_e} = \mathbf{X_e^0} + \beta_i \cdot \mathbf{X_e} \tag{4}$$

The first step, in (1), obtains a *relation context* (RC) $\mathbf{X_c}$ from the entity representations. For relation $r \in \mathcal{R}$, we extract its relation context as a concatenation of the mean entity representations for the head and the tail entities. By denoting the set of head and tail entities for relation r with H_r and T_r, we can thus express its computation as $(\mathbf{X_c})_i = \left[{}^1\!/_{|H_i|} \sum_{j \in H_i} (\mathbf{X_e})_j \| {}^1\!/_{|T_i|} \sum_{j \in T_i} (\mathbf{X_e})_j \right]$ where $\|$ denotes the concatenation operation. An entity occurring multiple times as the head is weighted equally to an entity occurring only once.

The second step, in (2), is the *dual graph attention* (DA). The attention scores on the dual graph α_{ij}^D are computed by dot product attention with leaky ReLU activation: $\alpha_{ij}^D = J_{ij} \cdot LeakyReLU(\mathbf{W_L}(\mathbf{X_c})_i + \mathbf{W_R}(\mathbf{X_c})_j)$. Notice that

$\mathbf{W_L}(\mathbf{X_c})_i + \mathbf{W_R}(\mathbf{X_c})_j = (\mathbf{W_L} \| \mathbf{W_R})^T((\mathbf{X_c})_i \| (\mathbf{X_c})_j)$, where $\|$ denotes the concatenation operation. In the published code, we further found a weight sharing mechanism for $\mathbf{W_L}$ and $\mathbf{W_R}$ implemented, decomposing the projection weight matrices as $\mathbf{W_L} = \mathbf{W'_L}\mathbf{W_C}$ and $\mathbf{W_R} = \mathbf{W'_R}\mathbf{W_C}$ with $\mathbf{W'_L}, \mathbf{W'_R} \in \mathbb{R}^{1\times h}, \mathbf{W_C} \in \mathbb{R}^{h\times 2d}$ being trainable parameters, and $\mathbf{W_C}$ shared between both projections. J_{ij} denotes a fixed triple-based relation similarity score computed as the sum of the Jaccard similarities of the head and tail entity set for relation r_i and r_j: $J_{ij} := {}^{|H_i \cap H_j|}/_{|H_i \cup H_j|} + {}^{|T_i \cap T_j|}/_{|T_i \cup T_j|}$. The softmax is then computed only over those relations, where $J_{ij} > 0$, i.e., pairs sharing at least one head or tail entity. In the implementation, this is implemented as dense attention with masking, i.e. setting $\alpha^D_{ij} = -\infty$ (or a very small value) for $J_{ij} = 0$. While this increases the required memory consumption to $\mathcal{O}(|\mathcal{R}|^2)$, the number of relations is usually small compared to the number of entities, cf. Table 1, and thus this poses no serious computational problem. With $\tilde{\alpha}^D_{ij}$ denoting the softmax output, the new relation representation finally is $(\mathbf{X_r})_i = ReLU\left(\sum_j \tilde{\alpha}^D_{ij}(\mathbf{X_r})_j\right)$.

In the third step, in (3), the entity representations are updated. To this end, a relation-specific scalar score is computed as $\alpha^r_i = LeakyReLU(\mathbf{W}\mathbf{X_r} + b)$ with trainable parameters W and b. Based upon the relation-specific scores, an attention score between two entities e_i, e_j with at least one relation between them is given as $\alpha^P_{ij} = \sum_{r\in\mathcal{T}_{ij}} \alpha^r_i$. These scores are normalized with a sparse softmax over all $\{j \mid \exists r \in \mathcal{R} : (e_i, r, e_j) \in \mathcal{T}\}$: $\tilde{\alpha}^P_{ij} = \text{softmax}_{j'}(\alpha^P_{ij'})_j$. The final output of the primal attention is $(\mathbf{X_e})_j = ReLU(\sum_i \tilde{\alpha}_{ij}(\mathbf{X_e})_j)$.

The fourth step, in (4), applies a skip connection from the initial representations to the current entity representation. The weight β_i is pre-defined ($\beta_1 = 0.1$, $\beta_2 = 0.3$) and not trained.

Standard Message Passing. The second part of the RDGCN consists of a sequence of GCN layers with highway layers. Each layer computes

$$\mathbf{X'_e} = ReLU(\mathbf{A}\mathbf{X_e}\mathbf{W}) \tag{5}$$

$$\beta = \sigma(\mathbf{W_g}\mathbf{X_e} + b_g) \tag{6}$$

$$\mathbf{X_e} = \beta \cdot \mathbf{X'_e} + (1 - \beta) \cdot \mathbf{X_e} \tag{7}$$

$\mathbf{A} \in \mathbb{R}^{|\mathcal{E}^L|\times|\mathcal{E}^L|}$ denotes the adjacency matrix of the primal graph. It is constructed by first creating an undirected, unweighted adjacency matrix where there is a connection between $e_i, e_j \in \mathcal{E}^L$ if there exists at least one triple $(e_i, r, e_j) \in \mathcal{T}^L$ for some relation $r \in \mathcal{R}^L$. Next, self-loops (e, e) are added for every entity $e \in \mathcal{E}^L$. Finally, the matrix is normalized by setting $\mathbf{A} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ with $\mathbf{D}$ denoting the diagonal matrix of node degrees. When investigating the published code, we further found out that the weight matrix $\mathbf{W}$ is constrained to be a diagonal matrix and initialized as an identity matrix.

Training. Let $\mathbf{x}^L_i$ denote the final entity representation for $e^L_i \in \mathcal{E}^L$ and anologously $\mathbf{x}^R_j$ for $e^R_j \in \mathcal{E}^R$. RDGCN is trained with a margin-based loss formulation.

It adopts a hard negative mining strategy, i.e., the set of negative examples for one pair is the top k most similar entities of one of the entities according to the similarity measure used for scoring. The negative l_1 distance is used as similarity, the margin is 1, $k = 10$, and the negative examples are updated every 10 epochs.

3.2 Deep Graph Matching Consensus (DGMC).

DGMC [7] also comprises two parts, which we name *enrichment* and *correspondence refinement*. The enrichment part is a sequence of GNN layers enriching the entity representations with information from their neighborhood. Each layer computes $\phi(\mathbf{X}) = ReLU(norm(\mathbf{A})\mathbf{X}\mathbf{W_1} + norm(\mathbf{A}^T)\mathbf{X}\mathbf{W_2} + \mathbf{X}\mathbf{W_3})$, where $\mathbf{A} \in \mathbb{R}^{|\mathcal{E}^L|\times|\mathcal{E}^L|}$ denotes the symmetrically normalized adjacency matrix (as for second part of RDGCN), *norm* the row-wise normalization operation, $\mathbf{X} \in \mathbb{R}^{\mathcal{E}^L \times d_{in}}$ the layer's input, and $\mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3} \in \mathbb{R}^{d_{in}\times d_{out}}$ trainable parameters of the layer. An optional batch normalization and dropout follow this layer. For the enrichment phase's final output, all individual layers' outputs are concatenated before a learned final linear projection layer reduces the dimension to d_{out}.

The second phase, the *correspondence refinement*, first calculates the $k = 10$ most likely matches in the other graph for each entity as a sparse correspondence matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{E}^L|\times|\mathcal{E}^R|}$, normalized using softmax. Next, it generates random vectors for each entity $\mathbf{R} \in \mathbb{R}^{|\mathcal{E}^L|\times d_{rnd}}$ and sends these vectors to the probable matches via the softmax normalized sparse correspondence matrix, $\mathbf{S}^T\mathbf{R} \in \mathbb{R}^{|\mathcal{E}^R|\times d_{rnd}}$. A GNN layer ψ as in phase one distributes these vectors in the neighborhood of the nodes: $\mathbf{Y}^R = \psi(\mathbf{S}^T\mathbf{R})$. A two-layer MLP predicts an update for the correspondence matrix, given the difference between the representations $\mathbf{Y}^L$ and $\mathbf{Y}^R$. This procedure is repeated for a fixed number of refinement steps $L = 10$.

4 Experiments

Experimental Setup. For the general evaluation setting and description of metrics, we refer to [3]. Here, we primarily use Hits@1 (H@1), which measures the correct entity's relative frequency of being ranked in the first position. When investigating the published code of both, RDGCN [17][7] and DGMC [7][8], we did not find any code for tuning the parameters, nor a train-validation split. Also, the papers themselves do not mention a train-validation split. Thus, it is unclear how they choose the hyperparameters without a test-leakage by directly optimizing the test set's performance. We thus decided to create a shared test-train-validation split used by all our experiments to enable a fair comparison. Since DGMC already uses PyTorch, we could use their published code and extend it with HPO code. RDGCN was re-implemented in PyTorch in our codebase. We

[7] https://github.com/StephanieWyt/RDGCN.
[8] https://github.com/rusty1s/deep-graph-matching-consensus/.

Table 3. Investigated hyperparameters for all methods. * denotes that these parameters share the same value range but were tuned independently.

Common	
Parameter	Choices
Optimizer	Adam
Similarity	{cos, dot, l1 (bound inverse), l1 (negative), l2 (bound inverse), l2 (negative)}
RDGCN	
Parameter	Choices
(entity embedding) normalization	{always-l2, initial-l2, never}
(number of) GCN layers	{0, 1, 2, 3}
(number of) interaction layers	{0, 1, 2, 3}
Interaction weights	$\{0.1, 0.2, \ldots, 0.6\}$
Trainable embeddings	{False, True}
Hard negatives	{no, yes}
Learning rate	$[10^{-4}, 10^{-1}]$
DGMC	
Parameter	Choices
ψ_1 / ψ_2 dimension*	$[32, 64, \ldots, 1024]$
ψ_1 / ψ_2 (number of) GCN layers*	{1, 2, 3, 4, 5}
ψ_1 / ψ_2 batch normalization*	{False, True}
ψ_1 / ψ_2 layer concatenation*	{False, True}
ψ_1 dropout	$[0.00, 0.05, \ldots, 1.0]$
ψ_2 dropout	0.0
Trainable embeddings	False
(entity embedding) normalization	{never, always-l1, always-l2}
Learning rate	$[10^{-3}, 10^{-1}]$
GCN-Align*	
Parameter	Choices
Model output dimension	$[32, 64, \ldots, (embeddingdimension)]$
(number of) GCN layers	{1, 2, 3}
Batch normalization	{False, True}
Layer concatenation	{False, True}
Final linear projection	{False, True}
Dropout	$\{0.0, 0.1, \ldots, 0.5\}$
Trainable embeddings	{False, True}
(entity embedding) normalization	{never, always-l1, always-l2}
(weight) sharing horizontal	{False, True}
Learning rate	$[10^{-3}, 10^{-1}]$

use the official train-test split for all datasets, which reserves 70% of the alignments for testing. We split the remaining part into 80% train alignments and 20% validation alignments.

We continued by tuning numerous model parameters (cf. Table 3) of all models on each of the datasets in Table 1 and each of the available initializations described in Sect. 2.2 to obtain sufficiently well-tuned configurations. We used random search due to its higher sample efficiency than grid search [1]. We additionally evaluate a baseline, which uses the GNN variant from DGMC without the neighborhood consensus refinement, coined *GCN-Align** due to its close correspondence to [16], and also evaluate the zero-shot performance of the initial node features.

For each tested configuration, we perform early stopping on validation H@1, i.e., select the epoch according to the best validation H@1. Across all tested configurations for a model-dataset-initialization combination, we then choose the best configuration according to validation H@1 and report the test performance in Table 4. We do not report performance for training on train+validation with the final configuration due to space restrictions. We decided to report performance when trained only on the train set to ensure that other works have performance numbers for comparison when tuning their own models.

4.1 Results

Table 4 presents the overall results. We can observe several points.

Table 4. Results in terms of H@1 for all investigated combinations of datasets, models, and initializations. Each cell represents the *test* performance of the best configuration of hyperparameters chosen according to *validation* performance.

DBP15k (JAPE)						
init	Wu [18]			Xu [20]		
subset	fr-en	ja-en	zh-en	fr-en	ja-en	zh-en
Zero Shot	79.47	63.48	56.07	83.70	65.64	59.40
GCN-Align*	81.81	67.45	57.94	86.74	67.65	60.32
RDGCN	86.91	**72.90**	66.44	86.82	74.35	**69.54**
DGMC	**89.35**	72.17	**69.98**	**90.12**	**76.60**	68.76

OpenEA				
init			Sun [15]	
subset	d-w	d-y	en-de	en-fr
Zero Shot	46.53	81.90	75.99	79.90
GCN-Align*	45.76	84.65	85.34	89.41
RDGCN	**64.28**	**98.41**	80.03	**91.52**
DGMC	51.29	88.60	**88.10**	89.40

WK3l15k			
init		BERT	
subset	en-de		en-fr
Zero Shot	85.55		77.27
GCN-Align*	85.92		**78.22**
RDGCN	**86.76**		78.05
DGMC	84.08		73.92

Zero-Shot Performance. Generally, there is an impressive Zero-Shot performance, ranging from 39.15% for `OpenEA d-w` to 83.85% `WK3l15k en-de`. Thus, even in the weakest setting, approximately 40% of the entities can be aligned solely from their label, without any sophisticated method. Consequently, this highlights that comparison against methods not using this information is unfair. For `DBP15k`, we can compare the initialization from Wu et al. [17], used, e.g., by RDGCN to the performance of the initialization by Xu et al. [7], used, e.g., by DGMC. We observe that Wu's initialization is 7–9% points stronger than Xu's initialization. For `OpenEA d-w` we obtain 39.15% zero-shot performance, despite the original labels of the `w` side being meaningless identifiers. This is only due to using attribute triples with a pre-defined set of "name" attributes, cf. Table 2.

Model Performance. When comparing the performance of both analyzed models, we can observe that they have a clear advantage over both baselines in two of three datasets. However, we cannot identify a single winner among them. Although the performance of DGMC dropped compared to the results reported originally[9], it still leads by about 3–4 points on almost all DBP15k subsets. Therefore, it confirms our observation that a smaller test set automatically leads to better results. Furthermore, we can see that different initialization with entity name also affects model performance, which especially applies to the ja-en subset for DGMC or fr-en for GCN-Align*. RDGCN has a clear advantage on the OpenEA subsets extracted from DBPedia with a margin of between 10 and 13 points on both subsets. Note that we significantly improved results of RDGCN on the OpenEA dataset through our extensive hyperparameter search compared to the original evaluation [15]. Interestingly, as can be seen in the next section, the main reason is *not* the exploiting of information about different relations. The WK3L15k dataset constitutes an interesting exception. The performance of the DGMC method, which is supposed to be robust against noise due to its correspondence refinement, is not better than the zero-shot results. While DGMC and GCN-Align* can improve the results, the improvement by 1–2 points does not look very convincing. From these results, we conclude that there exists no silver bullet for the task of EA, and the method itself is still a hyperparameter. At the same time, we see that the most realistic dataset poses a real challenge for SotA methods.

4.2 Ablation: RDGCN

We additionally present the results of an ablation study for some model parameters of RDGCN on the OpenEA datasets in Table 5. For each presented parameter and each possible value, we fix this one parameter and select the best configuration among all configurations with the chosen parameter setting according to validation H@1. The cell then shows the validation and test performance of this configuration. We highlight the best setting on the respective graph pair in

[9] As a general rule, the results improve by 1–2 points when trained on train+validation, and it is not going to change the picture.

Table 5. Ablation results for RDGCN on OpenEA datasets. The setting used by [17] is underlined. The first number is validation H@1, the second number test H@1. Bold highlights the best configuration. Please notice that due to the specialties of EA evaluation, the test and validation performance are *not* directly comparable [3].

Parameter	Value	Subset			
		d-w	d-y	en-de	en-fr
Normalization	Always	**84.06/64.28**	99.44/97.48	97.72/**93.56**	**96.89/91.52**
	Initial	82.67/62.58	**99.78**/98.41	97.67/93.02	95.56/89.50
	Never	78.39/61.77	99.72/**98.53**	**98.11**/80.03	95.44/90.14
GCN layers	0	57.33/50.79	92.33/83.83	**98.11**/80.03	92.22/86.94
	1	73.33/56.66	99.33/98.15	96.00/91.63	94.50/90.49
	2	78.39/61.77	99.56/98.16	97.72/**93.56**	**96.89/91.52**
	3	**84.06/64.28**	**99.78/98.41**	97.00/92.18	95.44/90.14
Interaction layers	0	78.11/60.53	99.72/**98.53**	97.72/**93.56**	95.33/89.08
	1	78.39/61.77	**99.78**/98.41	97.67/92.59	95.44/90.14
	2	82.67/62.58	99.56/98.16	**98.11**/80.03	**96.89/91.52**
	3	**84.06/64.28**	99.50/97.85	97.67/93.02	95.56/89.50
Trainable embeddings	No	**84.06/64.28**	99.72/**98.53**	97.72/**93.56**	**96.89/91.52**
	Yes	82.67/62.58	**99.78**/98.41	**98.11**/80.03	95.56/89.50
Similarity	Cos	82.67/62.58	99.56/98.16	**98.11**/80.03	95.56/89.50
	Dot	63.28/40.80	91.50/79.81	85.17/78.54	89.94/78.17
	l1 (inv.)	77.89/60.78	99.50/97.85	93.78/88.96	94.06/88.69
	l1 (neg.)	**84.06/64.28**	99.72/**98.53**	97.72/**93.56**	**96.89/91.52**
	l2 (inv.)	75.28/60.20	96.72/92.06	95.06/90.13	94.44/89.60
	l2 (neg.)	72.50/51.04	**99.78**/98.41	94.61/89.40	94.28/87.79
Hard negatives	No	82.67/62.58	**99.78/98.41**	**98.11**/80.03	**96.89/91.52**
	Yes	**84.06/64.28**	99.67/98.30	97.72/**93.56**	95.33/90.62

bold font. Note that the test performance numbers also coincide with the performance reported in Table 4 for OpenEA. We make the following interesting observations: for all but one graph pair, *always normalizing* the entity representations before passing them into the layers is beneficial. For `d-y`, where this is not the case, the difference in performance is small. For the *number of GCN layers*, we observe an increase in performance from 0 to 2 layers, and on some datasets (`d-w`, `d-y`) even beyond. Thus, aggregating the entities' neighborhood seems beneficial, highlighting the importance of the graph structure. For the *number of interaction layers*, which perform *relation-aware* message passing, we observe that for two of the four subsets (`d-y`, `en-de`) the best configuration does not use any interaction layer. However, the difference is small. None of the best configurations uses *trainable node embeddings*. The *negative* l_1 *similarity* is superior on all datasets, with most of the others being close to it. Using the dot product seems to be sub-optimal, maybe due to its unbound value range. Regarding *hard negative mining*, there is no clear tendency, but considering the hard negatives' expensive calculation (all-to-all kNN), its use might not be worthwhile.

Another observation is that sometimes there is a huge gap between the test performance for the best configuration according to validation performance and the best configuration according to test performance. For instance, if we had selected the hyperparameters according to test performance for `en-de`, we had obtained 93.53 H@1, while choosing them according to validation performance results in only 80.03 H@1 – a difference of 13.5% points. This difference emphasizes the need for a fair hyperparameter selection.

5 Conclusion

In this paper, we investigated state-of-the-art in Entity Alignment. Since we identified shortcomings in the commonly employed evaluation procedure, including the lack of validation sets for hyperparameter tuning and different initializations, we provided a fair and sound evaluation over a wide range of configurations. We additionally gave insight into the importance of individual components. Our results provide a strong, fair, and reproducible baseline for future works to compare against and offer deep insights into the inner workings of a GNN-based model.

We plan to investigate the identified weakness against noisy labelings in future work and increase the robustness. Moreover, we aim to improve the usage of relation type information in the message passing phase of models like RDGCN, which only use them in an initial entity representation refinement stage. For some datasets such as OpenEA d-y and en-de, optimal configurations did not consider the relational information. However, intuitively, this information should help to improve the structural description of entities. Potential improvements include establishing a relation matching between the two graphs or modifying the mechanism used to integrate relational information.

Acknowledgment. This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
2. Berrendorf, M., Faerman, E., Melnychuk, V., Tresp, V., Seidl, T.: Knowledge graph entity alignment with graph convolutional networks: lessons learned. In: Jose, J.M., et al. (eds.) ECIR 2020. LNCS, vol. 12036, pp. 3–11. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45442-5_1
3. Berrendorf, M., Faerman, E., Vermue, L., Tresp., V.: Interpretable and fair comparison of link prediction or entity alignment methods with adjusted mean rank. CoRR, abs/2002.06914 (2020)
4. Cao, Y., Liu, Z., Li, C., Liu, Z., Li, J., Chua, T.-S.: Multi-channel graph neural network for entity alignment. In: ACL (1), pp. 1452–1461. Association for Computational Linguistics (2019)

5. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (1), pp. 4171–4186. Association for Computational Linguistics (2019)
6. Dietz, L., Xiong, C., Dalton, J., Meij, E.: Special issue on knowledge graphs and semantics in text analysis and retrieval. Inf. Retr. J. **22**(3–4), 229–231 (2019)
7. Fey, M., Lenssen, J.E., Morris, C., Masci, J., Kriege, N.M.: Deep graph matching consensus. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. OpenReview.net (2020)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016, pp. 770–778. IEEE Computer Society (2016)
9. Li, C., Cao, Y., Hou, L., Shi, J., Li, J., Chua, T.-S.: Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model. In: EMNLP/IJCNLP (1), pp. 2723–2732. Association for Computational Linguistics (2019)
10. Mao, X., Wang, W., Xu, H., Lan, M., Wu, Y.: MRAEA: an efficient and robust entity alignment approach for cross-lingual knowledge graph. In: WSDM, pp. 420–428. ACM (2020)
11. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, 25–29 October 2014, A meeting of SIGDAT, a Special Interest Group of the ACL, pp. 1532–1543. ACL (2014)
12. Shi, X., Xiao, Y.: Modeling multi-mapping relations for precise cross-lingual entity alignment. In: EMNLP/IJCNLP (1), pp. 813–822. Association for Computational Linguistics (2019)
13. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. CoRR, abs/1505.00387 (2015)
14. Sun, Z., et al.: Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, 7–12 February 2020, pp. 222–229. AAAI Press (2020)
15. Sun, Z., et al.: A benchmarking study of embedding-based entity alignment for knowledge graphs. Proc. VLDB Endow. **13**(11), 2326–2340 (2020)
16. Wang, Z., Lv, Q., Lan, X., Zhang, Y.: Cross-lingual knowledge graph alignment via graph convolutional networks. In: EMNLP, pp. 349–357. Association for Computational Linguistics (2018)
17. Wu, Y., Liu, X., Feng, Y., Wang, Z., Yan, R., Zhao., D.: Relation-aware entity alignment for heterogeneous knowledge graphs. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, 10–16 August 2019, pp. 5278–5284. ijcai.org (2019)
18. Wu, Y., Liu, X., Feng, Y., Wang, Z., Zhao., D.: Jointly learning entity and relation representations for entity alignment. In: EMNLP/IJCNLP (1), pp. 240–249. Association for Computational Linguistics (2019)
19. Xu, H., et al.: High-order relation construction and mining for graph matching. CoRR, abs/2010.04348 (2020)

20. Xu, K., et al.: Cross-lingual knowledge graph alignment via graph matching neural network. In: Korhonen, A., Traum, D.R., Màrquez, L. (eds.) Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pp. 3156–3161. Association for Computational Linguistics (2019)
21. Yang, H.-W., Zou, Y., Shi, P., Lu, W., Lin, J., Sun, X.: Aligning cross-lingual entities with multi-aspect information. In: EMNLP/IJCNLP (1), pp. 4430–4440. Association for Computational Linguistics (2019)
22. Ye, R., Li, X., Fang, Y., Zang, H., Wang, M.: A vectorized relational graph convolutional network for multi-relational network alignment. In: IJCAI, pp. 4135–4141. ijcai.org (2019)
23. Zhang, Q., Sun, Z., Hu, W., Chen, M., Guo, L., Qu, Y.: Multi-view knowledge graph embedding for entity alignment. In: IJCAI, pp. 5429–5435. ijcai.org (2019)
24. Zhao, X., Zeng, W., Tang, J., Wang, W., Suchanek, F.: An experimental study of state-of-the-art entity alignment approaches. IEEE Trans. Knowl. Data Eng. (01), 1 (2020)
25. Zhu, Q., Zhou, X., Wu, J., Tan, J., Guo, L.: Neighborhood-aware attentional representation for multilingual knowledge graphs. In: IJCAI, pp. 1943–1949. ijcai.org (2019)

9 Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods

This chapter includes the following publications:

> **Max Berrendorf**, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. „Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods.“ In: *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE, Dec. 2020. DOI: 10.1109/wiiat50758.2020.00053

In addition, an extended abstract has been published as

> **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: *The 5th International Workshop on Deep Learning for Graphs (DL4G@WWW2020)* (2020)

and an extended technical report is available at

> **Max Berrendorf**, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. „On the Ambiguity of Rank-Based Evaluation of Entity Alignment or Link Prediction Methods.“ In: *CoRR* abs/2002.06914v2 (2020). arXiv: 2002.06914

The code is available at

> **Max Berrendorf**. *mberr/rank-based-evaluation: Zenodo*. Version 1.0.0. Oct. 2020. DOI: 10.5281/zenodo.4588898

Declaration of Authorship The research idea was proposed by Max Berrendorf and discussed with Laurent Vermue. Max Berrendorf and Evgeniy Faerman developed and conceptualized it further, and discussed with all co-authors. Max Berrendorf did the implementation, designed and conducted the experiments and evaluated their results. Max Berrendorf and Evgeniy Faerman wrote the manuscript.

2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT) | 978-1-6654-1924-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/WIIAT50758.2020.00053

Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods

Max Berrendorf*, Evgeniy Faerman*, Laurent Vermue† and Volker Tresp*‡
*Ludwig-Maximilians-Universität München, Munich, Germany
{berrendorf,faerman}@dbs.ifi.lmu.de
†Technical University of Denmark, Kongens Lyngby, Denmark
lauve@dtu.dk
‡Siemens AG, Munich, Germany
volker.tresp@siemens.com

***Abstract*—In this work, we take a closer look at the evaluation of two families of methods for enriching information from knowledge graphs: Link Prediction and Entity Alignment. In the current experimental setting, multiple different scores are employed to assess different aspects of model performance. We analyze the informativeness of these evaluation measures and identify several shortcomings. In particular, we demonstrate that all existing scores can hardly be used to compare results across different datasets. Therefore, we propose adjustments to the evaluation and demonstrate empirically how this supports a fair, comparable, and interpretable assessment of model performance.**

I. Introduction

Information retrieval systems often require information organized in an easily accessible and interpretable structure. Frequently, Knowledge Graphs (KGs) are used as an information source [9]. Consequently, the successful application of new information retrieval algorithms often depends on the completeness and quality of the information in KGs. *Link Prediction* (LP) [18] and *Entity Alignment* (EA) [4] are two disciplines with the goal to enrich information in KGs. LP makes use of existing information in a single KG by materializing latent links. The goal of EA is to align entities in different KGs, which facilitates the transfer of information between both or a fusion of multiple KGs to a single knowledgebase. Both disciplines work by assigning scores to potential candidates: LP methods compute scores for the facts in question at inference time and EA methods assign scores to candidate alignment pairs. Simple thresholding, or also more advanced assignment methods [15] for EA, can make use of these scores to predict new links or alignments.

During the evaluation, both, LP and EA, evaluate how the "true" entity is *ranked* relative to other candidate entities. Given a *rank* for each test instance, various metrics exist to obtain a single number quantifying the overall performance of an approach. In this paper, we analyze the whole evaluation procedure and make the following contributions:

1) We describe the intuition behind current aggregation scores and argue that they do not always provide a complete picture of the model performance. We show that this is an actual problem in the current evaluation setting, which sometimes may lead to wrong conclusions.

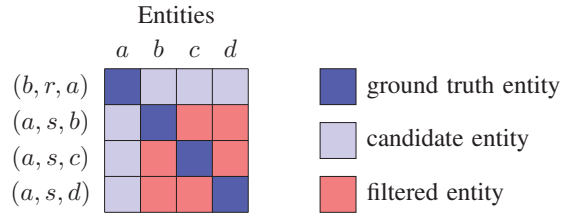


Fig. 1: Visualization of candidate sets for the filtered evaluation setting for link prediction (right side / tail prediction) on a toy example with triples $\{(b, r, a), (a, s, b), (a, s, c), (a, s, d)\}$. Depending on the presence of other triples with shared head-relation pairs, the number of considered candidate entities varies, and consequently the maximum possible rank. In this example, there are three triples starting with (a, s). When, e.g., triple (a, s, b) is evaluated c and d are ignored. Since only two entities remain, the rank cannot be larger than two.

2) We propose a new (adapted) evaluation score overcoming the problems of existing metrics.
3) We empirically demonstrate its usefulness for comparing Link Prediction results across datasets.

The remainder of the paper is structured as follows: In Section II, we discuss the rank definition and aggregation metrics summarizing individual ranks. In Section III, we point out the problems of current evaluation and introduce an adapted aggregation metric, which circumvents the shortcomings of existing aggregations. Afterwards, in Section IV, we discuss related work. Finally, in Section V, we demonstrate empirically the effects of our adaptations and conclude in Section VI.

II. Evaluation Framework

A. Rank for Link Prediction and Entity Alignment

a) Link Prediction: Let a single knowledge graph be represented as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of relations, and $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of triples. For the task of LP a set of given triples is usually divided in $\mathcal{T}_{train} \subseteq \mathcal{T}$ and $\mathcal{T}_{test} = \mathcal{T} \setminus \mathcal{T}_{train}$, where $\mathcal{T}_{test}$ is used to assess the model performance. A common evaluation protocol is to use every triple $(h, r, t) \in \mathcal{T}_{test}$, and perform left-side

978-1-6654-1924-6/20/$31.00 ©2020 IEEE
DOI 10.1109/WIIAT50758.2020.00053

and right-side prediction. For the right-side prediction, the score for every triple $\{(h, r, e) \mid e \in \mathcal{E}\}$ is computed and the entities e are sorted in decreasing order by the predicted scores. The *rank* of the "true" entity t is computed as the index in the resulting sorted list. The left-side prediction follows analogously. The final rank of the triple is computed as an average over both ranks, left-side and right-side. To account for the possibility of multiple existing links for a given head-relation / relation-tail pair, the filtered evaluation setting was introduced [5]: When scoring tail entities for a triple (h, r, t), all other entities $t \neq t' \in \mathcal{E}$ with triples $(h, r, t') \in (\mathcal{T}_{train} \cup \mathcal{T}_{test})$ are ignored, cf. Figure 1. Therefore, the performance does not decrease when other entities are scored higher than the currently considered one, as long as they are also true. The filtered evaluation protocol is the quasi-standard for link prediction on knowledge graphs, and unfiltered scores are rarely reported.

b) Entity Alignment: For this task, there are two knowledge graphs $\mathcal{G}_L = (\mathcal{E}_L, \mathcal{R}_L, \mathcal{T}_L)$ and $\mathcal{G}_R = (\mathcal{E}_R, \mathcal{R}_R, \mathcal{T}_R)$, and a set of aligned entities $\mathcal{A} \subseteq \mathcal{E}_L \times \mathcal{E}_R$. Analogous to the previous evaluation setting, the set of alignments is divided into $\mathcal{A}_{train} \subseteq \mathcal{A}$ and $\mathcal{A}_{test} = \mathcal{A} \setminus \mathcal{A}_{train}$. The common evaluation scheme [6], [7], [11], [21], [26]–[28], [30], [33], [35], [37], [38] now computes scores for every candidate pair $\{(a_L, e_R) \mid e_R \in \mathcal{E}_R, \exists a'_L \in \mathcal{E}_L : (a'_L, e_R) \in \mathcal{A}_{test}\}$, and determines the rank of the "true" score of (a_L, a_R). The right-side prediction is defined correspondingly. Notice, that only those entities are considered for which there exists an aligned entity from the other graph in the *test* part of the alignment.

B. Overall metrics

Given the set of individual rank scores $\mathcal{I}$, the following scores are commonly used as aggregation.

1) Hits @ k: The *Hits @ k* (H@k) score describes the fraction of hits, or fraction of instances, for which the "true" entity appears under the first k entities in the sorted list:

$$H@k := \frac{|\{r \in \mathcal{I} \mid r \leq k\}|}{|\mathcal{I}|}. \tag{1}$$

In the context of information retrieval, this metric is also known as Precision@k. One of the advantages of this metric is that it is easily interpretable. Since for many applications only the first outputs are taken into account, it can help to directly assess the method's applicability to the use-case. However, this metric does not distinguish the cases, where the rank is larger than k. Thus, the ranks $k + 1$ and $k + d$, where $d \gg 1$, have the same effect on the final score. Therefore, it is less suitable for the comparison of different models.

2) Mean Rank: The *mean rank* (MR) computes the mean over all individual ranks:

$$MR := \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} r. \tag{2}$$

The advantage of the MR score is that it is sensitive to any model performance changes. If the rank on the same evaluation set becomes better on average, the improvement is always reflected by the MR score. While the MR is still interpretable, it is necessary to keep the size of the candidate set in mind to assess the model performance and interpret its value: A MR of 10 might indicate strong performance for a candidate set size of 1,000,000, but for a candidate set of only 20 candidates it equal to the expected performance of a model with random scorings.

3) MRR: The *mean reciprocal rank* is still often reported along with other scores. It is defined as

$$MRR := \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} \frac{1}{r}. \tag{3}$$

While the MRR is less sensitive to outliers and has the property to be bounded in the range $(0, 1]$, it was shown that this metric has serious flaws and therefore should not be relied upon [10]. However, especially in LP codebases, the MRR is often used for early stopping. Presumably, the main reason for that is the behavior of the reciprocal function: While the Hits@k score ignores change among high rank values completely, MR values changes uniformly among the full value range. The MRR score, in contrast, is more affected by changes of low rank values than high ones, but it does not completely disregard them. Therefore, it can be considered as soft a version of Hits@k.

III. Our Evaluation Approach

A. Adjusted Mean Rank

While the *H@k* score enables assessments of the model's suitability for a use-case, the *MR* allows a more fine-grained comparison between different models. Both metrics are necessary to get the entire picture of the model performance, e.g. the evaluation in [31] demonstrates, that an excellent *H@k* score does not necessarily coincide with a good *MR*. However, since the *MR* score denotes the absolute position, it is not easily interpretable. Therefore, the comparison between experiments with different sizes of candidate sets is not easily possible with implications for the evaluation of both tasks.

a) Link Prediction: The results on datasets with a different number of entities are not directly comparable. However, comparability of performance on different datasets is important, for example, to assess the task complexity, choose benchmarks, or investigate model generalization. For instance, surprisingly good test scores can be an indication for test leakage, see e.g. [29]. Intuitively, the number of candidates is an important factor directly affecting the task complexity, while it is not the only factor.

b) Entity Alignment: While the comparison of the performance on different datasets is also difficult for EA, there is the additional problem that only those entities are considered as candidates, which occur in at least one *test* alignment. Therefore, the number of candidates depends on the size of the evaluation alignment set. Thus, results on the same dataset are not comparable for different train/test splits or between train and test sets. This can lead to various misinterpretations of results. For instance, in [17], [33], the authors show an experiment where they increase the training size step-wise

and evaluate the model on the rest of the data. Based on the score improvement, they conclude that the model benefits from additional training data. While this claim can still be true, we argue that another evaluation is necessary to support it. The necessary condition for such an evaluation is either independence on candidate set size or the same candidate set for all experiments. One possible solution would be to use all entities in the KG as candidates analogous to LP. However, this still would leave us with the unresolved problem of performance comparison across datasets. Therefore, we propose an adjustment to the *MR* score that assesses the model performance independently of the candidate set size.

c) Adjusted Mean Rank Index: Since we are interested in evaluating model performance, we start with the mean rank as our starting point. To compute a rank in LP and EA evaluation, we are given a list of scores $\mathcal{S} = [\beta_1, \ldots, \beta_C]$ for each test instance with $|\mathcal{C}| = C$, where $\mathcal{C}$ is a set of candidates. We denote the score of the "true" entity as α, and its position in the decreasingly sorted list as $rank(S, \alpha)$. If $\alpha, \beta_1, \ldots, \beta_C$ are i.i.d and drawn at random, and therefore the element can appear at any position with the same probability, the expected rank is also the middle of the sorted array:

$$\mathbb{E}[rank(\mathcal{S}, \alpha)] = \frac{1}{n}\sum_{i=1}^{|\mathcal{S}|} i = \frac{1}{2}(|\mathcal{S}| + 1) \tag{4}$$

Inspired by the Adjusted Rand Index (ARI) [22], we aim to adjust it for chance. Therefore, we compute the *expected mean rank* following the assumption that the individual ranks are independent:

$$\begin{aligned}\mathbb{E}\left[MR\right] &\overset{(2)}{=} \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} rank(\mathcal{S}_i, \alpha)\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[rank(\mathcal{S}_i, \alpha)\right] \\ &\overset{(4)}{=} \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathcal{S}_i| + 1}{2} = \frac{1}{2n}\sum_{i=1}^{n}(|\mathcal{S}_i| + 1)\end{aligned}$$

Now, we define the *adjusted mean rank* as the MR divided by its expected value:

$$AMR = \frac{MR}{\mathbb{E}\left[MR\right]} = \frac{2\sum_{i=1}^{n} r_i}{\sum_{i=1}^{n}(|\mathcal{S}_i| + 1)}$$

Finally, to obtain a measure where 1 corresponds to optimal performance, we transform the adjusted mean rank to *adjusted mean rank index* (AMRI) as follows:

$$AMRI = 1 - \frac{MR - 1}{\mathbb{E}\left[MR - 1\right]} = \frac{2\sum_{i=1}^{n}(r_i - 1)}{\sum_{i=1}^{n}(|\mathcal{S}_i|)} \tag{5}$$

Since $r_i - 1 \leq |\mathcal{S}| - 1$ the AMR has a bounded value range of $[-1, 1]$. A value of 1 corresponds to optimal performance where each individual rank is 1. A value of 0 indicates model performance similar to a model assigning random scores, or equal score to every candidate. The value is negative if the model performs worse than the constant-score model.

IV. Related Work

A special property of the *ranking* evaluation is that the candidate scores for each test instance are only required to be comparable within a single candidate set. The scores of candidates for another test instance may have a different value range, but since they are not compared with the candidates of other test instances, this does not affect the results. Therefore, ranking evaluation is appropriate for a setting where a human can evaluate model proposals. If, on the other hand, the decision has to be made automatically, e.g. using a fixed threshold, the *classification* setting is more appropriate. In the following, we review related approaches and demonstrate that classification and ranking evaluations are used interchangeably.

A. Triple Classification

In the LP task, we are given a pair of a head/tail entity $e \in \mathcal{E}$ and relation $r \in \mathcal{R}$, and rank a set of possible tail/head entities $e' \in \mathcal{E}$ according to the plausibility of the triple (e, r, e') / (e', r, e). In contrast, for *triple classification*, we aim at classifying whether a triple is true or false irrespective of the plausibility of other triples [12], [16], [25], [32], [34]. Consequently, a global threshold for the score of triples is required for the classification decision. If the threshold is chosen manually, classification metrics such as accuracy or F_1-measure can be used. Otherwise, the area under the precision-recall curve (PR-AUC), or receiver-operator curve (ROC-AUC) are used to summarize the performance over all possible decision thresholds. Link prediction and triple classification are sometimes evaluated alongside to demonstrate the effectiveness of novel knowledge graph embedding models across different tasks [14], [19], [25].

B. Ontology Matching

Ontology matching or *instance matching* is closely related to EA. Here we seek correspondences between instances of different ontologies based on different data properties of the instances. In contrast to EA, the vast majority of ontology matching approaches are unsupervised, i.e. there are no training alignments, but the instance features that are used for matching [3], [13], [20], [24]. The similarity is often fixed, e.g. to TF-IDF, and the methods optimize the matching process by pruning the candidate match space and selecting subsets of properties used for matching. Ontology matching approaches are evaluated in a classification setting with precision/recall/F_1-measure as evaluation score [1].

V. Experiments

In Table I, we compare the results of the *LP* evaluation in the filtered setting on two datasets, for which we used evaluation results from [31], and also computed results for MuRP [2] using their published code[1]. Given the AMRI score we can clearly conclude that all methods perform better than random. We also can compare the performance of the methods across

[1] https://github.com/ibalazevic/multirelational-poincare

TABLE I: Link prediction results

dataset	WN18RR		FB15k-237	
metric	MR	AMRI (%)	MR	AMRI (%)
DistMult [36]	7,000	65.8	500	93.0
ConvE [8]	4,412	78.4	241	96.6
TransE [5]	2,289	88.8	317	95.6
TransH [32]	2,126	89.6	219	97.0
R-GCN [23]	6,254	69.4	540	92.5
MuRP [2]	2,448	88.0	167	97.7

datasets and observe consistently worse performance on the *WN18RR* dataset. This difference is not only due to the larger number of entities in WN18RR ($\approx 45k$) compared to FB15k-237 ($\approx 15k$), but has to be caused by a different mechanism, e.g. the higher sparsity of WN18RR, or the richer relational patterns in FB15k-237. We leave the detailed analysis of dataset complexity for the future work.

VI. Conclusion

In this work, we address problems in the evaluation of LP and EA models for knowledge graphs. We thoroughly analyzed the current evaluation framework and identified several vulnerabilities. We demonstrated their causes and effects and showed how the problems can be mitigated by a simple adjustment for chance.

References

[1] Algergawy, A., Faria, D., Ferrara, A., Fundulaki, I., Harrow, I., Hertling, S., Jiménez-Ruiz, E., Karam, N., Khiat, A., Lambrix, P., Li, H., Montanelli, S., Paulheim, H., Pesquita, C., Saveta, T., Shvaiko, P., Splendiani, A., Thiéblin, É., Trojahn, C., Vatascinová, J., Zamazal, O., Zhou, L.: Results of the ontology alignment evaluation initiative 2019. In: OM@ISWC. CEUR Workshop Proceedings, vol. 2536, pp. 46–85. CEUR-WS.org (2019)

[2] Balazevic, I., Allen, C., Hospedales, T.M.: Multi-relational poincaré graph embeddings. In: NeurIPS. pp. 4465–4475 (2019)

[3] Belhadi, H., Akli-Astouati, K., Djenouri, Y., Lin, J.C.W.: Data mining-based approach for ontology matching problem. Applied Intelligence pp. 1–18 (2020)

[4] Berrendorf, M., Faerman, E., Melnychuk, V., Tresp, V., Seidl, T.: Knowledge graph entity alignment with graph convolutional networks: Lessons learned. arXiv preprint arXiv:1911.08342 (2019)

[5] Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS. pp. 2787–2795 (2013)

[6] Cao, Y., Liu, Z., Li, C., Liu, Z., Li, J., Chua, T.: Multi-channel graph neural network for entity alignment. In: ACL (1). pp. 1452–1461. ACL (2019)

[7] Chen, M., Tian, Y., Yang, M., Zaniolo, C.: Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In: IJCAI. pp. 1511–1517 (2017)

[8] Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: AAAI. pp. 1811–1818. AAAI Press (2018)

[9] Dietz, L., Xiong, C., Dalton, J., Meij, E.: Special issue on knowledge graphs and semantics in text analysis and retrieval. Inf. Retr. J. **22**(3-4), 229–231 (2019)

[10] Fuhr, N.: Some common mistakes in IR evaluation, and how they can be avoided. SIGIR Forum **51**(3), 32–41 (2017)

[11] Guo, L., Sun, Z., Hu, W.: Learning to exploit long-term relational dependencies in knowledge graphs. In: ICML. PMLR, vol. 97, pp. 2505–2514. PMLR (2019)

[12] Guo, S., Wang, Q., Wang, L., Wang, B., Guo, L.: Jointly embedding knowledge graphs and logical rules. In: EMNLP. pp. 192–202. The Association for Computational Linguistics (2016)

[13] Jiménez-Ruiz, E., Grau, B.C.: Logmap: Logic-based and scalable ontology matching. In: International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 7031, pp. 273–288. Springer (2011)

[14] Krompaß, D., Nickel, M., Jiang, X., Tresp, V.: Non-negative tensor factorization with rescal. In: Tensor Methods for Machine Learning, ECML workshop. pp. 1–10 (2013)

[15] Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)

[16] Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI. pp. 2181–2187. AAAI Press (2015)

[17] Mao, X., Wang, W., Xu, H., Lan, M., Wu, Y.: MRAEA: an efficient and robust entity alignment approach for cross-lingual knowledge graph. In: WSDM. pp. 420–428. ACM (2020)

[18] Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE **104**(1), 11–33 (2015)

[19] Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Icml. vol. 11, pp. 809–816 (2011)

[20] Niu, X., Rong, S., Wang, H., Yu, Y.: An effective rule miner for instance matching in a web of data. In: CIKM. pp. 1085–1094. ACM (2012)

[21] Pei, S., Yu, L., Hoehndorf, R., Zhang, X.: Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference. In: WWW. pp. 3130–3136. ACM (2019)

[22] Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association **66**(336), 846–850 (1971)

[23] Schlichtkrull, M.S., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: ESWC. pp. 593–607. Springer (2018)

[24] Shao, C., Hu, L., Li, J., Wang, Z., Chung, T.L., Xia, J.: Rimom-im: A novel iterative framework for instance matching. J. Comput. Sci. Technol. **31**(1), 185–197 (2016)

[25] Socher, R., Chen, D., Manning, C.D., Ng, A.Y.: Reasoning with neural tensor networks for knowledge base completion. In: NIPS. pp. 926–934 (2013)

[26] Sun, Z., Hu, W., Li, C.: Cross-lingual entity alignment via joint attribute-preserving embedding. In: ISWC (1). pp. 628–644. Springer (2017)

[27] Sun, Z., Hu, W., Zhang, Q., Qu, Y.: Bootstrapping entity alignment with knowledge graph embedding. In: IJCAI. pp. 4396–4402 (2018)

[28] Sun, Z., Wang, C., Hu, W., Chen, M., Dai, J., Zhang, W., Qu, Y.: Knowledge graph alignment network with gated multi-hop neighborhood aggregation. arXiv preprint arXiv:1911.08936 (2019)

[29] Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality. pp. 57–66 (2015)

[30] Trisedya, B.D., Qi, J., Zhang, R.: Entity alignment between knowledge graphs using attribute embeddings. In: AAAI. pp. 297–304. AAAI Press (2019)

[31] Wang, R., Li, B., Hu, S., Du, W., Zhang, M.: Knowledge graph embedding via graph attenuated attention networks. IEEE Access (2019)

[32] Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI. pp. 1112–1119. AAAI Press (2014)

[33] Wang, Z., Lv, Q., Lan, X., Zhang, Y.: Cross-lingual knowledge graph alignment via graph convolutional networks. In: EMNLP. pp. 349–357. ACL (2018)

[34] Xie, R., Liu, Z., Sun, M.: Representation learning of knowledge graphs with hierarchical types. In: IJCAI. pp. 2965–2971. IJCAI/AAAI Press (2016)

[35] Xu, K., Wang, L., Yu, M., Feng, Y., Song, Y., Wang, Z., Yu, D.: Cross-lingual knowledge graph alignment via graph matching neural network. In: ACL (1). pp. 3156–3161. ACL (2019)

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

[37] Zhang, Q., Sun, Z., Hu, W., Chen, M., Guo, L., Qu, Y.: Multi-view knowledge graph embedding for entity alignment. In: IJCAI. pp. 5429–5435 (2019)

[38] Zhu, Q., Zhou, X., Wu, J., Tan, J., Guo, L.: Neighborhood-aware attentional representation for multilingual knowledge graphs. In: IJCAI. pp. 1943–1949 (2019)

10 Conclusion

In this thesis, we presented advances in the area of KG enrichment, in particular, in the tasks of LP and EA. In the following, we summarize our primary contributions and outline promising future research directions.

- In Chapter 4, we proposed the novel task of active learning for EA. We formalized a labeling framework and demonstrated that existing state-of-the-art heuristics from classification settings could not obtain satisfying performance. We further proposed a novel active learning heuristic alongside several strong graph-centrality-based passive learning techniques and empirically demonstrated superior performance, in particular in the more relevant few label regime. Our methods thus allow to achieve strong EA performance with fewer labels, and hence enable its application at scale, or in settings where obtaining single matching entity pairs is expensive, e.g., since it requires domain experts. As future research directions, we envision extension to related graph matching tasks such as on road networks [72], or approximating graph edit distance [135]. Moreover, additional advances in particular for methods also making use of lingual features are of particular interest.

- In Chapter 5, we pioneered in the field of inductive LP with hyper-relational graphs and proposed a novel set of benchmark datasets. We could demonstrate absolute performance improvements of up to 6% Hits@10 compared to baselines without the qualifying information. Our findings thus enable LP with unseen entities to make use of the valuable information comprised by qualifying information present in modern rich KGs such as Wikidata, hence improving link prediction results in downstream applications. Future research directions encompass extensions to allow unseen relations and qualifiers, predicting qualifier pairs for given base triples, or extension from single-hop link prediction to the more complex query embedding. For the latter, we already have proposed an approach in [8].

- In Chapter 6, we introduced the Python library PyKEEN. The library encodes our understanding of individual components of KG embedding models, cf. Chapter 3, and is beneficial for users from different backgrounds ranging from a simple application via a command-line interface to LP researchers developing their components, as shown by e.g., [33, 35, 83]. The library is maintained and extended to this date, with numerous additional components being added since its 1.0 release.

- In Chapter 7, we analyzed the effect of individual components of KG embedding models, namely the interaction function, the loss function, the training assumption, and the explicit use of inverse triples, on LP performance. Our results have a solid

experimental foundation, and we provide several new state-of-the-art configurations. We also analyzed differences between predicting different relational patterns and conducted a large-scale reproducibility study. Our findings allow practitioners to select appropriate hyperparameters but also provide valuable insights for future research on LP. We also envision learning to predict suitable configurations from KG statistics and present relational patterns.

- In Chapter 8, we investigated the use of textual features in EA in a fair and consistent setting. We showed that the features alone achieve surprisingly strong zero-shot performance, fully explaining the observed alignment performance for some noisy datasets. On other datasets, we demonstrated the benefits of neighborhood aggregation through GNN layers. Besides providing surprising insights into the contribution of lingual features to the final EA performance, our findings also improve the intuition behind many of the relevant hyperparameters. In future work, improving the robustness against noise is an important direction to use the additional relational information better.
- In Chapter 9, we proposed a new rank-based metric that is adjusted for chance. Due to its implicit normalization by the number of candidates, it is comparable for settings with a varying number of candidates as encountered in filtered evaluation, or EA evaluation with a varying test set size. Our method improves the interpretability of rank-based evaluation and avoids pitfalls when dealing with different sizes of candidate sets. Future work should focus on further robustness against outlier ranks, e.g., caused by few false-positive triples. There have already been follow-up works building upon our findings [210].

Since machine learning for KGs is an extensive and active research field with numerous applications, there is a great potential for future research. We envision research along the following main axes for the future:

- In practice, KGs often do not remain static, but develop over time. This requires models to be able to cope with unseen entities, i.e., to be inductive. We developed approaches to this problem in the work presented in Chapter 5 for the task of LP. For EA, we investigated the use of textual features in Chapter 8. While these features in principle allow the models to be inductive, EA models have not been evaluated in this setting yet. An interesting research question is whether the part of the model processing the structural information, i.e., the GNN layers can be trained to be transferable across graph pairs. Also self-supervised pre-training of GNNs [109, 108] is a promising direction. Besides developing and improving models capable of this task, uncertainty quantification becomes a critical aspect. In particular with unseen entities, questions may arise about the trustworthiness of (link) predictions, crucial in safety-critical tasks where predicted links affect humans, but also impactful if false links may only have economic impact.
- Another interesting research question is the relation between (large-scale) language models and KG embedding models. Word embeddings from language models

have shown to improve entity alignment (cf., e.g., Chapter 8) and link prediction performance, e.g., [260]. Moreover, there are works on using KGs to improve contextual word embeddings [142, 177, 259]. There are also indications that language models can be used as KGs [221]. Also the automatic conversion from text to KGs an vice-versa is a promising research direction [192].

- Besides single-hop LP, more complex and expressive query patterns are an interesting direction. This task is often named *query embedding*, and in the recent year, we have witnessed promising works in this direction [60, 183, 184, 10]. These complex queries allow users to better describe their information needs directly. On the technical side, directly answering complex queries within a model allows uncertainties of individual steps to be implicitly taken into account in the latent representations, and complex dependency patterns to be considered. In contrast to multi-step methods that first perform link prediction to find (weighted) missing edges and then use a query engine for exact query processing to retrieve the answer entities, such implicit methods do not require calibrated scores and can also model latent knowledge for which, for example, no explicit entity exists. For extending these queries to support hyper-relational queries, we have already presented an approach in [8]. Further research is needed to be able to handle larger query classes and to further increase the efficiency and effectiveness of the methods.

- The comparable and fair evaluation of KG enrichment methods remains challenging. While there are initiatives for standardized evaluation protocols [107], they do not yet cover all use cases, e.g., for EA there are no large-scale datasets accepted across the community. With our contributions towards a normalized ranking metric, cf. Chapter 9, we have contributed to avoid false interpretation of results across differently sized datasets. However, this is only one source of incomparability, and other factors, such as different sources of side information in form of, e.g., labels used for initialization, need to be marked more precisely. Proper ablation studies are required to appropriately credit the individual components contribution towards final performance. We presented such in Chapters 7 and 8 and hope to set a good example.

In summary, we made contributions towards enabling EA at scale (Chapter 4), utilizing hyper-relational information in inductive settings (Chapter 5), isolating effects of individual LP components (Chapters 6 and 7) and label-based initializations EA (Chapter 8), and evaluating more interpretable and comparable (Chapter 9). Our publications and openly available codebases make us confident that these research directions will be successfully pursued in the future to enable further and improve applications with relational information.

Bibliography

[1] Amine Abou-Rjeili and George Karypis. „Multilevel algorithms for partitioning power-law graphs.“ In: *20th International Parallel and Distributed Processing Symposium (IPDPS 2006), Proceedings, 25-29 April 2006, Rhodes Island, Greece*. IEEE, 2006. DOI: 10.1109/IPDPS.2006.1639360. URL: https://doi.org/10.1109/IPDPS.2006.1639360.

[2] Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Fabian Flöck, and Jens Lehmann. „Detecting Linked Data quality issues via crowdsourcing: A DBpedia study.“ In: *Semantic Web* 9.3 (2018), pp. 303–335. DOI: 10.3233/SW-160239.

[3] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. „On the Surprising Behavior of Distance Metrics in High Dimensional Spaces.“ In: *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*. Ed. by Jan Van den Bussche and Victor Vianu. Vol. 1973. Lecture Notes in Computer Science. Springer, 2001, pp. 420–434. DOI: 10.1007/3-540-44503-X_27. URL: https://doi.org/10.1007/3-540-44503-X%5C_27.

[4] Mehdi Ali, **Max Berrendorf***, Charles Tapley Hoyt*, Laurent Vermue*, and Mikhail Galkin. *pykeen/benchmarking: Accompanying arXiv announcement*. Version v1.0. * equal contribution. June 2020. DOI: 10.5281/zenodo.3907252. URL: https://doi.org/10.5281/zenodo.3907252.

[5] 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: *CoRR* abs/2006.13365 (2020). * equal contribution. arXiv: 2006.13365.

[6] Mehdi Ali*, **Max Berrendorf***, Mikhail Galkin, Veronika Thost, Tengfei Ma, Volker Tresp, and Jens Lehmann. „Improving Inductive Link Prediction Using Hyper-relational Facts.“ In: *The Semantic Web – ISWC 2021* (2021). * equal contribution, awarded with Best Research Paper Award, pp. 74–92. DOI: 10.1007/978-3-030-88361-4_5.

[7] 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.

[8] Dimitrios Alivanistos, **Max Berrendorf**, Michael Cochez, and Mikhail Galkin. „Query Embedding on Hyper-relational Knowledge Graphs.“ In: (2021). arXiv: 2106.08166 [cs.AI].

[9] Carl Allen, Ivana Balazevic, and Timothy M. Hospedales. „Interpreting Knowledge Graph Relation Representation from Word Embeddings.“ In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=gLWj29369lW.

[10] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. „Complex Query Answering with Neural Link Predictors.“ In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=Mos9F9kDwkz.

[11] Martin Arjovsky, Soumith Chintala, and Léon Bottou. „Wasserstein GAN.“ In: *CoRR* abs/1701.07875 (2017). arXiv: 1701.07875. URL: http://arxiv.org/abs/1701.07875.

[12] Sören Auer, Viktor Kovtun, Manuel Prinz, Anna Kasprzik, Markus Stocker, and Maria Esther Vidal. „Towards a Knowledge Graph for Science.“ In: *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics - WIMS '18*. ACM Press, 2018. DOI: 10.1145/3227609.3227689.

[13] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. „Multi-relational Poincaré Graph Embeddings.“ In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 4465–4475. URL: https://proceedings.neurips.cc/paper/2019/hash/f8b932c70d0b2e6bf071729a4fa68dfc-Abstract.html.

[14] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. „TuckER: Tensor Factorization for Knowledge Graph Completion.“ In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Association for Computational Linguistics, 2019, pp. 5184–5193. DOI: 10.18653/v1/D19-1522. URL: https://doi.org/10.18653/v1/D19-1522.

[15] Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. „A2N: Attending to Neighbors for Knowledge Graph Inference.“ In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluis Màrquez. Association for Computational Linguistics, 2019, pp. 4387–4392. DOI: 10.18653/v1/p19-1431. URL: https://doi.org/10.18653/v1/p19-1431.

[16] Hannah Bast, Björn Buchhold, and Elmar Haussmann. „Semantic Search on Text and Knowledge Bases.“ In: *Foundations and Trends® in Information Retrieval* 10.1 (2016), pp. 119–271. DOI: 10.1561/1500000032.

[17] Peter W. Battaglia et al. „Relational inductive biases, deep learning, and graph networks.“ In: *CoRR* abs/1806.01261 (2018). arXiv: 1806.01261. URL: http://arxiv.org/abs/1806.01261.

[18] Christian Beecks and **Max Berrendorf**. „Optimal k-Nearest-Neighbor Query Processing via Multiple Lower Bound Approximations.“ In: *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*. Ed. by Naoki Abe, Huan Liu, Calton Pu, Xiaohua Hu, Nesreen K. Ahmed, Mu Qiao, Yang Song, Donald Kossmann, Bing Liu, Kisung Lee, Jiliang Tang, Jingrui He, and Jeffrey S. Saltz. IEEE, 2018, pp. 614–623. DOI: 10.1109/BigData.2018.8622493.

[19] Hiba Belhadi, Karima Akli-Astouati, Youcef Djenouri, and Jerry Chun-Wei Lin. „Data mining-based approach for ontology matching problem.“ In: *Appl. Intell.* 50.4 (2020), pp. 1204–1221. DOI: 10.1007/s10489-019-01593-3. URL: https://doi.org/10.1007/s10489-019-01593-3.

[20] Mikhail Belkin and Partha Niyogi. „Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering.“ In: *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. Ed. by Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani. MIT Press, 2001, pp. 585–591. URL: https://proceedings.neurips.cc/paper/2001/hash/f106b7f99d2cb30c3db1c3cc0fde9ccb-Abstract.html.

[21] Claus Bendtsen and Slavé Petrovski. *How data and AI are helping unlock the secrets of disease*. AstraZeneca Blog. https://www.astrazeneca.com/what-science-can-do/labtalk-blog/uncategorized/how-data-and-ai-are-helping-unlock-the-secrets-of-disease.html. Nov. 2019.

[22] **Max Berrendorf**. *mberr/rank-based-evaluation: Zenodo*. Version 1.0.0. Oct. 2020. DOI: 10.5281/zenodo.4588898.

[23] **Max Berrendorf** and Evgeniy Faerman. *mberr/ea-active-learning: Zenodo*. Version 1.0.1. Dec. 2020. DOI: 10.5281/zenodo.4588896.

[24] **Max Berrendorf**, Evgeniy Faerman, Valentyn Melnychuk, Volker Tresp, and Thomas Seidl. „Knowledge Graph Entity Alignment with Graph Convolutional Networks: Lessons Learned.“ In: *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*. Ed. by Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins. Vol. 12036. Lecture Notes in Computer Science. Springer, 2020, pp. 3–11. DOI: 10.1007/978-3-030-45442-5_1. URL: https://doi.org/10.1007/978-3-030-45442-5%5C_1.

[25] **Max Berrendorf**, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. „Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods.“ In: *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE, Dec. 2020. DOI: 10.1109/wiiat50758.2020.00053.

[26] **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: *The 5th International Workshop on Deep Learning for Graphs (DL4G@WWW2020)* (2020).

[27] **Max Berrendorf**, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. „On the Ambiguity of Rank-Based Evaluation of Entity Alignment or Link Prediction Methods.“ In: *CoRR* abs/2002.06914v2 (2020). arXiv: 2002.06914.

[28] **Max Berrendorf**, Ludwig Wacker, and Evgeniy Faerman. „A Critical Assessment of State-of-the-Art in Entity Alignment.“ In: *CoRR* abs/2010.16314 (2020). arXiv: 2010.16314.

[29] **Max Berrendorf**, Ludwig Wacker, and Evgeniy Faerman. „A Critical Assessment of State-of-the-Art in Entity Alignment.“ In: *Advances in Information Retrieval*. Ed. by Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani. Cham: Springer International Publishing, 2021, pp. 18–32. ISBN: 978-3-030-72240-1. DOI: 10.1007/978-3-030-72240-1_2.

[30] **Max Berrendorf**, Ludwig Wacker, and Evgeniy Faerman. *mberr/ea-sota-comparison: Zenodo*. Version v1.1.1. Dec. 2020. DOI: 10.5281/zenodo.4588894.

[31] **Max Berrendorf***, Evgeniy Faerman*, and Volker Tresp. „Active Learning for Entity Alignment.“ In: *The 5th International Workshop on Deep Learning for Graphs (DL4G@WWW2020)* (2020). * equal contribution. arXiv: 2001.08943.

[32] **Max Berrendorf***, Evgeniy Faerman*, and Volker Tresp. „Active Learning for Entity Alignment.“ In: *Advances in Information Retrieval*. Ed. by Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani. * equal contribution. Cham: Springer International Publishing, 2021, pp. 48–62. ISBN: 978-3-030-72113-8. DOI: 10.1007/978-3-030-72113-8_4.

[33] 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: *Bioinformatics* (May 2021). Ed. by Inanc Birol. DOI: 10.1093/bioinformatics/btab340.

[34] Kurt D. Bollacker, Robert P. Cook, and Patrick Tufts. „Freebase: A Shared Database of Structured General Human Knowledge.“ In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*. AAAI Press, 2007, pp. 1962–1963. URL: http://www.aaai.org/Library/AAAI/2007/aaai07-355.php.

[35] 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). arXiv: 2105.10488. URL: https://arxiv.org/abs/2105.10488.

[36] 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. DOI: 10.1007/s10994-013-5363-6. URL: https://doi.org/10.1007/s10994-013-5363-6.

[37] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. „Translating Embeddings for Modeling Multi-relational Data.“ In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger. 2013, pp. 2787–2795. URL: https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

[38] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. „Learning Structured Embeddings of Knowledge Bases.“ In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. Ed. by Wolfram Burgard and Dan Roth. AAAI Press, 2011. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3659.

[39] Felix Borutta, Julian Busch, Evgeniy Faerman, Adina Klink, and Matthias Schubert. „Structural Graph Representations based on Multiscale Local Network Topologies.“ In: *IEEE/WIC/ACM International Conference on Web Intelligence*. ACM, Oct. 2019. DOI: 10.1145/3350546.3352505.

[40] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. „Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting.“ In: *Graph Representation Learning and Beyond (GRL+) Workshop at the 37th International Conference on Machine Learning* (2020). eprint: 2006.09252. URL: https://grlplus.github.io/papers/75.pdf.

[41] Anna Breit, Simon Ott, Asan Agibetov, and Matthias Samwald. „OpenBioLink: a benchmarking framework for large-scale biomedical link prediction.“ In: *Bioinformatics* 36.13 (Apr. 2020). Ed. by Zhiyong Lu, pp. 4097–4098. DOI: 10.1093/bioinformatics/btaa274.

[42] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. „Geometric Deep Learning: Going beyond Euclidean data.“ In: *IEEE Signal Process. Mag.* 34.4 (2017), pp. 18–42. DOI: 10.1109/MSP.2017.2693418. URL: https://doi.org/10.1109/MSP.2017.2693418.

[43] Ling Cai, Bo Yan, Gengchen Mai, Krzysztof Janowicz, and Rui Zhu. „TransGCN: Coupling Transformation Assumptions with Graph Convolutional Networks for Link Prediction.“ In: *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019, Marina Del Rey, CA, USA, November 19-21, 2019*. Ed. by Mayank Kejriwal, Pedro A. Szekely, and Raphaël Troncy. ACM, 2019, pp. 131–138. DOI: 10.1145/3360901.3364441. URL: https://doi.org/10.1145/3360901.3364441.

[44] Yixin Cao, Zhiyuan Liu, Chengjiang Li, Zhiyuan Liu, Juanzi Li, and Tat-Seng Chua. „Multi-Channel Graph Neural Network for Entity Alignment.“ In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluis Màrquez. Association for Computational Linguistics, 2019, pp. 1452–1461. DOI: 10.18653/v1/p19-1140. URL: https://doi.org/10.18653/v1/p19-1140.

[45] Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. „Dual Quaternion Knowledge Graph Embeddings.“ In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 6894–6902. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16850.

[46] Valentina Anita Carriero, Aldo Gangemi, Maria Letizia Mancinelli, Ludovica Marinucci, Andrea Giovanni Nuzzolese, Valentina Presutti, and Chiara Veninata. „ArCo: The Italian Cultural Heritage Knowledge Graph.“ In: *Lecture Notes in Computer Science*. Springer International Publishing, 2019, pp. 36–52. DOI: 10.1007/978-3-030-30796-7_3.

[47] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. „Low-Dimensional Hyperbolic Knowledge Graph Embeddings.“ In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault. Association for Computational Linguistics, 2020, pp. 6901–6914. DOI: 10.18653/v1/2020.acl-main.617. URL: https://doi.org/10.18653/v1/2020.acl-main.617.

[48] Spencer Chang. *Scaling Knowledge Access and Retrieval at Airbnb*. AirBnB Medium Blog. https://medium.com/airbnb-engineering/scaling-knowledge-access-and-retrieval-at-airbnb-665b6ba21e95. Sept. 2018.

[49] Linlin Chao, Jianshan He, Taifeng Wang, and Wei Chu. „PairRE: Knowledge Graph Embeddings via Paired Relation Vectors.“ In: *CoRR* abs/2011.03798 (2020). arXiv: 2011.03798. URL: https://arxiv.org/abs/2011.03798.

[50] Michelle Cheatham, Adila Krisnadhi, Reihaneh Amini, Pascal Hitzler, Krzysztof Janowicz, Adam Shepherd, Tom Narock, Matt Jones, and Peng Ji. „The GeoLink knowledge graph.“ In: *Big Earth Data* 2.2 (Apr. 2018), pp. 131–143. DOI: 10.1080/20964471.2018.1469291.

[51] Jia Chen, Zhixu Li, Pengpeng Zhao, An Liu, Lei Zhao, Zhigang Chen, and Xiangliang Zhang. „Learning Short-Term Differences and Long-Term Dependencies for Entity Alignment.“ In: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I*. 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. 92–109. DOI: 10.1007/978-3-030-62419-4_6. URL: https://doi.org/10.1007/978-3-030-62419-4%5C_6.

[52] Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. „Co-training Embeddings of Knowledge Graphs and Entity Descriptions for Cross-lingual Entity Alignment.“ In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, July 2018. DOI: 10.24963/ijcai.2018/556.

[53] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. „Multilingual Knowledge Graph Embeddings for Cross-lingual Knowledge Alignment.“ In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Aug. 2017. DOI: 10.24963/ijcai.2017/209.

[54] Xuelu Chen, Muhao Chen, Changjun Fan, Ankith Uppunda, Yizhou Sun, and Carlo Zaniolo. „Multilingual Knowledge Graph Completion via Ensemble Knowledge Transfer.“ 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. Vol. EMNLP 2020. Findings of ACL. Association for Computational Linguistics, 2020, pp. 3227–3238. DOI: 10.18653/v1/2020.findings-emnlp.290. URL: https://doi.org/10.18653/v1/2020.findings-emnlp.290.

[55] Dawei Cheng, Fangzhou Yang, Xiaoyang Wang, Ying Zhang, and Liqing Zhang. „Knowledge Graph-based Event Embedding Framework for Financial Quantitative Investments.“ In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, July 2020. DOI: 10.1145/3397271.3401427.

[56] Boris V. Cherkassky, Andrew V. Goldberg, and Paul Martin. „Augment or Push: A Computational Study of Bipartite Matching and Unit-Capacity Flow Algorithms.“ In: *ACM J. Exp. Algorithmics* 3 (1998), p. 8. DOI: 10.1145/297096.297140. URL: https://doi.org/10.1145/297096.297140.

[57] Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. „AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies.“ In: *Proc. VLDB Endow.* 2.2 (2009), pp. 1586–1589. DOI: 10.14778/1687553.1687598. URL: http://www.vldb.org/pvldb/vol2/vldb09-1003.pdf.

[58] Marco Cuturi. „Sinkhorn Distances: Lightspeed Computation of Optimal Transport.“ In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger. 2013, pp. 2292–2300. URL: https://proceedings.neurips.cc/paper/2013/hash/af21d0c97db2e27e13572cbf59eb343d-Abstract.html.

[59] Jeff Dalgliesh. *How the Enterprise Knowledge Graph Connects Oil and Gas Data Silos*. Maana Blog. https://www.maana.io/blog/enterprise-knowledge-graph-connects-oil-gas-data-silos/. May 2016.

[60] Daniel Daza and Michael Cochez. „Message Passing Query Embedding.“ In: *ICML Workshop - Graph Representation Learning and Beyond*. 2020. URL: https://arxiv.org/abs/2002.02406.

[61] Alex DeJong, Radmila Bord, Will Dowling, Rinke Hoekstra, Ryan Moquin, Charlie O, Mevan Samarasinghe, Paul Snyder, Craig E. Stanley Jr., Anna Tordai, Michael Trefry, and Paul Groth. „Elsevier's Healthcare Knowledge Graph and the Case for Enterprise Level Linked Data Standards.“ In: *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018*. Ed. by Marieke van Erp, Medha Atre, Vanessa López, Kavitha Srinivas, and Carolina Fortuna. Vol. 2180. CEUR Workshop Proceedings. CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2180/paper-85.pdf.

[62] Caglar Demir, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. „A shallow neural model for relation prediction.“ In: *15th IEEE International Conference on Semantic Computing, ICSC 2021, Laguna Hills, CA, USA, January 27-29, 2021*. IEEE, 2021, pp. 179–182. DOI: 10.1109/ICSC50631.2021.00038. URL: https://doi.org/10.1109/ICSC50631.2021.00038.

[63] Caglar Demir and Axel-Cyrille Ngonga Ngomo. „Convolutional Complex Knowledge Graph Embeddings.“ In: *The Semantic Web - 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings*. Ed. by Ruben Verborgh, Katja Hose, Heiko Paulheim, Pierre-Antoine Champin, Maria Maleshkova, Óscar Corcho, Petar Ristoski, and Mehwish Alam. Vol. 12731. Lecture Notes in Computer Science. Springer, 2021, pp. 409–424. DOI: 10.1007/978-3-030-77385-4_24. URL: https://doi.org/10.1007/978-3-030-77385-4%5C_24.

[64] Caglar Demir and Axel-Cyrille Ngonga Ngomo. „Out-of-Vocabulary Entities in Link Prediction.“ In: *CoRR* abs/2105.12524 (2021). arXiv: 2105.12524. URL: https://arxiv.org/abs/2105.12524.

[65] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. „Convolutional 2D Knowledge Graph Embeddings.“ In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 1811–1818. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17366.

[66] Deepika Devarajan. *Happy Birthday Watson Discovery*. IBM Cloud Blog. https://www.ibm.com/blogs/bluemix/2017/12/happy-birthday-watson-discovery/. Dec. 2017.

[67] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.“ In: *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, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

[68] Laura Dietz, Alexander Kotov, and Edgar Meij. „Utilizing Knowledge Graphs for Text-Centric Information Retrieval.“ In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, June 2018. DOI: 10.1145/3209978.3210187.

[69] Xin Dong, Evgeniy Gabrilovich, Geremy 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: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. Ed. by Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani. ACM, 2014, pp. 601–610. DOI: 10.1145/2623330.2623623.

[70] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. „metapath2vec: Scalable Representation Learning for Heterogeneous Networks.“ In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 135–144. DOI: 10.1145/3097983.3098036. URL: https://doi.org/10.1145/3097983.3098036.

[71] Takuma Ebisu and Ryutaro Ichise. „TorusE: Knowledge Graph Embedding on a Lie Group.“ In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 1819–1826. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16227.

[72] Evgeniy Faerman, Otto Voggenreiter, Felix Borutta, Tobias Emrich, **Max Berrendorf**, and Matthias Schubert. „Graph Alignment Networks with Node Matching Scores." In: *Graph Representation Learning NeurIPS 2019 Workshop* (2019).

[73] Miao Fan, Qiang Zhou, Emily Chang, and Thomas Fang Zheng. „Transition-based Knowledge Graph Embedding with Relational Mapping Properties." In: *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation, PACLIC 28, Cape Panwa Hotel, Phuket, Thailand, December 12-14, 2014*. Ed. by Wirote Aroonmanakun, Prachya Boonkwan, and Thepchai Supnithi. The PACLIC 28 Organizing Committee and PACLIC Steering Committee / ACL / Department of Linguistics, Faculty of Arts, Chulalongkorn University, 2014, pp. 328–337. URL: https://www.aclweb.org/anthology/Y14-1039/.

[74] Michael Färber. „The Microsoft Academic Knowledge Graph: A Linked Data Source with 8 Billion Triples of Scholarly Data." In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*. Ed. by Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon. Vol. 11779. Lecture Notes in Computer Science. Springer, 2019, pp. 113–129. DOI: 10.1007/978-3-030-30796-7_8. URL: https://doi.org/10.1007/978-3-030-30796-7%5C_8.

[75] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F. Cruz, and Francisco M. Couto. „The AgreementMakerLight Ontology Matching System." In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences - Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings*. Ed. by Robert Meersman, Hervé Panetto, Tharam S. Dillon, Johann Eder, Zohra Bellahsene, Norbert Ritter, Pieter De Leenheer, and Dejing Dou. Vol. 8185. Lecture Notes in Computer Science. Springer, 2013, pp. 527–541. DOI: 10.1007/978-3-642-41030-7_38. URL: https://doi.org/10.1007/978-3-642-41030-7%5C_38.

[76] Jun Feng, Minlie Huang, Mingdong Wang, Mantong Zhou, Yu Hao, and Xiaoyan Zhu. „Knowledge Graph Embedding by Flexible Translation." In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*. Ed. by Chitta Baral, James P. Delgrande, and Frank Wolter. AAAI Press, 2016, pp. 557–560. URL: http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12887.

[77] Matthias Fey and Jan Eric Lenssen. „Fast Graph Representation Learning with PyTorch Geometric." In: *Representation Learning on Graphs and Manifolds Workshop at ICLR 2019* (2019). URL: http://arxiv.org/abs/1903.02428.

[78] Matthias Fey, Jan Eric Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. „Deep Graph Matching Consensus." In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=HyeJf1HKvS.

[79] Erwin Filtz. „Building and Processing a Knowledge-Graph for Legal Data.“ In: *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part II*. Ed. by Eva Blomqvist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler, and Olaf Hartig. Vol. 10250. Lecture Notes in Computer Science. 2017, pp. 184–194. DOI: 10.1007/978-3-319-58451-5_13. URL: https://doi.org/10.1007/978-3-319-58451-5%5C_13.

[80] Xiaoyi Fu, Xinqi Ren, Ole J. Mengshoel, and Xindong Wu. „Stochastic Optimization for Market Return Prediction Using Financial Knowledge Graph.“ In: *2018 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, Nov. 2018. DOI: 10.1109/icbk.2018.00012.

[81] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. „Fast rule mining in ontological knowledge bases with AMIE+.“ In: *VLDB J.* 24.6 (2015), pp. 707–730. DOI: 10.1007/s00778-015-0394-1. URL: https://doi.org/10.1007/s00778-015-0394-1.

[82] David Gale and Lloyd S Shapley. „College admissions and the stability of marriage.“ In: *The American Mathematical Monthly* 69.1 (1962), pp. 9–15.

[83] Mikhail Galkin, Jiapeng Wu, Etienne Denis, and William L. Hamilton. „NodePiece: Compositional and Parameter-Efficient Representations of Large Knowledge Graphs.“ In: *CoRR* abs/2106.12144 (2021). arXiv: 2106.12144. URL: https://arxiv.org/abs/2106.12144.

[84] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. „Hyperbolic Neural Networks.“ In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 5350–5360. URL: https://proceedings.neurips.cc/paper/2018/hash/dbab2adc8f9d078009ee3fa810bea142-Abstract.html.

[85] Junyang Gao, Xian Li, Yifan Ethan Xu, Bunyamin Sisman, Xin Luna Dong, and Jun Yang. „Efficient Knowledge Graph Accuracy Evaluation.“ In: *Proc. VLDB Endow.* 12.11 (2019), pp. 1679–1691. DOI: 10.14778/3342263.3342642. URL: http://www.vldb.org/pvldb/vol12/p1679-gao.pdf.

[86] Alberto Garcia-Durán, Antoine Bordes, and Nicolas Usunier. „Effective Blending of Two and Three-way Interactions for Modeling Multi-relational Data.“ In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I*. Ed. by Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo. Vol. 8724. Lecture Notes in Computer Science. Springer, 2014, pp. 434–449. DOI: 10.1007/978-3-662-44848-9_28. URL: https://doi.org/10.1007/978-3-662-44848-9%5C_28.

[87] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. „Neural Message Passing for Quantum Chemistry.“ In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1263–1272. URL: http://proceedings.mlr.press/v70/gilmer17a.html.

[88] Rafael S. Gonçalves, Matthew Horridge, Rui Li, Yu Liu, Mark A. Musen, Csongor I. Nyulas, Evelyn Obamos, Dhananjay Shrouty, and David Temple. „Use of OWL and Semantic Web Technologies at Pinterest.“ In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*. Ed. by Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon. Vol. 11779. Lecture Notes in Computer Science. Springer, 2019, pp. 418–435. DOI: 10.1007/978-3-030-30796-7_26. URL: https://doi.org/10.1007/978-3-030-30796-7%5C_26.

[89] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. „Generative Adversarial Nets.“ In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger. 2014, pp. 2672–2680. URL: https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

[90] Aditya Grover and Jure Leskovec. „node2vec: Scalable Feature Learning for Networks.“ In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. Ed. by Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi. ACM, 2016, pp. 855–864. DOI: 10.1145/2939672.2939754. URL: https://doi.org/10.1145/2939672.2939754.

[91] Saiping Guan, Xiaolong Jin, Yantao Jia, Yuanzhuo Wang, Huawei Shen, and Xueqi Cheng. „Self-Learning and Embedding Based Entity Alignment.“ In: *IEEE International Conference on Big Knowledge, ICBK 2017, Hefei, China, August 9-10, 2017*. IEEE Computer Society, 2017, pp. 33–40. DOI: 10.1109/ICBK.2017.15. URL: https://doi.org/10.1109/ICBK.2017.15.

[92] Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, Yantao Jia, Huawei Shen, Zixuan Li, and Xueqi Cheng. „Self-learning and embedding based entity alignment.“ In: *Knowl. Inf. Syst.* 59.2 (2019), pp. 361–386. DOI: 10.1007/s10115-018-1191-0.

[93] 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), pp. 1–1. DOI: 10.1109/tkde.2020.3028705.

[94] Ferras Hamad, Isaac Liu, and Xian Xing Zhang. *Food Discovery with Uber Eats: Building a Query Understanding Engine*. Uber Engineering Blog. https://eng.uber.com/uber-eats-query-understanding/. June 2018.

[95] William L. Hamilton, Zhitao Ying, and Jure Leskovec. „Inductive Representation Learning on Large Graphs." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 1024–1034. URL: https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

[96] Kazuo Hara, Ikumi Suzuki, Kei Kobayashi, Kenji Fukumizu, and Milos Radovanovic. „Flattening the Density Gradient for Eliminating Spatial Centrality to Reduce Hubness." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Ed. by Dale Schuurmans and Michael P. Wellman. AAAI Press, 2016, pp. 1659–1665. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12055.

[97] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. „Deep Residual Learning for Image Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.90.

[98] Qi He, Bee-Chung Chen, and Deepak Agarwal. *Building The LinkedIn Knowledge Graph*. LinkedIn Blog. https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph. Oct. 2016.

[99] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. „Learning to Represent Knowledge Graphs with Gaussian Embedding." In: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. Ed. by James Bailey, Alistair Moffat, Charu C. Aggarwal, Maarten de Rijke, Ravi Kumar, Vanessa Murdock, Timos K. Sellis, and Jeffrey Xu Yu. ACM, 2015, pp. 623–632. DOI: 10.1145/2806416.2806502. URL: https://doi.org/10.1145/2806416.2806502.

[100] Cory Henson, Stefan Schmid, Anh Tuan Tran, and Antonios Karatzoglou. „Using a Knowledge Graph of Scenes to Enable Search of Autonomous Driving Data." 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*. Ed. by Mari Carmen Suárez-Figueroa, Gong Cheng, Anna Lisa Gentile, Christophe Guéret, C. Maria Keet, and Abraham Bernstein. Vol. 2456. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 313–314. URL: http://ceur-ws.org/Vol-2456/paper84.pdf.

[101] Sven Hertling, Jan Portisch, and Heiko Paulheim. „Supervised ontology and instance matching with MELT." In: *Proceedings of the 15th International Workshop on Ontology Matching co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece),*

November 2, 2020. Ed. by Pavel Shvaiko, Jérôme Euzenat, Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, and Cássia Trojahn. Vol. 2788. CEUR Workshop Proceedings. CEUR-WS.org, 2020, pp. 60–71. URL: http://ceur-ws.org/Vol-2788/om2020%5C_LTpaper6.pdf.

[102] Marcel Hildebrandt, Swathi Shyam Sunder, Serghei Mogoreanu, Mitchell Joblin, Akhil Mehta, Ingo Thon, and Volker Tresp. „A Recommender System for Complex Real-World Applications with Nonlinear Dependencies and Knowledge Graph Context.“ In: *The Semantic Web*. Springer International Publishing, 2019, pp. 179–193. DOI: 10.1007/978-3-030-21348-0_12.

[103] Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. „Systematic integration of biomedical knowledge prioritizes drugs for repurposing.“ In: *eLife* 6 (Sept. 2017). DOI: 10.7554/elife.26726.

[104] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. „Improving neural networks by preventing co-adaptation of feature detectors.“ In: *CoRR* abs/1207.0580 (2012). arXiv: 1207.0580. URL: http://arxiv.org/abs/1207.0580.

[105] Sepp Hochreiter and Jürgen Schmidhuber. „Long Short-Term Memory.“ In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[106] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. „Knowledge Graphs.“ In: *CoRR* abs/2003.02320 (2020). arXiv: 2003.02320. URL: https://arxiv.org/abs/2003.02320.

[107] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. „Open Graph Benchmark: Datasets for Machine Learning on Graphs.“ 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. URL: https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

[108] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. „Strategies for Pre-training Graph Neural Networks.“ In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=HJlWWJSFDH.

[109] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. „GPT-GNN: Generative Pre-Training of Graph Neural Networks.“ In: *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. Ed. by Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash. ACM, 2020, pp. 1857–1867. DOI: 10.1145/3394486.3403237. URL: https://doi.org/10.1145/3394486.3403237.

[110] Thomas Hubauer, Steffen Lamparter, Peter Haase, and Daniel Markus Herzig. „Use Cases of the Industrial Knowledge Graph at Siemens.“ In: *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018*. Ed. by Marieke van Erp, Medha Atre, Vanessa López, Kavitha Srinivas, and Carolina Fortuna. Vol. 2180. CEUR Workshop Proceedings. CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2180/paper-86.pdf.

[111] Vassilis N. Ioannidis, Xiang Song, Saurav Manchanda, Mufei Li, Xiaoqin Pan, Da Zheng, Xia Ning, Xiangxiang Zeng, and George Karypis. *DRKG - Drug Repurposing Knowledge Graph for Covid-19*. https://github.com/gnn4dr/DRKG/. 2020.

[112] Sergey Ioffe and Christian Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.“ In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Ed. by Francis R. Bach and David M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[113] Robert Isele, Anja Jentzsch, and Christian Bizer. „Efficient Multidimensional Blocking for Link Discovery without losing Recall.“ In: *Proceedings of the 14th International Workshop on the Web and Databases 2011, WebDB 2011, Athens, Greece, June 12, 2011*. Ed. by Amélie Marian and Vasilis Vassalos. 2011. URL: http://webdb2011.rutgers.edu/papers/Paper%5C%2039/silk.pdf.

[114] Eric Jang, Shixiang Gu, and Ben Poole. „Categorical Reparameterization with Gumbel-Softmax.“ In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=rkE3y85ee.

[115] Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. „A latent factor model for highly multi-relational data.“ In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger. 2012, pp. 3176–3184. URL: https://proceedings.neurips.cc/paper/2012/hash/0a1bf96b7165e962e90cb14648c9462d-Abstract.html.

[116] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. „Knowledge Graph Embedding via Dynamic Mapping Matrix.“ In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computer Linguistics, 2015, pp. 687–696. DOI: 10.3115/v1/p15-1067.

[117] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. „Knowledge Graph Completion with Adaptive Sparse Transfer Matrix.“ In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Ed. by Dale Schuurmans and Michael P. Wellman. AAAI Press, 2016, pp. 985–991. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11982.

[118] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. „A Survey on Knowledge Graphs: Representation, Acquisition and Applications.“ In: *CoRR* abs/2002.00388 (2020). arXiv: 2002.00388. URL: https://arxiv.org/abs/2002.00388.

[119] Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. „Learning Visual Features from Large Weakly Supervised Data.“ In: *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Vol. 9911. Lecture Notes in Computer Science. Springer, 2016, pp. 67–84. DOI: 10.1007/978-3-319-46478-7_5. URL: https://doi.org/10.1007/978-3-319-46478-7%5C_5.

[120] Ademar Crotti Junior, Fabrizio Orlandi, Damien Graux, Murhaf Hossari, Declan O'Sullivan, Christian Hartz, and Christian Dirschl. „Knowledge Graph-Based Legal Search over German Court Cases.“ In: *The Semantic Web: ESWC 2020 Satellite Events*. Springer International Publishing, 2020, pp. 293–297. DOI: 10.1007/978-3-030-62327-2_44.

[121] Seyed Mehran Kazemi and David Poole. „SimplE Embedding for Link Prediction in Knowledge Graphs.“ In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 4289–4300. URL: https://proceedings.neurips.cc/paper/2018/hash/b2ab001909a8a6f04b51920306046ce5-Abstract.html.

[122] Thomas N. Kipf and Max Welling. „Semi-Supervised Classification with Graph Convolutional Networks.“ In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[123] Philip A. Knight. „The Sinkhorn-Knopp Algorithm: Convergence and Applications.“ In: *SIAM J. Matrix Anal. Appl.* 30.1 (2008), pp. 261–275. DOI: 10.1137/060659624. URL: https://doi.org/10.1137/060659624.

[124] Hyunwoong Ko, Paul Witherell, Yan Lu, Samyeon Kim, and David W. Rosen. „Machine learning and knowledge graph based design rule construction for additive manufacturing.“ In: *Additive Manufacturing* 37 (Jan. 2021), p. 101620. DOI: 10.1016/j.addma.2020.101620.

[125] Bhushan Kotnis and Vivi Nastase. „Analysis of the Impact of Negative Sampling on Link Prediction in Knowledge Graphs.“ In: *CoRR* abs/1708.06816 (2017). arXiv: 1708.06816. URL: http://arxiv.org/abs/1708.06816.

[126] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. „Angle-based outlier detection in high-dimensional data.“ In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. Ed. by Ying Li, Bing Liu, and Sunita Sarawagi. ACM, 2008, pp. 444–452. DOI: 10.1145/1401890.1401946. URL: https://doi.org/10.1145/1401890.1401946.

[127] Arun Krishnan. *Making search easier: How Amazon's Product Graph is helping customers find products more easily*. Amazon Blog. https://blog.aboutamazon.com/innovation/making-search-easier. Aug. 2018.

[128] Harold W Kuhn. „The Hungarian method for the assignment problem.“ In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[129] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. „Canonical Tensor Decomposition for Knowledge Base Completion.“ In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2869–2878. URL: http://proceedings.mlr.press/v80/lacroix18a.html.

[130] Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. „Word translation without parallel data.“ In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=H196sainb.

[131] 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. DOI: 10.3233/SW-140134. URL: https://doi.org/10.3233/SW-140134.

[132] Chengjiang Li, Yixin Cao, Lei Hou, Jiaxin Shi, Juanzi Li, and Tat-Seng Chua. „Semi-supervised Entity Alignment via Joint Knowledge Embedding Model and Cross-graph Model.“ In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Association for Computational Linguistics, 2019, pp. 2723–2732. DOI: 10.18653/v1/D19-1274. URL: https://doi.org/10.18653/v1/D19-1274.

[133] Qimai Li, Zhichao Han, and Xiao-Ming Wu. „Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning.“ In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 3538–3545. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16098.

[134] Weidong Li, Xinyu Zhang, Yaqian Wang, Zhihuan Yan, and Rong Peng. „Graph2Seq: Fusion Embedding Learning for Knowledge Graph Completion.“ In: *IEEE Access* 7 (2019), pp. 157960–157971. DOI: 10.1109/ACCESS.2019.2950230. URL: https://doi.org/10.1109/ACCESS.2019.2950230.

[135] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. „Graph Matching Networks for Learning the Similarity of Graph Structured Objects.“ In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 3835–3845. URL: http://proceedings.mlr.press/v97/li19d.html.

[136] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. „Gated Graph Sequence Neural Networks.“ In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://arxiv.org/abs/1511.05493.

[137] Xixun Lin, Hong Yang, Jia Wu, Chuan Zhou, and Bin Wang. „Guiding Cross-lingual Entity Alignment via Adversarial Knowledge Embedding.“ In: *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*. Ed. by Jianyong Wang, Kyuseok Shim, and Xindong Wu. IEEE, 2019, pp. 429–438. DOI: 10.1109/ICDM.2019.00053.

[138] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. „Learning Entity and Relation Embeddings for Knowledge Graph Completion.“ In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI Press, 2015, pp. 2181–2187. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

[139] Danyang Liu, Ting Bai, Jianxun Lian, Xin Zhao, Guangzhong Sun, Ji-Rong Wen, and Xing Xie. „News Graph: An Enhanced Knowledge Graph for News Recommendation.“ In: *Proceedings of the Second Workshop on Knowledge-aware and Conversational Recommender Systems, co-located with 28th ACM International Conference on Information and Knowledge Management, KaRS@CIKM 2019, Beijing, China, November 7, 2019*. Ed. by Vito Walter Anelli and Tommaso Di Noia. Vol. 2601. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 1–7. URL: http://ceur-ws.org/Vol-2601/kars2019%5C_paper%5C_01.pdf.

[140] Hanxiao Liu, Yuexin Wu, and Yiming Yang. „Analogical Inference for Multi-relational Embeddings.“ In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 2168–2178. URL: http://proceedings.mlr.press/v70/liu17d.html.

[141] Meng Liu, Hongyang Gao, and Shuiwang Ji. „Towards Deeper Graph Neural Networks.“ In: *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. Ed. by Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash. ACM, 2020, pp. 338–348. DOI: 10.1145/3394486.3403076. URL: https://doi.org/10.1145/3394486.3403076.

[142] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. „K-BERT: Enabling Language Representation with Knowledge Graph.“ In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 2901–2908. URL: https://aaai.org/ojs/index.php/AAAI/article/view/5681.

[143] Xiyang Liu, Huobin Tan, Qinghong Chen, and Guangyan Lin. „RAGAT: Relation Aware Graph Attention Network for Knowledge Graph Completion.“ In: *IEEE Access* 9 (2021), pp. 20840–20849. DOI: 10.1109/ACCESS.2021.3055529. URL: https://doi.org/10.1109/ACCESS.2021.3055529.

[144] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. „Rectifier nonlinearities improve neural network acoustic models.“ In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML'13. Atlanta, GA, USA: JMLR.org, 2013. URL: http://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

[145] Dhruv Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. „Exploring the Limits of Weakly Supervised Pretraining.“ In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part II*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair

Weiss. Vol. 11206. Lecture Notes in Computer Science. Springer, 2018, pp. 185–201. DOI: 10.1007/978-3-030-01216-8_12. URL: https://doi.org/10.1007/978-3-030-01216-8%5C_12.

[146] Tareq B. Malas, Roman Kudrin, Sergei Starikov, Dorien Peters, Marco Roos, Peter A. C. 't Hoen, and Kristina M. Hettne. „Semantic Knowledge Graph Network Features for Drug Repurposing." In: *Proceedings of the 10th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences (SWAT4LS 2017), Rome, Italy, December 4-7, 2017*. Ed. by Adrian Paschke, Albert Burger, Andrea Splendiani, M. Scott Marshall, Paolo Romano, and Valentina Presutti. Vol. 2042. CEUR Workshop Proceedings. CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-2042/paper12.pdf.

[147] Evangelos Maliaroudakis, Katarina Boland, Stefan Dietze, Konstantin Todorov, Yannis Tzitzikas, and Pavlos Fafalios. „ClaimLinker: Linking Text to a Knowledge Graph of Fact-checked Claims." In: *Companion Proceedings of the Web Conference 2021*. ACM, Apr. 2021. DOI: 10.1145/3442442.3458601.

[148] Stanislav Malyshev, Markus Krötzsch, Larry González, Julius Gonsior, and Adrian Bielefeldt. „Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph." In: *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part II*. Ed. by Denny Vrandecic, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl. Vol. 11137. Lecture Notes in Computer Science. Springer, 2018, pp. 376–394. DOI: 10.1007/978-3-030-00668-6_23. URL: https://doi.org/10.1007/978-3-030-00668-6%5C_23.

[149] Xin Mao, Wenting Wang, Huimin Xu, Man Lan, and Yuanbin Wu. „MRAEA: An Efficient and Robust Entity Alignment Approach for Cross-lingual Knowledge Graph." In: *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*. Ed. by James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang. ACM, 2020, pp. 420–428. DOI: 10.1145/3336191.3371804.

[150] Xin Mao, Wenting Wang, Huimin Xu, Yuanbin Wu, and Man Lan. „Relational Reflection Entity Alignment." In: *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. Ed. by Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux. ACM, 2020, pp. 1095–1104. DOI: 10.1145/3340531.3412001. URL: https://doi.org/10.1145/3340531.3412001.

[151] Cynthia Matuszek, John Cabral, Michael J. Witbrock, and John DeOliveira. „An Introduction to the Syntax and Content of Cyc." In: *Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, Papers from the 2006 AAAI Spring Symposium, Technical Report SS-06-05, Stanford, California, USA, March 27-29, 2006*. AAAI, 2006,

pp. 44–49. URL: http://www.aaai.org/Library/Symposia/Spring/2006/ss06-05-007.php.

[152] Edgar Meij. *Understanding News using the Bloomberg Knowledge Graph*. Invited talk at the Big Data Innovators Gathering (TheWebConf). Slides at https://speakerdeck.com/emeij/understanding-news-using-the-bloomberg-knowledge-graph. 2019.

[153] George A. Miller. „WordNet: A Lexical Database for English.“ In: *Commun. ACM* 38.11 (1995), pp. 39–41. DOI: 10.1145/219717.219748. URL: http://doi.acm.org/10.1145/219717.219748.

[154] Tom M. Mitchell et al. „Never-ending learning.“ In: *Commun. ACM* 61.5 (2018), pp. 103–115. DOI: 10.1145/3191513. URL: https://doi.org/10.1145/3191513.

[155] Sameh K. Mohamed, Vit Novácek, Pierre-Yves Vandenbussche, and Emir Muñoz. „Loss Functions in Knowledge Graph Embedding Models.“ In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019*. Ed. by Mehwish Alam, Davide Buscaldi, Michael Cochez, Francesco Osborne, Diego Reforgiato Recupero, and Harald Sack. Vol. 2377. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 1–10. URL: http://ceur-ws.org/Vol-2377/paper%5C_1.pdf.

[156] Elena Montiel-Ponsoda, Victor Rodriguez-Doncel, and Jorge Gracia. „Building the Legal Knowledge Graph for Smart Compliance Services in Multilingual Europe.“ In: *Proceedings of the 1st Workshop on Technologies for Regulatory Compliance co-located with the 30th International Conference on Legal Knowledge and Information Systems (JURIX 2017), Luxembourg, December 13, 2017*. Ed. by Victor Rodriguez-Doncel, Pompeu Casanovas, and Jorge González-Conejero. Vol. 2049. CEUR Workshop Proceedings. CEUR-WS.org, 2017, pp. 15–17. URL: http://ceur-ws.org/Vol-2049/02paper.pdf.

[157] Sebastian Mosbach, Angiras Menon, Feroz Farazi, Nenad Krdzavac, Xiaochi Zhou, Jethro Akroyd, and Markus Kraft. „Multiscale Cross-Domain Thermochemical Knowledge-Graph.“ In: *Journal of Chemical Information and Modeling* 60.12 (Nov. 2020), pp. 6155–6166. DOI: 10.1021/acs.jcim.0c01145.

[158] David Mrdjenovich, Matthew K. Horton, Joseph H. Montoya, Christian M. Legaspi, Shyam Dwaraknath, Vahe Tshitoyan, Anubhav Jain, and Kristin A. Persson. „propnet: A Knowledge Graph for Materials Science.“ In: *Matter* 2.2 (Feb. 2020), pp. 464–480. DOI: 10.1016/j.matt.2019.11.013.

[159] James Munkres. „Algorithms for the assignment and transportation problems.“ In: *Journal of the society for industrial and applied mathematics* 5.1 (1957), pp. 32–38.

[160] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. „Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs.“ In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluis Marquez. Association for Computational Linguistics, 2019, pp. 4710–4723. DOI: 10.18653/v1/p19-1466. URL: https://doi.org/10.18653/v1/p19-1466.

[161] Mojtaba Nayyeri, Sahar Vahdati, Can Aykul, and Jens Lehmann. „5* Knowledge Graph Embeddings with Projective Transformations.“ In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 9064–9072. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17095.

[162] Axel-Cyrille Ngonga Ngomo and Sören Auer. „LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data.“ In: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. Ed. by Toby Walsh. IJCAI/AAAI, 2011, pp. 2312–2317. DOI: 10.5591/978-1-57735-516-8/IJCAI11-385. URL: https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-385.

[163] Axel-Cyrille Ngonga Ngomo, Jens Lehmann, Sören Auer, and Konrad Höffner. „RAVEN - active learning of link specifications.“ In: *Proceedings of the 6th International Workshop on Ontology Matching, Bonn, Germany, October 24, 2011*. Ed. by Pavel Shvaiko, Jérôme Euzenat, Tom Heath, Christoph Quix, Ming Mao, and Isabel F. Cruz. Vol. 814. CEUR Workshop Proceedings. CEUR-WS.org, 2011. URL: http://ceur-ws.org/Vol-814/om2011%5C_Tpaper3.pdf.

[164] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. „EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming.“ In: *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*. Ed. by Elena Simperl, Philipp Cimiano, Axel Polleres, Óscar Corcho, and Valentina Presutti. Vol. 7295. Lecture Notes in Computer Science. Springer, 2012, pp. 149–163. DOI: 10.1007/978-3-642-30284-8_17. URL: https://doi.org/10.1007/978-3-642-30284-8%5C_17.

[165] 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: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 327–333. DOI: 10.18653/v1/N18-2053. URL: https://www.aclweb.org/anthology/N18-2053.

[166] 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.

[167] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. „Holographic Embeddings of Knowledge Graphs.“ In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Ed. by Dale Schuurmans and Michael P. Wellman. AAAI Press, 2016, pp. 1955–1961. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12484.

[168] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. „A Three-Way Model for Collective Learning on Multi-Relational Data.“ In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. Ed. by Lise Getoor and Tobias Scheffer. Omnipress, 2011, pp. 809–816. URL: https://icml.cc/2011/papers/438%5C_icmlpaper.pdf.

[169] Hao Nie, Xianpei Han, Le Sun, Chi Man Wong, Qiang Chen, Suhui Wu, and Wei Zhang. „Global Structure and Local Semantics-Preserved Embeddings for Entity Alignment.“ In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. Ed. by Christian Bessiere. ijcai.org, 2020, pp. 3658–3664. DOI: 10.24963/ijcai.2020/506. URL: https://doi.org/10.24963/ijcai.2020/506.

[170] Prakhar Ojha and Partha P. Talukdar. „KGEval: Accuracy Estimation of Automatically Constructed Knowledge Graphs.“ In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Association for Computational Linguistics, 2017, pp. 1741–1750. DOI: 10.18653/v1/d17-1183. URL: https://doi.org/10.18653/v1/d17-1183.

[171] Ekpe Okorafor and Atish Ray. *The path from data to knowledge*. Accenture Applied Intelligence Blog. https://www.accenture.com/us-en/insights/digital/data-to-knowledge. June 2019.

[172] Shichao Pei, Lu Yu, Guoxian Yu, and Xiangliang Zhang. „REA: Robust Cross-lingual Entity Alignment Between Knowledge Graphs.“ In: *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. Ed. by Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash. ACM, 2020, pp. 2175–2184. DOI: 10.1145/3394486.3403268. URL: https://doi.org/10.1145/3394486.3403268.

[173] Shichao Pei, Lu Yu, and Xiangliang Zhang. „Improving Cross-lingual Entity Alignment via Optimal Transport.“ In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 3231–3237. DOI: 10.24963/ijcai.2019/448. URL: https://doi.org/10.24963/ijcai.2019/448.

[174] Yanhui Peng, Jing Zhang, Cangqi Zhou, and Jian Xu. „Embedding-Based Entity Alignment Using Relation Structural Similarity.“ In: *2020 IEEE International Conference on Knowledge Graph, ICKG 2020, Online, August 9-11, 2020*. Ed. by Enhong Chen and Grigoris Antoniou. IEEE, 2020, pp. 123–130. DOI: 10.1109/ICBK50248.2020.00027.

[175] Jeffrey Pennington, Richard Socher, and Christopher Manning. „Glove: Global Vectors for Word Representation.“ In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014. DOI: 10.3115/v1/d14-1162.

[176] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. „DeepWalk: online learning of social representations.“ In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. Ed. by Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani. ACM, 2014, pp. 701–710. DOI: 10.1145/2623330.2623732. URL: https://doi.org/10.1145/2623330.2623732.

[177] Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. „Knowledge Enhanced Contextual Word Representations.“ In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Association for Computational Linguistics, 2019, pp. 43–54. DOI: 10.18653/v1/D19-1005. URL: https://doi.org/10.18653/v1/D19-1005.

[178] R. J. Pittman, Amit Srivastava, Sanjika Hewavitharana, Ajinkya Kale, and Saab Mansour. *Cracking the Code on Conversational Commerce*. eBay Blog. https://www.ebayinc.com/stories/news/cracking-the-code-on-conversational-commerce/. Apr. 2017.

[179] Jan Portisch and Heiko Paulheim. „Wiktionary matcher results for OAEI 2020.“ In: *Proceedings of the 15th International Workshop on Ontology Matching co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 2, 2020*. Ed. by Pavel Shvaiko, Jérôme Euzenat, Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, and Cássia Trojahn. Vol. 2788. CEUR Workshop Proceedings. CEUR-WS.org, 2020, pp. 225–232. URL: http://ceur-ws.org/Vol-2788/oaei20%5C_paper14.pdf.

[180] Wei Qian, Cong Fu, Yu Zhu, Deng Cai, and Xiaofei He. „Translating Embeddings for Knowledge Graph Completion with Relation Attention Mechanism.“ In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. Ed. by Jérôme Lang. ijcai.org, 2018, pp. 4286–4292. DOI: 10.24963/ijcai.2018/596. URL: https://doi.org/10.24963/ijcai.2018/596.

[181] Milos Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. „Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data.“ In: *J. Mach. Learn. Res.* 11 (2010), pp. 2487–2531. URL: http://portal.acm.org/citation.cfm?id=1953015.

[182] Nils Reimers and Iryna Gurevych. „Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.“ In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019. DOI: 10.18653/v1/d19-1410.

[183] Hongyu Ren, Weihua Hu, and Jure Leskovec. „Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings.“ In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=BJgr4kSFDS.

[184] Hongyu Ren and Jure Leskovec. „Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs.“ 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.

[185] Martin Ringsquandl, Evgeny Kharlamov, Daria Stepanova, Steffen Lamparter, Raffaello Lepratti, Ian Horrocks, and Peer Kroger. „On event-driven knowledge graph completion in digital factories.“ In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, Dec. 2017. DOI: 10.1109/bigdata.2017.8258105.

[186] Petar Ristoski and Heiko Paulheim. „RDF2Vec: RDF Graph Embeddings for Data Mining.“ In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*. Ed. by Paul Groth, Elena Simperl, Alasdair J. G. Gray, Marta Sabou, Markus Krötzsch, Freddy Lécué, Fabian Flöck, and Yolanda Gil. Vol. 9981. Lecture Notes in Computer Science. 2016, pp. 498–514. DOI: 10.1007/978-3-319-46523-4_30. URL: https://doi.org/10.1007/978-3-319-46523-4%5C_30.

[187] Maya Rotmensch, Yoni Halpern, Abdulhakim Tlimat, Steven Horng, and David Sontag. „Learning a Health Knowledge Graph from Electronic Medical Records.“ In: *Scientific Reports* 7.1 (July 2017). DOI: 10.1038/s41598-017-05778-z.

[188] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. „You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings.“ In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=BkxSmlBFvr.

[189] Alberto Santos, Ana R. Colaço, Annelaura B. Nielsen, Lili Niu, Philipp E. Geyer, Fabian Coscia, Nicolai J Wewer Albrechtsen, Filip Mundt, Lars Juhl Jensen, and Matthias Mann. „Clinical Knowledge Graph Integrates Proteomics Data into Clinical Decision-Making.“ In: (May 2020). DOI: 10.1101/2020.05.09.084897.

[190] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. „The Graph Neural Network Model.“ In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605. URL: https://doi.org/10.1109/TNN.2008.2005605.

[191] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. „Modeling Relational Data with Graph Convolutional Networks.“ In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*. Ed. by Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 593–607. DOI: 10.1007/978-3-319-93417-4_38. URL: https://doi.org/10.1007/978-3-319-93417-4%5C_38.

[192] Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. „An Unsupervised Joint System for Text Generation from Knowledge Graphs and Semantic Parsing.“ In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Association for Computational Linguistics, 2020, pp. 7117–7130. DOI: 10.18653/v1/2020.emnlp-main.577. URL: https://doi.org/10.18653/v1/2020.emnlp-main.577.

[193] Thomas Seidl and Hans-Peter Kriegel. „Optimal Multi-Step k-Nearest Neighbor Search.“ In: *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*. Ed. by Laura M. Haas and Ashutosh Tiwary. ACM Press, 1998, pp. 154–165. DOI: 10.1145/276304.276319. URL: https://doi.org/10.1145/276304.276319.

[194] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. „End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion.“ In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 3060–3067. DOI: 10.1609/aaai.v33i01.33013060. URL: https://doi.org/10.1609/aaai.v33i01.33013060.

[195] Sahand Sharifzadeh, Sina Moayed Baharlou*, **Max Berrendorf***, Rajat Koner, and Volker Tresp. „Improving Visual Relation Detection using Depth Maps.“ In: *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*. * equal contribution. IEEE, 2020, pp. 3597–3604. DOI: 10.1109/ICPR48806.2021.9412945. URL: https://doi.org/10.1109/ICPR48806.2021.9412945.

[196] Baoxu Shi and Tim Weninger. „ProjE: Embedding Projection for Knowledge Graph Completion.“ In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. Ed. by Satinder P. Singh and Shaul Markovitch. AAAI Press, 2017, pp. 1236–1242. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14279.

[197] Xiaofei Shi and Yanghua Xiao. „Modeling Multi-mapping Relations for Precise Cross-lingual Entity Alignment.“ In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Association for Computational Linguistics, 2019, pp. 813–822. DOI: 10.18653/v1/D19-1075. URL: https://doi.org/10.18653/v1/D19-1075.

[198] Saurabh Shrivastava. *Bring rich knowledge of people, places, things and local businesses to your apps*. Bing Blogs. https://blogs.bing.com/search-quality-insights/2017-07/bring-rich-knowledge-of-people-places-things-and-local-businesses-to-your-apps. July 2017.

[199] Amit Singhal. *Introducing the Knowledge Graph: things, not strings*. Google Blog. https://www.blog.google/products/search/introducing-knowledge-graph-things-not/. May 2012.

[200] Richard Sinkhorn and Paul Knopp. „Concerning nonnegative matrices and doubly stochastic matrices.“ In: *Pacific Journal of Mathematics* 21.2 (1967), pp. 343–348.

[201] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. „Reasoning With Neural Tensor Networks for Knowledge Base Completion.“ In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger. 2013, pp. 926–934. URL: https://proceedings.neurips.cc/paper/2013/hash/b337e84de8752b27eda3a12363109e80-Abstract.html.

[202] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. „Highway Networks.“ In: *CoRR* abs/1505.00387 (2015). arXiv: 1505.00387. URL: http://arxiv.org/abs/1505.00387.

[203] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. „Yago: a core of semantic knowledge.“ In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. Ed. by Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy. ACM, 2007, pp. 697–706. DOI: 10.1145/1242572.1242667. URL: https://doi.org/10.1145/1242572.1242667.

[204] Jian Sun, Yu Zhou, and Chengqing Zong. „Dual Attention Network for Cross-lingual Entity Alignment.“ In: *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*. Ed. by Donia Scott, Núria Bel, and Chengqing Zong. International Committee on Computational Linguistics, 2020, pp. 3190–3201. DOI: 10.18653/v1/2020.coling-main.284. URL: https://doi.org/10.18653/v1/2020.coling-main.284.

[205] Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. „Bootstrapping Entity Alignment with Knowledge Graph Embedding.“ In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, July 2018. DOI: 10.24963/ijcai.2018/611.

[206] Zequn Sun, Chengming Wang, Wei Hu, Muhao Chen, Jian Dai, Wei Zhang, and Yuzhong Qu. „Knowledge Graph Alignment Network with Gated Multi-Hop Neighborhood Aggregation.“ In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.01 (Apr. 2020), pp. 222–229. DOI: 10.1609/aaai.v34i01.5354.

[207] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. „RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space.“ In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=HkgEQnRqYQ.

[208] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. „A Re-evaluation of Knowledge Graph Completion Methods.“ In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 5516–5522. DOI: 10.18653/v1/2020.acl-main.489. URL: https://aclanthology.org/2020.acl-main.489.

[209] Andon Tchechmedjiev, Pavlos Fafalios, Katarina Boland, Malo Gasquet, Matthäus Zloch, Benjamin Zapilko, Stefan Dietze, and Konstantin Todorov. „ClaimsKG: A Knowledge Graph of Fact-Checked Claims.“ In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*. Ed. by Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon. Vol. 11779. Lecture Notes in Computer Science. Springer, 2019, pp. 309–324. DOI: 10.1007/978-3-030-30796-7_20. URL: https://doi.org/10.1007/978-3-030-30796-7%5C_20.

[210] Sudhanshu Tiwari, Iti Bansal, and Carlos R. Rivero. „Revisiting the Evaluation Protocol of Knowledge Graph Completion Methods for Link Prediction.“ In: *Proceedings of the Web Conference 2021*. ACM, Apr. 2021. DOI: 10.1145/3442381.3449856.

[211] Felice Tobin. *Thomson Reuters Launches first of its kind Knowledge Graph Feed allowing Financial Services customers to accelerate their AI and Digital Strategies.* Thomspon Reuters Press Release. https://www.thomsonreuters.com/en/press-releases/2017/october/thomson-reuters-launches-first-of-its-kind-knowledge-graph-feed.html. Oct. 2017.

[212] Kristina Toutanova and Danqi Chen. „Observed versus latent features for knowledge base and text inference." In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 57–66. DOI: 10.18653/v1/W15-4007. URL: https://aclanthology.org/W15-4007.

[213] Bayu Distiawan Trisedya, Jianzhong Qi, and Rui Zhang. „Entity Alignment between Knowledge Graphs Using Attribute Embeddings." In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 297–304. DOI: 10.1609/aaai.v33i01.3301297. URL: https://doi.org/10.1609/aaai.v33i01.3301297.

[214] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. „Complex Embeddings for Simple Link Prediction." In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 2071–2080. URL: http://proceedings.mlr.press/v48/trouillon16.html.

[215] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha P. Talukdar. „InteractE: Improving Convolution-Based Knowledge Graph Embeddings by Increasing Feature Interactions." In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 3009–3016. URL: https://aaai.org/ojs/index.php/AAAI/article/view/5694.

[216] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. „Composition-based Multi-Relational Graph Convolutional Networks." In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=BylA%5C_C4tPr.

[217] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. „Attention is All you Need." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna

M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[218] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. „Graph Attention Networks.“ In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=rJXMpikCZ.

[219] Denny Vrandecic and Markus Krötzsch. „Wikidata: a free collaborative knowledgebase.“ In: *Commun. ACM* 57.10 (2014), pp. 78–85. DOI: 10.1145/2629489. URL: https://doi.org/10.1145/2629489.

[220] Brian Walsh, Sameh K. Mohamed, and Vit Novácek. „BioKG: A Knowledge Graph for Relational Learning On Biological Data.“ In: *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. Ed. by Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux. ACM, 2020, pp. 3173–3180. DOI: 10.1145/3340531.3412776. URL: https://doi.org/10.1145/3340531.3412776.

[221] Chenguang Wang, Xiao Liu, and Dawn Song. „Language Models are Open Knowledge Graphs.“ In: *CoRR* abs/2010.11967 (2020). arXiv: 2010.11967. URL: https://arxiv.org/abs/2010.11967.

[222] 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. DOI: 10.1109/TKDE.2017.2754499.

[223] Ruijie Wang, Yuchen Yan, Jialu Wang, Yuting Jia, Ye Zhang, Weinan Zhang, and Xinbing Wang. „AceKG: A Large-scale Knowledge Graph for Academic Data Mining.“ In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. Ed. by Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang. ACM, 2018, pp. 1487–1490. DOI: 10.1145/3269206.3269252. URL: https://doi.org/10.1145/3269206.3269252.

[224] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. „Knowledge Graph Embedding by Translating on Hyperplanes.“ In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. Ed. by Carla E. Brodley and Peter Stone. AAAI Press, 2014, pp. 1112–1119. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[225] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. „Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks.“ In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff, David Chiang, Julia

Hockenmaier, and Jun'ichi Tsujii. Association for Computational Linguistics, 2018, pp. 349–357. DOI: 10.18653/v1/d18-1032. URL: https://doi.org/10.18653/v1/d18-1032.

[226] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. „Knowledge base completion via search-based question answering." In: *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*. Ed. by Chin-Wan Chung, Andrei Z. Broder, Kyuseok Shim, and Torsten Suel. ACM, 2014, pp. 515–526. DOI: 10.1145/2566486.2568032. URL: https://doi.org/10.1145/2566486.2568032.

[227] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. „Relation-Aware Entity Alignment for Heterogeneous Knowledge Graphs." In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 5278–5284. DOI: 10.24963/ijcai.2019/733. URL: https://doi.org/10.24963/ijcai.2019/733.

[228] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. „Jointly Learning Entity and Relation Representations for Entity Alignment." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Association for Computational Linguistics, 2019, pp. 240–249. DOI: 10.18653/v1/D19-1023. URL: https://doi.org/10.18653/v1/D19-1023.

[229] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. „Neighborhood Matching Network for Entity Alignment." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault. Association for Computational Linguistics, 2020, pp. 6477–6487. DOI: 10.18653/v1/2020.acl-main.578. URL: https://doi.org/10.18653/v1/2020.acl-main.578.

[230] Yuejia Xiang, Ziheng Zhang, Jiaoyan Chen, Xi Chen, Zhenxi Lin, and Yefeng Zheng. „OntoEA: Ontology-guided Entity Alignment via Joint Knowledge Graph Embedding." In: *CoRR* abs/2105.07688 (2021). arXiv: 2105.07688. URL: https://arxiv.org/abs/2105.07688.

[231] Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. „TransA: An Adaptive Approach for Knowledge Graph Embedding." In: *CoRR* abs/1509.05490 (2015). arXiv: 1509.05490. URL: http://arxiv.org/abs/1509.05490.

[232] Han Xiao, Minlie Huang, and Xiaoyan Zhu. „From One Point to a Manifold: Knowledge Graph Embedding for Precise Link Prediction." In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016,*

New York, NY, USA, 9-15 July 2016. Ed. by Subbarao Kambhampati. IJCAI/AAAI Press, 2016, pp. 1315–1321. URL: http://www.ijcai.org/Abstract/16/190.

[233] Han Xiao, Minlie Huang, and Xiaoyan Zhu. „TransG : A Generative Model for Knowledge Graph Embedding.“ In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. DOI: 10.18653/v1/p16-1219. URL: https://doi.org/10.18653/v1/p16-1219.

[234] Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard H. Hovy. „An Interpretable Knowledge Transfer Model for Knowledge Base Completion.“ In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Ed. by Regina Barzilay and Min-Yen Kan. Association for Computational Linguistics, 2017, pp. 950–962. DOI: 10.18653/v1/P17-1088. URL: https://doi.org/10.18653/v1/P17-1088.

[235] Zhiwen Xie, Runjie Zhu, Kunsong Zhao, Jin Liu, Guangyou Zhou, and Jimmy Xiangji Huang. „A Contextual Alignment Enhanced Cross Graph Attention Network for Cross-lingual Entity Alignment.“ In: *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*. Ed. by Donia Scott, Núria Bel, and Chengqing Zong. International Committee on Computational Linguistics, 2020, pp. 5918–5928. DOI: 10.18653/v1/2020.coling-main.520. URL: https://doi.org/10.18653/v1/2020.coling-main.520.

[236] Canran Xu and Ruijiang Li. „Relation Embedding with Dihedral Group in Knowledge Graph.“ In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluis Màrquez. Association for Computational Linguistics, 2019, pp. 263–272. DOI: 10.18653/v1/p19-1026. URL: https://doi.org/10.18653/v1/p19-1026.

[237] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. „How Powerful are Graph Neural Networks?“ In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km.

[238] Kun Xu, Linfeng Song, Yansong Feng, Yan Song, and Dong Yu. „Coordinated Reasoning for Cross-Lingual Knowledge Graph Alignment.“ In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 9354–9361. URL: https://aaai.org/ojs/index.php/AAAI/article/view/6476.

[239] Kun Xu, Liwei Wang, Mo Yu, Yansong Feng, Yan Song, Zhiguo Wang, and Dong Yu. „Cross-lingual Knowledge Graph Alignment via Graph Matching Neural Network.“ In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluis Màrquez. Association for Computational Linguistics, 2019, pp. 3156–3161. DOI: 10.18653/v1/p19-1304. URL: https://doi.org/10.18653/v1/p19-1304.

[240] Yexiang Xue, Yang Yuan, Zhitian Xu, and Ashish Sabharwal. „Expanding Holographic Embeddings for Knowledge Completion.“ In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 4496–4506. URL: https://proceedings.neurips.cc/paper/2018/hash/dd28e50635038e9cf3a648c2dd17ad0a-Abstract.html.

[241] Vincent K. C. Yan, Xiaodong Li, Xuxiao Ye, Min Ou, Ruibang Luo, Qingpeng Zhang, Bo Tang, Benjamin J. Cowling, Ivan Hung, Chung Wah Siu, Ian C. K. Wong, Reynold C. K. Cheng, and Esther W. Chan. „Drug Repurposing for the Treatment of COVID-19: A Knowledge Graph Approach.“ In: *Advanced Therapeutics* (May 2021), p. 2100055. DOI: 10.1002/adtp.202100055.

[242] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. „Embedding Entities and Relations for Learning and Inference in Knowledge Bases.“ In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6575.

[243] Hsiu-Wei Yang, Yanyan Zou, Peng Shi, Wei Lu, Jimmy Lin, and Xu Sun. „Aligning Cross-Lingual Entities with Multi-Aspect Information.“ In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Association for Computational Linguistics, 2019, pp. 4430–4440. DOI: 10.18653/v1/D19-1451. URL: https://doi.org/10.18653/v1/D19-1451.

[244] Jinzhu Yang, Wei Zhou, Lingwei Wei, Junyu Lin, Jizhong Han, and Songlin Hu. „RE-GCN: Relation Enhanced Graph Convolutional Network for Entity Alignment in Heterogeneous Knowledge Graphs.“ In: *Database Systems for Advanced Applications - 25th International Conference, DASFAA 2020, Jeju, South Korea, September 24-27, 2020, Proceedings, Part II*. Ed. by Yunmook Nah, Bin Cui, Sang-Won Lee, Jeffrey Xu Yu, Yang-Sae Moon, and Steven Euijong Whang. Vol. 12113. Lecture Notes in Computer Science. Springer, 2020, pp. 432–447. DOI: 10.1007/978-3-030-59416-9_26.

[245] Kai Yang, Shaoqin Liu, Junfeng Zhao, Yasha Wang, and Bing Xie. „COTSAE: CO-Training of Structure and Attribute Embeddings for Entity Alignment.“ In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 3025–3032. URL: https://aaai.org/ojs/index.php/AAAI/article/view/5696.

[246] Shihui Yang, Jidong Tian, Honglun Zhang, Junchi Yan, Hao He, and Yaohui Jin. „TransMS: Knowledge Graph Embedding for Complex Relations by Multidirectional Semantics.“ In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 1935–1942. DOI: 10.24963/ijcai.2019/268. URL: https://doi.org/10.24963/ijcai.2019/268.

[247] Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. „A Vectorized Relational Graph Convolutional Network for Multi-Relational Network Alignment.“ In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 4135–4141. DOI: 10.24963/ijcai.2019/574. URL: https://doi.org/10.24963/ijcai.2019/574.

[248] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. „Deep Sets.“ In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 3391–3401. URL: https://proceedings.neurips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html.

[249] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. „Neural Motifs: Scene Graph Parsing with Global Context.“ In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00611.

[250] Weixin Zeng, Xiang Zhao, Jiuyang Tang, Xinyi Li, Minnan Luo, and Qinghua Zheng. „Towards Entity Alignment in the Open World: An Unsupervised Approach.“ In: *Database Systems for Advanced Applications - 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11-14, 2021, Proceedings, Part I*. Ed. by Christian S. Jensen, Ee-Peng Lim, De-Nian Yang, Wang-Chien Lee, Vincent S. Tseng, Vana Kalogeraki, Jen-Wei Huang, and Chih-Ya Shen. Vol. 12681. Lecture Notes in Computer Science. Springer, 2021, pp. 272–289. DOI: 10.1007/978-3-030-73194-6_19. URL: https://doi.org/10.1007/978-3-030-73194-6%5C_19.

[251] Weixin Zeng, Xiang Zhao, Jiuyang Tang, and Xuemin Lin. „Collective Entity Alignment via Adaptive Features.“ In: *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 2020,

pp. 1870–1873. DOI: 10.1109/ICDE48307.2020.00191. URL: https://doi.org/10.1109/ICDE48307.2020.00191.

[252] Weixin Zeng, Xiang Zhao, Wei Wang, Jiuyang Tang, and Zhen Tan. „Degree-Aware Alignment for Entities in Tail.“ In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. Ed. by Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu. ACM, 2020, pp. 811–820. DOI: 10.1145/3397271.3401161. URL: https://doi.org/10.1145/3397271.3401161.

[253] Qingheng Zhang, Zequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. „Multi-view Knowledge Graph Embedding for Entity Alignment.“ In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Aug. 2019. DOI: 10.24963/ijcai.2019/754.

[254] Rui Zhang, Dimitar Hristovski, Dalton Schutte, Andrej Kastrin, Marcelo Fiszman, and Halil Kilicoglu. „Drug repurposing for COVID-19 via knowledge graph completion.“ In: *Journal of Biomedical Informatics* 115 (Mar. 2021), p. 103696. DOI: 10.1016/j.jbi.2021.103696.

[255] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. „Quaternion Knowledge Graph Embeddings.“ In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 2731–2741. URL: https://proceedings.neurips.cc/paper/2019/hash/d961e9f236177d65d21100592edb0769-Abstract.html.

[256] Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. „Interaction Embeddings for Prediction and Explanation in Knowledge Graphs.“ In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*. Ed. by J. Shane Culpepper, Alistair Moffat, Paul N. Bennett, and Kristina Lerman. ACM, 2019, pp. 96–104. DOI: 10.1145/3289600.3291014. URL: https://doi.org/10.1145/3289600.3291014.

[257] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. „Variational Reasoning for Question Answering With Knowledge Graph.“ In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 6069–6076. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16983.

[258] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. „Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction.“ In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 3065–3072. URL: https://aaai.org/ojs/index.php/AAAI/article/view/5701.

[259] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. „ERNIE: Enhanced Language Representation with Informative Entities.“ In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluis Màrquez. Association for Computational Linguistics, 2019, pp. 1441–1451. DOI: 10.18653/v1/p19-1139. URL: https://doi.org/10.18653/v1/p19-1139.

[260] Zhiyuan Zhang, Xiaoqian Liu, Yi Zhang, Qi Su, Xu Sun, and Bin He. „Pretrain-KGE: Learning Knowledge Representation from Pretrained Language Models.“ In: *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*. Ed. by Trevor Cohn, Yulan He, and Yang Liu. Vol. EMNLP 2020. Findings of ACL. Association for Computational Linguistics, 2020, pp. 259–266. DOI: 10.18653/v1/2020.findings-emnlp.25. URL: https://doi.org/10.18653/v1/2020.findings-emnlp.25.

[261] Shuangjia Zheng, Jiahua Rao, Ying Song, Jixian Zhang, Xianglu Xiao, Evandro Fei Fang, Yuedong Yang, and Zhangming Niu. „PharmKG: a dedicated knowledge graph benchmark for bomedical data mining.“ In: *Briefings in Bioinformatics* (Dec. 2020). DOI: 10.1093/bib/bbaa344.

[262] Bin Zhou, Jinsong Bao, Jie Li, Yuqian Lu, Tianyuan Liu, and Qiwan Zhang. „A novel knowledge graph-based optimization approach for resource allocation in discrete manufacturing workshops.“ In: *Robotics and Computer-Integrated Manufacturing* 71 (Oct. 2021), p. 102160. DOI: 10.1016/j.rcim.2021.102160.

[263] Xiaofei Zhou, Qiannan Zhu, Ping Liu, and Li Guo. „Learning Knowledge Embeddings by Combining Limit-based Scoring Loss.“ In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. Ed. by Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li. ACM, 2017, pp. 1009–1018. DOI: 10.1145/3132847.3132939.

[264] Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. „Iterative Entity Alignment via Joint Knowledge Embeddings.“ In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Aug. 2017. DOI: 10.24963/ijcai.2017/595.

[265] Qi Zhu, Hao Wei, Bunyamin Sisman, Da Zheng, Christos Faloutsos, Xin Luna Dong, and Jiawei Han. „Collective Multi-type Entity Alignment Between Knowledge Graphs.“ In: *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. Ed. by Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen. ACM / IW3C2, 2020, pp. 2241–2252. DOI: 10.1145/3366423.3380289.

[266] Qiannan Zhu, Xiaofei Zhou, Jia Wu, Jianlong Tan, and Li Guo. „Neighborhood-Aware Attentional Representation for Multilingual Knowledge Graphs.“ In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Aug. 2019. DOI: 10.24963/ijcai.2019/269.

[267] Renbo Zhu, Meng Ma, and Ping Wang. „RAGA: Relation-Aware Graph Attention Networks for Global Entity Alignment.“ In: *Advances in Knowledge Discovery and Data Mining - 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11-14, 2021, Proceedings, Part I*. Ed. by Kamal Karlapalem, Hong Cheng, Naren Ramakrishnan, R. K. Agrawal, P. Krishna Reddy, Jaideep Srivastava, and Tanmoy Chakraborty. Vol. 12712. Lecture Notes in Computer Science. Springer, 2021, pp. 501–513. DOI: 10.1007/978-3-030-75762-5_40. URL: https://doi.org/10.1007/978-3-030-75762-5%5C_40.

[268] Yao Zhu, Hongzhi Liu, Zhonghai Wu, and Yingpeng Du. „Relation-Aware Neighborhood Matching Model for Entity Alignment.“ In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 4749–4756. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16606.