electronics

MDPI

Article

Comprehensive Analysis of Knowledge Graph Embedding Techniques Benchmarked on Link Prediction

Ilaria Ferrari †, Giacomo Frisoni *,†, Paolo Italiani †, Gianluca Moro † and Claudio Sartori †

Department of Computer Science and Engineering (DISI), University of Bologna, Via dell'Università 50, I-47522 Cesena, Italy

* Correspondence: giacomo.frisoni@unibo.it

† All authors contributed equally to this paper.

Abstract: In knowledge graph representation learning, link prediction is among the most popular and influential tasks. Its surge in popularity has resulted in a panoply of orthogonal embedding-based methods projecting entities and relations into low-dimensional continuous vectors. To further enrich the research space, the community witnessed a prolific development of evaluation benchmarks with a variety of structures and domains. Therefore, researchers and practitioners face an unprecedented challenge in effectively identifying the best solution to their needs. To this end, we propose the most comprehensive and up-to-date study to systematically assess the effectiveness and efficiency of embedding models for knowledge graph completion. We compare 13 models on six datasets with different sizes, domains, and relational properties, covering translational, semantic matching, and neural network-based encoders. A fine-grained evaluation is conducted to compare each technique head-to-head in terms of standard metrics, training and evaluation times, memory consumption, carbon footprint, and space geometry. Our results demonstrate the high dependence between performance and graph types, identifying the best options for each scenario. Among all the encoding strategies, the new generation of translational models emerges as the most promising, bringing out the best and most consistent results across all the datasets and evaluation criteria.

Keywords: link prediction; knowledge graphs; knowledge graph completion; knowledge graph representation learning; graph neural networks; translational models

Citation: Ferrari, I.; Frisoni, G.; Italiani, P.; Moro, G.; Sartori, C. Comprehensive Analysis of Knowledge Graph Embedding Techniques Benchmarked on Link Prediction. *Electronics* **2022**, *11*, 3866. https://doi.org/10.3390/electronics11233866

Academic Editor: Stefanos Kollias

Received: 8 October 2022
Accepted: 15 November 2022
Published: 23 November 2022

1. Introduction

Knowledge graphs (KGs) have become a ubiquitous framework to represent entities and the relations that connect them. KGs provide a concise and intuitive abstraction for a variety of domains, where edges capture semantic relations between the entities inherent in social data, biological interactions, bibliographical citations and co-authorships, transport networks, etc. We consider the definition proposed in [1]:

> *A knowledge graph is a multi-relational graph composed of entities and relations that are regarded as nodes and different types of edges, respectively.*

More specifically, a KG is a set of resource description framework (RDF) triples, where each triple $(h, r, t) \in \mathbb{G}$ portrays a fact of the world, with h representing the head entity (or subject), r the relation (or predicate), and t the tail entity (or object).

At present, many open KGs are available online: among the most prominent ones, we have DBpedia [2], Freebase [3], Wikidata [4], and YAGO [5]. Despite their seemingly huge size, these knowledge bases are greatly incomplete. Since most knowledge bases are constructed manually or semi-automatically, a large number of implicit entities and relationships have yet to be discovered. For example, over 70% of people included in Freebase have no known birthplace, 99% have no known ethnicity, and 95% have no information on their parents [6]; in DBpedia, 60% of scientists are lacking a study area [7].

Accordingly, techniques exist to refine the KG, and they typically target improvements of the local knowledge without ingesting external content. Link prediction (LP) is one of the most common refinement (KG augmentation) solutions exploiting the existing facts in a KG to infer missing ones. As shown in Figure 1, LP infers the missing entity that completes an RDF triple (*h*, *r*, *?*) (tail prediction), (*?*, *r*, *t*) (head prediction), or predicts missing relations (*h*, *?*, *t*).

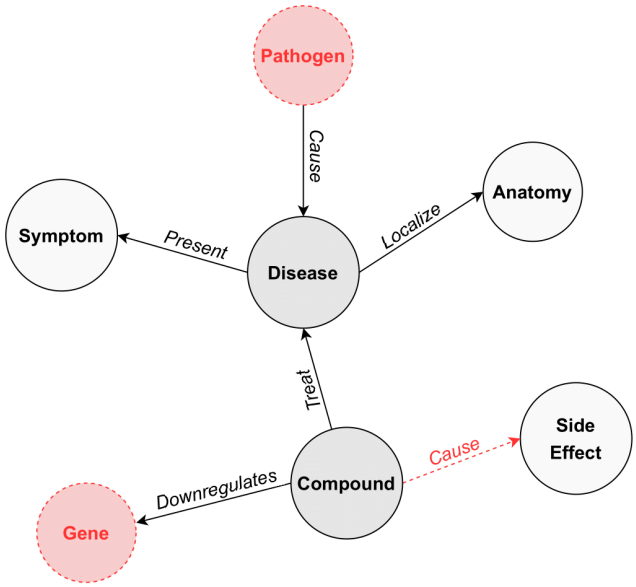


Figure 1. Example of link prediction with a missing head (*?*, *Cause*, *Disease*), tail (*Compound*, *Downgrades*, *?*), and relation (*Compound*, *?*, *Side Effect*). Red components denote potential relationships that may extend the knowledge graph completeness.

For instance, LP has been proven successful in drug discovery [8], where it is required to predict the interactions or missing links between drugs and their targets, including diseases, proteins, or other medicines. Question answering is another application worth mentioning, which has been substantially impacted by LP [9], including in natural language understanding contexts [10].

Although effective in representing structured data, the symbolic nature of triples makes KGs difficult to manipulate. For this reason, most LP models use KG elements first to learn low-dimensional representations—dubbed KG embeddings—and then employ them to mine new facts (i.e., the links between them). Versatile continuous representations can effectively solve computational complexity and data sparsity problems. In a few years, the community developed 150+ encoding techniques for KGs [11,12], outlining a complex solution landscape that still needs to be well-mapped and deciphered. Mechanisms in these models and the embedding vectors themselves vary greatly, making it challenging to comprehend and compare them. Current solutions differ in the core approach, supported input graphs, pattern modeling, dense space, additional information awareness, and complexity. Faced with this variety of choices, it is increasingly difficult for a practitioner to select the approach that best suits their needs, risking making the preference fall on common encoders rather than on alternatives supported by solid reasons. With the continuous development of new solutions, the current literature needs help to navigate individuals through trustworthy evidence on which method to adopt in a particular context. Our research aims to shed light on this issue, proposing the most comprehensive and up-to-date study to systematically assess the effectiveness and efficiency of embedding models for KG completion. In summary:

- We compare 13 highly representative encoders belonging to diverse taxonomical families. Following the success of KG representation learning in encoding structural

information, such models map entities and relations into a dense space following various strategies, namely translation, semantic matching, and neural network-based.
- We consider six of the most commonly employed datasets taken from four KGs with distinct relational properties.
- We evaluate predictions using a panoply of metrics and criteria, including mean rank, mean reciprocal rank, Hits@n, parameter number, training/evaluation time, and memory consumption.
- Besides considering more quality dimensions, we conduct new experiments for those techniques and datasets not yet covered in the literature, also offering hints in terms of software libraries for practical implementation—an aspect never dealt with by previous works.
- We publicly release all code, datasets, and hyperparameters to ensure maximum reproducibility and foster future research in this area.

The rest of the paper is organized as follows. Section 2 reviews related works and their differences compared to ours. Section 3 describes inspected models and datasets, also examining available software libraries and their characteristics. Section 4 details the experimental setup, from hardware settings to evaluation metrics. Section 5 presents all the results obtained with our analysis. Section 6 interprets the findings with respect to previous investigations. Finally, Section 7 concludes the discussion, outlines limitations, and outlines directions for future study.

2. Literature Survey

Some papers have already tried to delve into the broad understanding and breakdown of the emerged KG embedding methods.

Sharma et al. [13] scrutinized the KG encoding geometry (i.e., arrangement, length, and conicity of entity and relation vectors) and its correlation with task performance, model type, and hyperparameters.

Akrami et al. [14] re-assessed the effectiveness of KG completion methods by removing unrealistic triples. They proved the inadequacy of benchmarks where reverse, duplicated, and Cartesian product relations act as data leakage and redundancy, thus inflating the models' accuracy and rendering an LP scenario largely nonexistent in the real world.

Kadlec et al. [15] casted doubt on the claim that the performance improvements of recent models are due to architectural changes rather than different hyperparameters or training objectives. In particular, the authors showed that an appropriately tuned baseline (DistMult) can outperform most of its successors, suggesting a huge space for improvement even for the more complex models.

Tran et al. [16] analyzed KG encoding methods from an original perspective, representing each entity and relation by n multi-embedding vectors instead of single ones. Therefore, the triple (h, r, t) is represented by $\boldsymbol{h}^{(i)}, \boldsymbol{r}^{(k)}$ and $\boldsymbol{t}^{(j)}$, with $i, j, k \in \{1, \ldots, n\}$; all embedding vectors of h, t, and r interact with each other by trilinear products. They also propose a four-embedding model based on quaternion algebra.

At present, many reviews [1,11,17–21] have proposed thorough theoretically centered dissertations. Although testifying to the upsurge in KG embedding activities, they do not really answer which approach is preferable in practice. Such authors condense mainstream methods, findings, evaluation indicators, applications, and open problems but totally ignore the results on standard and diversified datasets. Instead, we conduct several experiments to broadly contrast the performance of encoding models, including unexplored combinations and original quality dimensions.

Most related to our work are systematic comparisons of KG embedding models for LP [12,22–24]. Lin et al. [22] first offered a quantitative review of KG embedding models in three knowledge acquisition tasks, including LP. Naturally, they only contemplated basic techniques, which were mainly transitional. Testbed datasets are poor, and judgment only counts on standard metrics. Rossi et al. [23] selected standard models for juxtaposed depiction and experimental analysis on several datasets. Nonetheless, they did not wrap

all the encoding families, remaining no longer updated with the latest techniques proposed in the literature and particularly lacking on the graph neural network (GNN) front. Other than LP quantitative metrics, they only measured the training and prediction times. Furthermore, they did not conduct experiments utilizing a unified environment configuration but adopted different coding frameworks, such as Python and C++. Wang et al. [12] point-by-point dissected more heterogeneous encoding techniques within a unified Pytorch environment, also considering the incorporation of additional information. In contrast, selected models were few, and tested benchmarks were only two and derived from the same KG. Zamini et al. [24] scanned a high number of recent and unconventional KG completion approaches, remarkably comprising 5+ GNN-based encoder–decoder architectures. On the other side, the final study appears weak and superficial, with LP results exclusively originating from previous publications and formed on two datasets only. Missing model-dataset values—merely based on two quantitative metrics—are not filled out through custom experiments. Importantly, we note some non-motivated discrepancies with original sources. Ultimately, as emphasized in Table 1, all the literature efforts mentioned above form an incomplete recommendation of what a practitioner should effectively use to guide their choices. It is unclear how they fit together and what valuable research directions one should pursue.

Taking one step forward, we extensively analyze the effectiveness and efficiency of 13 well-established KG embedding models for LP on six general and domain-specific open benchmarks. Our fine-grained inquiry is not limited to conservative metrics but also incorporates time/memory analysis, number of parameters, carbon footprint, and clustering visualization. All the experiments are conducted in a unified PyTorch environment, publicly releasing all code and hyperparameters for maximum reproducibility. We additionally provide hints about available software libraries and software ecosystems, a dimension poorly documented but vital for new practitioners. To the best of our knowledge, no past research article tackles the same variety of model categories (including GNNs), datasets, and assessment criteria. For each embedding approach, we trace the strengths and weaknesses in handling different real-world challenges. We hope our work can serve as a stepping stone to push research awareness and the proposal of new cutting-edge technologies.

Table 1. Model families, datasets, and evaluation criteria explored by the surveys most relevant to our work. Blue values in brackets indicate the number of methods for each family. * = except for results taken from previous papers.

	Model Families				**Datasets**						**Evaluation**						
	Translation	**Semantic Matching**	**CNN, RNN, Capsule**	**GNN**	**WN18 [25]**	**WN18RR [25]**	**FB15K [3]**	**FB15K-237 [3]**	**YAGO3-10 [5]**	**OGB-BioKG [26]**	**LP Metrics**	**Time Analysis**	**Memory Analysis**	**Carbon Footprint**	**Qualitative Analysis**	**Unified Dev Framework**	**Public Code**
Lin et al. [23]	✓(x6)	✓(x3)	✓(x1)		✓		✓				✓				✓	✓	
Rossi et al. [23]	✓(x5)	✓(x6)	✓(x5)		✓	✓	✓	✓	✓		✓	✓					✓
Wang et al. [12]	✓(x2)	✓(x6)	✓(x2)	✓(x1)			✓	✓			✓	✓				✓	
Zamini et al. [24]	✓(x8)	✓(x3)	✓(x8)	✓(x11)		✓		✓			✓						
Ours	✓(x6)	✓(x2)	✓(x2)	✓(x3)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓*	✓

3. Materials and Methods

3.1. Relation Patterns

To facilitate our understanding of the models described in Section 3.2, we formally define the possible relational patterns as a premise.

- *Symmetry*: a relation r is symmetric if $\forall(h,r,t) \in \mathbb{G} \implies (t,r,h) \in \mathbb{G}$; e.g., "marriage".
- *Antisymmetry*: a relation r is antisymmetric if $\forall(h,r,t) \in \mathbb{G} \implies (t,r,h) \notin \mathbb{G}$; e.g., "capital of".

- *Inversion*: a relation r_1 is inverse to relation r_2 if $\forall(h, r_1, t) \in \mathbb{G} \implies (t, r_2, h) \in \mathbb{G}$; e.g.,"has seller" and "is a seller for".
- *Composition*: a relation r_1 is composed of relation r_2 and relation r_3 if $\forall(x, r_2, y) \in \mathbb{G} \wedge (y, r_3, z) \in \mathbb{G} \implies (x, r_1, z) \in \mathbb{G}$; e.g., "my niece is my sister's daughter".

3.2. Models

A typical KG embedding technique generally consists of three steps: (i) defining the representation space, (ii) defining a scoring function, and (iii) learning representations. The first step specifies the form in which entities and relations are modeled in a continuous vector space. In the second step, a scoring function $f_r(h, t)$ is designated to measure the plausibility of each fact (h, r, t) and distinguish the correct triples from incorrect ones. In fact, since a KG contains only positive triples, negative triples are generated and added to datasets through corruption. Finally, the third step progressively updates entity and relation embeddings by solving an optimization problem that maximizes the total plausibility of observed facts.

This section delves into the functioning of all the implemented and tested KG encoders, sorted by increasing publication date. We categorized such techniques into three groups: translational, semantic matching, and neural network-based models. The complete taxonomy and timeline are delineated in Figures 2 and 3.

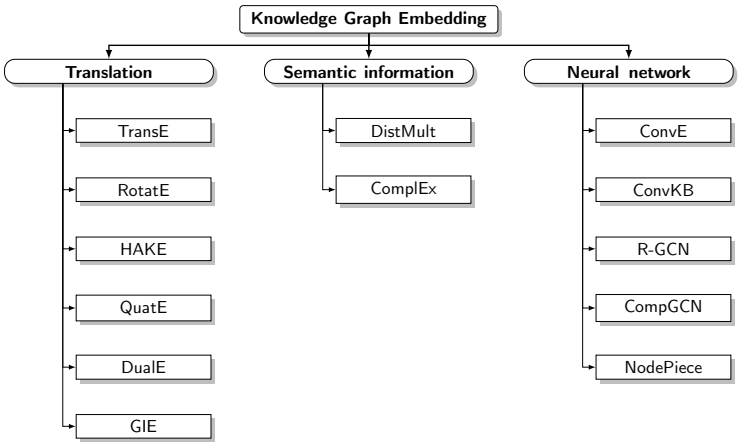


Figure 2. Taxonomy of the reviewed literature and analyzed KG embedding methods [27–39].

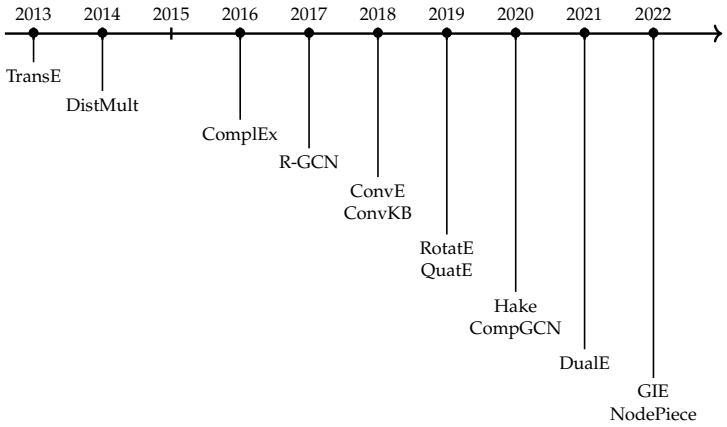


Figure 3. Publication timeline [27–39].

3.2.1. Translation-Based Models

Translational models—depicted in Figure 4—use distance-based scoring functions, treating the process of finding valid triples as translations of entities through relations. They are termed *"additive"*, as the vectors interact via addition and subtraction during

training. The contributions belonging to this class are generally simple and benefit from solid interpretability.

TransE [27] was the first and remains the most classic translational model. It represents entities and relations as vectors of the same Euclidean space $\mathbb{R}^d$, where a relation r is interpreted as a translation vector that connects the head entity h to the tail entity t with low error, $h + r \approx t$. The intuition comes from enforcing regularities observed in linguistic word2vec embeddings [40], where we can obtain $king + woman \approx queen + man$. The authenticity of the given triple is computed via a score function that is defined as:

$$f_r(h,t) = - \parallel h + r - t \parallel_{(l_1/l_2)}, \tag{1}$$

where l_1/l_2 are the norm constraints. Entity and relation vectors are learned through pairwise ranking loss. Given its simplicity and the small number of parameters, TransE is efficient and scalable but fails to model complex relations, such as one-to-many, many-to-one, many-to-many, and symmetric linkages. Moreover, it reckons on the Euclidean distance metric and thus assigns the same weight to each feature vector, resulting in poor flexibility since the knowledge representation accuracy may be affected by irrelevant dimensions. An effective strategy to overcome the disadvantages of TransE is that of allowing an entity to have distinct representations when involved in diverse relations. This intuition led to many researchers playing a part in the Trans series, in which improved models encompass TransH [41] (with relation-specific hyperplanes), TransR [42] (with relation-specific vector spaces), and TransD [43] (with different vectors for representing the projection of head and tail entities). We only keep TransE as the most representative one.

In non-symmetric relations, we cannot interchange subject and object entities, which requires having different embeddings (an increased number of parameters). Complex embeddings facilitate the joint learning of subject and object vectors while preserving the asymmetry of the relation. Following this line, RotatE [28] models entities and relations in the complex vector space $\mathbb{C}^d$ and defines each relation as an element-wise rotation from the head entity to the tail entity. The score function is formulated as below:

$$f_r(h,t) = - \parallel h \circ r - t \parallel, \tag{2}$$

where $\circ$ is the Hadamard (or element-wise) product. RotatE is scalable and linear in time and memory. It can infer various relational patterns, including symmetry, antisymmetry, inversion, and composition, but it fails in modeling hierarchy.

QuatE [30] extends the complex space into a four-dimensional hyper-complex space, $\mathbb{H}^d$. This model uses the Hamilton product ($\otimes$) to capture latent inter-dependencies and acquires a more meaningful rotational ability than RotatE. Indeed, it uses two rotating planes instead of one to preserve symmetric/antisymmetric, flipped, and combined relations. In practice, it unifies RotatE and ComplEx (explained in Section 3.2.2). The score function is described as follows:

$$f_r(h,t) = \parallel h \otimes \frac{r}{\mid r \mid} - t \parallel . \tag{3}$$

Hierarchy-aware knowledge graph embedding (HAKE) [29] models a semantic hierarchy of entities by polar coordinates, in which concentric circles can naturally reflect the order. This is divided into two parts: the modulus part, which aims to distinguish the different hierarchy levels; and the phase part, which seeks to determine entities at the same hierarchy level. The score function is defined by joining the modulus part (m) and the phase part (p):

$$f_r(h,t) = - \parallel h_m \circ r_m - t_m \parallel_2 - \gamma \parallel \sin(h_p + r_p - t_p)/2 \parallel_1, \tag{4}$$

where $\circ$ is the Hadamard product and $sin(\cdot)$ is an operation that applies the sine function to each element of the input.

DualE [31] aims to join the strengths of TransE, RotatE, and QuatE using the dual quaternion, which permits representing both translation and rotation. The entities and relations are embedded in a dual quaternion, which behaves like a "complex quaternion" with both real and imaginary parts that are quaternary. It can be placed in a translation-based model because the score function measures the distance between the head entity and the tail entity once the first is translated and rotated using the relation

$$f_r(h,t) = \langle h \otimes \frac{r}{|\, r \,|}, t \rangle, \tag{5}$$

where $\otimes$ denotes the Hamilton product and $\langle \cdot, \cdot \rangle$ defines the dual quaternion inner product. This model permits the inference of many relational patterns, including symmetry, antisymmetry, inversion, composition, and multiple relations.

Geometry interaction knowledge graph embedding (GIE) [32] learns chain, hierarchy, and ring spatial structures interactively between the Euclidean, hyperbolic, and hyperspherical spaces. The geometry interaction is essential for capturing the reliable structure of space and can be summarized in three steps. First, spatial embeddings are generated for the head entity h, including an embedding E_h in the Euclidean space, an embedding H_h in hyperbolic space, and an embedding S_h in hypersphere space. Second, H_h and S_h are mapped to the tangent space through a logarithmic map to facilitate geometry information propagation, which is necessary to pass information from one space to another. An attention mechanism aggregates the geometric message from Euclidean and tangent spaces. The interaction can be summarized as follows:

$$\text{Inter}(E_h, H_h, S_h) = \exp_0^c(\lambda_E E_h + \lambda_H \log_0^v(H_h)) + \lambda_S \log_0^u(S_h)), \tag{6}$$

where λ is an attention vector and $(\lambda_E, \lambda_H, \lambda_S) = \text{Softmax}(\lambda^T E_h, \lambda^T H_h, \lambda^T S_h)$. Here, we apply $\log_0^v(\cdot)$ and $\log_0^u(\cdot)$ to map H_h and S_h into the tangent space and propagate geometry information. Moreover, applying $\exp_0^c(\cdot)$ aims to form an approximate spatial structure for the interactive space.

The same procedure is used to integrate the information for the tail entity t, then it is possible to score the triple (h, r, t).

$$f_r(h,t) = -(d_c(\text{Inter}(E_h, H_h, S_h), t) + d_c(\text{Inter}(E_t, H_t, S_t), h)) + b, \tag{7}$$

where $Inter(\cdot)$ is defined in Equation (6), $d_c(\cdot, \cdot)$ represents a distance function and b is the bias which acts as a margin. GIE is capable of modeling many inference patterns, including symmetry, antisymmetry, inversion, composition, hierarchy, intersection, and mutual exclusion.

3.2.2. Semantic Matching-Based Models

Semantic matching models exploit similarity-based scoring functions (Figure 5). These are called *"multiplicative"* as the triple likelihood is quantified using the multiplication operator. The plausibility is measured by matching the latent semantics of entities and relations embodied in their vector space representations. In particular, for the models presented below, the scoring function is computed as a bilinear product:

$$f_r(h,t) = h \times r \times t, \tag{8}$$

where $h, t \in \mathbb{R}^d$ are, respectively, the embedding of the head and the tail, and the relation embedding is represented as a bidimensional matrix $r \in \mathbb{R}^{d \times d}$. This induces an entity-relation deep mutual information exchange to better achieve LP. Each model introduces specific additional constraints on the learned embedding.

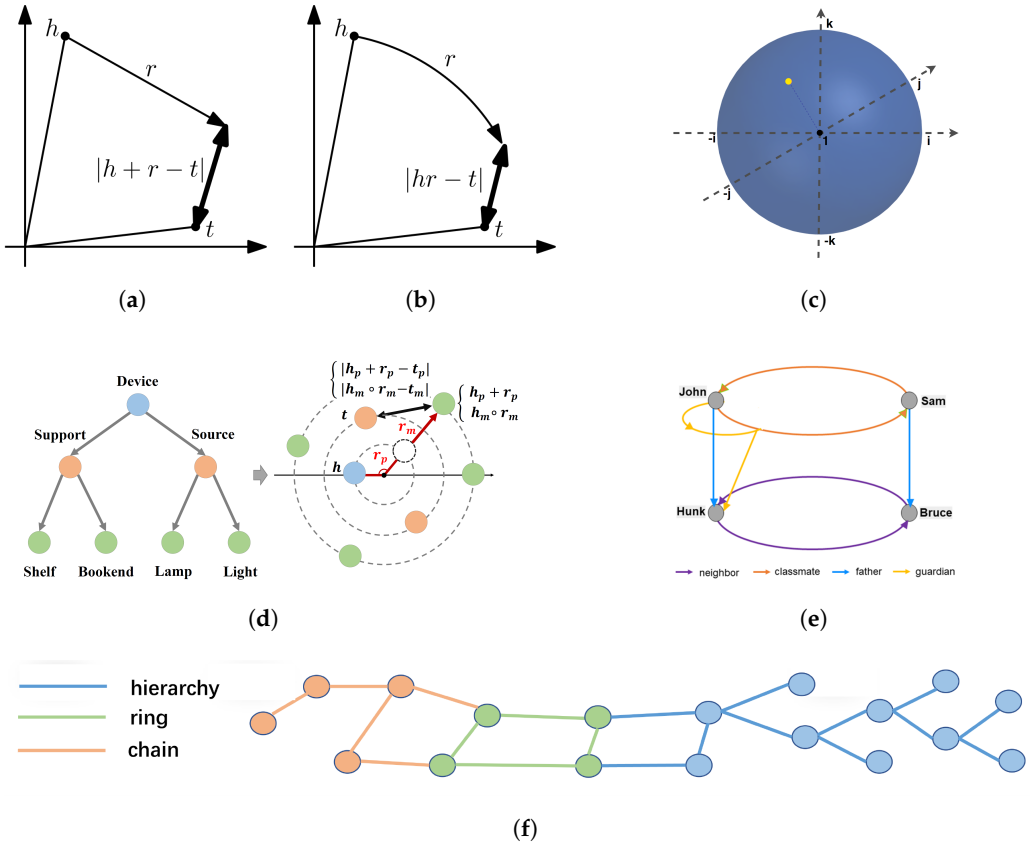


Figure 4. Illustration of transaction-based models. (**a**) TransE models relation r as a translation vector. (**b**) RotatE models r as a rotation in complex plane. Taken from [28]. (**c**) QuatE: the yellow point indicates the position of the unit quaternion. Taken from [30]. (**d**) Illustration of HAKE taken from [29]. The radial coordinate aims to model entities at different levels of the hierarchy (represented by distinct colors), and the angular coordinate aims to distinguish entities at the same level of the hierarchy. (**e**) The combination of translation and rotation in DualE. Taken from [31]. (**f**) Chain, hierarchy, and ring spatial structures of GIE. Orange entities of the KG show the chain structure, which is modeled in Euclidean space; green entities of the KG present a ring structure, which is depicted in the hypersphere space; blue entities of the KG exhibit hierarchical structures. Taken from [32].

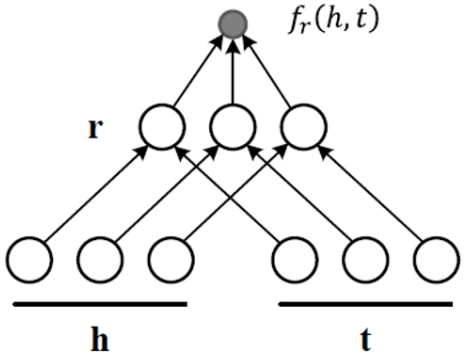


Figure 5. Illustration of DistMult and ComplEx semantic matching-based models. Taken from [12].

DistMult [33] restricts r to a diagonal matrix ($diag(\cdot)$), decreasing the parameter count to $\mathcal{O}(d)$ per relation and, therefore, the complexity of the algorithm. The scoring function is then defined as:

$$f_r(h,t) = h^T \mathrm{diag}(r) t. \tag{9}$$

This score captures pairwise interactions between the components of h and t along the same dimension only. Logistic loss is used for training the model. It is commutative and efficient in capturing symmetric relations but fails to handle antisymmetry due to its oversimplified nature.

ComplEx [34] is an extension of the DistMult model. It introduces complex-valued embeddings for entities and relations to better infer antisymmetric relations and excavate potential semantic associations. In ComplEx, h, r, t no longer lie in a real space but a complex space $\mathbb{C}^d$. The score is computed as below:

$$f_r(h,t) = \text{Real}(h^T \text{diag}(r)\bar{t}), \tag{10}$$

where $\bar{t}$ represents the complex conjugate of the tail entity and $Real(\cdot)$ denotes the real part of a complex relation. This function is no longer commutative, and triples from antisymmetric relations can collect different scores depending on the order of entities involved. Similar to DistMult, the logistic loss is used for training the model.

3.2.3. Based on Neural Networks

Over the years, neural networks have yielded unprecedented predictive performance in many fields, including natural language processing (NLP) and graph-based learning. Researchers have tried to exploit deep learning in KG embedding to model complex nonlinear projections in continuous low-dimensional space (Figure 6), fusing strong representation, associative storage, and generalization capabilities.

ConvE [35] is the first model to use a convolutional neural network (CNN) for the LP task; indeed, it is unquestionably considered a milestone by every survey in the literature. CNNs have the ability to extract multi-scale local space features and combine them to build an efficient representation. Unlike fully connected dense layers, CNNs can help capture complex nonlinear relationships by learning with significantly fewer parameters. ConvE adopts 2D convolution, outperforming 1D convolutional models to extract feature interactions between two embeddings. ConvE starts with a reshaping phase, where the head h and the relation r embeddings are first remodeled in a 2D representation ($\bar{h}$, $\bar{r}$) and then concatenated. The input matrix $[\bar{h}; \bar{r}]$ goes through a 2D convolution layer with a set of w filters that generates a feature map tensor. Subsequently, the tensor is vectorized ($vec(\cdot)$) and projected into a k-dimensional space through a linear transformation parameterized by the matrix W, where it is finally matched with the tail entity embedding t through an inner product. The scoring function is defined as follows:

$$f_r(h,t) = g(vec(g(concat(\bar{h}, \bar{r}) * w))W)t, \tag{11}$$

where $*$ represents convolution, $concat(\cdot,\cdot)$ is the concatenation operator, and g denotes a nonlinear function. ConvE achieves local relationships between different entities in multiple dimensions; however, it ignores the global relations between triple embeddings of the same dimension.

ConvKB [36] combines the idea proposed by ConvE with transitional models, representing one of the most helpful intuitions for LP. In particular, it removes the reshaping phase of ConvE and uses 1D convolution to maintain the translation characteristics of TransE, sufficient to capture global relationships and temporal attributes among entities. ConvKB represents the k-dimensional embedding of each triple (h, r, t) to a three-row matrix, in which each element is transformed as a row vector, $A = [h; r; t] \in \mathbb{R}^{3\times d}$. Then, the matrix A is fed into a convolution layer with a set of filters Ω that generates multiple feature maps. Finally, the feature maps are concatenated and used to calculate the score via dot product using the weight vector w. To summarize, the score function is calculated as below:

$$f_r(h,t) = concat(\sigma([h,r,t] * \Omega))w, \tag{12}$$

where $concat(\cdot,\cdot)$ is the concatenation operator and σ is some activation function, such as ReLU.

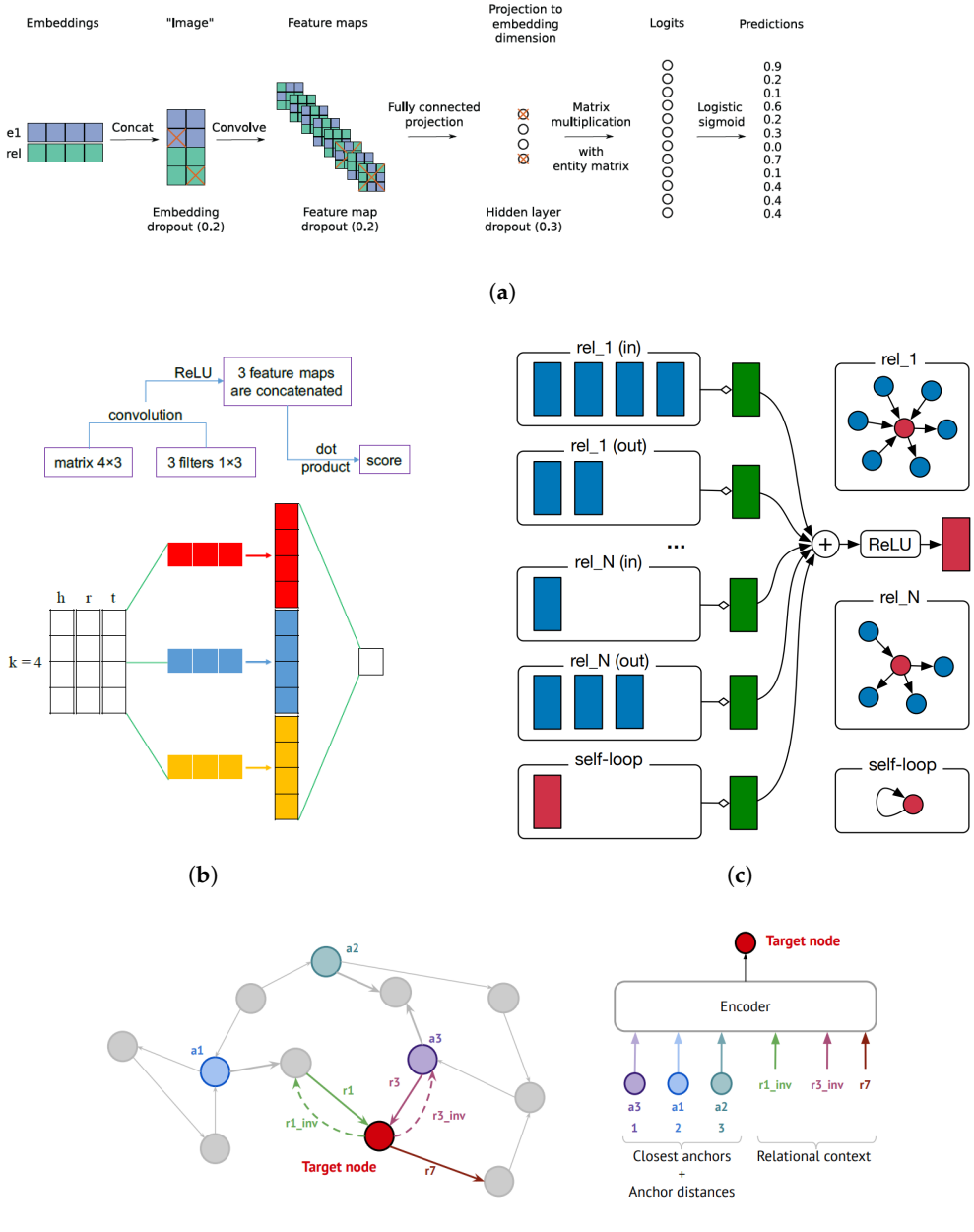


Figure 6. Illustration of neural network-based models. (**a**) In the ConvE model, the entity and relation embeddings are first remodeled in a 2D embedding and then concatenated (steps 1 and 2). The resulting matrix goes through a 2D convolution layer (step 3). Then, the feature map tensor is vectorized and projected into a k-dimensional space (step 4) and matched with all candidate tail-entity embedding t through an inner product (step 5). Taken from [35]. (**b**) The process involved in ConvKB with the embedding size $d = 4$, set of three filters Ω (for illustration purposes), and the activation function σ = ReLU. Taken from [36]. (**c**) Diagram showing the update of a single event-graph node (red) in the R-GCN model. Activations from neighboring nodes (dark blue) are gathered and then transformed for each relation type individually. The resulting representation (green) is accumulated in a (normalized) sum and passed through an activation function. Taken from [37]. (**d**) NodePiece tokenization strategy. Given three anchors $a1, a2, a3$, a target node can be tokenized into a hash of top-k closest anchors, their distances to the target node, and the relational context of outgoing relations from the target node. This hash sequence is passed through an injective encoder to obtain a unique embedding. Taken from [39].

Translational and semantic-matching models fail to cover the complex, inherently implicit information in the local neighborhood of a triple. Instead, they treat each triple independently. As a result, they cannot gain from graph structures to propagate information and make the embedding space smoother. More recently, GNNs have gained massive attention, recognized as efficient machine learning algorithms for irregular non-Euclidean structural graph data. In a GNN, the embedding of a node is learned by recursively aggregating and transforming the representation vectors of its neighboring nodes. Message passing in graphs made of cooperating nodes [44,45] is actually an established work mode borrowed from communication networks and distributed algorithms. Graph convolutional networks (GCNs) naturally extend CNNs to manipulate graphs. They can simultaneously perform end-to-end learning for node feature representations and downstream tasks. Researchers have proposed a spectrum of improved models based on GCNs. Among these, R-GCNs [37] are capable of modeling directed multi-relational data by providing relation-specific transformations. They operate according to the message-passing framework [46], where each entity representation $h_i^{(l+1)}$ in the $(l+1)$-th layer of the GNN is updated according to the messages sent from its neighboring nodes:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in\mathcal{N}_i^r}\frac{1}{|\mathcal{N}_i^r|}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right). \tag{13}$$

$\mathcal{N}_i^r$ represents the set of neighbors of node i under relation r, with W_r and W_0 applying relation and self-loop specific transformations, respectively. In the context of LP, the R-GCN acts as an encoder to obtain the latent feature representations of entities. Then, a decoder uses the tensor decomposition model DistMult to predict the labeled edge according to the learned representation, thereby computing plausibility scores for each triple.

CompGCN [38] comes to address the shortcomings of previously proposed GCNs, that, as R-GCN, are limited to learning only node representations (Figure 7). CompGCN, on the contrary, learns d-dimensional embeddings for both nodes $\boldsymbol{h}_v \in \mathbb{R}^d$ and relations $\boldsymbol{h}_r \in \mathbb{R}^d$. This upgrade makes the solution more suitable for tasks such as LP, solves over-parametrization, and enables the use of available features for edge initialization. The $h_v^{(l+1)}$ representation for node i after l CompGCN layers is given by:

$$h_i^{(l+1)} = \sigma\left(\sum_{(j,r)\in\mathcal{N}_i}\boldsymbol{W}_{dir(r)}\phi\left(h_j^{(l)}, h_r^{(l)}\right)\right), \tag{14}$$

where $\boldsymbol{h}_j^0$ and $\boldsymbol{h}_r^0$ are the initial node $\boldsymbol{x}_v$ and relation $\boldsymbol{z}_r$ features, respectively. $h_r^{(l)}$ denotes the representation of a predicate r after l layers, $\phi(\cdot)$ is a non-parameterized operation such as subtraction or multiplication, and $\boldsymbol{W}_{dir(r)}$ is a direction (i.e., direct, inverse, self-loop) specific parameter. As anticipated, CompGCN updates the relation embeddings as follows:

$$\boldsymbol{h}_r^{(l+1)} = \boldsymbol{W}_{\text{rel}}^{(l)}\,\boldsymbol{h}_r^{(l)}, \tag{15}$$

where $\boldsymbol{W}_{\text{rel}}^{(l)}$ is the l-th layer relation transformation matrix. DistMult is used as the default scoring function.

Drawing inspiration from subword embeddings in NLP, NodePiece [39] tokenizes entities into smaller atoms to parameterize. Instead of storing huge entity embedding matrices, the authors suggest learning a fixed-size vocabulary of atoms, such that any node can be derived from a sequence of them—thereby generalizing towards unseen entities, in the same way as words are constructed from a sequence of subword units. Given a directed graph $G = (N, E, R)$ consisting of nodes $n \in N$, edges $e \in E$, and relation types $r \in R$, NodePiece learns a vocabulary of atoms V with $(|V| \ll |N|)$, consisting of anchor nodes $a \in A$ (a subset of existing nodes in the graph that can be selected randomly or according

to some centrality measures), and all the relation types such that $V = A + R$. Once we defined V, each node n can be tokenized into a hash(n) using k nearest anchors a_i, their k corresponding distances z_i, and a set of m outgoing relations r_i emitted from n:

$$\text{hash}(n) = \left[\{a_1, a_2, ..., a_k\} + \{z_{a_1}, z_{a_2}, ..., z_{a_k}\}, \{r_1, r_2, ..., r_m\}\right] \in \mathbb{R}^{(k+m)\times d}. \quad (16)$$

Subsequently, an encoding function enc : $\mathbb{R}^{(k+m)\times d} \rightarrow \mathbb{R}^d$ (i.e., 2-layer multilayer perceptron MLP) is applied to hash(n), outputting a d-dimensional embedding for node n. RotatE is used as a link prediction decoder.

In Table 2, the space of the embedding and the space complexity for each model are summarized.

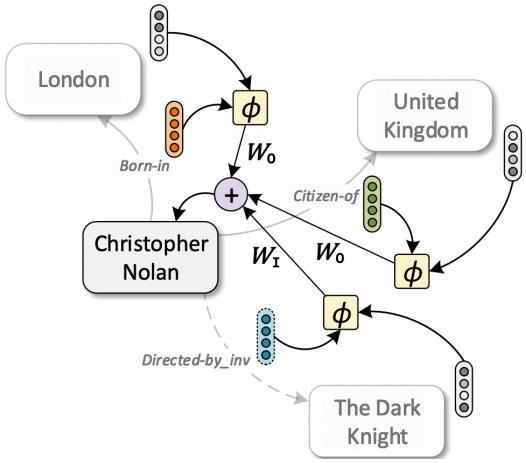


Figure 7. Overview of CompGCN showcasing how the embedding of the central node *Christopher Nolan* is updated. First, a composition operation $\phi(\cdot)$ over each neighboring node and the corresponding edge is performed. The composed embeddings are then convolved with specific filters $\mathbf{W}_O$ and $\mathbf{W}_I$ for direct and inverse relations, respectively; self-loops are omitted for clarity. Finally, the messages from all neighbors are aggregated to obtain an updated embedding of the central node. Re-elaborated from [38].

Table 2. Embedding dimensions and space complexity for the models included in our analysis. N_e and N_r, respectively, denote the number of entities and relation types, i.e., $N_e = |\mathcal{E}|$ and $N_r = |\mathcal{R}|$; d is the embedding dimension; K is the number of layers; $\mathcal{B}$ represents the number of bases; T is the number of filters; and m and n are the dimensions of the filter.

Model	Entity Embedding	Relation Embedding	Space Complexity
TransE [27]	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$	$\mathbf{r} \in \mathbb{R}^d$	$\mathcal{O}(N_e d + N_r d)$
RotatE [28]	$\mathbf{h}, \mathbf{t} \in \mathbb{C}^d$	$\mathbf{r} \in \mathbb{C}^d$	$\mathcal{O}(N_e d + N_r d)$
QuatE [30]	$\mathbf{h}, \mathbf{t} \in \mathbb{H}^d$	$\mathbf{r} \in \mathbb{H}^d$	$\mathcal{O}(N_e d + N_r d)$
HakE [29]	$\mathbf{h}_m, \mathbf{t}_m \in \mathbb{R}^d$ $\mathbf{h}_p, \mathbf{t}_p \in [0, 2\pi)^d$	$\mathbf{r}_m \in \mathbb{R}^d$ $\mathbf{r}_p \in [0, 2\pi)^d$	$\mathcal{O}(N_e d + N_r d)$
GIE [32]	$\mathbf{h}, \mathbf{t} \in \mathbb{G}^d$	$\mathbf{r} \in \mathbb{G}^d$	-
DualE [31]	$\mathbf{h}, \mathbf{t} \in \mathbb{H}^d_{dual}$	$\mathbf{r} \in \mathbb{H}^d_{dual}$	$\mathcal{O}(N_e d + N_r d)$
DistMult [33]	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$	$\mathbf{r} \in \mathbb{R}^d$	$\mathcal{O}N_e d + N_r d)$
ComplEx [34]	$\mathbf{h}, \mathbf{t} \in \mathbb{C}^d$	$\mathbf{r} \in \mathbb{C}^d$	$\mathcal{O}(N_e d + N_r d)$
R-GCN [37]	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$	$\mathbf{r} \in \mathbb{R}^d$	$\mathcal{O}(\mathcal{B}Kd^2 + N_r \mathcal{B}K)$
ConvE [35]	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$	$\mathbf{r} \in \mathbb{R}^d$	$\mathcal{O}(N_e d + N_r d + T_{mn} + Td_e(2d_{e_m} - m + 1)(d_{e_n} - n + 1))$
ConvKB [36]	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$	$\mathbf{r} \in \mathbb{R}^d$	$\mathcal{O}(N_e d + N_r d + 4T)$
CompGCN [38]	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$	$\mathbf{r} \in \mathbb{R}^d$	$\mathcal{O}(Kd^2 + \mathcal{B}d + N_r \mathcal{B})$
NodePiece [39]	-	-	-

3.3. Datasets

We evaluate the models shown in Section 3.2 on the LP task using 6 benchmarks sampled from 4 different KGs, including a real-world biomedical graph. Each dataset

has different relation patterns—formally defined in Section 3.1—that must be inferred to achieve good results [28]. All tested datasets are openly published and maintained by communities or research institutions.

WordNet (WN) [25], first released in 1995, is a large lexical database defining conceptual-semantic and lexical relations between word forms or between synsets—sets of synonyms. The WN18 (https://everest.hds.utc.fr/lib/exe/fetch.php?media=en:wordnet-mlj12.tar.gz (accessed on 18 September 2022)) dataset is an English WordNet subset with 40,943 entities and 18 relations. An example triple is *(presentation, derivationally_related_form, present)*. The main relation patterns are symmetry, antisymmetry, and inversion. The WN18RR (https://github.com/louisccc/KGppler/raw/master/datasets/WN18RR.tar.gz (accessed on 18 September 2022)) dataset is a subset of WN18 where the inverse relations are deleted, and the main relation pattern is symmetry. It is also characterized by entities at different levels of the hierarchy, e.g., "mammal" and "dog", "run", and "move".

Freebase (FB) [3] is a large collaborative knowledge base composed of a large number of world facts, particularly with triples about movies, actors, awards, sports, and sports teams, such as *(Anthony Asquith, location, London)* and *(Nobuko Otowa, profession, actor)*. Two benchmarks are created from this KG. FB15k (https://everest.hds.utc.fr/lib/exe/fetch.php?media=en:fb15k.tgz (accessed on 18 September 2022)) was one of the first datasets used for the LP task and mainly includes symmetry, antisymmetry, and inversion pattern. Precisely, it contains only those Freebase entities that are also available in Wikipedia based on the wiki-links database (https://code.google.com/archive/p/wiki-links/ (accessed on 18 September 2022)) and have at least 100 appearances in Freebase; the relations must also have at least 100 instances. This includes reified relations, meaning the addition of intermediate nodes as compound value types for transforming n-ary relations with $n > 2$ into multiple binary relations. We underline that WN18 and FB15 were introduced by the same authors. FB15k-237 (https://github.com/louisccc/KGppler/raw/master/datasets/fb15k-237.tgz (accessed on 18 September 2022)) is a subset of FB15k obtained by deleting inverse relations. The main relation pattern is composition.

YAGO [5] is a large semantic KG derived from Wikipedia, WordNet, WikiData, GeoNames, and other data sources. YAGO3 is the multilingual extension of YAGO and currently covers more than 17 million entities (persons, organizations, cities, etc.), containing more than 150 million facts about these entities. We used the standard dataset YAGO3-10 (https://github.com/louisccc/KGppler/raw/master/datasets/YAGO3-10.tar.gz (accessed on 18 September 2022)), a subset of YAGO3 which only includes entities with a minimum of 10 in/out-degree. Most of the triples deal with the descriptive attributes of people, such as citizenship, gender, profession, and marital status.

OGBL-BioKG (https://ogb.stanford.edu/docs/linkprop/#ogbl-biokg (accessed on 18 September 2022)) [26] is a domain-specific KG created from a large number of biomedical data repositories, buckled down to drug design. It contains a total of 93,773 entities divided into 5 types: diseases (10,687 nodes), proteins (17,499 nodes), drugs (10,533 nodes), side effects (9969 nodes), and protein functions (45,085 nodes). The dataset contains 51 relations, among which 47 are symmetric since they link entities of the same type (drug–drug, protein–protein, function–function).

Elaborate details of each dataset are showcased in Table 3.

Table 3. Statistics on datasets composition.

Dataset	#Entity	#Relation	Triples			Properties		
			#Train	#Valid	#Test	Reified	Test Leakage	Multiple Domains
WN18 [25]	40,943	18	141,442	5000	5000		✓	
WN18RR [25]	40,943	11	86,835	3034	3134			
FB15k [3]	14,951	1345	483,142	50,000	59,071	✓	✓	✓
FB15k-237 [3]	14,541	237	272,115	17,535	20,466	✓		✓
YAGO3-10 [5]	123,182	37	1,079,040	5000	5000			✓
OGB-BioKG [26]	93,773	51	4,762,677	162,870	162,886			

3.4. Software Libraries

Research has notoriously boosted the open source campaign. Over time, several software libraries have been developed to train and evaluate KG embeddings. Table 4 lists them, together with their main characteristics.

Libraries such as PyKEEN and Pykg2vec offer a wide scope of models, and are thus ideal for large comparative studies such as ours. However, this requirement may not be fundamentally relevant where we only need high-performance accuracy. In that case, hyperparameter optimization, multi-GPU support, and distributed training may be pivotal, for which each library—except for OpenKE and TorchKGE—responds to different needs. From a scalability perspective, it is worth mentioning that PyTorch-BigGraph is designed for massive KGs and does not perform well for small graph instances.

Different libraries support different frameworks, but PyTorch stands out as the most popular. If the user only knows TensorFlow, it may be more appropriate to use Pykg2vec, GraphVite, or muKG.

We also note some additional features. Pykg2vec supplies information charts to visualize model performance metrics and embeddings, making it possible to picture the latent representation of entities and relations on the 2D plane using t-SNE dimensionality reduction. LibKGE allows for in-depth automatic hyperparameter optimization at a dataset level, also exploring the model decision and the score function. PyKEEN offers automated memory optimization: if the input batch size is too large, it is divided into sub-batches for execution. μKG supports joint representation learning over multi-source KGs and multiple embedding tasks.

Table 4. An overview of the functionalities offered by central software libraries (determined in September 2022). HPO refers to hyperparameter optimization, ES for early stopping, MGPU for multi-GPUs support, DTR for distributed training, and * indicates that multi-GPUs support is only available using the PyTorch Lighting integration.

Library	#Models	#Datasets	HPO	ES	MGPU	DTR	Framework
OpenKE [47]	10 9	8	-	-	-	-	PyTorch TensorFlow
Pykg2vec (v0.0.52) [48]	29	10	✓	✓	-	-	PyTorch TensorFlow
GraphVite (v0.2.2) [49]	6	5	-	-	✓	-	PyTorch
PyTorch-BigGraph (v1.0.0) [50]	4	1	-	-	✓	✓	PyTorch
DGL-KE (v0.1.1) [51]	6	5	-	-	✓	✓	PyTorch MXNet
LibKGE [52]	10	9	✓	✓	-	-	PyTorch
TorchKGE (v0.17.5) [53]	11	7	-	-	-	-	PyTorch
PyKEEN (v1.9.0) [54]	44	36	✓	✓	✓*	-	PyTorch PyTorch Lighting
muKG [55]	13	14	-	✓	✓	✓	PyTorch TensorFlow
NeuralKG [56]	19	4	✓	✓	-	-	PyTorch Lighting

4. Experimental Setup

4.1. Hardware Settings

All of our experiments, including the training and evaluation of each model, are performed on a workstation having one Nvidia GeForce RTX3090 GPU with 24 GB of dedicated memory, 64 GB of RAM, and an AMD EPYC 7443 24-Core CPU.

4.2. Implementation Details

We train all models introduced in Section 3.2 on all datasets presented in Section 3.3, except those whose results are already available. We choose diverse repositories to train each model appropriately, actually using two of the software libraries proposed in Section 3.4 and PyTorch in both cases. For TransE, RotatE, DistMult, ComplEx, ConvE, ConvKB, CompGCN, and NodePiece, we use PyKEEN, which faithfully observes each model's architectural construction. Even if QuatE is supported by PyKEEN, we prefer the Pykg2vec

implementation to avoid memory costs exceeding our GPU capacity. Considering that we can see DualE as an extension of QuatE, we also adopt Pykg2vec in this case. Not yet distributed in the employed library version, we re-implement DualE following the original code provided by the authors (https://github.com/Lion-ZS/DualE (accessed on 18 September 2022)). For R-GCN, we operate with PyTorch Geometric. For Hake (https://github.com/MIRALab-USTC/KGE-HAKE (accessed on 18 September 2022)) and GIE (https://github.com/Lion-ZS/GIE (accessed on 18 September 2022)), we use the official source code, adding the datasets not available in the distribution.

4.3. Hyperparameters

KG completion methods are sensitive to hyperparameter tuning, and each dataset must define different hyperparameters. Usually, the authors of a model define a search space made up of acceptable values for each hyperparameter, and then search for the best combination. However, a grid search or even a Bayesian search of the optimal configuration has high costs, requiring several computation hours to conclude each training. To overcome this problem, we refer to the optimal hyperparameters reported in the main paper or other related works, resorting—when not possible (e.g., datasets not yet tested)—to extensive tuning aware of authors' hints. In Appendix A, we report the best hyperparameters we found. It is worth mentioning that for OGB-BioKG, we set the embedding dimension, the batch size, and the negative sample size to the same value for each model reckoning on the examples given by the OGB team (https://github.com/snap-stanford/ogb (accessed on 18 September 2022)). All the models are trained for 100 epochs, a small number compared to some studies with other sets of models and datasets. This choice is guided by the massive size of OGB-BioKG and the correlated training/evaluation costs. As such, the maximum training time asked by a model is ≈2 days, and—even if the results presented in Section 5 are not optimal—we ensure fair performance comparison.

4.4. Training and Evaluation Protocols

The essence of the KG completion task is how to apply observed triples in an incomplete graph to predict whether the unobserved triples are established. Each model is trained with negative sampling. For each observed triple, ω negative ones are sampled by corrupting either its tail or head. Regarding the prediction phase, each target triple in the test set is contaminated by replacing either its head or tail with all dictionary entities in turn, as proposed in [27]. The only exception is OGB-BioKG, where we alter each test triple by replacing its head or tail with randomly sampled 1000 negative entities (500 for its head and 500 for its tail) [26]. We evaluate how the target triple ranks against all the negatives, expecting a higher plausibility for the former. We operate on a filtered scenario so that valid triples outscoring the target one are not considered mistakes and therefore skipped when computing the rank, as proposed in [27].

All the experiments were tracked with Weights & Biases (https://wandb.ai/site (accessed on 18 September 2022)) (≈600 total training hours).

4.5. Evaluation Metrics

Our evaluation practices follow three standard metrics based on the prediction ranks Q, checking whether the correct answers can be ranked before synthetic negatives.

Hits@n accounts for the proportion of ground-truth triples appearing in the first n top-ranked triples.

$$Hits@n = \frac{1}{|Q|} \sum_{q \in Q} \mathbb{1}[q \leq n], \tag{17}$$

where $\mathbb{1}[q \leq n]$ yields 1 if q takes values between 1 and n, otherwise 0. The value bound of Hits@n is between 0 and 1. We consider $n \in \{1, 3, 10\}$. Higher scores indicate better results.

Mean Rank (MR) is the average value of predictions/recommendations among all candidates; therefore, the lower it is, the better the model results are:

$$MR = \frac{1}{|Q|} \sum_{q \in Q} q. \tag{18}$$

Its score bound is always between 1 and the total number of entities.

Mean Reciprocal Rank (MRR) scores the predicted triples based on whether they are true or not. If the first predicted triple is true, its score is 1, and the second true score is $\frac{1}{2}$, and so on. When the n-th triple is established, its score is $\frac{1}{n}$. The final MRR value is the sum of all scores. In other words, MRR is the average of the reciprocal of the obtained ranks, and it is preferable to MR because of its greater robustness to outliers. The calculation formula is:

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{q}. \tag{19}$$

Its value is always between 0 and 1; the higher it is, the better the model results.

These three indicators assess the performance of a KG completion model from different aspects. If Hits@n represents the ability of the model to correctly predict the relationship between triples, the MR and MRR indices reflect the ranking of the correct triples.

5. Results

5.1. Effectiveness

Tables 5–8 illustrate the model results in datasets with different graph characteristics, namely FB15K and WN18, which exploit the model capability of inferring symmetry, antisymmetry, and inversion pattern; FB15K-237 and WN18RR show that the method can model symmetry, antisymmetry, and composition; YAGO3-10 and OGB-BioKG reveal whether the model is efficient on larger graphs. Figure 8 depicts a complete performance overview. We can see that the most recent solutions, such as DualE, GIE, and HakE, yield the best results. We then perform a detailed analysis of each dataset.

Table 5. Results on the WN18 dataset and FB15K dataset; † refers to ours. **Bold** and underline denote the best and second best scores.

Datasets	**WN18**[25]					**FB15K** [3]				
Metric	**Mean Rank**	**MRR**	**Hits@n (%)**			**Mean Rank**	**MRR**	**Hits@n (%)**		
			1	**3**	**10**			**1**	**3**	**10**
TransE [27]	-	0.495	11.3	88.8	94.3	-	0.463	29.7	57.8	74.9
RotatE [28]	184	0.947	93.8	95.3	96.1	32	0.699	58.5	78.8	87.2
QuatE [30]	388	0.949	94.1	95.4	96.0	41	0.770	70.0	82.1	87.8
HakE † [29]	304	0.934	91.9	94.7	95.7	128	0.714	63.9	76.8	83.4
GIE † [32]	-	0.943	93.1	94.9	96.2	-	0.805	75.2	84.1	89.4
DualE [31]	**156**	**0.952**	**94.6**	**95.6**	96.2	**21**	**0.813**	**76.6**	**85.0**	**89.6**
DistMult [33]	655	0.822	72.8	91.4	93.6	42	0.524	54.6	73.3	82.4
ComplEx [34]	-	0.941	93.6	94.5	94.7	-	0.692	59.9	75.9	84.0
R-GCN [37]	-	0.819	69.7	92.9	**96.4**	-	0.651	54.1	73.6	82.5
ConvE [35]	504	0.942	93.5	94.7	95.5	64	0.745	67.0	80.1	87.3
ConvKB † [36]	516	0.824	74.3	89.6	94.5	109	0.567	46.2	62.9	76.2
CompGCN † [38]	539	0.676	51.8	81.6	92.1	69	0.419	27.0	49.8	71.2
NodePiece † [39]	633	0.435	26.3	48.6	64.7	420	0.148	7.83	15.8	28.8

Table 6. Results on WN18RR and FB25K-237; † refers to our results. **Bold** and underline denote the best and second best scores.

Datasets	**WN18RR** [25]					**FB15K-237** [3]				
Metrics	**Mean Rank**	**MRR**	**Hits@n (%)**			**Mean Rank**	**MRR**	**Hits@n (%)**		
			1	**3**	**10**			**1**	**3**	**10**
TransE † [27]	2775	0.2173	3.6	36.4	49.6	334	0.218	5.52	25.7	40.7
RotatE [28]	3277	0.476	42.8	49.2	57.1	185	0.338	20.5	32.8	48.0
QuatE [30]	3472	0.481	43.6	50.0	56.4	176	0.311	22.1	34.2	49.5
HakE [29]	-	**0.497**	**45.2**	**51.6**	58.2	-	0.346	25.0	38.1	54.2
GIE [32]	-	0.491	**45.2**	50.5	57.5	-	0.362	**27.1**	**40.1**	55.2
DualE [31]	**2270**	0.492	44.4	51.3	**58.4**	**91**	0.365	26.8	40.0	**55.9**
DistMult [33]	5100	0.437	39	44	49	254	0.241	15.5	26.3	41.9
ComplEx [34]	5261	0.436	41.0	46	51	339	0.247	15.8	27.5	42.8
R-GCN [37]	-	0.39	33.8	43.1	49	-	0.248	15.3	25.8	41.4
ConvE [35]	5277	0.46	39.0	43.0	48.0	246	0.316	23.9	35.0	49.1
ConvKB [36]	2554	0.248	-	-	52.5	257	**0.396**	-	-	51.7
CompGCN [38]	3533	0.479	44.3	49.4	54.6	197	0.355	26.4	39.0	53.5
NodePiece [39]	-	0.403	-	-	51.5	-	0.256	-	-	42.0

Table 7. Results on YAGO3-10; † refers to ours. **Bold** and underline denote the best and second best scores.

Datasets	**YAGO3-10**[5]				
Metric	**Mean Rank**	**MRR**	**Hits@n (%)**		
			1	**3**	**10**
TransE † [27]	**1156**	0.118	6.7	12.8	21.2
RotatE [28]	1767	0.495	40.2	55.0	67.0
QuatE † [30]	1281	0.260	7.5	38.5	57.6
HakE [29]	-	0.545	46.2	59.6	69.4
GIE [32]	-	**0.579**	**50.5**	**61.8**	**70.9**
DualE † [31]	2068	0.280	9.6	41.6	59.1
DistMult [33]	5926	0.340	24.0	38.0	54.0
ComplEx [34]	6351	0.360	26.0	40.0	55.0
R-GCN [37]	-	0.12	6	11.3	21.1
ConvE [35]	2792	0.520	45.0	56.0	66.0
ConvKB † [36]	2626	0.419	32.3	47.9	58.6
CompGCN † [38]	1591	0.227	12.6	25.6	44.2
NodePiece [39]	-	0.247	-	-	48.8

On WN18 and FB15k, DualE achieves the most satisfactory performance, with QuatE and GIE giving comparable results. Competitive scores are delivered by HakE, ComplEx, ConvE, and RotatE. We highlight the poor effectiveness of TransE, justified by its incapability to infer symmetry patterns. DistMult performs particularly poorly on FB15k since it tends to give identical scores to true triples (h, r, t) and their opposite (t, r, h), thus being unable to model antisymmetry and inversion patterns. This shortcoming also clearly affects the R-GCN and CompGCN scores, bearing in mind that DistMult is used as a decoder to perform LP. Nevertheless, both GNNs behave significantly better on WN18 where there are only 18 types of relations; therefore, it is easy to calculate and generalize. On the other hand, FB15K has 1345 types of relations, increasing computational complexity exponentially.

On WN18RR, TransE obtains scarce results justified by symmetry as the main relation pattern. With this dataset's hierarchical structure, HakE turns out to be the best-performing technique. Despite the peculiar attributes of WN18RR, that match a hierarchical-aware

solution such as HakE, DualE, and GIE yield similar results, closely followed by QuatE, CompGCN, RotatE, and ConvE.

On FB15k-237, DualE and GIE once again prove their superiority, with ComplEx losing ground because of its inability to infer the composition pattern. HakE performs the worst because many triples do not lead to hierarchy. Promising scores are additionally provided by ConvKB, followed by CompGCN, HakE, RotatE, ConvE, and QuatE.

Table 8. Our results on OGB-BIOKG. **Bold** and underline denote the best and second best scores.

Datasets	**OGB-BIOKG**[26]				
Metric	**Mean Rank**	**MRR**	**Hits@n (%)**		
			1	**3**	**10**
TransE [27]	222	0.154	4.9	17.7	37.1
RotatE [28]	**146**	0.296	13.9	37.6	60.8
QuatE [30]	2664	0.270	21.5	29.4	37.4
HakE [29]	333	0.290	16.2	33.8	56.4
GIE [32]	-	0.431	31.7	48.4	65.9
DualE [31]	502	0.336	9.1	20.5	54.5
DistMult [33]	824	0.103	2.4	11.6	26.3
ComplEx [34]	1964	0.219	12.8	24.8	41.2
R-GCN [37]	-	**0.636**	**51.1**	**71.3**	**88.4**
ConvE [35]	444	0.276	17.2	31.8	48.9
ConvKB [36]	476	0.206	12.5	22.3	37.1
CompGCN [38]	-	-	-	-	-
NodePiece [39]	201	0.230	13.4	34.3	58.7

On YAGO3-10, the best performance is given from GIE and Hake because of the semantic hierarchy property of the dataset. We can also assume that these models act well on large graphs, while TransE, QuatE, DualE, and R-GCN face some issues. As for the latter, it has to learn additional weight networks for every connection, consequently making it notoriously non-adaptable for enormous graphs. Conversely, ConvE, ConvKB, and RotatE do not seem to experience a drop in performance despite the peculiar characteristics of this dataset.

On OGB-BIOKG, as we explained in Section 4, the training was performed for only 100 epochs; thus, as we expected, the results are lower than the ones performed by the OGB teams (https://ogb.stanford.edu/docs/leader_linkprop/#ogbl-biokg(accessed on 18 September 2022))—even if we used the same hyperparameters. The best method is R-GCN, followed by GIE, NodePiece, and RotatE, which prove that these methods are exceptionally capable of inferring symmetric relations, which is the main relation pattern of this dataset. For the same reason, DualE and HakE achieve good performance as well, while TransE continues to produce low results. The low-grade outcomes of DistMult are a surprise because we expected this method to perform well on symmetric patterns. We hypothesize that the cause may be related to the hyperparameters' configuration. This is proven by the fact that R-GCN achieves outstanding scores using DistMult as a decoder. The absence of results for CompGCN is justified by the impossibility of training in it on our hardware due to an out-of-memory error (even with the substantial decrease in the batch size, number of layers, and embedding dimension).

5.2. Efficiency

Figure 9 illustrates, for each model, the time in hours spent for the training and evaluation on the OGB-BIOKG dataset, where we conducted all the experiments from scratch. We observe that transactional models have long training times except for TransE and Hake, which have efficiency as one of their strengths. In comparison, neural network methods appear to be strikingly fast. The evaluation time—which covers the full ranking generation in both head and tail predictions for all the facts—is short for every method except for QuatE, DualE, and Hake.

We evaluate the memory consumption (10) considering a batch size of 512 to allow a fair comparison. CompGCN, ComplEx, DualE, and QuatE, are the most memory-intensive solutions, followed by RotatE, R-GCN, and NodePiece, all exceeding the 12 GB standard memory size, with CompGCN even failing to fit our GPU requirements, as explained in Section 5.1. We point out that GNN-based solutions require a considerable amount of memory, even if they are characterized by a reduced number of parameters (see Table 9). This might be caused by the optimization of GNN computational graphs notoriously suffering from (i) redundant neural operator computation; (ii) inconsistent thread mapping; and (iii) excessive intermediate data [57].

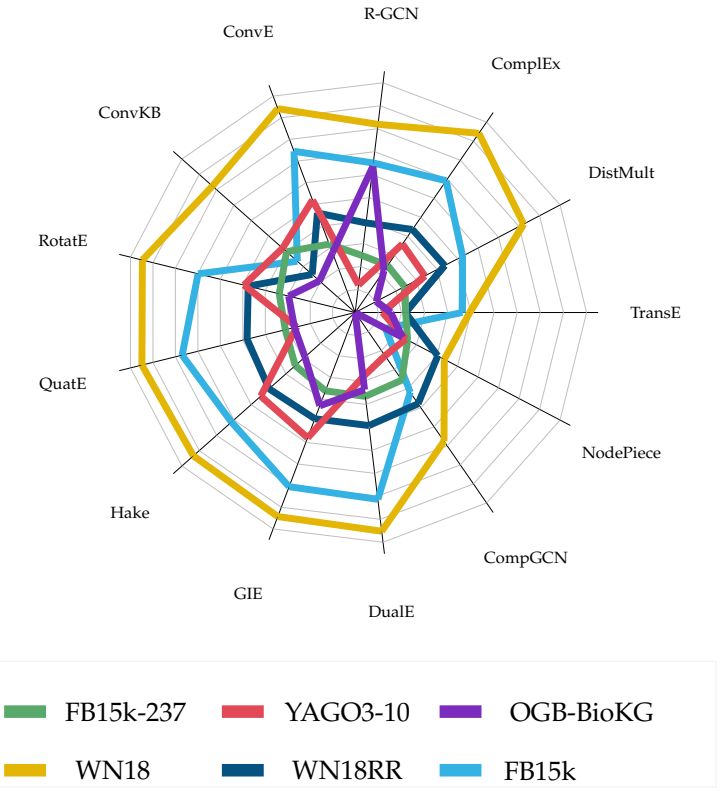


Figure 8. Performance of models on all datasets using MRR [3,5,25,26].

We estimate the carbon footprint with specialized software (http://green-algorithms.org/ (accessed on 18 September 2022)) not interfering with training (Table 10). The value is highly dependent on the server location, the power draw of GPU, and the jobs' total time. The ranking of the models is almost the same as the times, modified only by GPU usage. All the algorithms use over 90% of the GPU power, except for ConvE and Hake, which reach 20% and 70%, respectively. For these reasons, the greenest algorithm is ConvE.

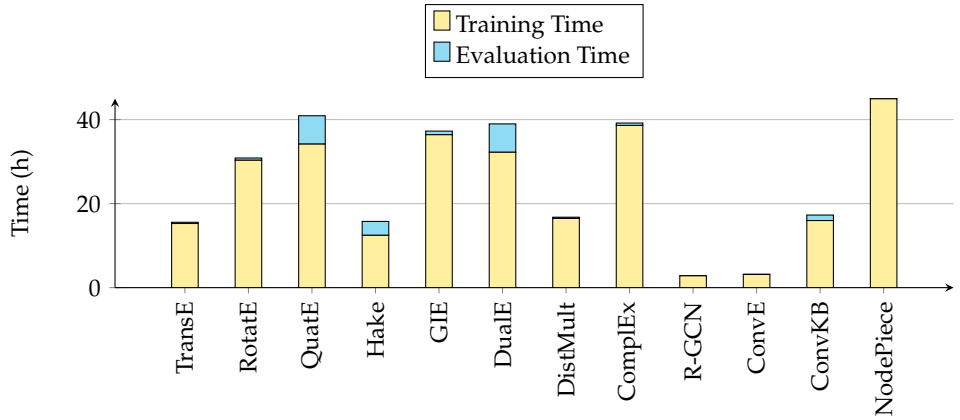


Figure 9. Training and evaluation times (stacked) for each model on OGB-BioKG.

Table 9. Number of parameters of each model on OGBBioKG. Top: translational models. Center: semantic matching models. Bottom: neural network-based models. The architectural cost is highlighted by red gradients (the deeper, the greater).

#params OGBBioKG	TransE [27]	RotatE [28]	QuatE [29]	HakE [30]	GIE [32]	DualE [31]
	187,648,000	375,296,000	750,242,000	375,602,306	93,849,502	750,242,000
	DistMult [33]	Complex [34]				
	187,648,000	375,296,000				
	R-GCN [37]	ConvE [35]	ConvKB [36]	CompGCN [38]	NodePiece [39]	
	52,513,000	20,943,359	187,776,257	18,986,250	2,761,400	

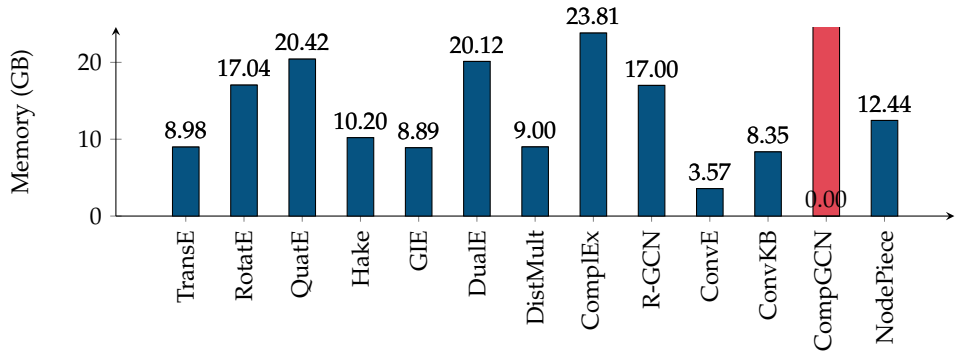


Figure 10. Memory consumption for each model on OGB-BioKG. Red bars denote incomplete experiments due to out-of-memory errors.

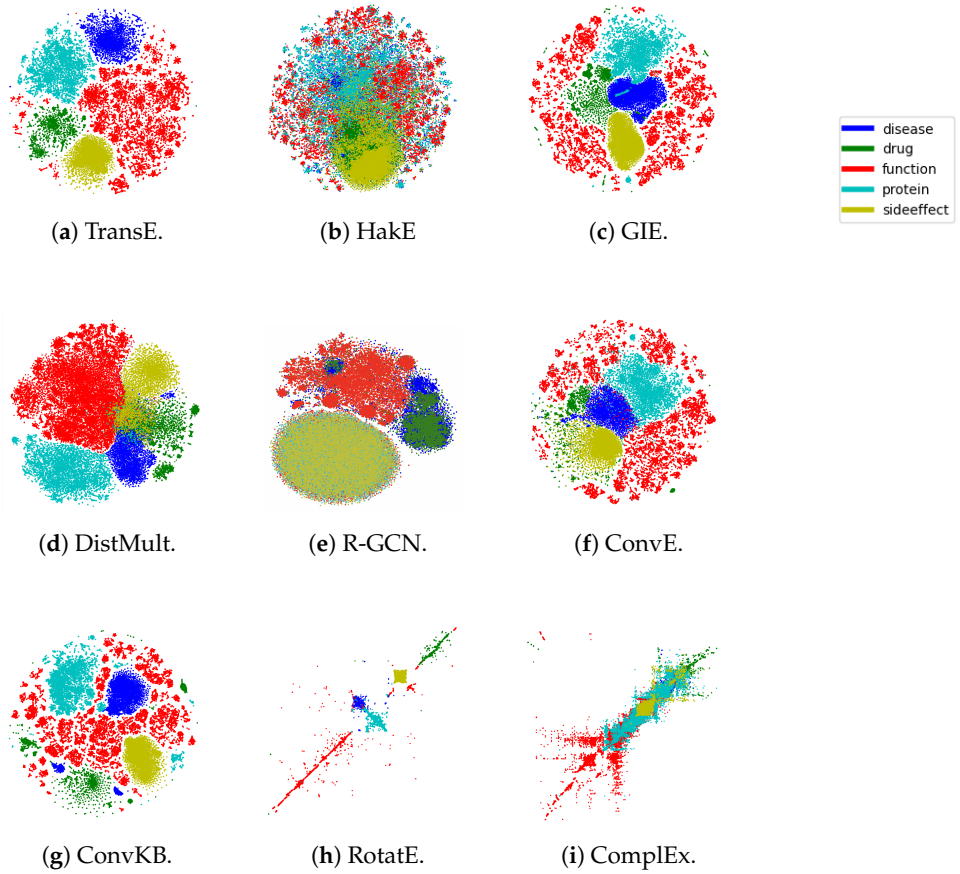
Table 10. Carbon footprint. The green nature of a model is highlighted with green gradients (the deeper, the greater).

Emissions (g CO_2e)	TransE [27]	RotatE [28]	QuatE [29]	HakE [30]	GIE [32]	DualE [31]
	2960	5802	8216	2165	6740	7472
	DistMult [33]	Complex [34]				
	3223	7384				
	R-GCN [37]	ConvE [35]	ConvKB [36]	CompGCN [38]	NodePiece [39]	
	505	82	3943	-	9451	

5.3. Embedding Visualization

KGE models embed the entities and relations of the KG into dense features that are semantically meaningful. These representations are learned using gradient descent at a particular target objective loss; therefore, they are characterized by a black-box nature, unlike other approaches based on association rule mining and graph features. To provide an interpretable analysis of learned entity representations, we include a visual study of such embeddings, projected into a 2-dimensional space using t-SNE [58]. The 93,773 distinct entities of OGB-BioKG are displayed in this compressed space and colored according to their

type (i.e., disease, drug, function, protein, and side effect). We point out to the reader that RotatE (Figure 11h) and ComplEx (Figure 11i) learn embeddings in complex space while all the others in real space. Furthermore, We don't show QuatE and DualE because their embeddings are in a 4- and 8-dimensional space, respectively. From these visualizations, we can appreciate how entities of the same type, encoded by TransE, RotatE, GIE, ConvE, and ConvKB, are clustered together, achieving good separation between classes. This result should be an indicator of the quality of the learned features; however, in some cases, it contrasts with the link prediction performances. Indeed, R-GCN shows poor separation between entity types, contrasting its excellent link prediction scores. Nevertheless, we underline that t-SNE produces an approximation, passing from high to low dimensional space. This can indeed cause a loss of important information and might explain this discrepancy.



(**a**) TransE. (**b**) HakE (**c**) GIE.

(**d**) DistMult. (**e**) R-GCN. (**f**) ConvE.

(**g**) ConvKB. (**h**) RotatE. (**i**) ComplEx.

Figure 11. Entity embeddings compressed in 2D spaces using t-SNE.

6. Discussion

Our analysis highlights that there is no absolute best model, but the effectiveness strictly depends on the KG properties.

Semantic matching models show interesting results on WN18 and WN18RR. Nevertheless, as regards FB15k and FB15k-237, their performances are partially compromised by their inability to infer the composition pattern and antisymmetry/inversion patterns for ComplEx and DistMult, respectively. Neural network-based models are characterized by the most unstable results, with significant differences even between them, highlighting the importance of architectural choices. The new generation of translational models shows the most consistent results, obtaining state-of-the-art performances on almost all datasets and metrics. The foremost translation operation, working in Euclidean space and introduced in [27], has seemingly given way to roto-translation, hyperbolic, hyperspherical, complex, and quaternion spaces. The experimental quality of this family leads us to believe that it is currently the most promising direction. However, it is essential to underline that such

models require a long training time. In terms of efficiency, neural network-based models appear better, particularly on large graphs.

Table 11 showcases model consistency across different benchmarks by averaging the ranking of each model on each dataset. The translation models, particularly GIE, DualE, Hake, and RotatE, are very consistent, with the exception of TransE that has some limitation in modeling the different patterns featured in the datasets. Several approaches based on neural networks, such as RGCN and ConvE, reported very high performance, but were not consistent across different benchmarks. One of the worst is NodePiece. The poor results can possibly be improved by adopting the inductive variant of the model, which uses CompGCN as the encoder. This solution is not taken into consideration in our analysis because it is used to perform inductive LP while we trained all our models on transductive settings. The main benefit of the selected variant (NodePiece + RotatE) is the drastic reduction in the number of parameters, particularly on large-scale KGs.

The best model in terms of effectiveness is GIE, while the leading model in terms of efficiency is ConvE, which has low memory and time costs, in addition to being the greenest algorithm. The best trade-off between effectiveness and efficiency is given by Hake, which occupies top positions across all datasets and is one of the fastest above the massive OGB-BIOKG. Its good behavior can be attributed to its ability to lead most triples to hierarchy.

Based on their experiments on FB15k and FB15k-237, Wang et al. [12] concluded that RotatE outperforms all other models. We showed that RotatE actually achieves competitive results; however, our more inclusive study demonstrates that there are improved solutions capable of significantly pushing the state of the art on different datasets. In line with Zamini et al. [24], we grasp the great capabilities of QuatE and DualE on WN18RR and FB15k-237; in contrast, we capture the improved results of HakE and R-GCN on other testbeds.

Table 11. Average dataset-specific ranking of each model.

	GIE [32]	**DualE [31]**	**HAKE [29]**	**RotatE [28]**	**ConvE [35]**	**QuatE [30]**	**ConvKB [36]**	**CompGCN [38]**	**ComplEx [34]**	**R-GCN [37]**	**DistMult [33]**	**NodePiece [39]**	**TransE [27]**
Avg. Ranking	2.83	3.17	3.83	5.00	5.50	5.67	7.67	7.67	8.17	8.67	9.83	10.67	12.33

7. Conclusions

In this paper, we shed light on the current state of knowledge graph representation learning solutions for link prediction, a domain of research revolving around automatically tracking down missing connections in a multi-relational graph. We propose the most comprehensive and up-to-date study comparing the effectiveness and efficiency of 13 encoding techniques on six popular benchmarks. The investigated works represent different strategies proposed over the years, while datasets capture different scalability and relational modeling needs (symmetry, antisymmetry, inversion, composition). We investigate the circumstances where models perform satisfactorily and pinpoint the areas where research still has room for improvement. Our analysis relies on open source libraries and fully replicable experiments, inspecting dimensions such as automatic metric scores, training/evaluation times, and architectural parameters. We hope that our contributions could serve as a solid base for future research, guiding practitioners in the field to better identify the model that best suits their requirements.

7.1. Limitations and Future Directions

It goes without saying that the presented article is not devoid of some limitations, the overcoming of which is left to future work. Based on our findings, we suggest the following research directions.

7.1.1. More Recent Models

During our literature analysis, we encountered a few top-tier venue contributions—published last year—that captured our interest. However, we were not able to include them in this article for at least one of the following problems: (i) non-available code [59,60]; (ii) computational needs exceeding our academic hardware configuration [61]; (iii) ad hoc architectures for capturing special graph properties, requiring specific datasets and preventing a fair and horizontal comparison with the other representation learning techniques.

7.1.2. Embedding with Auxiliary Information

The models analyzed thus far learn embeddings that are only required to be compatible with the structured facts observed in the KG-based dataset, and hence might not be predictive enough for LP. Thereupon, there is an increasing number of studies that focus on generalizing KG representation learning by incorporating additional semantic information, e.g., entity types, relation paths, literals (e.g., number and text attributes), visual information, uncertain information, and even logical rules. These multi-model embeddings can act as a catalyst for further improving task performance. For instance, literary descriptions and images may give proof of an individual's age and occupation, while uncertain embedding models may simultaneously preserve relational facts and their confidence in non-deterministic KGs. We refer the reader to [1,62] for insights.

7.1.3. Data Augmentation

Data augmentation is the process of modifying or enriching a dataset with additional data and preserving labels. Graph data augmentation (GDA) methods for link prediction (LP) mostly modify the graph structure to reduce overfitting and improve model generalizability (performance invariance across datasets), e.g., node/edge dropping/perturbation [63], multi-scale views via the iterative aggregation of similar nodes [64], literal incorporation [65], creation of counterfactual links [66], and the introduction of silver fake data [67]. To date, most GDA methods have been centered on node- and graph-level tasks [68], outlining a recent trend still under-explored for LP. Given the scarceness of openly available LP-oriented GDA contributions, their poor support from existing libraries, and their demand for ad hoc modified benchmark graphs, we did not include them in our study.

7.1.4. Transfer Learning

Transfer learning aims to reuse prior knowledge gained from abundant data (e.g., the relevant parts of a pre-trained model) to solve a different but related problem. Consequently, it is mostly used to increase performance in similar task scenarios characterized by fewer data. Increasing training data through GDA is a way to unlock transfer learning for LP. As highlighted by Nayyeria et al. [69], graph transfer learning methods have been specifically designed to work on graph-structured data, but they consider entity similarities and are not directly applicable to LP. Similarly to GDA, transductive/inductive out-of-graph LP is a novel research area with few published relevant works [70–72]. We also underline that the majority of existing KG representation learning techniques assume that all test entities appear during training: they need to be frequently re-trained on the whole graph in case of dynamic settings or to be ported towards new graphs with the same set of entities and relation types. Furthermore, following the success of cross-domain learning in NLP tasks [73–78], we believe that future LP research should delve into the combination of multiple training resources to verify the existence of benefits from knowledge sharing in models' parameters. Few-shot learning is another technique proposed for learning in case the amount of training data is low, which has previously been shown to have significant performance in image vision tasks. To the best of our knowledge, no attempts have been made to evaluate few-shot learning strategies on link prediction, therefore we advocate for this novel research direction.

7.1.5. Scalability

As we learned from the results on YAGO3-10 in Section 5.1, scalability is imperative in a large-scope knowledge graph. Existing embedding models capturing rich interactions in relational data are often limited in this sense. Inversely, efficient models are often considerably less expressive. A few solutions (i.e., HolE [79] and ExpressGNN [80]), try to condense the best of both worlds. We leave the adoption of this family of scalable methods to future work.

7.1.6. Dynamic Knowledge

Almost all LP methods assume that the KG is static, but in many real-world scenarios, facts change over time. Many triples are not perpetually factual, e.g., gene profiles after new scientific discoveries [81], or the CEO of a company after administrative changes. Plus, new information is constantly created and might be infused into KGs progressively, e.g., financial news for stock market prediction [82], scientific publications, social network posts, and e-commerce reviews. Emerging solutions have enhanced prediction accuracy by incorporating timestamps to allow learning the dynamic evolution of knowledge. Future comparative analysis should include temporal KG completion models, where facts become quadruples due to the additional time dimension to catch. We refer the reader to [83] for a complete survey of existing literature. We also point up that dynamic relational knowledge poses obstacles in terms of query resolution, where the user may expect the automatic update of the answer in the case of newly added facts satisfying the search condition, asking for incremental views [84].

7.1.7. Link Prediction on Semantic Parsing Graphs

We evaluate the models' LP performance solely on single large KGs curated by human experts. Nevertheless, KGs are not the only source of graph data. Graph extraction from text corpora has long been studied in NLP, originating from preliminary research in latent topic modeling [85,86], word relevance estimation [87], deep metric learning [88], and algebraic-driven semantic relationships [89–93]. Nowadays, semantic parsing provides an efficient and simple method to automatically convert natural language utterances into logical forms. Several graph-based formalisms enable semantic parsing from text, i.e., abstract meaning representation (AMR) [94], event extraction [78,95], semantic dependency parsing [96], and universal conceptual cognitive annotation [97]. For instance, AMR graphs (Figure 12) represent high-level linguistically grounded semantic relations between abstract concepts. Consequently, sentences with a similar meaning should return similar AMR graphs, boosting artificial text evaluation [98]. In recent years, the community emphasized the need to integrate such structured knowledge into neural language models [99] for sundry tasks, such as neural machine translation [100–102], abstractive summarization [103–109], question answering [110–112] and recommendation systems [113–115]. Graph representation learning constitutes a key toolbox for acquiring relational control over knowledge embeddings. We argue that natural language understanding is one of the sectors where these methods can best unleash their expressive powers, representing an evolution of more traditional word embeddings. For instance, having dense formal representations is becoming more and more indispensable to avoid hallucinations in conversational agents and produce more factual and semantically coherent summaries. Notably, authors have devised unsupervised and inductive methods for mapping small semantic parsing graphs into low-dimensional vectors (whole-graph granularity), reflecting their structural and semantic similarities [116]. For these reasons, we find the direction of evaluating the LP on semantic parsing graphs compelling and still never-explored.

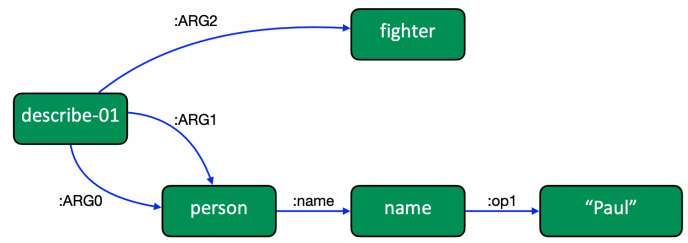


Figure 12. Example of AMR graph obtained from the sentence: "Paul's description of himself: a fighter". Taken from [117].

Author Contributions: Conceptualization, G.F., G.M. and C.S.; methodology, G.F. and G.M.; software, I.F. and P.I.; validation, I.F., G.F. and P.I.; formal analysis, I.F., G.F., G.M. and P.I.; investigation, I.F., G.F. and P.I.; resources, G.M.; data curation, I.F., G.F. and P.I.; writing—original draft preparation, I.F., G.F. and P.I.; writing—review and editing, G.M. and C.S.; visualization, I.F., G.F., and P.I.; supervision: G.M. and C.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: For replication purposes, code and datasets are publicly available at https://github.com/disi-unibo-nlp/kg-emb-link-pred (accessed on 18 September 2022).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GNN	Graph Neural Network
GCN	Graph Convolutional Network
R-GCN	Relational Graph Convolutional Network
KG	Knowledge Graph
KGE	Knowledge Graph Embedding
LP	Link Prediction

Appendix A

Table A1 reports the hyperparameter settings used for each model in our experiments.

Table A1. Hyperparameters used to train the models in our work. $*$ marks the final picked values. *B*: batch size, *Ep*: training epochs or *Steps*: training steps, *D*: embedding dimension, *LR*: learning rate, *NSS*: negative sample size (default: 1), γ: margin, *AdvTemp*: temperature in adversarial negative sampling, *Reg*: regularization method (default: L2), λ: regularization, *Opt*: optimizer (default: Adam), *Mod*: modulus weight, *Phase*: phase weight, *Drop*: dropout rate (in: input, h: hidden layer, feat: features), *K*: number of convolutional filters, *N*: number of GCN layer, $|A|$: anchors, *k*: anchors per node, *m*: relational context.

Model	Dataset	Hyperparams
TransE [27]	WN18RR [25]	B:256, Ep:{3000$*$, 1000}, D:50, LR:LR:{5 $\times 10^{-5}*$, 0.01}, γ:5
	FB15K-237 [3]	B: 256, Ep:{3000$*$, 1000}, D:{100$*$, 50}, LR:{5 $\times 10^{-5}*$, 0.01, 0.1},γ:1
	YAGO3-10 [5]	B:256, Ep:3000, D:100, LR:{5 $\times 10^{-5}*$, 5 $\times 10^{-4}$}, γ:5
	OGB-BIOKG [26]	B:512, Ep:100, D:2000, NSS:128, LR: 5 $\times 10^{-5}$, γ:20, AdvTemp:1.0
RotatE [28]	OGB-BIOKG [26]	B:512, Ep:100, D:2000, NSS:128, LR: 5 $\times 10^{-5}$, γ:20, AdvTemp:1.0
QuatE [30]	YAGO3-10 [5]	B:{512$*$, 9300}, Ep:1000, D:{100$*$, 200}, LR: 0.1, γ: 1.0, λ:0.1
	OGB-BIOKG [26]	B:512, Ep:100, D:1000, NSS:128, LR: 5 $\times 10^{-5}$, γ:20

Table A1. *Cont.*

Model	Dataset	Hyperparams
HakE [29]	WN18[25]	B:1024, Steps:180000, D:500, NSS:128, LR:0.0002, γ:24.0, AdvTemp:1.0, Mod:1.0, Phase:0.5
	FB15K [3]	B:1024, Steps:180000, D:500, NSS:128, LR:0.0002, γ:24.0, AdvTemp:1.0, Mod:1.0, Phase:0.5
	OGB-BIOKG [26]	B:512, Steps:300000, D:2000, NSS:128, LR:0.0002, γ:24.0, AdvTemp:1.0, Mod:1.0, Phase:0.5
GIE [32]	WN18 [25]	B:2000, Ep:100, D:2000, LR:0.01, Opt:Adagrad, Reg:N3, RegWeight: 0.1
	FB15K [3]	B:2000, Ep:100, D:2000, LR:0.01, Opt:Adagrad, Reg:N3, RegWeight: 0.1
	OGB-BIOKG [26]	B:2000, Ep:100, D:2000, LR:0.01, Opt:Adagrad, Reg:N3, RegWeight: 5×10^{-3}
DualE [31]	YAGO3-10 [5]	B:512, Ep:1000, D:100, LR: 0.1, Opt:adagrad, γ: 1.0, λ:0.25
	OGB-BIOKG [26]	B:512, Ep:100, D:1000, NSS:128, LR:5×10^{-5}, γ: 20.0, λ:0.1
DistMult [33]	OGB-BIOKG [26]	B:512, Ep:100, D:2000, NSS:128, LR:0.001 γ:{500$*$,1}, AdvTemp:{1.0$*$, 0}
ComplEx [34]	OGB-BIOKG [26]	B:512, Ep:100, D:2000, NSS:128, LR:0.001, γ:500, AdvTemp:1.0
R-GCN [37]	FB15K [3]	Ep:50, D:500, LR:0.01, λ:0.1, Drop:{h:0.2}
	WN18 [25]	Ep:50, D:500, LR:0.01, λ:0.1, Drop:{h:0.2}
	WN18RR [25]	B:512, Ep:1000, D:150, LR:0.001, λ:0.0001, Reg:L3, Drop:{h:0.2}
	FB15K-237 [3]	Ep:50, D:500, LR:0.01, λ:0.1, Drop:{h:0.2}
	YAGO3-10 [5]	B:512, Ep:1000, D:150, LR:0.001, λ:0.0001, Reg:L3, Drop:{h:0.2}
	OGB-BIOKG [26]	B:65536, Ep:100, D:500, LR:0.001, NSS:1000, Drop:{h:0.2}
ConvE [35]	OGB-BIOKG [26]	B:512, Ep:100, D:2000, NSS:128, LR:0.0001, γ:20, AdvTemp:1.0, Drop:{in:0.1;h:0.2;feat:0.2}
ConvKB [36]	WN18 [25]	B:256, Ep:100, D:50, LR:0.0001, K:64, Drop:{h:0.5}
	FB15K [3]	B:256, Ep:100, D:50, LR:0.0001, K:{64 $*$, 50}, Drop:{h:{0.5 $*$, 0}
	OGB-BIOKG [26]	B:512, Ep:100, D:2000, NSS:128, LR:0.0001, γ:20, AdvTemp:1.0, K:64, Drop:{h:0.5}
CompGCN [38]	WN18 [25]	B:512, Ep:200, D:200, LR:0.001, γ:40, N:2, Drop:{h:0.1}
	FB15K [3]	B:512, Ep:200, D:200, LR:0.001, γ:40, N:2, Drop:{h:0.1}
	YAGO3-10 [5]	B:2048, Ep:100, D:200, LR:0.001, γ:40, N:2, Drop:{h:0.1}
NodePiece [39]	WN18 [25]	B:512, D: 200, Ep:600, LR:{0.00025$*$,0.0005}, NSS: 10, γ:50, AdvTemp:1.0, Drop:{h:0.1}, $\|A\|$:{1000$*$,500}, k:{20$*$,50}, m:{5$*$,4}
	FB15K [3]	B:512, D: 200, Ep:500, LR:{0.00025$*$,0.0005}, NSS: 10, γ:{50$*$,15}, AdvTemp:1.0, Drop:{h:0.1}, $\|A\|$:1000, k:20, m:{5$*$,15}
	OGB-BIOKG [26]	B:512, D: 200, Ep:500, LR:0.0001, NSS: 128, γ:50, AdvTemp:1.0, Drop:{h:0.1}, $\|A\|$:100, k:10, m:6

References

1. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Philip, S.Y. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Networks Learn. Syst.* **2021**, *33*, 494–514.
2. Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; Ives, Z. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 722–735.
3. Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada, 10–12 June 2008; pp. 1247–1250.
4. Vrandečić, D.; Krötzsch, M. Wikidata: A Free Collaborative Knowledgebase. *Commun. ACM* **2014**, *57*, 78–85. https://doi.org/10.1145/2629489.
5. Suchanek, F.M.; Kasneci, G.; Weikum, G. Yago: A core of semantic knowledge. In Proceedings of the WWW, Banff, AB, Canada, 8–12 May 2007; ACM: New York, NY, USA, 2007; pp. 697–706.
6. West, R.; Gabrilovich, E.; Murphy, K.; Sun, S.; Gupta, R.; Lin, D. Knowledge base completion via search-based question answering. In Proceedings of the 23rd International Conference on World Wide Web, Seoul, Korea, 7–11 April 2014; pp. 515–526.
7. Krompaß, D.; Baier, S.; Tresp, V. Type-Constrained Representation Learning in Knowledge Graphs. In *Proceedings of the ISWC (1)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2015; Volume. 9366, pp. 640–655.
8. Abbas, K.; Abbasi, A.; Dong, S.; Niu, L.; Yu, L.; Chen, B.; Cai, S.M.; Hasan, Q. Application of network link prediction in drug discovery. *BMC Bioinform.* **2021**, *22*, 187.
9. Chen, X.; Hu, Z.; Sun, Y. Fuzzy Logic Based Logical Query Answering on Knowledge Graphs. In Proceedings of the AAAI, Virtually, 22 February–1 March 2022; AAAI Press: Palo Alto, CA, USA, 2022; pp. 3939–3948.
10. Yasunaga, M.; Bosselut, A.; Ren, H.; Zhang, X.; Manning, C.D.; Liang, P.; Leskovec, J. Deep Bidirectional Language-Knowledge Graph Pretraining. In Proceedings of the Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 November 2022.
11. Dai, Y.; Wang, S.; Xiong, N.N.; Guo, W. A Survey on Knowledge Graph Embedding: Approaches, Applications and Benchmarks. *Electronics* **2020**, *9*, 750. https://doi.org/10.3390/electronics9050750.
12. Wang, M.; Qiu, L.; Wang, X. A survey on knowledge graph embeddings for link prediction. *Symmetry* **2021**, *13*, 485.
13. Sharma, A.; Talukdar, P.; et al. Towards understanding the geometry of knowledge graph embeddings. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; Volume 1: Long Papers, pp. 122–131.
14. Akrami, F.; Saeef, M.S.; Zhang, Q.; Hu, W.; Li, C. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, Portland, OR, USA, 14–19 June 2020; pp. 1995–2010.

15. Kadlec, R.; Bajgar, O.; Kleindienst, J. Knowledge base completion: Baselines strike back. *arXiv* **2017**, arXiv:1705.10744.
16. Tran, H.N.; Takasu, A. Analyzing knowledge graph embedding methods from a multi-embedding interaction perspective. *arXiv* **2019**, arXiv:1903.11406.
17. Wang, Q.; Mao, Z.; Wang, B.; Guo, L. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 2724–2743.
18. Chen, Z.; Wang, Y.; Zhao, B.; Cheng, J.; Zhao, X.; Duan, Z. Knowledge graph completion: A review. *IEEE Access* **2020**, *8*, 192435–192456.
19. Choudhary, S.; Luthra, T.; Mittal, A.; Singh, R. A survey of knowledge graph embedding and their applications. *arXiv* **2021**, arXiv:2107.07842.
20. Garg, S.; Roy, D. A Birds Eye View on Knowledge Graph Embeddings, Software Libraries, Applications and Challenges. *arXiv* **2022**, arXiv:2205.09088.
21. Hamilton, W.L.; Ying, R.; Leskovec, J. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* **2017**, *40*, 52–74.
22. Lin, Y.; Han, X.; Xie, R.; Liu, Z.; Sun, M. Knowledge Representation Learning: A Quantitative Review. *arXiv* **2018**, arXiv:1812.10901.
23. Rossi, A.; Barbosa, D.; Firmani, D.; Matinata, A.; Merialdo, P. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Trans. Knowl. Discov. Data (TKDD)* **2021**, *15*, 14.
24. Zamini, M.; Reza, H.; Rabiei, M. A Review of Knowledge Graph Completion. *Information* **2022**, *13*, 396.
25. Miller, G.A. WordNet: a lexical database for English. *Commun. ACM* **1995**, *38*, 39–41.
26. Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 22118–22133.
27. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. *Adv. Neural Inf. Process. Syst.* **2013**, *26* , 2787–2795.
28. Sun, Z.; Deng, Z.H.; Nie, J.Y.; Tang, J. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv* **2019**, arXiv:1902.10197.
29. Zhang, Z.; Cai, J.; Zhang, Y.; Wang, J. Learning hierarchy-aware knowledge graph embeddings for link prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 3065–3072.
30. Zhang, S.; Tay, Y.; Yao, L.; Liu, Q. Quaternion knowledge graph embeddings. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.
31. Cao, Z.; Xu, Q.; Yang, Z.; Cao, X.; Huang, Q. Dual quaternion knowledge graph embeddings. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtually, 2–9 February 2021; Volume 35, pp. 6894–6902.
32. Cao, Z.; Xu, Q.; Yang, Z.; Cao, X.; Huang, Q. Geometry Interaction Knowledge Graph Embeddings. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 28 February–1 March 2022.
33. Yang, B.; Yih, W.T.; He, X.; Gao, J.; Deng, L. Embedding entities and relations for learning and inference in knowledge bases. *arXiv* **2014**, arXiv:1412.6575.
34. Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; Bouchard, G. Complex embeddings for simple link prediction. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 2071–2080.
35. Dettmers, T.; Minervini, P.; Stenetorp, P.; Riedel, S. Convolutional 2d knowledge graph embeddings. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
36. Nguyen, D.Q.; Nguyen, T.D.; Nguyen, D.Q.; Phung, D. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv* **2017**, arXiv:1712.02121.
37. Schlichtkrull, M.; Kipf, T.N.; Bloem, P.; Berg, R.v.d.; Titov, I.; Welling, M. Modeling relational data with graph convolutional networks. In Proceedings of the European Semantic Web Conference, Crete, Greece, 3–7 June 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 593–607.
38. Vashishth, S.; Sanyal, S.; Nitin, V.; Talukdar, P. Composition-based multi-relational graph convolutional networks. *arXiv* **2019**, arXiv:1911.03082.
39. Galkin, M.; Denis, E.; Wu, J.; Hamilton, W.L. NodePiece: Compositional and Parameter-Efficient Representations of Large Knowledge Graphs. In Proceedings of the International Conference on Learning Representations, Virtually, 25–29 April 2022.
40. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of the ICLR (Workshop Poster), Scottsdale, AZ, USA, 2–4 May 2013.
41. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. Knowledge Graph Embedding by Translating on Hyperplanes. In Proceedings of the AAAI, Quebec City, QC, USA, 27–31 July 2014; AAAI Press: Palo Alto, CA, USA, 2014; pp. 1112–1119.
42. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu, X. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In Proceedings of the AAAI, Austin, TX, USA, 25–30 January 2015; AAAI Press: Palo Alto, CA, USA, 2015; pp. 2181–2187.
43. Ji, G.; He, S.; Xu, L.; Liu, K.; Zhao, J. Knowledge Graph Embedding via Dynamic Mapping Matrix. In *Proceedings of the ACL (1)*; The Association for Computer Linguistics: New York, NY, USA, 2015; pp. 687–696.
44. Lodi, S.; Moro, G.; Sartori, C. Distributed data clustering in multi-dimensional peer-to-peer networks. In Proceedings of the Database Technologies 2010, Twenty-First Australasian Database Conference (ADC 2010), Brisbane, Australia, 18–22 January 2010; Proceedings; *CRPIT*; Shen, H.T., Bouguettaya, A., Eds.; Australian Computer Society: Sydney, Australia, 2010; Volume 104, pp. 171–178.

45. Cerroni, W.; Moro, G.; Pirini, T.; Ramilli, M. Peer-to-Peer Data Mining Classifiers for Decentralized Detection of Network Attacks. In Proceedings of the Twenty-Fourth Australasian Database Conference, ADC 2013, Adelaide, Australia, 29 January–1 February 2013; *CRPIT*; Wang, H., Zhang, R., Eds.; Australian Computer Society: Sydney, Australia, 2013; Volume 137, pp. 101–108.
46. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural message passing for quantum chemistry. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 1263–1272.
47. Han, X.; Cao, S.; Lv, X.; Lin, Y.; Liu, Z.; Sun, M.; Li, J. Openke: An open toolkit for knowledge embedding. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Brussels, Belgium, 31 October–4 November 2018; pp. 139–144.
48. Yu, S.Y.; Rokka Chhetri, S.; Canedo, A.; Goyal, P.; Faruque, M.A.A. Pykg2vec: A Python Library for Knowledge Graph Embedding. *arXiv* **2019**, arXiv:1906.04239.
49. Zhu, Z.; Xu, S.; Tang, J.; Qu, M. Graphvite: A high-performance cpu-gpu hybrid system for node embedding. In Proceedings of the The World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 2494–2504.
50. Lerer, A.; Wu, L.; Shen, J.; Lacroix, T.; Wehrstedt, L.; Bose, A.; Peysakhovich, A. Pytorch-biggraph: A large scale graph embedding system. *Proc. Mach. Learn. Syst.* **2019**, *1*, 120–131.
51. Zheng, D.; Song, X.; Ma, C.; Tan, Z.; Ye, Z.; Dong, J.; Xiong, H.; Zhang, Z.; Karypis, G. DGL-KE: Training Knowledge Graph Embeddings at Scale. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xi'an, China, 25–30 July 2020; SIGIR '20; Association for Computing Machinery: New York, NY, USA, 2020; pp. 739–748.
52. Broscheit, S.; Ruffinelli, D.; Kochsiek, A.; Betz, P.; Gemulla, R. LibKGE - A Knowledge Graph Embedding Library for Reproducible Research. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, 16–20 November 2020; pp. 165–174.
53. Boschin, A. TorchKGE: Knowledge Graph Embedding in Python and PyTorch. In Proceedings of the International Workshop on Knowledge Graph: Mining Knowledge Graph for Deep Insights, 2020. Virtual Event, 24 August 2020.
54. Ali, M.; Berrendorf, M.; Hoyt, C.T.; Vermue, L.; Sharifzadeh, S.; Tresp, V.; Lehmann, J. PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *J. Mach. Learn. Res.* **2021**, *22*, 1–6.
55. Luo, X.; Sun, Z.; Hu, W. μKG: A Library for Multi-source Knowledge Graph Embeddings and Applications. In Proceedings of the ISWC, Hangzhou, China, 23–27 October 2022.
56. Zhang, W.; Chen, X.; Yao, Z.; Chen, M.; Zhu, Y.; Yu, H.; Huang, Y.; et al. NeuralKG: An Open Source Library for Diverse Representation Learning of Knowledge Graphs. *arXiv* **2022**, arXiv:2202.12571.
57. Zhang, H.; Yu, Z.; Dai, G.; Huang, G.; Ding, Y.; Xie, Y.; Wang, Y. Understanding gnn computational graph: A coordinated computation, io, and memory perspective. *Proc. Mach. Learn. Syst.* **2022**, *4*, 467–484.
58. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
59. Zhou, Z.; Wang, C.; Feng, Y.; Chen, D. JointE: Jointly utilizing 1D and 2D convolution for knowledge graph embedding. *Knowl.-Based Syst.* **2022**, *240*, 108100.
60. Le, T.; Le, N.; Le, B. Knowledge Graph Embedding by Relational Rotation and Complex Convolution for Link Prediction. *Expert Syst. Appl.* **2022**, *214* 119–122.
61. Shen, J.; Wang, C.; Gong, L.; Song, D. Joint language semantic and structure embedding for knowledge graph completion. *arXiv* **2022**, arXiv:2209.08721.
62. Gesese, G.A.; Biswas, R.; Sack, H. A Comprehensive Survey of Knowledge Graph Embeddings with Literals: Techniques and Applications. In Proceedings of the DL4KG@ESWC, CEUR-WS.org, CEUR Workshop, Portoroz, Slovenia, 2 June 2019, Volume 2377, pp. 31–40.
63. Luo, D.; Cheng, W.; Yu, W.; Zong, B.; Ni, J.; Chen, H.; Zhang, X. Learning to drop: Robust graph neural network via topological denoising. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, Jerusalem, Israel, 8–12 March 2021; pp. 779–787.
64. Cai, L.; Ji, S. A multi-scale approach for graph link prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 3308–3315.
65. Wang, J.; Ilievski, F.; Szekely, P.; Yao, K.T. Augmenting Knowledge Graphs for Better Link Prediction. *arXiv* **2022**, arXiv:2203.13965.
66. Zhao, T.; Liu, G.; Wang, D.; Yu, W.; Jiang, M. Learning from counterfactual links for link prediction. In Proceedings of the International Conference on Machine Learning, PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 26911–26926.
67. Wu, X.; Zhang, Y.; Yu, J.; Zhang, C.; Qiao, H.; Wu, Y.; Wang, X.; Wu, Z.; Duan, H. Virtual data augmentation method for reaction prediction. *Sci. Rep.* **2022**, *12*, 17098.
68. Ding, K.; Xu, Z.; Tong, H.; Liu, H. Data augmentation for deep graph learning: A survey. *arXiv* **2022**, arXiv:2202.08235.
69. Nayyeri, M.; Cil, G.M.; Vahdati, S.; Osborne, F.; Rahman, M.; Angioni, S.; Salatino, A.; Recupero, D.R.; Vassilyeva, N.; Motta, E.; et al. Trans4E: Link prediction on scholarly knowledge graphs. *Neurocomputing* **2021**, *461*, 530–542.
70. Baek, J.; Lee, D.B.; Hwang, S.J. Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 546–560.
71. Han, X.; Huang, Z.; An, B.; Bai, J. Adaptive transfer learning on graph neural networks. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 565–574.

72. Dai, D.; Zheng, H.; Luo, F.; Yang, P.; Chang, B.; Sui, Z. Inductively representing out-of-knowledge-graph entities by optimal estimation under translational assumptions. *arXiv* **2020**, arXiv:2009.12765.
73. Domeniconi, G.; Moro, G.; Pasolini, R.; Sartori, C. Cross-domain Text Classification through Iterative Refining of Target Categories Representations. In Proceedings of the KDIR 2014–Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, Rome, Italy, 21–24 October 2014; Fred, A.L.N., Filipe, J., Eds.; SciTePress: Setubal, Portugal, 2014; pp. 31–42. https://doi.org/10.5220/0005069400310042.
74. Domeniconi, G.; Moro, G.; Pagliarani, A.; Pasolini, R. Markov Chain based Method for In-Domain and Cross-Domain Sentiment Classification. In Proceedings of the KDIR 2015—International Conference on Knowledge Discovery and Information Retrieval, Part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015), Lisbon, Portugal, 12–14 November 2015; Fred, A.L.N., Dietz, J.L.G., Aveiro, D., Liu, K., Filipe, J., Eds.; SciTePress: Setubal, Portugal, 2015; Volume 1, pp. 127–137. https://doi.org/10.5220/0005636001270137.
75. Domeniconi, G.; Masseroli, M.; Moro, G.; Pinoli, P. Cross-organism learning method to discover new gene functionalities. *Comput. Methods Programs Biomed.* **2016**, *126*, 20–34. https://doi.org/10.1016/j.cmpb.2015.12.002.
76. Domeniconi, G.; Moro, G.; Pagliarani, A.; Pasolini, R. On Deep Learning in Cross-Domain Sentiment Classification. In Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Funchal, Madeira, Portugal, 1–3 November 2017; Fred, A.L.N., Filipe, J., Eds.; SciTePress: Setubal, Portugal, 2017; Volume 1, pp. 50–60. https://doi.org/10.5220/0006488100500060.
77. Moro, G.; Pagliarani, A.; Pasolini, R.; Sartori, C. Cross-domain & In-domain Sentiment Analysis with Memory-based Deep Neural Networks. In Proceedings of the IC3K 2018, Seville, Spain, 18–20 September 2018; SciTePress: Setubal, Portugal, 2018; Volume 1, pp. 127–138. https://doi.org/10.5220/0007239101270138.
78. Frisoni, G.; Moro, G.; Balzani, L. Text-to-Text Extraction and Verbalization of Biomedical Event Graphs. In Proceedings of the 29th International Conference on Computational Linguistics, Gyeongju, Republic of Korea, 12–17 October 2022; International Committee on Computational Linguistics: Gyeongju, Republic of Korea, 2022; pp. 2692–2710.
79. Nickel, M.; Rosasco, L.; Poggio, T. Holographic embeddings of knowledge graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
80. Zhang, Y.; Chen, X.; Yang, Y.; Ramamurthy, A.; Li, B.; Qi, Y.; Song, L. Efficient probabilistic logic reasoning with graph neural networks. *arXiv* **2020**, arXiv:2001.11850.
81. Domeniconi, G.; Masseroli, M.; Moro, G.; Pinoli, P. *Discovering New Gene Functionalities from Random Perturbations of Known Gene Ontological Annotations*; INSTICC Press: Lisboa, Portugal, 2014; pp. 107–116. https://doi.org/10.5220/0005087801070116.
82. Fabbri, M.; Moro, G. Dow Jones Trading with Deep Learning: The Unreasonable Effectiveness of Recurrent Neural Networks. In Proceedings of the 7th International Conference on Data Science, Technology and Applications, DATA 2018, Porto, Portugal, 26–28 July 2018; Bernardino, J., Quix, C., Eds.; SciTePress: Setubal, Portugal, 2018; pp. 142–153. https://doi.org/10.5220/0006922101420153.
83. Cai, B.; Xiang, Y.; Gao, L.; Zhang, H.; Li, Y.; Li, J. Temporal Knowledge Graph Completion: A Survey. *arXiv* **2022**, arXiv:2201.08236.
84. Moro, G.; Sartori, C. Incremental maintenance of multi-source views. In Proceedings of the Twelfth Australasian Database Conference, ADC2001, Bond University, Queensland, Australia, 29 January–1 February 2001; Orlowska, M.E., Roddick, J.F., Eds.; IEEE Computer Society: Washington, DC, USA, 2001; pp. 13–20. https://doi.org/10.1109/ADC.2001.904459.
85. Domeniconi, G.; Moro, G.; Pasolini, R.; Sartori, C. Iterative Refining of Category Profiles for Nearest Centroid Cross-Domain Text Classification. In Proceedings of the IC3K 2014, Rome, Italy, 21–24 October 2014; Revised Selected Papers; Springer: Berlin/Heidelberg, Germany, 2014; Volume 553, pp. 50–67. https://doi.org/10.1007/978-3-319-25840-9_4.
86. Domeniconi, G.; Semertzidis, K.; López, V.; Daly, E.M.; Kotoulas, S.; Moro, G. A Novel Method for Unsupervised and Supervised Conversational Message Thread Detection. In Proceedings of the DATA 2016—5th International Conference on Data Science and Its Applications, Lisbon, Portugal, 24–26 July 2016; SciTePress: Setubal, Portugal, 2016; pp. 43–54. https://doi.org/10.5220/0006001100430054.
87. Domeniconi, G.; Moro, G.; Pasolini, R.; Sartori, C. A Comparison of Term Weighting Schemes for Text Classification and Sentiment Analysis with a Supervised Variant of tf.idf. In *Proceedings of the DATA (Revised Selected Papers)*; Springer: Berlin/Heidelberg, Germany, 2015; Volume 584, pp. 39–58. https://doi.org/10.1007/978-3-319-30162-4_4.
88. Moro, G.; Valgimigli, L. Efficient Self-Supervised Metric Information Retrieval: A Bibliography Based Method Applied to COVID Literature. *Sensors* **2021**, *21*, 6430. https://doi.org/10.3390/s21196430.
89. Domeniconi, G.; Moro, G.; Pagliarani, A.; Pasini, K.; Pasolini, R. Job Recommendation from Semantic Similarity of LinkedIn Users' Skills. In Proceedings of the ICPRAM 2016, Rome, Italy, 24–26 February 2016; SciTePress: Setubal, Portugal, 2016; pp. 270–277. https://doi.org/10.5220/0005702302700277.
90. Frisoni, G.; Moro, G.; Carbonaro, A. Learning Interpretable and Statistically Significant Knowledge from Unlabeled Corpora of Social Text Messages: A Novel Methodology of Descriptive Text Mining. In Proceedings of the DATA 2020—Proc. 9th International Conference on Data Science, Technology and Applications, Online, 7–9 July 2020; SciTePress: Setubal, Portugal, 2020; pp. 121–134.
91. Frisoni, G.; Moro, G. Phenomena Explanation from Text: Unsupervised Learning of Interpretable and Statistically Significant Knowledge. In *Proceedings of the DATA (Revised Selected Papers)*; Springer: Berlin/Heidelberg, Germany, 2020; Volume 1446, pp. 293–318. https://doi.org/10.1007/978-3-030-83014-4_14.

92. Frisoni, G.; Moro, G.; Carbonaro, A. Towards Rare Disease Knowledge Graph Learning from Social Posts of Patients. In Proceedings of the RiiForum, Athens, Greece, 15–17 April 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 577–589. https://doi.org/10.1007/978-3-030-62066-0_44.
93. Frisoni, G.; Moro, G.; Carbonaro, A. Unsupervised Descriptive Text Mining for Knowledge Graph Learning. In Proceedings of the IC3K 2020—12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Budapest, Hungary, 2–4 November 2020; SciTePress: Setubal, Portugal, 2020; Volume 1, pp. 316–324.
94. Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; Schneider, N. Abstract meaning representation for sembanking. In Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, Sofia, Bulgaria, 8–9 August 2013; pp. 178–186.
95. Frisoni, G.; Moro, G.; Carbonaro, A. A Survey on Event Extraction for Natural Language Understanding: Riding the Biomedical Literature Wave. *IEEE Access* **2021**, *9*, 160721–160757. https://doi.org/10.1109/ACCESS.2021.3130956.
96. Oepen, S.; Kuhlmann, M.; Miyao, Y.; Zeman, D.; Cinková, S.; Flickinger, D.; Hajic, J.; Uresova, Z. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Denver, CO, USA, 4–5 June 2015; pp. 915–926.
97. Abend, O.; Rappoport, A. Universal conceptual cognitive annotation (UCCA). In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Sofia, Bulgaria, 4–9 August 2013; pp. 228–238.
98. Frisoni, G.; Carbonaro, A.; Moro, G.; Zammarchi, A.; Avagnano, M. NLG-Metricverse: An End-to-End Library for Evaluating Natural Language Generation. In Proceedings of the 29th International Conference on Computational Linguistics, Gyeongju, Republic of Korea, 12–17 October 2022; International Committee on Computational Linguistics: Gyeongju, Republic of Korea, 2022 ; pp. 3465–3479.
99. Colon-Hernandez, P.; Havasi, C.; Alonso, J.; Huggins, M.; Breazeal, C. Combining pre-trained language models and structured knowledge. *arXiv* **2021**, arXiv:2101.12294.
100. Yin, Y.; Meng, F.; Su, J.; Zhou, C.; Yang, Z.; Zhou, J.; Luo, J. A novel graph-based multi-modal fusion encoder for neural machine translation. *arXiv* **2020**, arXiv:2007.08742.
101. Xu, M.; Li, L.; Wong, D.; Liu, Q.; Chao, L.S.; et al. Document graph for neural machine translation. *arXiv* **2020**, arXiv:2012.03477.
102. Song, L.; Gildea, D.; Zhang, Y.; Wang, Z.; Su, J. Semantic neural machine translation using AMR. *Trans. Assoc. Comput. Linguist.* **2019**, *7*, 19–31.
103. Huang, L.; Wu, L.; Wang, L. Knowledge Graph-Augmented Abstractive Summarization with Semantic-Driven Cloze Reward. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, 5–10 July 2020; Jurafsky, D., Chai, J., Schluter, N., Tetreault, J.R., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 5094–5107. https://doi.org/10.18653/v1/2020.acl-main.457.
104. Zhu, C.; Hinthorn, W.; Xu, R.; Zeng, Q.; Zeng, M.; Huang, X.; Jiang, M. Enhancing Factual Consistency of Abstractive Summarization. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, 6–11 June 2021; Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tür, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., Zhou, Y., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 718–733. https://doi.org/10.18653/v1/2021.naacl-main.58.
105. An, C.; Zhong, M.; Chen, Y.; Wang, D.; Qiu, X.; Huang, X. Enhancing Scientific Papers Summarization with Citation Graph. In Proceedings of the 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, 2–9 February, 2021; AAAI Press: Palo Alto, CA, USA, 2021; pp. 12498–12506.
106. Ji, X.; Zhao, W. SKGSUM: Abstractive Document Summarization with Semantic Knowledge Graphs. In Proceedings of the International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, 18–22 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8. https://doi.org/10.1109/IJCNN52387.2021.9533494.
107. Moro, G.; Ragazzi, L. Semantic Self-Segmentation for Abstractive Summarization of Long Legal Documents in Low-Resource Regimes. In Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Virtual Event, 22 February–1 March 2022; AAAI Press: Palo Alto, CA, USA, 2022; pp. 1–9.
108. Moro, G.; Ragazzi, L.; Valgimigli, L.; Freddi, D. Discriminative Marginalized Probabilistic Neural Method for Multi-Document Summarization of Medical Literature. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland, 22–27 May 2022; Association for Computational Linguistics: Dublin, 2022; pp. 180–189. https://doi.org/10.18653/v1/2022.acl-long.15.
109. Frisoni, G.; Italiani, P.; Boschi, F.; Moro, G. Enhancing Biomedical Scientific Reviews Summarization with Graph-based Factual Evidence Extracted from Papers. In Proceedings of the 11th International Conference on Data Science, Technology and Applications, DATA 2022, Lisbon, Portugal, 11–13 July 2022; Cuzzocrea, A., Gusikhin, O., van der Aalst, W.M.P., Hammoudi, S., Eds.; SCITEPRESS: Setubal, Portugal, 2022; pp. 168–179. https://doi.org/10.5220/0011354900003269.
110. Han, J.; Cheng, B.; Wang, X. Open domain question answering based on text enhanced knowledge graph with hyperedge infusion. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online, 16–20 November 2020; pp. 1475–1481.
111. Feng, Y.; Chen, X.; Lin, B.Y.; Wang, P.; Yan, J.; Ren, X. Scalable multi-hop relational reasoning for knowledge-aware question answering. *arXiv* **2020**, arXiv:2005.00646.

112. Yasunaga, M.; Ren, H.; Bosselut, A.; Liang, P.; Leskovec, J. QA-GNN: Reasoning with language models and knowledge graphs for question answering. *arXiv* **2021**, arXiv:2104.06378.
113. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 974–983.
114. Guo, Z.; Wang, H. A deep graph neural network-based mechanism for social recommendations. *IEEE Trans. Ind. Informatics* **2020**, *17*, 2776–2783.
115. Chen, C.; Zhang, M.; Ma, W.; Liu, Y.; Ma, S. Jointly non-sampling learning for knowledge graph enhanced recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xi'an, China, 25–30 July 2020; pp. 189–198.
116. Frisoni, G.; Moro, G.; Carlassare, G.; Carbonaro, A. Unsupervised Event Graph Representation and Similarity Learning on Biomedical Literature. *Sensors* **2022**, *22*, 3. https://doi.org/10.3390/s22010003.
117. Wu, L.; Chen, Y.; Shen, K.; Guo, X.; Gao, H.; Li, S.; Pei, J.; Long, B. Graph neural networks for natural language processing: A survey. *arXiv* **2021**, arXiv:2106.06090.