symmetry

MDPI

Article

A Survey on Knowledge Graph Embeddings for Link Prediction

Meihong Wang, Linling Qiu and Xiaoli Wang *

School of Informatics, Xiamen University, Xiamen 361000, China; wangmh@xmu.edu.cn (M.W.); qiulinling@stu.xmu.edu.cn (L.Q.)
* Correspondence: xlwang@xmu.edu.cn

Abstract: Knowledge graphs (KGs) have been widely used in the field of artificial intelligence, such as in information retrieval, natural language processing, recommendation systems, etc. However, the open nature of KGs often implies that they are incomplete, having self-defects. This creates the need to build a more complete knowledge graph for enhancing the practical utilization of KGs. Link prediction is a fundamental task in knowledge graph completion that utilizes existing relations to infer new relations so as to build a more complete knowledge graph. Numerous methods have been proposed to perform the link-prediction task based on various representation techniques. Among them, KG-embedding models have significantly advanced the state of the art in the past few years. In this paper, we provide a comprehensive survey on KG-embedding models for link prediction in knowledge graphs. We first provide a theoretical analysis and comparison of existing methods proposed to date for generating KG embedding. Then, we investigate several representative models that are classified into five categories. Finally, we conducted experiments on two benchmark datasets to report comprehensive findings and provide some new insights into the strengths and weaknesses of existing models.

Keywords: link prediction; knowledge graph embedding; knowledge graph completion; survey

Citation: Wang, M.; Qiu, L.; Wang, X. A Survey on Knowledge Graph Embeddings for Link Prediction. *Symmetry* **2021**, *13*, 485. https://doi.org/10.3390/sym13030485

Academic Editor: Theodore E. Simos

Received: 8 February 2021
Accepted: 5 March 2021
Published: 16 March 2021

1. Introduction

Knowledge graphs (KGs) have been widely used to store structured semantic information for tasks of artificial intelligence. Technically speaking, a knowledge graph is based on big data, which is one of the forms of big data applications. Many open KGs have been constructed, e.g., Freebase [1], DBpedia [2] and YAGO [3]. They often contain a large number of facts constructed using billions of entities and relations, which are represented as nodes and the edges linking these nodes, respectively. Each fact is represented as a triple (h, r, t), where h is a head entity, t is a tail entity and r is the relation between them. KGs have been applied in many areas, such as question answering [4], recommendation systems [5] and information retrieval [6]. However, KGs have self-defects; they are always incomplete. First, existing KGs are often incomplete, as it is difficult to incorporate all the concepts that humans have [6]. Second, real-world data are often dynamic and evolving, which leads to difficulty in constructing correct and complete KGs [7]. Therefore, it is a challenging task to automatically construct a more complete KG, which is often formulated as the link-prediction problem [8]. The goal of link prediction is to predict missing information (links or relations) between the entities in KGs. Figure 1 shows an illustrated example of link prediction. The solid lines in the left figure are existing relations, and the dotted lines are possible relations. The different colors in the right figure represent various possible relations, which are calculated by the link-prediction task.

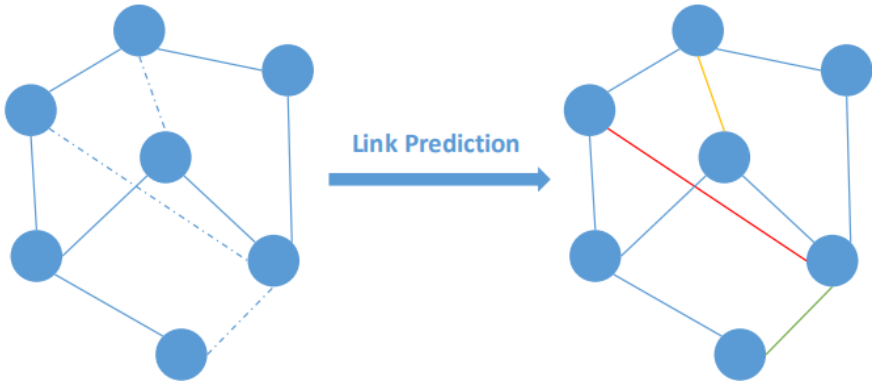

Figure 1. An example of link prediction.

To solve the link-prediction problem, various techniques have been proposed, including decomposition-based methods, path-based methods and embedding-based methods [9]. Decomposition-based models are based on potential semantic information, and their entity–relation triples are encoded in tensors [10]. They may involve many parameters, and such models may have low efficiency and poor scalability. Path-based models consider a path from a to b via a sequence of edges; the earliest models include random walks and the path-ranking algorithm (PRA) [11]. The larger the step size is, the larger the optimal solution space, but the computational complexity is higher. To address the above issues, embedding-based methods transform a KG into a low-dimensional vector space while preserving its underlying semantics [12]. Among them, knowledge graph embedding (KGE) models, which learn semantic representations of entities and relations, have significantly advanced the state of the art in the past few years [13]. Therefore, this paper focuses on a comprehensive survey of KGE models for link prediction.

KGE models embed entities and relations into a low-dimensional vector space while preserving the structure of the KG and its underlying semantic information. These models can be effectively applied to link prediction [14]. In this paper, we comprehensively survey existing KGE models and categorize them into three groups: translational-distance-based models, semantic-matching-based models and neural-network-based models. The first group is also denoted as a group of additive models, such as TransE [13], TransH [15], TransM [16] and TransR [17]. Inspired by word2vec [18], which allows word vectors to capture the semantic information of words with translation invariance, TransE [13] regards the relations in KGs as translation vectors. Given a triple (h, r, t), the relation r translates the head entity h to the tail entity t. It defines a scoring function to measure the correctness of the triple in the embedding space. However, these models are reported to have low expressive power without capturing semantic information [19]. The second group of multiplicative models includes DistMult [20] and Complex [21], which can outperform the additive models by capturing more semantic information [19]. These models first embed entities and relations into a unified continuous vector space and then define a scoring function to measure its authenticity. However, these early models only consider each individual fact, while their intrinsic associations are neglected, which is not sufficient for capturing deeper semantics for better embedding. The third group are neural-network-based models, such as ConvE [22], ConvKB [23], HypER [24], CompGCN [25], SACN [26] and CNN-BiLSTM [27]. These models consider the type of entity or relation, temporal information, path information and substructure information. The use of convolutional neural networks or attention mechanisms also helps to generate better embeddings.

At present, there are many surveys of KGE models, such as [6,28–33]. They summarized, analyzed and compared the relevant KGE models from different angles, such as the models themselves, the training strategies, and the research directions. Rossi et al. [28] classified models into three categories: tensor decomposition models, geometric models and deep learning models. For these three categories, they selected typical models for detailed description, experimental result analysis and comparison. However, there is no

overall classification and summary of the KGE models proposed in recent years in this paper, and the selected models are few, which cannot cover all types of KGE models. In particular, many KGE models fusing external information have been proposed in recent years, in which this information is diversified. However, this model does not better classify and summarize from the perspective of integrated information. In addition, the experiment of this survey was not reproduced in a unified environment configuration but used different coding frameworks, such as Python and C++. Dai et al. [19] described these models with two categories: triplet fact-based representation learning models and description-based representation learning models. Regarding additional information, they considered only these two aspects and also did not provide an overall table. They also conducted experiments on representative models and provided a detailed comparison, which is similar to Rossi et al. [28]'s. Ji et al. [31] divided the research on KGs into four categories: knowledge representation learning, knowledge acquisition, temporal KGs and knowledge-aware applications. We synthesized the previous surveys' ideas; we summarize these models of KG embedding proposed over nearly three years into a classification table, which is intuitive, and we analyze the correlations among these models from a more fine-grained perspective, which involves our five main lines. In addition, our experiments on some representative models were conducted in a unified environment, including the server type and programming language (pytorch). Finally, we compare and analyze the results in detail, including the performance and training time. In this survey paper, the focus is on the analysis of different embedding approaches, and their advantages and drawbacks in handling different challenges are highlighted. Moreover, a review of the applications of representative KGE models is provided, with experiments conducted specifically on the link-prediction task. The contributions of this paper are summarized as follows:

- This paper provides a theoretical analysis and comparison of existing KGE methods for generating KG embeddings for link prediction in KGs.
- Several representative models in each category are also analyzed and compared along five main lines.
- We conducted experiments on two benchmark datasets to report comprehensive findings and provide new insights into the strengths and weaknesses of existing models. We also provide new insights into existing techniques that are beneficial for future research.

The main contents of the rest of the article are as follows: Section 2 introduces the concept of the knowledge graph and knowledge graph embedding, as well as the definition of the link-prediction task; Section 3 mainly presents the two types of categories of models and a detailed introduction on representative models; Section 4 presents the experiment and comparative analysis of representative models; Section 5 is the conclusion.

2. Preliminaries and Problem Definition

2.1. Preliminaries

In essence, knowledge graphs (KGs) are semantic networks that reveal the correlations between entities, which have the abilities of analysis and reasoning like human beings. A knowledge graph is similar to a knowledge base, describing information from different perspectives. The knowledge graph tends to consider the graph structure, while the knowledge base tends to be displayed in the textual form of reasoning and explanation [31]. Their differences and connections are shown in Figures 2 and 3. Previous literature has proposed many definitions of KGs. In this paper, we cite a widely accepted definition proposed in [31], shown as Definition 1. Following previous literature, we mathematically define a KG as $G = (v, \varepsilon)$, where $v = \left\{v_1, v_2, \cdots, v_{|v|}\right\}$ is a set of nodes (entities) and $\varepsilon \subseteq v \times v$ is a set of edges (relations) [34]. An adjacency matrix $A \in R^{|v| \times |v|}$ shows where $A[i][j] = 0$ if $(v_i, v_j) \notin \varepsilon$; otherwise, $A[i][j] \in R_+$ represents the weight of the edge. A degree matrix $D \in R^{|v| \times |v|}$ is a diagonal matrix where $D[i][i] = \sum_{j=1}^{|v|} A[i][j]$ represents the degree of v_i.

A knowledge graph is composed of facts in the world, which exists in the form of triples (h, r, t), with h, r and t representing the head entity, relation and tail entity, respectively. The relations in the world are plural. For example, some relations are symmetric (e.g., marriage) while others are antisymmetric (e.g., filiation); some relations are the inverse of other relations (e.g., hypernym and hyponym); and some relations may be composed by others (e.g., my mother's husband is my father). It is critical to find ways to model and infer these patterns, i.e., symmetry/antisymmetry, inversion and composition, from observed facts in order to predict missing links [35].

(Albert Einstein, BornIn, German Empire)
(Albert Einstein, SonOf, Hermann Einstein)
(Albert Einstein, GraduateFrom, University of Zurich)
(Albert Einstein, WinnerOf, Nobel Prize in Physics)
(Albert Einstein, ExpertIn, Physics)
(Nobel Prize in Physics, AwardIn, Physics)
(The theory of relativity, TheoryOf, Physics)
(Albert Einstein, SupervisedBy, Alfred Kleiner)
(Alfred Kleiner, ProfessorOf, University of Zurich)
(The theory of relativity, ProposedBy, Albert Einstein)
(Hans Albert Einstein, SonOf, Albert Einstein)

Figure 2. Knowledge base.

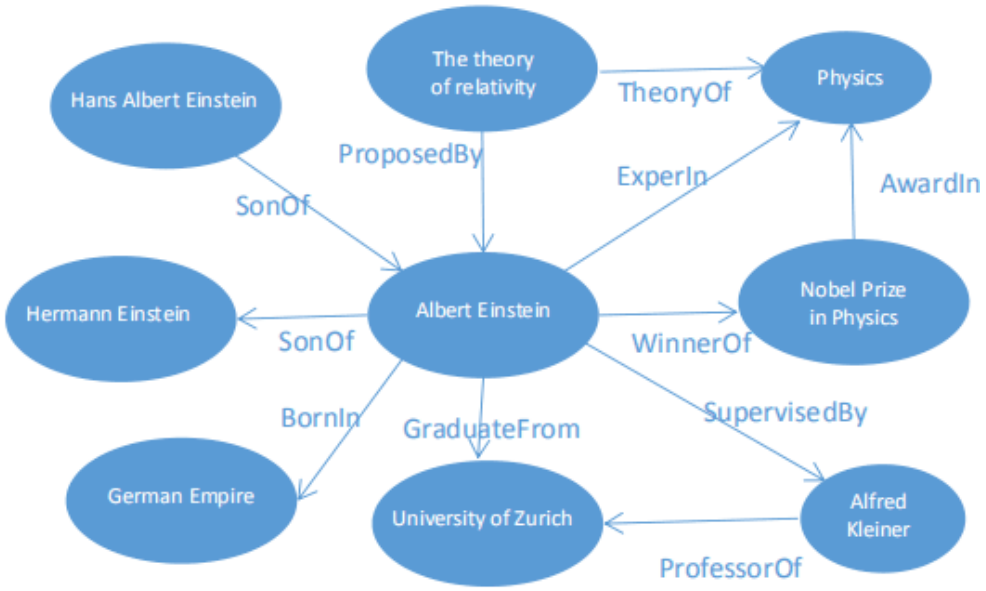

Figure 3. Knowledge graph.

In order to improve knowledge graphs, we use knowledge graph embedding (KGE) technology, which represents entities and relations as low-dimensional continuous vectors. We cite the definition of a KGE proposed in [31] as shown in Definition 2.

Definition 1 (Knowledge Graph (KG)). *A knowledge graph is a multirelational graph composed of entities and relations that are regarded as nodes and different types of edges, respectively.*

Definition 2 (Knowledge Graph Embedding (KGE)). *Knowledge graph embedding is a technology for mapping the content of entities and relations in a knowledge graph to continuous low-dimensional vector space.*

2.2. Link Prediction

Link prediction is one of the most common tasks for evaluating the performance of KGE, which has no formal definition. It has two subtasks: entity prediction, also called entity ranking, for predicting missing entities, i.e., predicting h given $(?, r, t)$ or t given $(h, r, ?)$, and relation prediction for predicting missing relations, i.e., predicting r given $(h, ?, t)$, where "?" represents a missing entity or relation for a triple (h, r, t). For link-prediction models, the goal is to predict the missing relations among entities using the existing relations in a KG. It replaces the relation of each test triple with all the relations in the KG to obtain the negative samples. It is also defined as an entity-sorting task. Then, it determines whether the new triple, which is not observed in the KG, is valid according to the scoring function. In KGE technology, the entities are generally regarded as vectors, while the relations are regarded as operations in the vector space, which are used to calculate the relationships between entities. First, the entities and relations are mapped into a continuous low-dimensional vector space. Then, a scoring function is defined to measure the reliability of the triples, and a higher score indicates that the triple is more likely to be true. Finally, a loss function is defined to optimize the total reliability of all the triples in the KG. For evaluation, it is a common practice to record the ranks of the correct answers in such ordered lists to see whether the correct answers are ranked before the incorrect ones. The commonly used evaluation metrics in the study are the mean rank (the average of the predicted ranks), mean reciprocal rank (the average of the reciprocal ranks), and Hits@n (the proportion of ranks larger than n).

2.3. Research Questions

As described above, there are many models for KGE, but there are still many challenges in obtaining better entity and relation embeddings for link prediction. Some (but not all) of the challenges found in the survey are as follows:

1. It is difficult to model the heterogeneity of graphs.
2. There are few studies on dynamic graphs, which are better able to reflect the real world.
3. How to incorporate prior knowledge to obtain deep semantics should be determined.
4. How to capture multi-hop neighbor information should be determined.
5. It has been argued that many models struggle to perform on hierarchical graphs such as WN18RR [8].

This paper focuses on the classification and comparative analysis of various models, aiming to determine the differences and breakthrough points of challenges to provide better guidance for future research.

3. Embedding Models for Link Prediction

We first performed a comprehensive investigation on the KGE models proposed in recent years. Then, we explored the three categories of KGE models for link prediction based on the investigation, including translation-distance-based models, semantic-matching-based models and neural-network-based models. These three categories reflect the mainstream models of KGE development in three main stages. The detailed models proposed in recent years and their categories are shown in Table 1. Then, we analyze how they work, and the challenges and the differences between them. Finally, in order to better study the correlations between various models (possibly cross-category models), we classify all these models from a more fine-grained perspective, proposing five main lines. The five

main lines can help readers to more intuitively understand the more in-depth correlations between the models.

Table 1. Models and their categories.

Categories	Subcategories	Models
Based on translation distance	TransE and its extensions	TransE [13], TransH [15], TransM [16], TransR [17], TransD [36], TransA [37], TranSparse [38], ManifoldE [39], STransE [40], TransX-FT [41], TransX-DT [42], TransHR [43], CirE [44], GTrans [45], TransCore [46], TransF [47], TransGH [48], AEM [49], EMT [50], TransX-SYM [51], TransMS [52], KGLG [53], TransL [54].
	Gaussian embedding	KG2E [55], TransG [56].
	Others	KGE Continual Learning [57], TorusE [58], QuatE [59], RotatE [35], HAKE [60], DeCom [61], MobiusE [62], QuaR [63].
Based on semantic information	No additional information	RESCAL [64], DistMult [20], Hole [65], ComplEx [21], ANALOGY [66], ComplEx-N3 [67], TuckER [68], TriModel [69], CrossE [70], HolEx [71], MEI [72].
	Fusing additional information	**Entity/relation types:** TKRL [73], SSE [74], Att-Model+Types [75], ETE [76], TransT [77], Bilinear+TR [78], TransET [79], KGE via weighted score [80], RecKGC [81], RPE [82]. **Relation paths:** PTransE [83], Att-Model+Types [75], TransP [84], PaSKoGE [85], PTransD [86], DPTransE [87], abstract paths for KGC [88], ELPKG [9], PTranSparse [89], CoKE [90], RPE [82], RW-LMLM [91], KBAT [92], GAATs [93], path-based reasoning [27]. **Textual descriptions:** DKRL [94], SSP [95], KDCoE [96], CATT [97], textural association [98], open-world extension for KGC [99], EDGE [100], TransW [101]. **Logic rules:** KALE [102], lppTransX [103], BiTransX+ [104], ELPKG [9], X-lc [105], RUGE [106], TARE [107], logic rule powered KGE [108], SoLE [109], GPFL [110], CBR [111], probabilistic case-based reasoning [112]. **Entity attributes:** KBLRN [113], TransEA [114], LiteralE [115], MARINE [116], AKRL [117]. **Temporal:** Time-aware link prediction [118], co-evolution of event and KGs [119], Know-Evolve [120], iTransA [121], HyTE [122], ATiSE [123], QCHyTE [124], TDG2E [125], T-GAP [126], TeMP [127]. **Structure:** GAKE [128], APP [129], ORC [130], KGC by embedding correlations [131], TCE(2017) [132], TCE(2018) [133], Graph2Seq [134], HRS [135], SA-KGE [136], TransS [137], S2E [138], AKSE [139], AggrE [140]. **Constraints:** ComplEx+NNE+AER [141], RPE [82], CoRelatE [142]. **Negative sampling:** TransR-PNC [143], TSLRF [144], TransE-ANS [145]. **Fake triples:** KGE via fake triples [146], AWML [147]. **Order:** RKGE [148], RW-LMLM [91]. **Concepts:** TransC [149], KEC [150], TransFG [151]. **Background:** SimplE [152], SimplE+ [153].
Based on neural network	No additional information	SME [154], NTN [155], MLP [156], NAM [157], R-GCNs [158], DSKG [159], SENN [160], TBNN [105], NTransGH [161], ConvE [22], ConvKB [23], ConnectER [162], FRS [163], HypER [24], CompGCN [25], TransGate [106], R-MeN [164], TransGCN [165], VR-GCN [166], InteractE [167], KBAT [92], BDRAN [168], BDR+CA [169], wRAN [170], RA-GCN [171], path-based reasoning [72], MultiView [172], KGEL [173], GAEAT [174], GRL [175].

Table 1. *Cont.*

Categories	Subcategories	Models
	Fusing additional information	ProjE [176], ProjFE [177], ProjR [178], SACN [26], CNN-BiLSTM [27], ConMask [179], MIA Model [180],TKGE [181], a semi-supervised model for KGE [182], GAN based on Wasserstein [183], LAN [184], TransAt [185], KANE [186], context-aware temporal KGE [187], CACL [188], DKGE [189], G-DRNN [190], MTKGNN [191], LogicENN [192], TECRL [193], PATHCON [194], HARP [195].

3.1. Translation-Distance-Based Models

These models are additive models, which usually use distance-based functions to define the scoring function for link prediction. After the relation is translated, the distance is used to measure the credibility between two entities. For this category, we divide these models into three subcategories: TransE and its extensions, Gaussian embedding and others. The models of TransE and its extensions are those that extend TransE [13]. TransE [13] is one of the earliest models for link prediction, which uses the relation for translating the head entity to a tail entity. There are many extensions of TransE. The models of Gaussian embedding are those considering the uncertainties of entities and relations by using a probability function. The models of others are those similar to TransE; they do not belong to distance-based models in essence, but their idea is the same as that of TransE, such as using rotations, Lie groups, quaternions and MobiusE rings for translation.

TransE [13] is a well-known, early and simple model that regards a relation as a translation from a head entity to a tail entity. It uses a distance scoring function as follows: $f(h,t) = \|h + r - t\|_{\frac{1}{2}}$. TransE is the earliest translation-based embedding model, and it has difficulty dealing with multirelational graphs; it is limited by its simple translation operation as well as its lack of a discrimination policy for all kinds of relations. In recent years, many variants of TransE have been proposed, such as **TransH** [15] and **TransR** [17]. TransH introduces a hyperplane, and TransR uses a relation-specific space to handle different relations, excavating more semantic information. The **TransHR** [43] model focuses on the embedding of hyperrelational data. These models are similar in nature; the only improvement is in translating the head entities to tail entities.

TransMS [52] regards the head entity, relation and tail entity as a subject, predicate and object, respectively, as in a sentence. In this way, it considers the semantics between the head entity and the relation as well as the semantics between the relation and the tail entity, which is not done in previous models. It uses the nonlinear function $p(e,r)$ (this is the tanh function) instead of a linear function to translate the semantics, that is, to translate both h and r to t, obtaining the final tail-entity embedding vector as $t_\perp = P(p(h,r),t)$ and obtaining the head-entity embedding vector from the converse semantic transfer $h_\perp = P(p(-t,r),h)$. In addition, it defines the bias vector $\alpha \cdot g(h,t)$ to transmit the semantic information of both h and t to r, where α is an added dimension for relations and $g(h,t)$ is a function concerning h and t, and then obtains the final relation embedding vector as $r_\perp = G(r, \partial \cdot g(h,t))$. It chooses the tanh function as the nonlinear function $p(e,r)$, so the entity and relation embedding vectors are as Equations (1)–(3):

$$h_\perp = tanh(-t \otimes r) \otimes h = -tanh(t \otimes r) \otimes h \tag{1}$$

$$t_\perp = tanh(h \otimes r) \otimes t \tag{2}$$

$$r_\perp = r + \partial \cdot (h \otimes t) \tag{3}$$

Finally, the score function is defined as in Equation (4):

$$\| - \tanh(t \otimes r) \otimes h + r + \partial \cdot (h \otimes t) - \tanh(h \otimes r) \otimes t\|_2^2 \tag{4}$$

Gaussian embedding models consider the uncertainties of entities and relations and model them as random variables. There are two models of this kind: **KG2E** [55] and **TransG** [56]. KG2E regards entities and relations as random vectors drawn from multivariate Gaussian distributions and scores a triple using the distance between the two random vectors. The TransG model also models entities with Gaussian distributions, using a mixture of Gaussian distributions to obtain multiple semantics. Gaussian embedding models take the uncertainties of the entities and relations into account, but this results in a complex model. Other methods such as **TorusE** [58], **QuatE** [59], **RotatE** [35] and **MobiusE** [62], using Lie groups, quaternions, rotations and MobiusE rings, are similar to TransE [13]. They do not belong to distance-based models in essence, but their idea is the same as that of TransE. Given a triple (h, r, t), they all map the head entity h to the tail entity t through the relation r, but the specific mapping operations about r are different, so this paper puts them in this category.

RotatE [35] can model and infer various relation patterns, including symmetry (antisymmetry), inversion and composition. Specifically, it defines each relation as a rotation from the head entity to the tail entity in a complex vector space. It provides a novel self-adversarial negative-sampling technique for efficiently and effectively training RotatE. Inspired by Euler's identity, RotatE maps the head and tail entities h and t to complex embeddings such as $\mathbf{h}, \mathbf{t} \in \mathbb{C}^k$; then, it defines a mapping function induced by each relation r as an elementwise rotation from the head entity h to the tail entity t. For the triple $(\mathrm{h}, \mathrm{r}, \mathrm{t})$, RotatE yields $\mathbf{t} = \mathbf{h} \circ \mathbf{r}$, , where $|r_i| = 1$ and $\circ$ is the Hadamard (or elementwise) product. For each element in the embedding, we have $t_i = h_i r_i$. Here, this method constrains the modulus of each element of $\mathbf{r} \in \mathbb{C}^k$, i.e., $r_i \in \mathbb{C}$, to be $|r_i| = 1$. By doing so, r_i takes the form $e^{i\theta_{r,i}}$, which corresponds to a counterclockwise rotation by $\theta_{r,i}$ radians about the origin of the complex plane and affects only the phases of the entity embeddings in the complex vector space. This is the origin of the "rotation". The distance function of RotatE is $d_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|$. By defining each relation as a rotation in the complex vector space, RotatE can model and infer all three types of relation patterns introduced above. It summarizes the pattern modeling and inference abilities of several models as shown in Table 2.

Table 2. The pattern modeling and reasoning ability of several models.

Model	Symmetry	Antisymmetry	Inversion	Composition
SE	✗	✗	✗	✗
TransE	✗	✓	✓	✓
TransX	✓	✓	✗	✗
DistMult	✓	✗	✗	✗
ComplEx	✓	✓	✓	✗
RotatE	✓	✓	✓	✓

HAKE [60] considers the semantic hierarchy of entities by using polar coordinates, in which concentric circles can naturally reflect the hierarchy. Additionally, it considers whether the entities are at the same level of the hierarchy, which consists of two parts, the modulus and the phase, used to distinguish the two types of entities. The modulus focuses on entities at different levels of the hierarchy; modulus information is used to model the entities as follows: $h_m \circ r_m = t_m, h_m, t_m \in R^k, r_m \in R^k_+$. Here, h_m, r_m, t_m are the head-/tail-entity embedding and the relation embedding, and $\circ$ is the Hadamard (or elementwise) product. The corresponding distance function is $d_{r,m}(h_m, t_m) = \|h_m \circ r_m - t_m\|_2$. The phase focuses on entities at the same level of the hierarchy; phase information is used to distinguish the entities as follows: $(h_p + r_p) \bmod 2\pi = t_p, h_p, r_p, t_p \in [0, 2\pi)^k$. Here, h_p, r_p, t_p are the head-/tail-entity embedding and the relation embedding. The corresponding distance function is $d_{r,p}(h_p, t_p) = \left\|\sin\left((h_p + r_p - t_p)/2\right)\right\|_1$. The above two parts correspond to the radial and angular parts of polar coordinates, respectively. This method

maps each entity e to $[e_m; e_p]$, where e_m, e_p are generated by the two parts and $([h_m]_i, [h_p]_i)$ is a 2D point in polar coordinates. The final score function is as in Equation (5):

$$\begin{aligned} f_r(h,t) &= d_r(h,t) = -d_{r\cdot m}(h,t) - \lambda d_{r,p}(h,t) \\ &= -\|h_m \circ r_m - t_m\|_2 - \lambda \left\|\sin\left((h_p + r_p - t_p)/2\right)\right\|_h \end{aligned} \tag{5}$$

All of these translation-based models use the margin-based pairwise ranking loss function to measure the scores of triples $(\mathbf{h}, \mathbf{r}, \mathbf{t})$ as the L_n distance between $\mathbf{h} + \mathbf{r}$ and $\mathbf{t}$. Therefore, the differences among these models are in how they translate a head entity $\mathbf{h}$ to a tail entity $\mathbf{t}$ by the relation $\mathbf{r}$. Their loss function is as in Equation (6):

$$\Lambda = \sum_{(h,r,t)\in D} \sum_{\left(h',r',t'\right)\in D'} \left[f_t(\mathbf{h},\mathbf{t}) + \gamma - f_r\left(\mathbf{h}',\mathbf{t}'\right)\right]_+ \tag{6}$$

where $[\chi]_+$ denotes the positive value of χ, and $\gamma > 0$ is a margin hyperparameter. Although the embedding operation of this type of model is relatively simple, the expressiveness is not sufficient due to the direct addition of $\mathbf{h} + \mathbf{r}$ and the lack of different semantic information.

3.2. Semantic Information-Based Models

Semantic information-based models usually use similarity-based functions to define scoring functions for traditional semantic-matching models or introduce additional information to mine more knowledge for recently proposed models. Thus, these models are divided into two subcategories: models with additional information and models without additional information. The latter traditional models match the latent semantics of entities and relation embeddings to measure the plausibility of a triple. The traditional semantic-based models focus only on the information of the triple itself and do not fuse any additional information, as **DistMult** [20] and **Complex** [21] do. These models suffer from higher computational complexity. The former recently proposed models fuse various additional information to obtain better performance to mine deeper semantic information at the bottoms of graphs. The additional information includes path information, order information, concepts, entity attributes, entity types and so on.

TransW [101] uses word embeddings to compose knowledge graph embeddings and learns a function mapping from the word embedding space to the knowledge embedding space. Entities and relations are represented in the form of linear combinations of word embeddings in this model, which can detect unknown facts, as Equations (7)–(9):

$$h = \sum_{i=0}^{n} h_i \otimes w_{hi} + b_h \tag{7}$$

$$t = \sum_{i=0}^{m} t_i \otimes w_{ti} + b_t \tag{8}$$

$$r = \sum_{i=0}^{p} r_i \otimes w_{ri} + b_r \tag{9}$$

where n, m, p are the numbers of words in entities h and t and relations r; $\mathbf{h}_i, \mathbf{r}_i, \mathbf{t}_i$ are the word embeddings for the i-th word in the corresponding h, r, t, respectively; w_{hi}, w_{ri}, w_{ti} are the i-th connection vectors for $\mathbf{h}, \mathbf{r}, \mathbf{t}$, respectively; $\otimes$ denotes the Hadamard product; and $b_h, b_r, b_t \in R^k$ are the bias parameters. The score function is as in Equation (10):

$$\left\| \left(\sum h_i \otimes w_{hi} + b_h\right) + \sum r_i \otimes w_{ri} - \left(\sum t_i \otimes w_{ti} + b_t\right) \right\|_{\frac{l_1}{2}}^2 \tag{10}$$

The **TransC** [149] model combines structural information with entity concepts (referring to the categories of entities) to improve KGE models and introduces a novel type of

semantic similarity to measure the distinctness of entity semantics by using the concept information. The relations in this model consist of two concept sets: the head concept set C_r^{head} and tail concept set C_r^{tail}. The semantic similarity of a relation and a head entity and of the relation and a tail entity is used to measure the distinction of entity semantic with concept information that is defined as in Equations (11) and (12), separately:

$$sim(r_{head}, h) = \frac{\left|C_r^{head} \cup C_h\right|}{\left|C_r^{head}\right|} \tag{11}$$

$$sim(r_{tail}, h) = \frac{\left|C_r^{tail} \cup C_t\right|}{\left|C_r^{tail}\right|} \tag{12}$$

The semantic similarity of the head entity and tail entity is as in Equation (13):

$$sim(h, t) = \frac{|C_h \cup C_t|}{|C_h|} \tag{13}$$

TransC regards each entity concept as a concept vector and each entity as a set of concept vectors, and the likelihood of a vector representation for the triple $\mathrm{P}(fact, h, r, t)$ (it represents the possibility that triple (h, r, t) is a fact, which means that the triple (h,r,t) satisfies the $h + r \approx t$ principle) is defined as in Equation (14):

$$\mathrm{P}(fact, h, r, t) = \sum_{i=1}^{|C_h|} \sum_{j=1}^{|C_t|} w_{h,i} w_{t,j} f_r(h_i, t_j) \tag{14}$$

where $|C_h|$, $|C_t|$ are the number of concepts of the head entity h and tail entity t; $\left\{w_{h,1}, \ldots, w_{h,|C_h|}\right\}, \left\{w_{t,1}, \ldots, w_{t,|C_t|}\right\}$ are the distributions of random variables of h and t; and $f_r(h_i, t_j)$ is the likelihood of the component with the i-th concept vector h_i of the head entity h and the j-th concept vector t_j of the tail entity t. The score function is defined as follows: $f_r(h_i, t_j) = \left\|h_i + r - t_j\right\|_l$.

PTransE [83] is a translation-based model, and it introduces contextual information by using multiple-step relation paths to extend TransE, which treats relation paths as transformations of representation learning between entities. TransE regards a relation as a translation vector of the head- and tail-entity vector. Its scoring function is defined as $E(h, r, t) = \|h + r - t\|$. PTransE integrates another score item into the triple's score for the multiple-step relation path, given as $G(h, r, t) = E(h, r, t) + E(h, P, t)$. $E(h, r, t)$ is defined as in TransE, modeling the relations in the triple (h, r, t) between entities and relations directly; $E(h, P, t)$ models multiple-step relationships between the entities and relations of each triple (h, r, t) in a multiple-step path, defined as $E(h, P, t) = \frac{1}{Z} \sum_{p \in P(h,t)} R(p|h, t) E(h, p, t)$, where $\sum_{p \in P(h,t)} R(p|h, t)$ is a normalized factor, for measuring the reliability of the relational path, and PTransE proposes a path-constraint resource allocation (PCRA) algorithm for it. $E(h, p, t)$ is used as a scoring function to define the triple (h, r, t), differing from TransE in leveraging a semantic combinatorial model recurrent neural network (RNN) to combine the relational path p, defined as $E(h, p, t) = \|p - (t - h)\| = \|p - r\| = E(p, r)$. Moreover, PTransE considers the relation path in only one direction. To address this problem, bidirectional relations including both the forward and the inverse direction are added to the KGs, such as $e_1 \xrightarrow[T]{BornInCity} e_2 \xrightarrow[T]{CityOfCountry^{-1}} e_3$ (BornInCity and CityOfCountry both represent relations, both constituting bidirectional relations. CityOfCountry is the inverse relation of BornInCity. If A is the BornInCity of B, then B is the CityOfCountry of A.). To improve the computational efficiency, the path length is limited to a maximum of three steps, and only the relationship paths with reliability scores greater than 0.01 will be selected.

Bilinear+TR [78] improves RESCAL [64] by introducing a regularization factor into the loss function, which is used to take entity types into account. In RESCAL, entities are expressed as vectors $x \in \mathbb{R}^d$, and relations are expressed as matrices $W \in \mathbb{R}^{d \times d}$. Then, the triple (h, r, t) is represented as a score given by $s_c(s, r, t) = x_s^T W_r x_t$. This is similar to tensor factorization; these vectors and matrices are learned by a loss function that compares a correct triple with an incorrect triple. RESCAL uses a max-margin loss function as in Equations (15) and (16):

$$J(\Theta) = \sum_{i=1}^{N} \sum_{t' \in N(t)} \mathrm{mm}\left(\sigma(s_{c_i}), \sigma\left(s'_{c_i}\right)\right) \tag{15}$$

$$\mathrm{mm}\left(\sigma(s_{c_i}), \sigma\left(s'_{c_i}\right)\right) = \max\left[0, 1 - \sigma(s_{c_i}) + \sigma\left(s'_{c_i}\right)\right] \tag{16}$$

where there are N positive instances, and the positive and negative instances are scored as $s_{c_i} = s_c(s_i, r_i, t_i)$ and $s'_{c_i} = s_c(s_i, r_i, t'_i)$, respectively. N_t is the set of incorrect targets, and σ is the sigmoid function. In bilinear+TR, let s_{cat} be the type of entity s and r_{cat} be the relation between s and s_{cat}. For a query q with head entity s and tail entity t, the regularizer is defined as in Equation (17):

$$\begin{aligned} R(\Theta, q) := & \\ & \sum_{\substack{s'_{cat} \in N(s_{cat}) \\ s_{cat} \in T(s_{cat})}} mm\left(\sigma_{s_c}(s, r_{cat}, s_{cat}), \sigma_{s_c}\left(s, r_{cat}, s'_{cat}\right)\right) \\ + & \sum_{\substack{t'_{cat} \in N(t_{cat}) \\ t_{cat} \in T(t_{cat})}} mm\left(\sigma_{s_c}(t, r_{cat}, t_{cat}), \sigma_{s_c}\left(t, r_{cat}, t'_{cat}\right)\right) \end{aligned} \tag{17}$$

where $T(s_{cat})$ and $T(t_{cat})$ are the sets of correct categories for head s and tail t, respectively. mm is the maximum margin loss described above. Then, the complete objective function is minimized as in Equation (18):

$$J(\Theta) = \sum_{i=1}^{N} \sum_{t'_i \in T(q_i)} \mathrm{mm}\left(q_i, t_i, t'_i\right) + \alpha R(\Theta, q_i) \tag{18}$$

where the hyperparameter α, $\alpha \geq 0$, controls the impact of the regularizer terms and $T(q_i)$ is the set of negative targets for query q_i, where q_i corresponds to query $(s_i, r_i, ?)$.

RW-LMLM is a novel approach for link prediction that consists of a random walk algorithm for KGs (RW) and a language model-based link-prediction model (LMLM). The paths output from RW are regarded as pseudosentences for LMLM training. RW can capture the semantic and syntactic information in KGs by considering the entities, relations and order information of the paths, in contrast to DeepWalk (which considers only the entities and relations). Therefore, the entities in the paths are in head-to-tail form, and the relations are in the middle, for example, $e_0 \xrightarrow{r_0} e_1 \xrightarrow{r_1} \cdots \xrightarrow{r_{l-1}} e_l$. LMLM uses a multilayer transformer decoder language model instead of word2vec models (continuous bag-of-words encoders or skip-grams). The standard language model usually defines a probability distribution over a sequence of tokens: $P(w_1, w_2, \ldots, w_n) = \prod_i P(w_i \mid w_1, \ldots, w_{i-1})$. The goal of language modeling is to maximize this probability. The conditional probabilities $P(w_i \mid w_1, \ldots, w_{i-1})$ can be learned by neural networks. The objective of RW-LMLM is to maximize the following probability as in Equation (19):

$$P(e_1, e_2, \ldots, e_l) = \prod_{i=1}^{l} P(e_i \mid (e_0, r_0), \ldots, (e_{i-1}, r_{i-1})) \tag{19}$$

LiteralE [115] introduces the literature information, giving priority to numerical literature. A general model with additional information adds a text-dependent merging term to the output of the scoring function to merge text indirectly. LiteralE directly incorporates literature information through a learnable parameterized function. There are two forms of triples in LiteralE: relations between entities and relations of entities and literals. This method obtains global information via the latent feature model, learning low-dimensional, latent representations, that is, embeddings. To evaluate the results, the basic dataset was extended here by adding literal information, and experiments were carried out on the basic models DistMult, ComplEx and ConvE. Except for embedding entities for transformation through a core function $g : R^H \times R^{N_d} \rightarrow R^H$, the scoring function is the same as in the basic models; that is, it transforms the entities into $e_i^{lit} = g(e_i, l_i)$. The definition of function g is critical for LiteralE. In addition to having learnability and flexibility, it should also be able to independently determine whether additional information is useful so that a judgment of whether to merge or ignore it can be made. To define g, LiteralE is inspired by the RNN gating strategy, and the gated recurrent unit (GRU) strategy is used to transform entities to ensure that the output vector and entity embedding have the same dimension. However, the model introduces a certain parameter overhead, which is proportional to the number of relationships in the KG.

Since the traditional model was put forward, semantic-matching-based models have been continuously mining deeper semantic information, including path information, entity types, contextual information and so on, to obtain more accurate representations for KGE.

3.3. Neural Network-Based Models

Neural networks have been a promising solution in many fields and have the abilities of self-study, associative storage and high-speed optimization. Traditional distance-based and semantic-matching-based models cannot meet the requirements of KGE. Recently, to obtain better and more effective entity and relation embeddings, a neural-network model was also introduced into KGE to propagate neighborhood information. These models are also divided into two subcategories: models with additional information and models without additional information. The additional information also includes path information, order information, concepts and so on.

ConvE [22] is the first model to use the convolutional neural network (CNN) framework for KG completion. Compared with fully connected neural networks, CNNs capture complex relationships with very few parameters by learning nonlinear features. ConvE uses embedded 2D convolution to predict missing links in KGs, which is the simplest multilayer convolutional network structure in these models for link prediction. Notably, 2D convolution outperforms 1D convolution in terms of extracting feature interactions between two embeddings. ConvE obtains local relationships in different dimensions between entities through a convolution layer and a fully connected layer, which ignores the global relationships between triple embeddings of the same dimension. ConvE first reshapes the head-entity embedding and relation embedding, concatenating them into an input matrix for the 2D convolution layer, which then returns a feature map tensor. Then, the tensor is vectorized and projected into k-dimensional space through a linear transformation parameterized by the matrix W and is finally matched with the tail-entity embedding through an inner product. Its scoring function is as in Equation (20):

$$\psi_r(e_s, e_o) = f(vec(f([\overline{e_s}; \overline{r_r}] * w))W)e_o \tag{20}$$

where $r_r \in R^k$ is a relation parameter depending on r, and $\overline{e_s}$ and $\overline{r_r}$ are the 2D reshapings of the head-entity embedding and relation embedding, respectively. The most important characteristic of ConvE is that the score it produces is defined by an embedded 2D convolution. Moreover, ConvE introduces a 1-N scoring program that takes a head-entity–relation pair and matches all the tail entities simultaneously, which is different from other models that use the 1-1 scoring program; thus, much evaluation time is saved.

Removing the reshaping operation from ConvE, **ConvKB** [23] uses 1D convolution to maintain the translation characteristics of TransE, which is sufficient for capturing the global relationships and transitional characteristics between entities. It represents the k-dimensional embedding of each triple (v_h, v_r, v_t) as a three-column matrix $A = [v_h, v_r, v_t] \in R^{k\times 3}$ and then feeds it into the convolutional layer, where there are multiple filters of the same 1X3 shape with the ability to extract the global relationships among the same-dimensional entries of an embedding triple. These filters operate on each row of the input matrix to obtain different feature maps: $\mathbf{v} = [v_1, v_2, \ldots, v_k] \in R^k$, where $v_i = g(\omega \cdot A_{i,:} + b)$, $A_{i,:} \in R^{1\times 3}$ is the i-th row of A; $\omega \in R^{1\times 3}$ is the filter used to examine the global relationships between the same-dimensional entries of embedding triples and to retrieve the transitional characteristics in transition-based models; $b \in R$ represents the bias; and g is the activation function. Then, these feature maps are concatenated into a triple feature vector and calculated with a weight vector $\mathbf{w}$ via the dot product to obtain the score of the triple, which is used to determine the validation of the triple. Its scoring function is as in Equation (21):

$$f(h, r, t) = concat(g([v_h, v_r, v_t] * \Omega)) \cdot \mathbf{w} \tag{21}$$

where Ω is a set of filters, Ω and $\mathbf{w}$ are shared parameters, $*$ is the convolution operator, and *concat* is the concatenation operator.

HypER [24] introduces hypernetworks based on ConvE to generate convolutional filter weights for each relation. The hypernetworks can be used to achieve weight sharing across layers and dynamically synthesize weights given inputs. The differences between HypER and ConvE are as follows: (1) ConvE uses 2D filters to construct convolution operators for the entity and relation embeddings after reshaping and concatenating, while HypER uses 1D relation-specific filters to handle entity embeddings, which simplifies the interaction between entities and relational embeddings. (2) The interaction between entities and relations in ConvE is affected by how they are reshaped and concatenated before being fed into the convolutional layers, while HypER uses a convolution operator for head-entity embeddings with a set of relation-specific filters F_r, which is created by hypernetwork H from the relation embeddings. The hypernetwork is a fully connected layer. (3) The feature maps obtained from ConvE and HypER by the convolution operator are all projected into a k-dimensional space and vectorized; however, the difference is that ConvE uses a linear transformation parameterized by $\mathbf{w}$, while HypER uses a weight matrix W to which the ReLU activation function is applied. (4) Finally, both methods calculate the tail-entity embeddings via the inner product to obtain a scoring vector for matching. The scoring function is as in Equation (22):

$$\begin{aligned} \phi_r(e_1, e_2) &= f(vec(e_1 * F_r)W)e_2 \; = \\ & f\Big(vec\Big(e_1 * vec^{-1}(w_r H)\Big)W\Big)e_2 \end{aligned} \tag{22}$$

where the *vec* operator transforms the matrix to a vector, the vec^{-1} operator reshapes a vector to a matrix, and the nonlinearity f is chosen to be a rectified linear unit (ReLU).

In summary, the vector embedding of each relation is projected through a fully connected layer in HypER, the result of which is reshaped, and then, a set of convolutional filter weight vector relations is provided for each layer. HypER implements nonlinear (quadratic) combinations of entity and relation embeddings rather than the linear combinations (weighted sums) in ConvE, which gives HypER the advantages of a much richer expressive capacity and fewer parameters. HypER can also be regarded as a factorization model.

Previous models introduced contextual information directly through one-way paths or text information, which ignores the important effects of various connection patterns between entities and unnecessary paths or unimportant words; this weakens the embedding of entities, resulting in the insufficient capture of the relationships between triples.

R-GCN [158] utilizes a relational graph convolutional network (GCN) to model highly multiple-relational data explicitly, restoring the information of the 1-hot neighborhood facts around the entity. For link-prediction tasks, a relational graph convolutional network (R-GCN) can be regarded as an autoencoder consisting of an encoder and a decoder. The encoder generates a latent feature representation of the entity, while the decoder scores the triple according to the representation generated by the encoder. In the encoder, there is an R-GCN using the idea of a graph convolutional network, which can fully encode the structural information of the entities by considering all types of relations among the entity connections, including the in and out relations. The forward update of the entity is as in Equation (23):

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in N_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right) \tag{23}$$

where N_i^r is the set of neighbors of node i under relation $r \in R$ and $c_{i,r}$ is a problem-specific normalization constant that can be learned or set to a number in advance (such as $c_{i,r} = \left|N_i^r\right|$). In an R-GCN, a base decomposition and block diagonal decomposition are proposed to solve the problem of the height parameters. In the decoder, the DistMult factorization model is selected as the scoring function, and each relationship is associated with a diagonal matrix. Then, the scoring function of the triples is as follows: $f(s,r,o) = e_s^T R_r e_o$.

ProjE [176] complements the missing information in a KG by learning the joint embedding of the entities and edges and making modifications to the loss function, thus improving the KGE model through trivial changes to the network architecture and eliminating complex feature engineering. By means of a learned combinatorial operator, the embedding vectors of head entities and relations are combined into a target vector, which also distributes the projection of the candidate entities to obtain the order list, and the top-ranking candidates are the correct entities. Compared with TransE, ProjE saves many transformation matrix calculations because of the combination operations, which are defined as $e \oplus r = D_e e + D_r r + b_c$, where D_e, D_r are diagonal matrices that serve as the global entity and relationship weights, respectively, and $b_c \in R^k$ is the combination bias. Then, the embedding projection function is defined as $h(e,r) = g\left(W^c f(e \oplus r) + b_p\right)$, where f and g are activation functions, $W^c \in R^{s\times k}$ is the candidate-entity matrix (s is the number of candidate entities), b_p is the projection bias, and $h(e,r)$ represents the ranking score vector, whose elements measure the similarity between the candidate entity in W^c and the combined input embedding $e \oplus r$. TransE defines the combination operation as addition and the similarity operation as distance. Similarly, ProjE defines the combination operation as a global linear layer and the similarity operation as a dot product. Moreover, it uses a collective ranking loss for the list of candidate entities (or relations) with two proposed methods: ProjE-pointwise and ProjE-listwise. The former uses sigmoid and tanh as the activation functions for g and f, respectively. Therefore, the ranking score is defined as $h(e,r)_i = sigmoid\left(W_{[i,:]}^c \tanh(e \oplus r) + b_p\right)$, where $W_{[i,:]}^c$ represents the i^{th} candidate in the candidate-entity matrix. The latter uses softmax and tanh as the activation functions, and its ranking score is defined as in Equation (24):

$$h(e,r)_i = \frac{\exp\left(W_{[i,:]}^c \tanh(e \oplus r) + b_p\right)}{\sum_j \exp\left(W_{[i,:]}^c \tanh(e \oplus r) + b_p\right)} \tag{24}$$

ProjE improves the loss function to Equation (25):

$$\Gamma = -\sum_{i\in\{i|y_i=1\}} \log(h(e,r)_i) - \sum_m E_{j\sim P_y} \log\left(1 - h(e,r)_j\right) \tag{25}$$

where e and r are the input embedding vectors, $y \in R^s$ is a binary label vector such that $y_i = 1$ means that candidate i is a positive label, and m is the number of negative samples drawn from a negative candidate distribution $E_{j \sim P_y}$.

ProjR [178] takes into account the diversity of structures and represents entities in terms of their different relational contexts and different entity locations; it combines TransR and ProjE to obtain these two representations by defining a unique combination operator for each relation. It can model N-1, N-N, 1-N-1 and one-relation circle structures with the advantage of a combination operator design. Before the similarity operator, TransR projects the representation of the head and tail entities into the relation-specific space through a relational projection matrix. ProjE projects the candidate-entity vectors into the representation space of the input entity relation pairs. Similarly, ProjR defines the scoring function in two parts: a combination operator and a similarity operator. To obtain representations of entities in different relational contexts, a combination operation $C_r(h)$ is applied to each relation r: $C_r(h) = c_{hr} = \tanh(D_r^e h + D_r^r r + b_c)$, where $D_r^e \in R^{d \times d}$ is a diagonal matrix defined for the linear transformation of the head entity under relation r, $D_r^r \in R^{d \times d}$ is a diagonal matrix defined for the linear transformation of relation r, $b_c \in R^d$ is a global bias vector, and $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ is a nonlinear activation function in which the output is restricted to (-1,1).To obtain the representations of entities at different entity locations, the tail-entity vector $\mathbf{t} \in R^d$, rather than projection, is used directly for the similarity operation as $S(h, r, t) = \sigma(\mathbf{t} \cdot c_{hr})$, where $\sigma(z) = \frac{1}{1+e^{-z}}$ is used to restrict the final output to (0,1) as the confidence score.

On the basis of ProjE, **ProjFE** [177] improves the combination operator by adding the fuzzy-membership degree, which is used to measure the degree of confidence that an entity belongs to a certain concept, to improve the performance of the model with different degrees of positive and negative samples. Because a large number of translation-matrix calculations are omitted, the model has a very small number of parameters. Unlike previous models, ProjFE uses binary vectors to represent fuzzy embeddings for projection work. ProjFE has the same combination operations as ProjE, except that it adds the fuzzy-membership degree μ_e, μ_r for fuzzy entities and relationships, where the fuzzy-membership degree is defined as $\mu = e - \left(\frac{x-a}{\sigma}\right)^2, x \in (0, \infty), a = (a_1, a_2, ..., a_n), a_i \in [0.5, 1]$. The combination operator is defined as $\mathbf{e} \oplus \mathbf{r} = \mu_e D_e \mathbf{e} + \mu_r D_r \mathbf{r} + b_c = \left(e - \left(\frac{x_e - a}{\sigma}\right)^2\right) D_e \mathbf{e} + \left(e - \left(\frac{x_r - a}{\sigma}\right)^2\right) D_r \mathbf{r} + b_c$. The scoring function of ProjFE is defined as in Equation (26):

$$S(e)_i = s(\mu_c D_c r(e \oplus r) + b_s) = sigmoid\left(\left(e - \left(\frac{x_c - a}{\sigma}\right)^2\right) W_{[i,:]}^c \operatorname{Re} LU(\mathbf{e} \oplus \mathbf{r}) + b_s\right) \quad (26)$$

where D_c is a candidate-entity matrix, μ_c is the candidate fuzzy degree, b_s is the calculated bias, and $W_{[i,:]}^c$ is the matrix of the i^{th} candidate entity.

Kipf et al. proposed a simple but effective hierarchical-propagation rule running directly on a graph with a neural-network model, a **GCN**. This is the idea of a local first-order approximation derived from spectral convolution that motivates the convolution structure. It can be scaled linearly on the edge of the graph to learn hidden layer representations for encoding the local graphical structures and features of nodes. Its hierarchical-propagation rules are as follows: $H^{(l+1)} = \sigma\left(\overline{D}^{-\frac{1}{2}} \overline{A} \overline{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right)$, where $\overline{A} = A + I_N$ is the adjacency matrix of a graph with self-connections, I_N is the identity matrix, and $\overline{D}_{ii} = \sum_j \overline{A_{ij}}$ and $W^{(l)}$ is a layer-specific trainable weight matrix. The GCN is a spectral method, and the convolution theorem on the graph is used to define the graph convolution in the spectral domain. Many spatial methods have been proposed, and their main idea is to define node similarity by an aggregation function in the node domain.

Weighted GCN, known as **SACN** [26], is an end-to-end structure-aware convolutional network that takes into account node connectivity, node attributes and relation types

simultaneously. The model also defines an encoder and decoder. In the encoder part, by introducing weights for different types of relations, SACN improves the GCN model to obtain a weighted graph convolutional network (WGCN). It makes different trade-offs for different types of relations when aggregating, so the multirelational graph can be regarded as multiple single-relational graphs, where each subgraph contains a specific type of relation. The decoder, called Conv-TransE, removes the reshaping operation from the input entity and relation embeddings and lets the convolutional filters operate directly on input entities and relations in the same dimension. Thus, the translation properties of TransE remain, while the same prediction performance as ConvE is maintained. Furthermore, SACN treats entity attributes as another type of node, called attribute nodes, which have similar representations and operations as nodes. Its propagation process is defined for node v_i as in Equation (27):

$$h_i^{l+1} = \sigma\left(\sum_{j \in N_i} \alpha_t^l h_j^l W^l + h_i^l W^l\right) \tag{27}$$

CompGCN [25] is a novel GCN that uses composition operators from KGE methods by jointly embedding both entities and relations in a relational graph. For a given entity, it considers the outgoing edges of the original, inverse and self relations simultaneously via the ϕ composition operator, defined as $e_o = \phi(e_s, e_r)$, where the ϕ operator is restricted to nonparameterized operations. The updating process in CompGCN is as follows: $h_v = f\left(\sum_{(u,r)\in N(v)} W_{\lambda(r)}\phi(e_u, e_r)\right)$, where e_u and e_r are the initial features for entity u and relation r, $N(v)$ is the set of immediate neighbors of v for the outgoing edges, and $w_{\lambda(r)} \in R^{d_1 \times d_2}$ is a relation-type specific parameter for the original w_O, inverse w_I, and self w_S relations, separately. Simultaneously, the relation embedding is transformed to $h_r = w_{rel} z_r$, where $w_{rel} \in R^{d_1 \times d_2}$ is a transformation matrix that projects relations to the same space as entities; z_r is defined as a set of learnable basis vectors as $z_r = \sum_{b=1}^{B} \alpha_{br} v_b$, where $\alpha_{br} \in R$ is the relation- and basis-specific learnable scale weight. In this paper, the ϕ operator is calculated as follows, inspired by TransE, DistMult and HolE, to obtain the score:

Subtraction: $\phi(e_s, e_r) = e_s - e_r$.

Multiplication: $\phi(e_s, e_r) = e_s * e_r$.

Circular correlation: $\phi(e_s, e_r) = e_s \cdot e_r$.

CompGCN is a general GCN-based model that addresses the shortcomings of over-parameterization by sharing a relation embedding calculated using basis decomposition across layers.

3.4. Connections between Typical Models

The category above is macrolevel, and there may be progressive correlations between these models of different categories. Based on the coarse-grained categories above, we made a more fine-grained category describing some representative models along five main lines. In this way, we can understand the relationships between these models in depth. These models include the representative models of the previous categories, including CNN-based models, GCN-based models, TransE extensions and so on, some of which also contain additional information.

In the first line, the introduced models are based on a CNN, as shown in Figure 4.

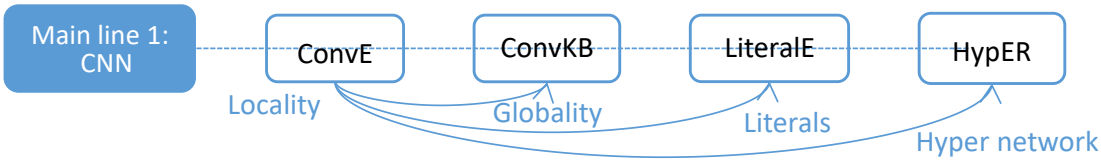

Figure 4. Main line 1: based on a CNN.

In the second line, the introduced models contain contextual information, as shown in Figure 5. The detailed descriptions are as follows:

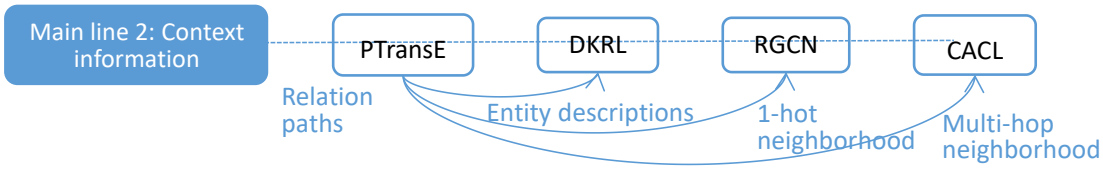

Figure 5. Main line 2: contextual information.

In the third line, the introduced models all concern tensor or matrix production, as shown in Figure 6. The detailed descriptions are as follows:

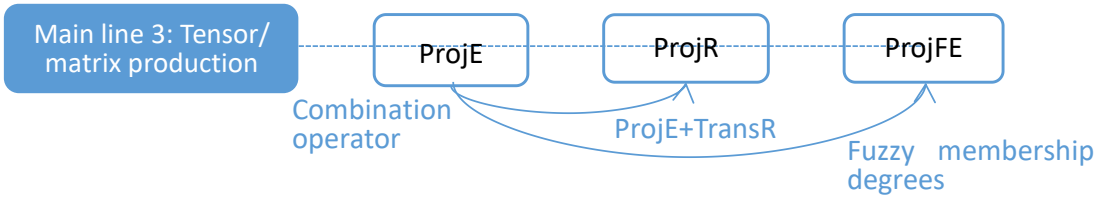

Figure 6. Main line 3: tensor or matrix production.

In the fourth line, the introduced models are all based on a GCN, as shown in Figure 7. The detailed descriptions are as follows:

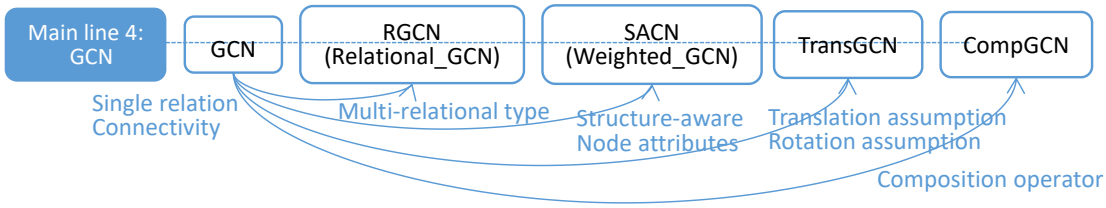

Figure 7. Main line 4: based on a GCN.

In the fifth line, the introduced models are all extensions of TransE, as shown in Figure 8. The detailed descriptions are as follows:

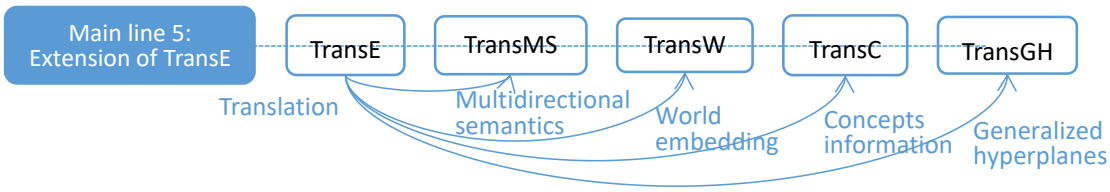

Figure 8. Main line 5: extensions of TransE.

4. Experiments

4.1. Experimental Settings

We conducted all the experiments targeting the link-prediction task in a unified environment with a T640 Dell server, the Ubuntu 16.04 system, and an NVIDIA-SMI 418.67 GPU. The implementation of these model architectures was based on the PyTorch framework. We also used the code provided in the original paper with the same settings to obtain the best results.

4.2. Dataset

We selected two datasets for our experiments, FB15k and FB15-237, which are the standard datasets commonly used in this field. FB15k is a subset of Freebase, a large-scale knowledge graph containing general knowledge facts. FB15k-237 is a subset of FB15k, where inverse relations are deleted. Therefore, these datasets could be used to evaluate the performance of the model more comprehensively. The details of these two datasets are shown in Table 3.

Table 3. Statistics of the experimental datasets.

Dataset	Entity	Relation	Triple	Train	Valid	Test
FB15k	14,951	1345	592,213	483,142	50,000	59,071
FB15k-237	14,541	237	310,116	272,115	17,535	20,466

4.3. The Implemented Models

Based on the previous categories of the existing models and the analysis of the five main lines, we selected several representative models for conducting the experiments, including CNN-based, GCN-based, semantic-matching and TransE extensions, some of which also fuse some important additional information. Therefore, these selected models are representative for gaining deep insight into the existing models, and they are conventional, recently proposed or highly cited. The implemented models are described below.

SACN [26] exploits a graph convolutional neural network and integrates relation type information as well as node attribute information, which is not limited to the traditional method of embedding based only on triple information. **HypER** [24] improves the ConvE model by providing a simple calculation method for sparsity and leveraging a parameter binding mechanism, which uses a hypernetwork to perform weight sharing. **Bilinear+TR** [78] introduces a type regularizer into the loss function, which fully considers the type information of entities. **RW-LMLM** [91] considers paths with three aspects of information: entities, relations and order information. It draws on the random walk algorithm and semantic-based models. **LiteralE** [115] introduces textual information as the attribute information of entities. **SimplE** [152] encodes background knowledge into an embedding by parameter sharing. It embeds relations and their inverse relations separately. **HAKE** [60] refers to the idea of polar coordinates, and it considers the hierarchical information of semantics. **RotatE** [35] replaces the traditional translation operation with a rotation operation, which can be used to distinguish various relations, such as symmetry, antisymmetry and composition. **ConvE** [22] is the first model to utilize the CNN framework for KG completion. It uses embedded 2D convolution to predict missing links in KGs. **DistMult** [20] and **ComplEx** [21] are traditional semantic-matching models based on tensor decomposition. ComplEx models asymmetric relations. The information on these models is summarized in Table 4.

4.4. Performance Analysis

As shown in Table 5, **RotatE** [35] and **HypER** [24] outperform the other models, which indicates that the rotation operation used in translational-distance models and the hypernetwork used in CNN-based models play important roles in improving performance. RotatE uses the complex space and mines different types of relations (symmetry/antisymmetry, inversion and composition); thus, different aspects of semantic information are modeled well by this integration strategy. HypER and **ConvE** [22] are based on a CNN; the former improves the latter by a hypernetwork, which can be used to perform weight sharing across layers and dynamically synthesize weights given inputs. **RW-LMLM** [91] takes into account both the order information and random walk algorithm, and it has the capability of dealing with underlying semantic information. **SACN** [26] also performs well among these models; it uses the relation types and entity attributes in the GCN model structure. On the whole, the top-performing models are all based on neural networks (GCNs or

CNNs), from which we can conclude that the advanced neural-network structure, with its ability to generate rich and expressive feature embeddings, is helpful in the KGE task. The performance of conventional models such as translation models (TransE) and semantic models (DistMult and ComplEx) is not good. **SimplEx** [152] has low performance for FB15k-237 and high performance for FB15k because FB15k contains inverse relations and SimplEx can model the inverse relations appropriately. For **HAKE** [60], we believe that the polar coordinates may have great benefits because of their particular structure, which enables them to mine considerable hidden semantic structure information. In terms of **Bilinear+TR** [78] and **LiteralE** [115], we can see that adding the entity type, text and other information helps to improve performance.

Table 4. Implemented models.

Categories	Model	Embedding Space	Additional Information	Scoring Function
Based on translation distance	RotatE	Complex space	None	$-\|h \circ r - t\|$
	HAKE	Vector space	None	$\|h_m \circ r_m - t_m\|_2 + \lambda\|\sin((h_p + r_p - t_p)/2)\|_1$
Based on semantic information	RW-LMLM	Matrix space	Order information	$soft\max(h_n^i W_h W_E^T)$
	SimplE	Vector space	Background knowledge	$\frac{1}{2}\left(\left\langle h_{e_i}, v_r, t_{e_j}\right\rangle + \left\langle h_{e_j}, v_{r^{-1}}, t_{e_i}\right\rangle\right)$
	LiteralE	Vector space	Literal information	$f_X(g(e_i, l_i), g(e_j, l_j), r_k)$
	Bilinear+TR	Tensor space	Entity types	$x_S^T W_r x_t$
	ComplEx	Complex space	None	$\mathrm{Re}(\langle h, r, t\rangle)$
	DistMult	Vector space	None	$(\langle h, r, t\rangle)$
Based on neural network	SACN	Vector space	Relation types	$f(vec(\mathbf{M}(e_s, e_r))\mathbf{W})e_o$
	ConvE	Vector space	None	$f(vec(f([\overline{e_s}; \overline{r_r}] * w))W)e_o$
	HypER	Vector space	None	$f(vec(e_1 * vec^{-1}(w_r H))W)e_2$

As for **LiteralE** [115], it does not obtain only good results. For FB15k, only ComplEx obtains good results upon adding literal information, while for FB15-237, only DistMult improves slightly after adding literal information. For ConvE, a neural-network model, adding literal data does not achieve better results but worse results. LiteralE combines literal vectors (only numerical information is involved in this model) and an entity embedding as the input for training. DistMult uses a simple bilinear formula and matrix multiplication to learn embeddings. Its scoring function can only capture pairwise interactions of the same dimension between entities. Therefore, this simple embedding can only deal with symmetric relations. We suspect that this is the reason that literal information does not work. However, for FB15k-237, the result of DistMult is slightly improved due to the deletion of the inverse relationship. Because ComplEx introduces a complex vector space and can deal with asymmetric relations, it has a good response to literal information for FB15k but not FB15k-237. For ConvE, we believe that the neural-network model is able to aggregate the domain information well, so it is not sensitive to the addition of literal vectors. It performs even worse, which we guess is because of the large number of parameters from LiteralE and itself. In addition, we added textual information on the basis of numerics for LiteralE (LiteralE+text+DistMult), and the experimental results show that the performance was similarly worse. We speculate that simply adding textual information to the entity embedding of the input for training does not play a very important role. We should continuously aggregate effective domain information in the process of training and try to reduce the number of parameters. Moreover, the text information should not be only

numerical information but should also include the entity type, entity attribute, path and other additional information.

Compared with traditional models, the latest models have their own advantages. Generally, they have achieved better results, benefiting from their own unique model structures and sampling technologies or adding important additional information. It is safe to conclude that the models using additional information and taking advantage of neural networks have better performance.

Table 5. Results for FB15k-237 (left) and FB15k (right).

Model	FB15k-237				FB15k			
	Hits@1	**Hits@3**	**Hits@10**	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**	**MRR**
RotatE	0.2471	0.3802	0.5370	0.3432	0.7387	0.8240	0.8797	0.7905
HAKE	0.2561	0.3871	0.5488	0.3523	0.5745	0.7614	0.8482	0.6809
Bilinear+ TR-TransE	0.1288	0.1524	0.1912	0.1503	0.1522	0.1584	0.1695	0.1604
Bilinear+ TR-bilinear	0.2370	0.3549	0.4855	0.3246	0.4119	0.5683	0.6586	0.5064
DistMult	0.2129	0.3215	0.4635	0.2953	0.3055	0.4230	0.5149	0.3785
ComplEx	0.2014	0.3165	0.4526	0.2851	0.4407	0.5366	0.6168	0.5028
LiteralE+ text+DistMult	0.2192 ↓	0.3285 ↓	0.4651 ↓	0.3013 ↓	0.2730 ↓	0.3537 ↓	0.4431 ↓	0.3323 ↓
LiteralE+ DistMult	0.2209	0.3279	0.4628	0.3022	0.2917 ↓	0.3820 ↓	0.4758 ↓	0.3555 ↓
LiteralE+ ComplEx	0.1936 ↓	0.3029 ↓	0.4397 ↓	0.2753 ↓	0.5138	0.6102	0.6963	0.5777
LiteralE+ ConvE	0.2092 ↓	0.3277 ↓	0.4675 ↓	0.2963 ↓	0.6055 ↓	0.7014 ↓	0.7768 ↓	0.6667 ↓
SimplE	0.0895	0.1701	0.3170	0.1623	0.6593	0.7758	0.8434	0.7283
RW-LMLM	0.2249	0.3396	0.4859	0.3109	0.6767	0.7947	0.8648	0.7466
ConvE	0.2212	0.3371	0.4787	0.3070	0.6250	0.7254	0.7956	0.6874
SACN	0.2522	0.3699	0.5154	0.3400	-	-	-	-
HypER	0.2482	0.3629	0.5080	0.3335	0.7005	0.8167	0.8799	0.7668

4.5. Training Time Analysis

Different model architectures critically affect the computational workload. Concerning the training time consumption, we analyzed these models in detail. The results obtained are consistent with the previous model description. See Table 6. Generally, the time efficiencies of the models based on neural networks are relatively low due to the introduction of more parameters.

1. For CNN-based models, the initial model ConvE, which introduces numerous parameters because it uses an embedded 2D convolution, is very time-consuming for training. Similarly, for LiteralE, the introduction of additional information and its complex model structure lead to some additional parameter overhead. While HypER utilizes a 1D relation-specific filter and a nonlinear (quadratic) combination of entity and relation embeddings via hypernetworks to perform weight sharing, it has many fewer parameters than ConvE, so it saves much training time.
2. Semantic matching models such as DistMult and ComplEx all suffer from longer training times.
3. Translational distance models such as HAKE and RotatE all have shorter training times because the translational-distance model has a relatively simple model structure and scoring function without too many parameters.

4. The bilinear+TR model has the shortest training time, with a type regularizer incorporated into the loss function, which fully considers the type information of entities. The times of the linear models are short, but their performance is not good.
5. LiteralE introduces some overhead in terms of the number of parameters compared to the base method, leading to a long training time. This is due to the choice of the core function g, which takes an entity's embedding and a literal vector as inputs and maps them to a vector of the same dimension as the entity embedding. Thus, it can make much effort in this step to choose a better function.

Table 6. Training time used for FB15k.

Categories	Models	Training Time
Based on translation distance	HAKE	4.8908 h
	RotatE	6.9708 h
Based on semantic information	Bilinear+TR_bilinear	1.8083 h
	Bilinear+TR_transE	1.8935 h
	SimplE	4.4111 h
	LiteralE-ComplEx	11.1810 h
	DistMult	31.5515 h
	RW-LMLM	33.6141 h
	ComplEx	38.1560 h
	LiteralE-DistMult	42.3798 h
	LiteralE-text-DistMult	44.9654 h
Based on neural network	HypER	8.9695 h
	ConvE	50.8799 h
	LiteralE-ConvE	60.1813 h

4.6. Suggestions for Improvement

Based on the previous model descriptions and experimental results, we can conclude that for factorization-based models, the sparsity and parameters are the key factors to be considered. It is revealed that reducing the computational complexity brought about by sparsity and conducting parameter sharing will greatly improve the overall performance. For translational-distance models, translating the head entity to the tail entity is critical for obtaining a concise and efficient scoring function. Additional efforts in the embedding space will also make a difference. Furthermore, it is nontrivial to exploit deeper semantic information and a better model structure to help improve the performance of models by using a neural network and adding additional information.

5. Conclusions

Knowledge graph embedding (KGE), as the technology of embedding entities and relations into a low-dimensional continuous vector space, has made remarkable progress in offering precise, effective and structural representation of information in many fields. This paper reviewed the main technologies of KGE, categorized the existing models into two types based on whether or not they use additional information besides facts, and then overviewed the advantages and disadvantages of representative models in each category. We focused on the task of link prediction and carried out experiments on several typical models in a unified environment. Through the analysis of the experimental results, we found that different model architectures enjoyed unique advantages in different facets. According to our research on KGE models and the analysis and comparison of the experimental results, we can roughly summarize two points about how to improve the performance of KGE models, as follows:

- Neural network models with excellent structure and a small number of parameters have good performance. Especially, the graph convolution neural network has a strong ability to mine the underlying semantics of knowledge graphs. In addition, if the node information of a multi-hop domain can be aggregated, the accuracy of the model in specific tasks can be greatly improved.

- Models with additional information, such as node attributes, node types, relationship types, prior knowledge and so on, have better performance.

We hope that this survey can provide researchers with new insights and a stepping stone to help them conduct research better. Of course, there are some limitations in our survey. Due to space constraints, we will conduct in-depth research into the following aspects in future research:

1. This survey only focused on the link prediction of KGE; we will research more tasks of knowledge graph completion in the future, such as entity prediction, entity classification and triple classification.
2. This survey only used two datasets (FB15k and FB15k-237) for the experiments; we will use more knowledge graph datasets, such as WN18, WN18RR and FB13.
3. This survey only focused on static graphs; we will explore new model architectures, such as dynamic graphs and heterogeneous graphs.
4. The categories we proposed for KGE models may not be the perfect ones; we will attempt to mine new category strategies for KGE models.

Author Contributions: Conceptualization, M.W., L.Q. and X.W.; methodology, M.W. and X.W.; software, L.Q.; validation, M.W., L.Q. and X.W.; formal analysis, M.W., L.Q. and X.W.; investigation, M.W. and L.Q.; resources, M.W. and X.W.; data curation, L.Q.; writing—original draft preparation, L.Q.; writing—review and editing, M.W. and X.W.; visualization, L.Q.; supervision, M.W. and X.W.; project administration, M.W.; funding acquisition, M.W. and X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fujian Provincial Department of Science and Technology under Grant No. 2019H0001, the National Natural Science Foundation of China under Grant No. 61702432, the Fundamental Research Funds for Central Universities of China under Grant No. 20720180070, and the International Cooperation Projects of Fujian Province in China under Grant No. 2018I0016.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research was funded by the Fujian Provincial Department of Science and Technology under Grant No. 2019H0001, the National Natural Science Foundation of China under Grant No. 61702432, the Fundamental Research Funds for Central Universities of China under Grant No. 20720180070, and the International Cooperation Projects of Fujian Province in China under Grant No. 2018I0016.

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

References

1. Bollacker, K.D.; Evans, C.; Paritosh, P.; Sturge, T.; Taylor, J. *Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge*; SIGMOD: Vancouver, BC, Canada, 2008; pp. 1247–1250.
2. Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D. Mendes, P. N.; Hellmann, S.; Morsey, M.; Kleef, P. V.; Auer, S.; et al. *DBpedia—A Large-Scale, Multilingual Knowledge base Extracted from Wikipedia*; Semantic Web, Springer, 2015; Volume 6, pp. 167–195.
3. Mahdisoltani, F.; Biega, J.A.; Suchanek, F.M. YAGO3: A Knowledge Base from Multilingual Wikipedias. In Proceedings of the CIDR, Asilomar, CA, USA, 4–7 January 2015.
4. Wang, R.; Wang, M.; Liu, J.; Chen, W.; Cochez, M.; Decker, S. Leveraging Knowledge Graph Embeddings for Natural Language Question Answering. In Proceedings of the DASFAA 2019, Chiang Mai, Thailand, 22–25 April 2019; pp. 659–675.
5. Musto, C.; Basile, P.; Semeraro, G. Embedding Knowledge Graphs for Semantics-aware Recommendations based on DBpedia. In Proceedings of the UMAP 2019, Larnaca, Cyprus, 9–12 June 2019; pp. 27–31.
6. 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.
7. Cai, H.; Zheng, V.W.; Chang, K.C. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Trans. Knowl. Data Eng.* **2017**, *30*, 1616–1637.
8. Siddhant, A. A Survey on Graph Neural Networks for Knowledge Graph Completion. *arXiv* **2020**, arXiv:2007.12374, .

9. Ma, J.; Qiao, Y.; Hu, G.; Wang, Y.; Zhang, C.; Huang, Y.; Sangaiah, A.K.; Wu, H.; Zhang, H.; Ren, K. ELPKG: A High-Accuracy Link Prediction Approach for Knowledge Graph Completion. *Symmetry* **2019**, *11*, 1096.
10. Chang, K.; Yih,W.; Yang, B.; Meek, C. Typed Tensor Decomposition of Knowledge Bases for Relation Extraction. In Proceedings of the EMNLP, Doha, Qatar, 25–29 October 2014; pp. 1568–1579.
11. Lao, N.; Mitchell, T.; Cohen, W.W. Random Walk Inference and Learning in A Large Scale Knowledge Base. In Proceedings of the EMNLP, Edinburgh, UK, 27–31 July 2011; pp. 529–539.
12. Lu, F.; Cong, P.; Huang, X. Utilizing Textual Information in Knowledge Graph Embedding: A Survey of Methods and Applications. *IEEE Access* **2020**, *8*, 92072–92088.
13. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating Embeddings for Modeling Multi-Relational Data. In Proceedings of the NIPS, Lake Tahoe, NV, USA, 5–8 December 2013.
14. Minervini, P.; d' Amato, C.; Fanizzi, N.; Esposito, F. Efficient Learning of Entity and Predicate Embeddings for Link Prediction in Knowledge Graphs. *URSW@ISWC*, **2015**, 26–37.
15. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. *Knowledge Graph Embedding by Translating on Hyperplanes*; AAAI Press: Palo Alto, CA, USA, 2014; pp. 1112–1119.
16. Fan, M.; Zhou, Q.;Chang, E.; Zheng, T.F. Transition-based Knowledge Graph Embedding with Relational Mapping Properties. In Proceedings of the PACLIC, Phuket, Thailand, 12–14 December 2014; pp. 328–337.
17. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu, X. *Learning Entity and Relation Embeddings for Knowledge Graph Completion*; AAAI Press: Palo Alto, CA, USA, 2015; pp. 2181–2187.
18. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of the NIPS, Lake Tahoe, NV, USA, 5–8 December 2013; pp. 3111–3119.
19. Liu, Z.; Sun, M.; Lin, Y.; Xie, R. Knowledge Representation Learning: A Review. *J. Comp. Res. Develop.* **2016**, 247–261.
20. Yang, B.; Yih, W.; He, X.; Gao, J.; Deng, L. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In Proceedings of the ICLR (Poster), San Diego, CA, USA, 7–9 May 2015.
21. Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; Bouchard, G. *Complex Embeddings for Simple Link Prediction*; ICML: New York City, NY, USA, 2016; pp. 2071–2080.
22. Dettmers, T.; Minervini, P.; Stenetorp, P.; Riedel, S. *Convolutional 2D Knowledge Graph Embeddings*; AAAI Press: Palo Alto, CA, USA, 2017; pp. 1811–1818.
23. Nguyen, D.Q.; Nguyen, T.D.; Nguyen, D.Q.; Phung, D.Q. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In Proceedings of the NAACL-HLT, New Orleans, LA, USA, 1–6 June 2018; pp. 327–333.
24. Balazevic, I.; Allen, C.; Hospedales, T.M. Hypernetwork Knowledge Graph Embeddings. In Proceedings of the ICANN (Workshop), Munich, Germany, 17–19 September 2019; pp. 553–565.
25. Vashishth, S.; Sanyal, S.; Nitin, V.; Talukdar, P.P. Composition-based Multi-Relational Graph Convolutional Networks. In Proceedings of the ICLR, Addis Ababa, Ethiopia, 26–30 April 2020.
26. Shang, C.; Tang, Y.; Huang, J.; Bi, J.; He, X.; Zhou, B. *End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion*; AAAI Press: Palo Alto, CA, USA, 2019; pp. 3060–3067.
27. Jagvaral, B.; Lee, W.; Roh, J.S.; Kim, M.S.; Park, Y.T. Path-based reasoning approach for knowledge graph completion using CNN-BiLSTM with attention mechanism. *Expert Syst. Appl.* **2020**, *142*, 112960.
28. Rossi, A.; Barbosa, D.; Firmani, D.; Matinata, A.; Merialdo, P. Knowledge graph embedding for link prediction: A comparative analysis. *TKDD* **2021**, *15*, 1–49.
29. Dai, Y.; Wang, S.; Xiong, N. N.; Guo, W. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics* **2020**, *9*, 750.
30. Chen, X.; Jia, S.; Xiang, Y. A review: Knowledge reasoning over knowledge graph. *Expert Syst. Appl.* **2020**, *141*, 112948.1–112948.21.
31. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Yu, P.S. A Survey on Knowledge Graphs: Representation, Acquisition and Applications. *arXiv* **2020**, arXiv:2002.00388.
32. Lin, Y.; Han, X.; Xie, R.; Liu, Z.; Sun, M. Knowledge Representation Learning: A Quantitative Review. *arXiv* **2018**, arXiv:1812.10901.
33. Nguyen, D.Q. An overview of embedding models of entities and relationships for knowledge base completion. *arXiv* **2017**, arXiv:1703.08098.
34. Kazemi, S.M.; Goel, R.; Jain, K.;Kobyzev, I.; Sethi, A.; Forsyth, P.; Poupart, P. Representation Learning for Dynamic Graphs: A Survey. *J. Mach. Learn. Res.* **2020**, *21*, 1–73.
35. Sun, Z.; Deng, Z.H.; Nie, J.Y. Tang, J. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In Proceedings of the ICLR(Poster), New Orleans, LA, USA, 6–9 May 2019.
36. Ji, G.; He, S.; Xu, L.; Liu, K.; Zhao, J. *Knowledge Graph Embedding via Dynamic Mapping Matrix*; ACL: Beijing, China, 2015; pp. 687–696.
37. Jia, Y.; Wang, Y.; Lin, H.; Jin, X.; Cheng, X. *Locally Adaptive Translation for Knowledge Graph Embedding*; AAAI: Phoenix, AZ, USA, 2016; pp. 992–998.
38. Ji, G.; Liu, K.; He, S.; Zhao, J. *Knowledge Graph Completion with Adaptive Sparse Transfer Matrix*; AAAI Press: Palo Alto, CA, USA, 2016; pp. 985–991.
39. Xiao, H.; Huang, M.; Zhu, X. From One Point to a Manifold: Knowledge Graph Embedding for Precise Link Prediction. In Proceedings of the IJCAI, New York, NY, USA, 9–15 July 2016; pp. 1315–1321.

40. Nguyen, D.Q.; Sirts, K.; Qu, L.; Johnson, M. STransE: A novel embedding model of entities and relationships in knowledge bases. In Proceedings of the HLT-NAACL, San Diego, CA, USA, 21 May 2016; pp. 460–466.
41. Feng, J.; Huang, M.; Wang, M.; Zhou, M.; Hao, Y.; Zhu, X. Knowledge Graph Embedding by Flexible Translation. In Proceedings of the KR, Cape Town, South Africa, 25–29 April 2016; pp. 557–560.
42. Chang, L.; Zhu, M.; Gu, T.; Bin, C.; Qian, J.; Zhang, J. Knowledge graph embedding by dynamic translation. *IEEE Access* **2017**, *5*, 20898–20907.
43. Zhang, C.; Zhou, M.; Han, X.; Hu, Z.; Ji, Y. Knowledge Graph Embedding for Hyper-Relational Data. *J. Tsinghua Univ. Nat. Sci. Ed.* **2017**, *22*, 185–197.
44. Du, Z.; Hao, Z.; Meng, X.; Wang, Q. CirE: Circular Embeddings of Knowledge Graphs. In Proceedings of the DASFAA, Suzhou, China, 27–30 May 2017; pp. 148–162.
45. Tan, Z.; Zhao, X.; Fang, Y.; Xiao, W. GTrans: Generic knowledge graph embedding via multi-state entities and dynamic relation spaces. *IEEE Access* **2018**, *6*, 8232-8244.
46. Zhu, J.; Jia, Y.; Xu, J.; Qiao, J.; Cheng, X. Modeling the Correlations of Relations for Knowledge Graph Embedding. *Comput. Sci. Technol.* **2018**, *33*, 323–334, .
47. Do, K.; Tran, T.; Venkatesh, S. Knowledge Graph Embedding with Multiple Relation Projections. In Proceedings of the ICPR, Beijing, China, 20–24 August 2018; pp. 332–337.
48. Zhu, Q.; Zhou, X.; Tan, J.; Liu, P.; Guo, L. Learning Knowledge Graph Embeddings via Generalized Hyperplanes. In Proceedings of the ICCS, Wuxi, China, 11–13 June 2018; pp. 624–638.
49. Geng, Z.; Li, Z.; Han, Y. A Novel Asymmetric Embedding Model for Knowledge Graph Completion. In Proceedings of the ICPR, Beijing, China, 20–24 August 2018; pp. 290–295.
50. Zhang, Y.; Du, Z.; Meng, X. EMT: A Tail-Oriented Method for Specific Domain Knowledge Graph Completion. In Proceedings of the PAKDD, Macau, China, 14–17 April 2019; pp. 514–527.
51. Yao, J.; Zhao, Y. Knowledge Graph Embedding Bi-vector Models for Symmetric Relation. In *Chinese Intelligent Systems Conference*; Springer: Singapore, 2019.
52. Yang, S.; Tian, J.; Zhang, H.; Yan, J.; He, H.; Jin, Y. TransMS: Knowledge Graph Embedding for Complex Relations by Multidirectional Semantics. In Proceedings of the IJCAI, Macao, China, 10–16 August 2019; pp. 1935–1942.
53. Ebisu, T.; Ichise, R. Generalized Translation-Based Embedding of Knowledge Graph. *IEEE Trans. Knowl. Data Eng.* **2020**, *32*, 941–951.
54. Cui, Z.; Liu, S.; Pan, L.; He, Q. Translating Embedding with Local Connection for Knowledge Graph Completion. In Proceedings of the AAMAS, Auckland, New Zealand, 9–13 May 2020; pp. 1825–1827.
55. He, S.; Liu, K.; Ji, G.; Zhao, J. Learning to Represent Knowledge Graphs with Gaussian Embedding. In Proceedings of the CIKM, Melbourne, VIC, Australia, October 19-23, 2015; pp. 623–632.
56. Xiao, H.; Huang, M.; Hao, Y.; Zhu, X. TransG: A Generative Mixture Model for Knowledge Graph Embedding. *ACL* **2015**, *1*, 2316–2325.
57. Song, H. J.; Park, S. B. Enriching translation-based knowledge graph embeddings through continual learning. *IEEE Access* **2018**, *6*, 60489–60497.
58. Ebisu, T.; Ichise, R. *TorusE: Knowledge Graph Embedding on a Lie Group*; AAAI Press: Palo Alto, CA, USA, 2018; pp. 1819–1826.
59. Zhang, S.; Tay, Y.; Yao, L.; Liu, Q. Quaternion Knowledge Graph Embeddings. *NeurIPS* **2019**.
60. Zhang, Z.; Cai, J.; Zhang, Y.; Wang, J. Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction. In Proceedings of the AAAI 2020, New York, NY, USA, 7–12 February 2020; pp. 3065–3072.
61. Kong, X.; Chen, X.; Hovy, E.H. Decompressing Knowledge Graph Representations for Link Prediction. *arXiv* **2019**, arXiv:1911.04053.
62. Chen, Y.; Liu, J.; Zhang, Z.; Wen, S.; Xiong, W. MobiusE: Knowledge Graph Embedding on Mobius Ring. *arXiv* **2021**, arXiv-2101.
63. Chen, H.; Wang, W.; Li, G.; Shi, Y. A quaternion-embedded capsule network model for knowledge graph completion. *IEEE Access* **2020**, *8*, 100890–100904.
64. Nickel, M.; Tresp, V.; Kriegel, H.P. A Three-Way Model for Collective Learning on Multi-Relational Data. ICML, Washington, D.C., USA, 28 June–2 July 2011; pp. 809–816.
65. Nickel, M.; Rosasco, L.; Poggio, T.A. *Holographic Embeddings of Knowledge Graphs*; AAAI: Phoenix, AZ, USA, 2016; pp. 1955–1961.
66. Liu, H.; Wu, Y.; Yang, Y. *Analogical Inference for Multi-Relational Embeddings*; ICML: Sydney, NSW, Australia, 2017; pp. 2168–2178.
67. Lacroix, T.; Usunier, N.; Obozinski, G. Canonical Tensor Decomposition for Knowledge Base Completion. In Proceedings of the ICML, Vienna, Austria, 23–31 July 2018; pp. 2869–2878.
68. Balazevic, I.; Allen, C.; Hospedales, M,T. *TuckER: Tensor Factorization for Knowledge Graph Completion*; EMNLP/IJCNLP: Hong Kong, China, 2019; pp. 5184–5193.
69. Mohamed, S.K.; Novácek, V. Link Prediction Using Multi Part Embeddings. In Proceedings of the ESWC, Portoroz, Slovenia, 2–6 June 2019; pp. 240–254.
70. Zhang, W.; Paudel, B.; Zhang, W.; Bernstein, A.; Chen, H. *Interaction Embeddings for Prediction and Explanation in Knowledge Graphs*; WSDM: Melbourne, VIC, Australia, 2019; pp. 96–104.
71. Xue, Y.; Yuan, Y.; Xu, Z.; Sabharwal, A. Expanding Holographic Embeddings for Knowledge Completion. *NeurIPS*, **2018**.

72. Tran, H.N.; Takasu, A. Multi-Partition Embedding Interaction with Block Term Format for Knowledge Graph Completion. In Proceedings of the ECAI, Copenhagen, Denmark, 19–24 July 2020; pp. 833–840.
73. Xie, R.; Liu, Z.; Sun, M.g. Representation Learning of Knowledge Graphs with Hierarchical Types. In Proceedings of the IJCAI, New York, NY, USA, 9–15 July 2016; pp. 2965–2971.
74. Guo, S.; Wang, Q.; Wang, B.; Wang, L.; Guo, L. SSE: Semantically Smooth Embedding for Knowledge Graphs. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 884–897.
75. Jiang, X.; Wang, Q.; Qi, B.; Qiu, Y.; Li, P.; Wang, B. Attentive Path Combination for Knowledge Graph Completion. In Proceedings of the ACML, Seoul, Korea, 15–17 November 2017; pp. 590–605.
76. Moon, C.; Jones, P.; Samatova, N.F. Learning Entity Type Embedding for Knowledge Graph Completion. In Proceedings of the CIKM, Singapore, 6–10 November 2017; pp. 2215–2218.
77. Ma, S.; Ding, J.; Jia, W.; Wang, K.; Guo, M. TransT: Type-Based Multiple Embedding Representations for Knowledge Graph Completion. In Proceedings of the ECML/PKDD, Skopje, Macedonia, 18–22 September 2017; pp. 717–733.
78. Kotnis, B.; Nastase, V. *Learning Knowledge Graph Embeddings with Type Regularizer*; K-CAP: Austin, TX, USA, 2017; pp. 1–4.
79. Rahman, M.M.; Takasu, A. Knowledge Graph Embedding via Entities' Type Mapping Matrix. In Proceedings of the ICONIP, Siem Reap, Cambodia, 13–16 December 2018.
80. Zhou, B.; Chen, Y.; Liu, K.; Zhao, J. Relation and Fact Type Supervised Knowledge Graph Embedding via Weighted Scores. In Proceedings of the CCL, Kunming, Chinapp, 18–20 October 2019; pp. 258–267.
81. Ma, J.; Zhong, M.; Wen, J.; Chen, W.; Zhou, X.; Li, X. RecKGC: Integrating Recommendation with Knowledge Graph Completion. In Proceedings of the ADMA, Dalian, China, 21–23 November 2019; pp. 250–265.
82. Lin, X.; Liang, Y.; Giunchiglia, F.; Feng, X.; Guan, R. Relation path embedding in knowledge graphs. *Neur. Comput. Appl.* **2019**, *31*, 5629–5639.
83. Lin, Y.; Liu, Z.; Luan, H.B.; Sun, M.; Rao, S.; Liu, S. Modeling Relation Paths for Representation Learning of Knowledge Bases. *EMNLP* **2015**.
84. Zeng, P.; Tan, Q.; Meng, X.; Zhang, H.; Xu, J. Modeling Complex Relationship Paths for Knowledge Graph Completion. *IEICE Transact.* **2018**, *101-D* 1393–1400.
85. Jia, Y.; Wang, Y.; Jin, X.; Cheng, X. Path-specific knowledge graph embedding. *Knowl. Based Syst.* **2018**, *151*, 37–44.
86. Xiong, S.; Huang, W.; Duan, P. Knowledge Graph Embedding via Relation Paths and Dynamic Mapping Matrix. In Proceedings of the ER Workshops, Xi'an, China, 22–25 October 2028; pp. 106–118.
87. Zhang, M.; Wang, Q.; Xu, W.; Li, W.; Sun, S. Discriminative Path-Based Knowledge Graph Embedding for Precise Link Prediction. In Proceedings of the ECIR, Grenoble, France, 26–29 March 2018.
88. Nastase, V.; Kotnis, B. Abstract Graphs and Abstract Paths for Knowledge Graph Completion. In Proceedings of the *SEM@NAACL-HLT 2019, Minneapolis, MN, USA, 6–7 June 2019.
89. Sun, J.; Xu, G.; Cheng, Y.; Zhuang, T. Knowledge Map Completion Method Based on Metric Space and Relational Path. *ICCSE* **2019**, 108–113.
90. Wang, Q.; Huang, P.; Wang, H.; Dai, S.; Jiang, W.; Liu, J.; Lyu, Y.; Zhu, Y.; Wu, H. CoKE: Contextualized Knowledge Graph Embedding. *arXiv* **2019** arXiv:1911.02168.
91. Wang, C.; Yan, M.; Yi, C.; Sha, Y. Capturing Semantic and Syntactic Information for Link Prediction in Knowledge Graphs. In Proceedings of the ISWC, Auckland, New Zealand, 26–30 October 2019; pp. 664–679.
92. Nathani, D.; Chauhan, J.; Sharma, C.; Kaul, M. Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. In Proceedings of the ACL 2019, Florence, Italy, 28 July–2 August 2019.
93. Wang, R.; Li, B.; Hu, S.; Du, W.; Zhang, M. Knowledge Graph Embedding via Graph Attenuated Attention Networks. *IEEE Access* **2020**, *8*, 5212–5224.
94. Xie, R.; Liu, Z.; Jia, J.; Luan, H.; Sun, M. *Representation Learning of Knowledge Graphs with Entity Descriptions*; AAAI Press: Palo Alto, CA, USA, 2016; pp. 2659–2665.
95. Xiao, H.; Huang, M.; Meng, L.; Zhu, X. *SSP: Semantic Space Projection for Knowledge Graph Embedding with Text Descriptions*; AAAI Press: Palo Alto, CA, USA, 2017; pp. 3104–3110.
96. Chen, M.; Tian, Y.; Chang, K.-W.; Skiena, S.; Zaniolo, C. Co-training Embeddings of Knowledge Graphs and Entity Descriptions for Cross-Lingual Entity Alignment. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; pp. 3998–4004.
97. Zhao, M.; Zhao, Y.; Xu, B. Knowledge Graph Completion via Complete Attention between Knowledge Graph and Entity Descriptions. In Proceedings of the CSAE, Sanya, China, 22–24 October 2019.
98. Veira, N.; Keng, B.; Padmanabhan, K.; Veneris, A.G. Unsupervised Embedding Enhancements of Knowledge Graphs using Textual Associations. In Proceedings of the IJCAI, Macao, China, 10–16 August 2019; pp. 5218–5225.
99. Shah, H.; Villmow, J.; Ulges, A.; Schwanecke, U.; Shafait, F. *An Open-World Extension to Knowledge Graph Completion Models*; AAAI Press: Palo Alto, CA, USA, 2019; pp. 3044–3051.
100. Wang, S.; Jiang, C. Knowledge graph embedding with interactive guidance from entity descriptions. *IEEE Access* **2019**, *7*, 156686–156693.
101. Ma, L.; Sun, P.; Lin, Z.; Wang, H. Composing Knowledge Graph Embeddings via Word Embeddings. *arXiv* **2019**, arXiv:1909.03794.
102. Guo, S.; Wang, Q.; Wang, L.; Wang, B.; Guo, L. Jointly embedding knowledge graphs and logical rules. In Proceedings of the EMNLP, Austin, TX, USA, 1–4 November 2016; pp. 192–202.

103. Yoon, H.-G.; Song, H.-J.; Park, S.-B.; Park, S.-Y. A Translation-Based Knowledge Graph Embedding Preserving Logical Property of Relations. In Proceedings of the HLT-NAACL, San Diego, CA, USA, 21 May 2016; pp. 907–916.
104. Du, J.; Qi, K.; Wan, H.; Peng, B.; Lu, S.; Shen, Y. Enhancing Knowledge Graph Embedding from a Logical Perspective. In Proceedings of the JIST, Gold Coast, Australia, 10–12 November 2017; pp. 232–247.
105. Han, X.; Zhang, C.; Sun, T.; Ji, Y.; Hu, Z. A triple-branch neural network for knowledge graph embedding. *IEEE Access* **2018**, *6*, 76606–76615.
106. Yuan, J.; Gao, N.; Xiang, J. *TransGate: Knowledge Graph Embedding with Shared Gate Structure*; AAAI Press: Palo Alto, CA, USA, 2019; pp. 3100–3107.
107. Wang, M.; Rong, E.; Zhuo, H.; Zhu, H. Embedding Knowledge Graphs Based on Transitivity and Asymmetry of Rules. In Proceedings of the PAKDD, Melbourne, VIC, Australia, 3–6 June 2018; pp. 141–153.
108. Wang, P.; Dou, D.; Wu, F.; Silva, N.; Jin, L. Logic Rules Powered Knowledge Graph Embedding. *arXiv* **2019**, arXiv:1903.03772.
109. Zhang, J.; Li, J. Enhanced Knowledge Graph Embedding by Jointly Learning Soft Rules and Facts. *Algorithms*, **2019**, *12*, 12.
110. Gu, Y.; Guan, Y.; Missier, P. Towards Learning Instantiated Logical Rules from Knowledge Graphs. *arXiv* **2020**, arXiv-2003.
111. Das, R.; Godbole, A.; Dhuliawala, S.; Zaheer, M.; McCallum, A. *A Simple Approach to Case-Based Reasoning in Knowledge Bases*; AKBC: San Francisco, CA, USA, 2020.
112. Das, R.; Godbole, A.; Monath, N.; Zaheer, M.; McCallum, A. Probabilistic Case-based Reasoning for Open-World Knowledge Graph Completion. *arXiv* **2020**, arXiv:2010.03548.
113. García-Durán, A.; Niepert, M. KBLRN: End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features. In Proceedings of the UAI, Monterey, CA, USA, 6–10 August 2018; pp. 372–381.
114. Wu, Y.; Wang, Z. Knowledge Graph Embedding with Numeric Attributes of Entities. *Rep4NLP@ACL* **2018**, 132, pp. 132-136.
115. Kristiadi, A.; Khan, M.A.; Lukovnikov, D.; Lehmann, J.; Fischer, A. Incorporating Literals into Knowledge Graph Embeddings. In Proceedings of the ISWC, Auckland, New Zealand, 26–30 October 2019; pp. 347–363.
116. Feng, M.-H.; Hsu, C.-C.; Li, C.-T.; Yeh, M.-Y.; Lin, S.-D. MARINE: Multi-relational Network Embeddings with Relational Proximity and Node Attributes. *WWW* **2019**, 470–479.
117. Zhang, Z.; Cao, L.; Chen, X.; Tang, W.; Xu, Z.; Meng, Y. Representation Learning of Knowledge Graphs With Entity Attributes. *IEEE Access* **2020**, 7435–7441.
118. Jiang, T.; Liu, T.; Ge, T.; Sha, L.; Li, S.; Chang, B.; Sui, Z. Encoding Temporal Information for Time-Aware Link Prediction. In Proceedings of the EMNLP, Austin, TX, USA, 1–4 November 2016.
119. Esteban, C.; Tresp, V.; Yang, Y.; Baier, S.; Krompass, D. Predicting the co-evolution of event and Knowledge Graphs. *FUSION* **2016**.
120. Trivedi, R.; Dai, H.; Wang, Y.; Song, L. Know-evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. In Proceedings of the ICML, Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 3462–3471.
121. Jia, Y.; Wang, Y.; Jin, X.; Lin, H.; Cheng, X. Knowledge Graph Embedding: A Locally and Temporally Adaptive Translation-Based Approach. *ACM Trans. Web* **2018**, *12*, 8:1–8:33
122. Dasgupta, S.S.; Ray, S.N.; Talukdar, P.P. HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding. In Proceedings of the EMNLP, Jeju, Korea, October 31–November 4, 2018; pp. 2001–2011.
123. Xu, C.; Nayyeri, M.; Alkhoury, F.; Lehmann, J.; Yazdi, H.S. Temporal Knowledge Graph Completion Based on Time Series Gaussian Embedding. In Proceedings of the ISWC, Athens, Greece, 2–6 November, 2020; pp. 654–671.
124. Chen, S.; Qiao, L.; Liu, B.; Bo, J.; Cui, Y.; Li, J. Knowledge Graph Embedding Based on Hyperplane and Quantitative Credibility. In Proceedings of the MLICOM, Nanjing, China, 24–25 August 2019; pp. 583–594.
125. Tang, X.; Yuan, R.; Li, Q.; Wang, T.; Yang, H.; Cai, Y.; Song, H. Timespan-Aware Dynamic Knowledge Graph Embedding by Incorporating Temporal Evolution. *IEEE Access* **2020**, 6849–6860.
126. Jung, J.; Jung, J.; Kang, U. T-GAP: Learning to Walk across Time for Temporal Knowledge Graph Completion. *arXiv* **2020**, arXiv:2012.10595.
127. Wu, J.; Cao, M.; Cheung, J.K.; Hamilton, W.L. TeMP: Temporal Message Passing for Temporal Knowledge Graph Completion. *EMNLP* **2020**, (1), 5730–5746.
128. Feng, J.; Huang, M.; Yang, Y.; Zhu, X. GAKE: Graph Aware Knowledge Embedding. In Proceedings of the COLING, Osaka, Japan, 11–16 December 2016.
129. Zhou, C.; Liu, Y.; Liu, X.; Liu, Z.; Gao, J. *Scalable Graph Embedding for Asymmetric Proximity*; AAAI Press: Palo Alto, CA, USA, 2017; pp. 2942–2948.
130. Zhang, W. Knowledge Graph Embedding with Diversity of Structures. In Proceedings of the WWW (Companion Volume), Perth, Australia, 3–7 April 2017.
131. Pal, S.; Urbani, J. Enhancing Knowledge Graph Completion By Embedding Correlation. In Proceedings of the CIKM, Singapore, 6–10 November 2017; pp. 2247–2250.
132. Shi, J.; Gao, H.; Qi, G.; Zhou, Z. Knowledge Graph Embedding with Triple Context. In Proceedings of the CIKM, Singapore, 6–10 November 2017; pp. 2299–2302.
133. Gao, H.; Shi, J.; Qi, G.; Wang, M. Triple context-based knowledge graph embedding. *IEEE Access* **2018**, *6*, 58978-58989.
134. Li, W.; Zhang, X.; Wang, Y.; Yan, Z.; Peng, R. Graph2Seq: Fusion Embedding Learning for Knowledge Graph Completion. *IEEE Access* **2019**, 157960–157971.

135. Zhang, Z.; Zhuang, F.; Qu, M.; Lin, F.; He, Q. Knowledge Graph Embedding with Hierarchical Relation Structure. *EMNLP* **2018**.
136. Han, X.; Zhang, C.; Guo, C.; Sun, T.; Ji, Y. *Knowledge Graph Embedding Based on Subgraph-Aware Proximity*; AAAI Press: Palo Alto, CA, USA, 2018; pp. 306–318.
137. Tan, Y.; Li, R.; Zhou, J.; Zhu, S. Knowledge Graph Embedding by Translation Model on Subgraph. In Proceedings of the HCC, Mérida, Mexico, 5–7 December 2018; pp. 269–280.
138. Zhang, Y.; Yao, Q.; Chen, L. Neural Recurrent Structure Search for Knowledge Graph Embedding. Technical report, International Workshop on Knowledge Graph@KDD, 2019.
139. Wan, G.; Du, B.; Pan, S.; Wu, J. Adaptive knowledge subgraph ensemble for robust and trustworthy knowledge graph completion. *World Wide Web* **2020**, *23*, 471–490.
140. Qiao, Z.; Ning, Z.; Du, Y.; Zhou, Y. Context-Enhanced Entity and Relation Embedding for Knowledge Graph Completion. *arXiv* **2020**, arXiv:2012.07011.
141. Ding, B.; Wang, Q.; Wang, B.; Guo, L. Improving Knowledge Graph Embedding Using Simple Constraints. In Proceedings of the ACL, Trujillo, Perupp, 13–16 November 2019; pp. 110–121.
142. Huang, Y.; Xu, K.; Wang, X.; Sun, H.; Lu, S.; Wang, T.; Zhang, X. CoRelatE: Modeling the Correlation in Multi-fold Relations for Knowledge Graph Embedding. In Proceedings of the ICLR, New Orleans, LO, USA, 6–9 May 2019.
143. Kanojia, V.; Maeda, H.; Togashi, R.; Fujita, S. Enhancing Knowledge Graph Embedding with Probabilistic Negative Sampling. *WWW (Companion Volume)* **2017**.
144. Niu, J.; Sun, Z.; Zhang, W. Enhancing Knowledge Graph Completion with Positive Unlabeled Learning. In Proceedings of the IICPR, Beijing, China, 20–24 August 2018; pp. 296–301.
145. Qin, S.; Rao, G.; Bin, C.; Chang, L.; Gu, T.; Xuan, W. Knowledge Graph Embedding Based on Adaptive Negative Sampling. In Proceedings of the ICPCSEE, Guilin, China, 20–23 September 2019; pp. 551–563.
146. Yan, Z.; Peng, R.; Wang, Y.; Li, W. Enhance knowledge graph embedding via fake triples. In Proceedings of the IJCNN, Budapest, Hungary, 2019; pp. 1–7.
147. Guo, C.; Zhang, C.; Han, X.; Ji, Y. AWML: Adaptive weighted margin learning for knowledge graph embedding. *J. Intell. Inf. Syst.* **2019**, *53*, 167–197.
148. Yuan, J.; Gao, N.; Xiang, J.; Tu, C.; Ge, J. Knowledge Graph Embedding with Order Information of Triplets. In Proceedings of the PAKDD, Macau, China, 14–17 April 2019; pp. 476–488.
149. Wang, Y.; Liu, Y.; Zhang, H.; Xie, H. Leveraging Lexical Semantic Information for Learning Concept-Based Multiple Embedding Representations for Knowledge Graph Completion. *APWeb/WAIM* **2019**, 382–397.
150. Guan, N.; Song, D.; Liao, L. Knowledge graph embedding with concepts. *Knowl.-Based Syst.* **2019**, *164*, 38–44.
151. Yu, Y.; Xu, Z.; Lv, Y.; Li, J. TransFG: A Fine-Grained Model for Knowledge Graph Embedding. In Proceedings of the WISA, Qingdao, China, September 20–22 2019.
152. Kazemi, S.M.; Poole, D. SimplE Embedding for Link Prediction in Knowledge Graphs. *NeurIPS* **2018**.
153. Fatemi, B.; Ravanbakhsh, S.; Poole, D. *Improved Knowledge Graph Embedding Using Background Taxonomic Information*; AAAI Press: Palo Alto, CA, USA, 2019; pp. 3526–3533.
154. Bordes, A.; Glorot, X.; Weston, J.; Bengio, Y. A semantic matching energy function for learning with multi-relational data. *Mach. Learn.* **2014**, *94*, 233–259.
155. Socher, R.; Chen, D.; Manning, C.D.; Ng, A.Y. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In Proceedings of the NIPS, Lake Tahoe, NV, USA, 5–8 December 2013; pp. 926–934.
156. Dong, X.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; Zhang, W. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In Proceedings of the KDD, New York, NY, USA, 24–27 August 2014; pp. 601–610.
157. Liu, Q.; Jiang, H.; Ling, Z.H.; Wei, S.; Hu, Y. Probabilistic Reasoning via Deep Learning: Neural Association Models. *arXiv* **2016**, arXiv:1603.07704.
158. Schlichtkrull, M.S.; Kipf, T.N.; Bloem, P.; Berg, R.v.d.; Titov, I.; Welling, M. Modeling Relational Data with Graph Convolutional Networks. In Proceedings of the ESWC, Crete, Greece, 3–7 June 2018; pp. 593–607.
159. Guo, L.; Zhang, Q.; Ge, W.; Hu, W.; Qu, Y. DSKG: A Deep Sequential Model for Knowledge Graph Completion. In Proceedings of the CCKS, Tianjin, China, 14–17 August 2018; pp. 65–77.
160. Guan, S.; Jin, X.; Wang, Y.; Cheng, X. Shared Embedding Based Neural Networks for Knowledge Graph Completion. In Proceedings of the CIKM, Turin, Italy, 22–26 October, 2018; pp. 247–256.
161. Zhu, Q.; Zhou, X.; Zhang, P.; Shi, Y. A neural translating general hyperplane for knowledge graph embedding. *J. Comput. Sci* **2019**, *30*, 108–117.
162. Huang, Z.; Li, B.; Yin, J. Knowledge Graph Embedding by Learning to Connect Entity with Relation. *APWeb/WAIM* **2018**, (1), 400-414.
163. Wang, L.; Lu, X.; Jiang, Z.; Zhang, Z.; Li, R.; Zhao, M.; Chen, D. FRS: A simple knowledge graph embedding model for entity prediction. *Math. Biosci. Eng.*, **2019**, *16*, 7789–7807.
164. Nguyen, D.Q.; Nguyen, T.D.; Phung, D.Q. Relational Memory-based Knowledge Graph Embedding. CoRR **2019**.
165. Cai, L.; Yan, B.; Mai, G.; Janowicz, K.; Zhu, R. TransGCN: Coupling Transformation Assumptions with Graph Convolutional Networks for Link Prediction. In Proceedings of the K-CAP, Marina Del Rey, CA, USA, 2019; pp. 131–138.

166. Ye, R.; Li, X.; Fang, Y.; Zang, H.; Wang, M. A Vectorized Relational Graph Convolutional Network for Multi-Relational Network Alignment. In Proceedings of the IJCAI, Macao, China, 10–16 August 2019; pp. 4135–4141.
167. Vashishth, S.; Sanyal, S.; Nitin, V.; Agrawal, N.; Talukdar, P.P. *InteractE: Improving Convolution-Based Knowledge Graph Embeddings by Increasing Feature Interactions*; AAAI Press: Palo Alto, CA, USA, 2020; pp. 3009–3016.
168. Hu, K.; Liu, H.; Zhan, C.; Tang, Y.; Hao, T. *A Bi-Directional Relation Aware Network for Link Prediction in Knowledge Graph*. In Proceedings of the NCAA, Springer Nature, Shenzhen, China, 3–5 July 2020; pp. 259–271.
169. Hu, K.; Liu, H.; Zhan, C.; Tang, Y.; Hao, T. *Learning Knowledge Graph Embedding with a Bi-Directional Relation Encoding Network and a Convolutional Autoencoder Decoding Network*; Neural Computing and Applications, Springer, 2021; pp. 1–17.
170. Zhang, N.; Deng, S.; Sun, Z.; Chen, J.; Zhang, W.; Chen, H. Relation Adversarial Network for Low Resource Knowledge Graph Completion. In Proceedings of the WWW, Taipei, Taiwan, 20–24 April 2020.
171. Tian, A.; Zhang, C.; Rang, M.; Yang, X.; Zhan, Z. RA-GCN: Relational Aggregation Graph Convolutional Network for Knowledge Graph Completion. In Proceedings of the ICMLC, Shenzhen China, 15–17 February 2020; pp. 580–586.
172. Jiang, W.; Guo, M.; Chen, Y.; Li, Y.; Xu, J.; Lyu, Y.; Zhu, Y. Multi-view Classification Model for Knowledge Graph Completion. In Proceedings of the AACL/IJCNLP, Suzhou, China, 4–7 December 2020.
173. Zeb, A.; Haq, A. U.; Zhang, D.; Chen, J.; Gong, Z. KGEL: A novel end-to-end embedding learning framework for knowledge graph completion. *Expert Syst. Appl.* **2021**, *167*, 114164.
174. Han, Y.; Fang, Q.; Hu, J.; Qian, S.; Xu, C. GAEAT: Graph Auto-Encoder Attention Networks for Knowledge Graph Completion. In Proceedings of the CIKM, New York, NY, USA, 2020; pp. 2053–2056.
175. Wang, Q.; Ji, Y.; Hao, Y.; Cao, J. GRL: Knowledge graph completion with GAN-based reinforcement learning. *Knowl.-Based Syst.* **2020**, *209*, 106421.
176. Shi, B.; Weningr, T. *ProjE: Embedding Projection for Knowledge Graph Completion*; AAAI Press: Palo Alto, CA, USA, 2017; pp. 1236–1242.
177. Liu, H.; Bai, L.; Ma, X.; Yu, W.; Xu, C. ProjFE: Prediction of fuzzy entity and relation for knowledge graph completion. *Applied Soft Computing* **2019**, *81*, 105525.
178. Zhang, W.; Li, J.; Chen, H. ProjR: Embedding Structure Diversity for Knowledge Graph Completion. In Proceedings of the NLPCC, Hohhot, China, 26–30 August 2018; pp. 145–157.
179. Shi, B.; Weninger, T. *Open-World Knowledge Graph Completion*; AAAI Press: Palo Alto, CA, USA, 2018; pp. 1957–1964.
180. Fu, C.; Li, Z.; Yang, Q.; Chen, Z.; Fang, J.; Zhao, P.; Xu, J. Multiple Interaction Attention Model for Open-World Knowledge Graph Completion. *WISE* **2019**, 630–644.
181. Nie, B.; Sun, S. Knowledge graph embedding via reasoning over entities, relations, and text. *Future Gener. Computer Syst.* **2019**, *91*, 426–433.
182. Zhu, J.; Zheng, Z.; Yang, M.; Fung, G.P.C.; Tang, Y. A semi-supervised model for knowledge graph embedding. *Data Min. Knowl. Discov.* **2020**, *34*, 1–20.
183. Dai, Y.; Wang, S.; Chen, X.; Xu, C.; Guo, W. Generative adversarial networks based on Wasserstein distance for knowledge graph embeddings. *Knowl.-Based Syst.* **2020**, *190*, 105165.
184. Wang, P.; Han, J.; Li, C.; Pan, R. *Logic Attention Based Neighborhood Aggregation for Inductive Knowledge Graph Embedding*; AAAI Press: Palo Alto, CA, USA, 2019; pp. 7152–7159.
185. Qian, W.; Fu, C.; Zhu, Y.; Cai, D.; He, X. Translation Embeddings for Knowledge Graph Completion with Relation Attention Mechanism. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; pp. 4286–4292.
186. Liu, W.; Cai, H.; Cheng, X.; Xie, S.; Yu, Y.; Zhang, H. Learning High-order Structural and Attribute information by Knowledge Graph Attention Networks for Enhancing Knowledge Graph Embedding. *arXiv* **2019**, arXiv:1910.03891.
187. Liu, Y.; Hua, W.; Xin, K.; Zhou, X. Context-Aware Temporal Knowledge Graph Embedding. In Proceedings of the WISE, Hong Kong, China, 26–30 November 2019.
188. Oh, B.; Seo, S.; Lee, K.-H. Knowledge Graph Completion by Context-Aware Convolutional Learning with Multi-Hope Neighborhoods. In Proceedings of the CIKM, Turin, Italy, 22–26 October 2018; pp. 257–266.
189. Wu, T.; Khan, A.; Gao, H.; Li, C. Efficiently Embedding Dynamic Knowledge Graphs. *arXiv* **2019**, arXiv:1910.06708.
190. Han, X.; Zhang, C.; Ji, Y.; Hu, Z. A Dilated Recurrent Neural Network-Based Model for Graph Embedding. *IEEE Access* **2019**, 32085–32092.
191. Tay, Y.; Luu, A.T.; Phan, M.C.; Hui, S.C. Multi-task Neural Network for Non-discrete Attribute Prediction in Knowledge Graphs. In Proceedings of the CIKM 2017, Singapore, 6–10 November 2017.
192. Nayyeri, M.; Xu, C.; Lehmann, J.; Yazdi, H.S. LogicENN: A Neural Based Knowledge Graphs Embedding Model with Logical Rules. *arXiv* **2019**, arXiv:1908.07141.
193. Zhao, F.; Xu, T.; Jin, L.; Jin, H. Convolutional Network Embedding of Text-enhanced Representation for Knowledge Graph Completion. *IEEE Int. Things J.* **2020**.
194. Wang, H.; Ren, H.; Leskovec, J. Entity Context and Relational Paths for Knowledge Graph Completion. *arXiv* **2020**, arXiv:2002.06757.
195. Wang, Y.; Zhang, H. HARP: A Novel Hierarchical Attention Model for Relation Prediction. *TKDD* **2021**, *15*, 1–22.