A Survey on Knowledge Graph Embeddings with Literals: Which model links better Literal-ly?

Genet Asefa Gesese [*], Russa Biswas, Mehwish Alam, and Harald Sack
FIZ Karlsruhe - Leibniz Institute for Information Infrastructure & Institute for Applied Informatics and Formal Description Systems (AIFB), Karlsruhe Institute of Technology, Karlsruhe Germany
E-mails: genet-asefa.gesese@fiz-karlsruhe.de, russa.biswas@fiz-karlsruhe.de, mehwish.alam@fiz-karlsruhe.de, harald.sack@fiz-karlsruhe.de

Abstract. Knowledge Graphs (KGs) are composed of structured information about a particular domain in the form of entities and relations. In addition to the structured information KGs help in facilitating interconnectivity and interoperability between different resources represented in the Linked Data Cloud. KGs have been used in a variety of applications such as entity linking, question answering, recommender systems, etc. However, KG applications suffer from high computational and storage costs. Hence, there arises the necessity for a representation able to map the high dimensional KGs into low dimensional spaces, i.e., embedding space, preserving structural as well as relational information. This paper conducts a survey of KG embedding models which not only consider the structured information contained in the form of entities and relations in a KG but also its unstructured information represented as literals such as text, numerical values, images, etc. Along with a theoretical analysis and comparison of the methods proposed so far for generating KG embeddings with literals, an empirical evaluation of the different methods under identical settings has been performed for the general task of link prediction.

Keywords: Knowledge Graphs, Knowledge Graph Embeddings, Knowledge Graph Embeddings with Literals, Link Prediction, Survey

1. Introduction

Knowledge Graphs (KGs) have become quite crucial for storing structured information. There has been a sudden attention towards using KGs for various applications mainly in the area of artificial intelligence. For instance, in a more general sense, KGs can be used to support decision making process and to improve different machine learning applications such as question answering [1], recommender systems [2], and relation extraction [3]. Some of the most popular publicly available general purpose KGs are DBpedia [4], Wikidata [5], and YAGO [6]. These general purpose KGs often consist of huge amount of facts constructed using millions of entities (represented as nodes) and relations (as edges connecting these nodes).

Although KGs are effective in representing structured data, there exist some issues which hinder their efficient manipulation such as i) different KGs are usually based on different rigorous symbolic frameworks and this makes it hard to utilize their data in other applications [7] and ii) the fact that a significant number of important graph algorithms needed for the efficient manipulation and analysis of graphs have proven to be NP-complete [8]. In order to address these issues and use a KG more efficiently, it is beneficial to transform it into a low dimensional vector space while preserving its underlying semantics. To this end, various attempts have been made so far to learn vector representations (embeddings) for KGs.

[*]Corresponding author. E-mail: genet-asefa.gesese@fiz-karlsruhe.de.

As discussed in [9], a typical KG embedding approach, which uses only structured information from the KG, generally follows three steps: (i) determining the form of entity and relation representations, (ii) defining a scoring function, and (iii) learning entity and relation representations. In the first step, the forms in which entities and relations are represented in the vector space are determined. Entities can be represented as vectors or modeled as multivariate Gaussian distributions whereas relations can be encoded as operations, matrices, tensors, multivariate Gaussian distributions, or mixtures of Gaussians. Once the form of the entities are determined, in the second step, a scoring function which measures the plausibility of a triple is defined. The main goal is to enable the model to assign higher score to true triples and lower scores to false/negative/corrupted triples. Thus, in order to achieve this, the third step solves an optimization problem which maximizes the plausibility of true facts in order to learn the embeddings of entities and relations. Note that the method used to generate false/negative/-corrupted triples has an impact on the performance of a model. The various negative triple generation methods and their differences are discussed in detail in [10].

Among the different embedding approaches proposed so far, TransE [11] is, to the best of our knowledge, the very first attempt to use distance-based scoring function to learn KG embedding. Given a triple $< h, r, t >$ where h and t are head and tail entities respectively and r is a relation, TransE represents h, r, and t as vectors $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$ respectively by modeling the relation r as a translation vector which connects the vectors $\mathbf{h}$ and $\mathbf{t}$. The problem with TransE is that it fails to model certain type of relations such as one-to-many or many-to-one. In order to address such limitations, different embedding techniques which are extension of TransE or are entirely new have been proposed. However, most of the existing approaches, including the current state-of-the-art models such as ConvE [12], are structure-based embeddings which do not make use of any literal information i.e., only triples consisting of entities connected via properties are usually considered. This is a major disadvantage because information encoded in the literals will be left unused when capturing the semantics of a certain entity.

Literals can bring advantages to the process of learning KG embeddings in two major ways:

1. *Learning embeddings for novel entities:* Novel entities are the entities which are not linked to any other entity in the KG but have literal values associated with them such as their *textual description, numeric literals, and images*. In most existing structure-based embedding models, it is not possible to learn embeddings for such novel entities. However, this can be addressed by utilizing the information represented in literals to learn embeddings. For example, considering the dataset FB15K-20 [13], which is a subset of Freebase, the entity `'/m/0gjd61t'` is a novel entity which does not occur in any of the training triples, but it has a description given as follows in the form <*subject, relation, object*>.

```
</m/0gjd61t, http://rdf.freebase.com/
    ns/common.topic.description, "
    Vincent Franklin is an English
    actor best known for his roles
    in comedy television programmes
    ...">
```

   In order to learn the embedding for this particular entity (i.e., `/m/0gjd61t`), the model can make use of the entity's textual description. DKRL [13] is one of those approaches which provide embeddings for novel entities using their descriptions.
2. *Improving the representation of entities in structure based embedding models:* Literals play a vital role in improving the representation learning where an entity is required to appear in at least a minimum number of relational triples. For example, taking into consideration only the information provided in a sample KG presented in Figure 1, which is extracted from DBpedia, it is not possible to tell apart the entities `dbr:Gina_Torres`, `dbr:Patrick_J.Adams`, and `dbr:Sarah_Rafferty` from one another. This is the case due to the fact that the only information that is available regarding these entities in this KG is that they all star in the series `dbr:Suits_(season_1)` and this is not enough to know which entities are similar to each other and which are not. Therefore, if some KG embedding model is trained using only this KG, it is not possible to get good representations for the entities `dbr:Gina_Torres`, `dbr:Patrick_J.Adams`, and `dbr:Sarah_Rafferty`. However, having the model trained with more triples containing literal values for these entities, as shown in Figure 2, would improve the embeddings for the entities. For instance, based on the values of the data relation `dbr:birthDate`, it

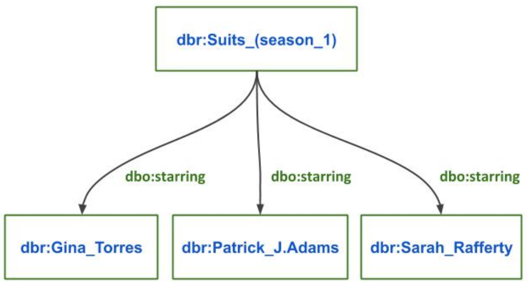


Fig. 1. A small fraction of triples taken from the KG DBpedia [4].

is possible to deduce the fact that `dbr:Sarah_Rafferty` and `dbr:Gina_Torres` are almost the same age but they are both older than `dbr:Patrick_J.Adams`. On the other hand, the images of the entities along with the textual descriptions (`dbo:abstract`) would allow us to infer the entities' gender, i.e., `dbr:Sarah_Rafferty` and `dbr:Gina_Torres` are female and `dbr:Patrick_J.Adams` is male. The above example indicates that the use of literals along with their respective entities would add more semantics so that similar entities can be represented close to each other in the vector space while those dissimilar are further apart.

Recently, some approaches have been proposed which leverage the information present in literals to learn KG embeddings. The types of literals considered in these embedding methods are either text, numeric, images, or multi-modal literals, i.e., a combination of more than one medium of information. These methods use different techniques in order to incorporate the literals into the KG embeddings. However, data typed literals are not addressed in these KG embedding models and surveys that are conducted on KG embeddings. The main challenge with data typed literals, such as date and time, is that they require additional semantics to be represented in KG embeddings.

This survey analyses different embedding approaches, which make use of literals, and highlights their advantages and drawbacks in handling different challenges such as multi-valued data properties/relations, data typed literals, and units of literals. A review of the different applications used for model evaluation by different KG embedding models is also presented. Furthermore, experiments with some of the models have been conducted specifically on the link prediction task. The contribution of this paper is summarized as follows:

1. A detailed analysis of the existing literal enriched KG embedding models and their approaches. In addition, the models are categorized into different classes based on the type of literals used.
2. An evaluation oriented comparison of the existing models on the link prediction task is performed under same experimental settings.
3. The research gaps in the area of KG embeddings in using literals are indicated which can open directions for further research.

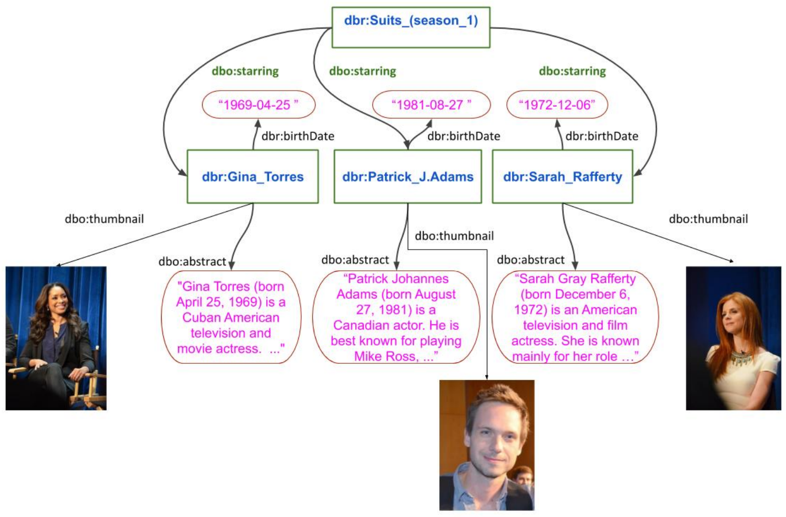
The rest of this paper is organized as follows: Section 2 presents a brief overview of related work. In Section 3, the problem formulation including definitions, preliminaries, types of literals and research questions are provided while Section 4 analyses different KG embedding techniques with literals is discussed. Section 5 reviews different tasks used to train or evaluate the embedding models is given. Section 6 discusses the experiment conducted with the existing KG embedding models with literals on the link prediction task. Finally, concluding remarks summarize our findings on KGs with literals and are presented along with future directions in Section 7.

2. Related Work

This section describes the state-of-the-art algorithms proposed for generating KG embeddings. It also gives a brief overview of the surveys already published following these lines and what is lacking in those studies.

A brief overview of the most popular KG embedding techniques, including the state-of-the-art approaches are short listed in Table 1. The categories presented in this table are inspired by a previous survey work [9] for the models without literals (column 1). These categories are created based on the methods used by the models, i.e., translation distance, semantic matching, entity types, relation paths, logical rules, temporal information, and graph structures. We have also categorized the techniques which use literals with respect to the same set of categories. Since a detailed discussion on these categories on the models without literals has already been presented in [9], in the current study, the main focus lies on analysing the models which make use of literals. The standard KG embedding techniques which are extended by the models with literals are listed in Table 2.

Few attempts have been made to conduct surveys on the techniques and applications of KG embed-



"Gina Torres (born April 25, 1969) is a Cuban American television and movie actress. ..."
"Patrick Johannes Adams (born August 27, 1981) is a Canadian actor. He is best known for playing Mike Ross, ..."
"Sarah Gray Rafferty (born December 6, 1972) is an American television and film actress. She is known mainly for her role ..."

Fig. 2. A small fraction of triples with literals taken from the KG DBpedia [4].

Table 1
KG embedding models and their categories.

Categories - based on the method used	**Models without literals**	**Models with Literals**
Translational Distance	TransE [11] and its extensions: TransH [14] TransR [15], TransD [16], TranSparse [17], TransA [18] etc.	TransEA [19], DKRL [13], IKRL [20], Jointly(desp) [21], Jointly [22], SSP [23], KDCoE [24], EAKGAE [25]
Semantic Matching	RESCAL [26] and Its Extensions: DistMult [27], HolE [28], ComplEx [29], and etc. Semantic Matching with Neural Networks: SME [30], NTN [31], MLP [32], and etc.	LiteralE [33], MKBE [34], MTKGNN [35], KGlove with literals [36], Extended RESCAL [37], LiteralE with blocking [38]
Entity Types	SSE [39], TKRL [40], Type constrained representation learning [41], Rules incorporated KG completion models [42], TRESCAL [43], Entity Hierarchy Embedding [44]	Extended RESCAL [37]
Relation Paths	PTransE [45], Traversing KGs in Vector Space [46], RTRANSE [47], Compositional vector space [48], Reasoning using RNN [49], Context-dependent KG embedding [50]	KBLRN [51]
Logical Rules	Rules incorporated KG completion models [42], Large-scale Knowledge Base Completion [52], KALE [53], Logical Background Knowledge for Relation Extraction [54], and etc.	
Temporal Information	Time-Aware Link Prediction [55], co-evolution of event and KGs [56], Know-evolve [57]	
Graph Structures	GAKE [58], Link Prediction in Multi-relational Graphs [59]	KBLRN [51]

dings [9, 61, 62]. The survey [61] is conducted on factorization based, random walk based, and deep learning based network embedding approaches such as DeepWalk, Node2vec, and etc. [9, 62] discuss only RESCAL [26] and KREAR [63] as methods which use attributes of entities for KG embeddings, and focus mostly on the structure-based embedding methods, i.e., methods using non-attributive triples, for ex-

Table 2
KG embedding models with literals and their corresponding base models.

Models with literals	The standard models they extend
Extended RESCAL [37]	RESCAL [26]
Jointly(desp) [21]	TransE [11]
DKRL [13]	TransE [11]
Jointly [22]	TransE [11]
SSP [23]	TransE [11]
KDCoE [24]	TransE [11]
KGlove with literals [36]	KGlove
LiteralE [33]	DistMult [27], ComplEx [29], ConvE [12]
TransEA [19]	TransE [11]
IKRL [20]	TransE [11]
MTKGRL [60]	TransE [11]
EAKGAE [25]	TransE [11]
MKBE [34]	DistMult [27], ConvE [12]

ample, translation based embedding models listed in Table 1. However, RESCAL is a matrix-factorization method for relational learning which encodes each object/data property as a slice of the tensor leading to an increase in the dimensionality of the tensor automatically. This method suffers from efficiency issues if literals are utilized while generating KG embeddings. Similarly, KREAR only considers those data properties which have categorical values, i.e., fixed number of values and ignores those which take any random literals as values. One of the recent surveys [64] summarizes the methods proposed so far on refining KGs. However, this survey does not confine itself to embedding techniques and also does not consider most of the approaches which are making use of literals. Another very recent related study [65], discusses different aspects of KG embedding models such as model architectures, training strategies, and hyperparameter optimization but it takes into consideration only those models without literals.

None of the surveys mentioned above include all the existing KG embedding models which make use of literals, such as the ones categorized as models incorporating information represented in literals in Table 1. To the best of our knowledge, this is the first attempt to analyse the algorithms proposed so far for generating KG embeddings using literals. In this paper, discussions on the type of literals, the embedding approaches, and the applications/tasks on which the embedding models are evaluated are given. A categorization of the models based on the type of literals they use is also provided.

This survey is an extension of an already published short survey [66]. The major difference between the two versions is that (i) this survey contains a much more detailed theoretical analysis of the KG embedding models with literals proposed so far, and (ii) it performs empirical evaluation of the discussed models under the same experimental settings under the example of link prediction.

3. Problem Formulation

This section briefly introduces the fundamentals of KGs and KG embeddings followed by a formal definition of KG embeddings with literals. It also poses various research questions about why conducting this study is a stepping stone for future development.

3.1. Preliminaries

Knowledge Graphs. Knowledge Graphs (KGs) consist of a set of triples $K \subseteq E \times R \times (E \cup L)$, where E is a set of resources referred to as entities, L a set of literals, and R a set of relations. An entity is identified by a URI which represents a real-world object or an abstract concept. A relation (or property) is a binary predicate and a literal is a string, date, or number eventually followed by its data type. For a triple `<h, r, t>`, *h* is a subject, *r* is a relation and *t* is an object. The subject and object are often referred to as *head* and *tail* entity respectively. The triples consisting of literals as objects are often referred to as *attributive triples*.

Relations (or Properties). Based on the nature of the objects, relations are classified into two main categories:

- **Object Relation** links an entity to another entity. E.g., in the triple `<dbr:Albert_Einstein, dbo:field, dbr:Physics>`, both `dbr:Albert_Einstein` and `dbr:Physics` are entities, the relation `dbo:field` is an *Object Relation*.
- **Data Type Relation** links an entity to its values, i.e., literals. For example, in `<dbr:Albert_Einstein, dbo:birthDate, "1879-03-14">`, where `"1879-03-14"` is a literal value, the relation `dbo:birthDate` is a *Data Type Relation*.

3.2. Types of Literals

Literals in a KG encode additional information which is not captured by the entities or relations. There are different types of literals present in the KGs:

- **Text Literals:** A wide variety of information can be stored in KGs in the form of free text such as names, labels, titles, descriptions, comments, etc. In most of the KG embedding models with literals, text information is further categorized into ***Short text*** and ***Long text***. The literals which are fairly short such as for relation like names, titles, labels, etc. are considered as *Short text*. On the other hand, for strings that are much longer such as descriptions of entities, comments, etc. are considered as *Long text* and are usually provided in natural language.
- **Numeric Literals:** Information encoded as integers, float and so on such as height, date, population, etc. also provide useful information about an entity. It is worth considering the numbers as distinct entities in the embedding models, as it has its own semantics to be covered which cannot be covered by string distance metrics. For instance, 777 is more similar to 788 than 77.
- **Units of Measurement:** Numeric literals often denote units of measurements to a definite magnitude. For example, Wikidata property `wdt:P2048` ("height") takes values in mm, cm, m, km, inch, foot and pixel. Hence, discarding the units and considering only the numeric values without normalization results in loss of semantics, especially if units are not comparable, e.g., units of length and units of weight.
- **Image Literals:** Images also provide latent useful information for modelling the entities. For example, a person's details such as age, gender, etc. can be deduced via visual analysis of an image depicting the person.
- **Other Types of Literals:** Useful information encoded in the form of other literals such as external URIs which could lead to an image, text, audio or video files.

3.3. Research Questions

As it can be seen from the above discussion that the information represented in the KGs is diverse, modelling these entities is a challenging task. The challenges which are further targeted in this study are given as follows:

- **RQ1** – *How can structured (triples with object relations) and unstructured information (attributive triples) in the KGs be combined into the representation learning?*
- **RQ2** – *How can the heterogeneity of the types of literals present in the KGs be captured and combined into representation learning?*

4. Knowledge Graph Embeddings with Literals

This section investigates KG embedding models with literals divided into the following different categories based on the types of literals utilized: (i) Text, (ii) Numeric, (iii) Image, and (iv) Multi-modal. A KG embedding model which makes use of at least two types of literals providing complementary information is considered as multi-modal. In the subsequent sections, a description of the models for each of the previously described categories analyzing their similarities and differences, followed by a discussion of potential drawbacks are provided.

4.1. Models with Text Literals

In this section, seven KG embedding models utilizing text literals are discussed, namely, Extended RESCAL [37], Jointly(desp) [21], DKRL [13], Jointly [22], SSP [23], KDCoE [24], and KGloVe with literals [36]. A detailed description followed by a summary presenting the comparison of these models is given along with their drawback. Moreover, in order to show the differences between the models based on complexity, the number of parameters of each model is presented in Table 3.

Extended RESCAL aims to improve the original RESCAL approach by extending its algorithm to process literal values more efficiently and to deal with the drawback of sparsity that accompanies tensors. In the original RESCAL approach, relational data is modeled as a three-way tensor X of size $n \times n \times m$, where n is the number of entities and m is the number of relations. An entry $X_{ijk} = 1$ denotes the existence of the triple with i-th entity as a subject, k-th relation as a predicate, and j-th entity as an object. If X_{ijk} is set to 0, it indicates that the triple doesn't exist. A new approach for tensor factorization is proposed which is performed on X. For further details refer to [37]. If attributive triples have to be modeled in such a way, then the literals will be taken as entities even if they cannot occur as subject in the triples. Including literals may lead to an increment in the runtime since a larger tensor has to be factorized.

In contrast to the original algorithm, the extended RESCAL algorithm handles the attributive triples in a separate matrix. The matrix factorization is performed jointly with the tensor factorization of the non-attributive triples. The attributive triples containing only text literals are encoded in an entity-attribute matrix D in such a way that the rows are entities and the columns are $<$ *data type relation*, *value* $>$ pairs. Given a triple with a textual data type such as `rdfs:label` or `yago:hasPreferredMeaning`, one or more such pairs are created by tokenizing and stemming the text in the object literal. The matrix D is then factorized into $D \approx AV$ with A and V being the latent-component representations of entities and attributes respectively. Despite the advantage that this approach has for handling multi-valued literals, it does not consider the sequence of words of the literal values. Note that Extended RESCAL represents RDF(S) data in such a way that there is no distinction drawn among A-Box and T-Box, i.e., both classes and instances are modeled equally as entities in a tensor. The T-Box is rather taken as soft constraints instead of letting them impose hard constraints on the model.

Jointly(Disp) is an approach which jointly learns embeddings of KGs and a text corpus of entity descriptions, i.e, it uses an alignment model to make sure the entities, relations, and words are represented in the same vector space. This approach consists of three components, namely, knowledge model, text model, and alignment model. The knowledge model is used to capture the semantics of the structured information from the KG. Given a triple $< h, r, t >$, the model defines the plausibility of the triple, same as in [67]:

$$Pr(h|r,t) = \frac{exp\{z(h,r,t)\}}{\sum_{\tilde{h} \in I} exp\{z(\tilde{h},r,t)\}}, \tag{1}$$

where $z(h,r,t) = b - 0.5 \cdot \|h + r - t\|_2^2$, $b = 7$. Analogously, $Pr(r|h,t)$ and $Pr(t|h,r)$ are defined.

Then, the loss function of the knowledge model is defined as follows:

$$L_K = \sum_{(h,r,t)} [\log Pr(h|r,t) + \log Pr(t|h,r) + \log Pr(r|h,t)]. \tag{2}$$

The text model adopts the same assumption made in [67] that is if two words occur in the same context then there is a relation between them. Based on this assumption, the text model defines the probability of a pair of words w and v co-occurring in a text window as follows:

$$Pr(w|v) = \frac{exp\{z(w,v)\}}{\sum_{\tilde{w} \in V} exp\{z(\tilde{w},v)\}}, \tag{3}$$

where $z(w,v) = b - 0.5 \cdot \|\mathbf{w} - \mathbf{v}\|_2^2$. Then, the loss function of the text model is given as:

$$L_T = \sum_{(w,v)} \log Pr(w|v). \tag{4}$$

The role of the third component, the alignment model, is to put the embeddings of the entities, relations, and words into the same vector space. This submodel works by utilizing entity descriptions to align these embeddings. For every word w in the description of entity e, the conditional probability of predicting w given e is defined as :

$$Pr(w|e) = \frac{exp\{z(e,w)\}}{\sum_{\tilde{w} \in V} exp\{z(e,\tilde{w})\}}, \tag{5}$$

where $z(e,w) = b - 0.5 \cdot \|\mathbf{e} - \mathbf{w}\|_2^2$. The entity vector $\mathbf{e}$ in Eq 5 is the same as the entity vector appearing in Eq 1, i.e., an entity has a single unified representation which captures the semantics from both the structured KG and the entity descriptions. $Pr(e|w)$ is defined analogously. Based on the definition given in Eq 5, the loss function of the alignment model is defined as:

$$L_A = \sum_{e \in \mathcal{E}} \sum_{w \in D_e} [\log Pr(w|e) + \log Pr(e|w)], \quad (6)$$

where $\mathcal{E}$ and D_e denote the set of entities and the description of the entity e respectively.

By adopting the joint embedding framework in [67], the main loss of Jointly(desp) is defined as follows:

$$L(\{e_i\}, \{r_j\}, \{w_l\}) = L_K + L_T + L_A. \quad (7)$$

DKRL extends TransE [11] by utilizing the descriptions of entities. For each entity e, two kinds of vector representations are learned, i.e., structure-based e_s and description-based e_d. These two kinds of entity representations are learned simultaneously into the same vector space but not forced to be unified so that novel entities with only descriptions can be represented. In order to achieve this, given a certain triple $< h, r, t >$ the energy function of the DKRL model is defined as:

$$\begin{aligned} E = &||h_s + r - t_s|| + ||h_d + r - t_d|| \\ &+ ||h_s + r - t_d|| + ||h_d + r - t_s||, \end{aligned} \quad (8)$$

where h_s and t_s are the structure-based representations, and h_d and t_d are the description-based representations of their corresponding entities.

In order to learn structure-based representations, the TransE approach is directly applied which considers the relation in a triple as the translation from the head entity to the tail entity. On the other hand, Continuous Bag of Words (CBOW) and a deep Convolutional Neural Network (CNN) model have been used to generate the description-based representations of the head and tail entities. In case of CBOW, short text is generated from the description based on keywords and their corresponding word embeddings are summed up to generate the entity embedding. In the CNN model, after preprocessing the description, pretrained word vectors from Wikipedia are provided as input. This CNN model has five layers and after every convolutional layer pooling is applied to decrease the parameter space of CNN and filter noises. Max-pooling is applied for the first pooling and mean pooling for the last one. The activation function used is either tanh or ReLU. The CNN model works better than CBOW because it preserves the sequence of words.

In order to train DKRL, the following margin-based score function is considered as an objective function and minimized using a standard back propagation using stochastic gradient descent (SGD)

$$\begin{aligned} L = \sum_{(h,r,t) \in T} \sum_{(h',r',t') \in T'} &max(\gamma + d(h + r, t) \\ &- d(h' + r', t'), 0), \end{aligned} \quad (9)$$

where $\gamma > 0$ is a margin hyperparameter, d is a dissimilarity function and T' is the set of corrupted triples. The representation of the entities can be either structure-based or description-based.

Jointly [22] learns KG embeddings by leveraging entity descriptions. Specifically, it learns a joint embedding of an entity by combining its structure-based and description-based representations with a gating mechanism. The gate is used to find balance between the structure-based and the description-based representations. For a certain entity a representation can be encoded from its descriptions by converting the description into fixed length vector. In Jointly, different text encoders have been used such as bag-of-words, LSTM, and Attentive LSTM.

For an entity e, its joint representation $\mathbf{e}$ is a linear interpolation between its structure-based representation ($\mathbf{e_s}$) and description-based representation ($\mathbf{e_d}$), which is defined as:

$$\mathbf{e} = g_e \odot \mathbf{e_s} + (1 - g_e) \odot \mathbf{e_d}, \quad (10)$$

where $\odot$ is an element-wise multiplication and g_e is a gate to balance the two information sources (structure and text) which is computed as $g_e = \alpha(\tilde{g_e})$ with $g_e = \tilde{g_e} \in \mathcal{R}^d$ being real-value vector stored in a lookup table.

The entity descriptions are encoded using either bag-of-words, LSTM, or Attentive LSTM (ALSTM) encoders in order to generate text-based representation for the corresponding entities. On the other hand, to better model the structure-based embedidngs, entities and relations can be pre-trained with any existing KG embedding models, such as TransE.

Jointly's score function is inspired by TransE and defined as follows:

$$\begin{aligned} f(h, r, t; d_h, d_t) = &\|(\mathbf{g}_h \odot \mathbf{h}_s + (1 - \mathbf{g}_h) \\ \odot \mathbf{h}_d) + r - (g_t \odot \mathbf{h}_t &+ (1 - \mathbf{g}_t) \odot \mathbf{t}_d)\|_2^2. \end{aligned} \quad (11)$$

where $\mathbf{h}_s$, $\mathbf{h}_d$, and $\mathbf{g}_h$ are the head entity's structure-based embedding, description-based embedding, and gate respectively whereas $\mathbf{t}_s$, $\mathbf{t}_d$, and $\mathbf{g}_t$ are the tail entity's structure-based embedding, description-based embedding, and gate respectively.

SSP (Semantic Space Projection) [23] is a joint embedding model which learns from both structured/symbolic triples and textual descriptions. Differently from DKRL and Jointly(Desp), where first-order constraints which are weak in capturing the correlation of textual descriptions and symbolic triples are applied, SSP follows the principle that triple embedding is considered always as the main procedure and textual descriptions must interact with triples in order to learn better representation. Therefore, triple embedding is projected onto a semantic subspace such as a hyperplane to allow strong correlation by adopting quadratic constraint.

SSP applies the following scoring function:

$$f_r(h,t) = -\lambda\|\mathbf{e} - \mathbf{s}^T\mathbf{e}\mathbf{s}\|_2^2 + \|\mathbf{e}\|_2^2, \tag{12}$$

where

$$\mathbf{e} \doteq \mathbf{h} + \mathbf{r} - \mathbf{t}, \tag{13}$$

and

$$\mathbf{s} \doteq \frac{\mathbf{s_h} + \mathbf{s_t}}{\|\mathbf{s_h} + \mathbf{s_t}\|}. \tag{14}$$

Note that λ is a suitable hyper-parameter, $\mathbf{h}$ and $\mathbf{t}$ are the structure (symbolic triples) based embedding of the head and tail entities respectively, $\mathbf{s_h}$ and $\mathbf{s_t}$ are the semantic vectors generated from the textual descriptions of the head and tail entities respectively. SSE adopts the Non-negative Matrix Factorization (NMF) topic model to generate description-based semantic vectors for entities ($\mathbf{s_h}$ and $\mathbf{s_t}$), i.e., by treating each entity description as a document and taking the topic distribution of the document as the representation of the corresponding entity.

SSP provides two different settings for training which are referred to as **Std** and **Joint**. In Std, a pre-trained topic model with NMF is used to obtain description-based semantic vectors. These description-based vectors are fixed during training but the other parameters are optimized. On the other hand, in the Joint setting the topic model is also learnt simultaneously with the KG embeddings instead of using a fixed pre-trained vectors.

KDCoE focuses on the creation of an alignment between entities of multilingual KGs by creating new Inter-Lingual Links (ILLs) based on an embedding approach which exploits entity descriptions. The model uses a weakly aligned multilingual KG for semi-supervised cross-lingual learning. It performs co-training of a multilingual KG embedding Model (KGEM) and a multilingual entity Description Embedding Model (DEM) iteratively in order for each model to propose a new ILL alternately. KGEM is composed of two components, i.e., a knowledge model and an alignment model, to learn embeddings based on structured information from the KGs (the non attributive triples). Given a set of languages $\mathcal{L}$, a separate k$_1$-dimensional embedding space $\mathbb{R}_L^{k_1}$ is used for each language $L \in \mathcal{L}$ to represent the corresponding relations R_L and entities E_L. In order to learn the embeddings for R_L and E_L, the knowledge model adopts TransE and thus uses hinge loss as its objective function. On the other hand, a linear-transformation-based technique which has the best performance in case of cross-lingual inferences is adopted for the alignment model. This technique employs the following objective function:

$$S_A = \sum_{(e,e')\in I(L_i,L_j)} \|M_{ij}\mathbf{e} - \mathbf{e}'\|_2, \tag{15}$$

where $I(L_i, L_j)$ is ILLs between the languages L_i and L_j, and M_{ij} is a $k_1 \times k_1$ matrix used as a linear transformation on entity vectors from L_i to L_j.

Let S_K be the hinge loss function used by the knowledge model, the KGEM model then minimizes $S_{KG} = S_K + \alpha S_A$, where α is a positive hyperparameter. In case of DEM model, an attentive gated recurrent unit encoder (AGRU) is used to encode the multilingual entity descriptions. DEM applies multilingual word embeddings in order to capture the semantic information of multilingual entity descriptions from the word level. The two models, i.e., KGEM and DEM, are iteratively co-trained in order for each model to propose a new ILL alternately.

KGloVe with literals is an experimental attempt to incorporate entity descriptions in KGloVe KG embedding approach. The experiment is conducted on DBpedia considering the abstracts and comments of entities as their descriptions. The main goal is to extract named entities from the textual description and for every entity in the text, to replace those words representing it with the entity itself and then take its neighbouring words and entities as its context. The approach works by creating two co-occurrence matrices independently and then by merging them at the end so that a joint embedding can be performed. The first matrix is generated using the same technique as in KGloVe [68], i.e., by performing Personalized PageRank (PPR) on the (weighted) graph followed by the same optimisation used in the GloVe [69] approach.

In order to create the second matrix, the Named Entity Recognition (NER) task is performed on the entity description text using the list of entities and predicates of the KG as an input. The NER step employs a simple exact string matching technique which leads to numerous drawbacks such as missing entities due to having different keywords with the same semantics. All the English words that do not match any entity labels are added to the entity-predicate list. Then GloVe co-occurrence for text is applied to the modified text (i.e., DBpedia abstract and comments) using the entity-predicate and word list as input. Finally, the two co-occurrence matrices are summed up together to create a single unified matrix. The proposed approach has been evaluated on classification and regression tasks and the result indicates that for most of the classifiers used, except SVM, the approach does not bring significant improvement to KGloVe. However, the approach can be improved using parameter tuning with extensive experiments.

Summary The basic differences between these models lie in the methods used to exploit the information given in the text literals and combine them with structure-based representation. One major advantage of KDCoE over text literal based embedding models is that it considers descriptions present in multilingual KGs. Also, both DKRL and KDCoE embedding models are designed to perform well for the novel entities, which have only attributive triples in the KGs. Jointly(Desp) aligns KG embeddings and word embeddings on word level, which may lead to losing some semantic information on phrase or sentence level. Jointly applies a gating mechanism which allows to automatically find a balance between the structural and textual information. It also uses an LSTM encoder which enables the model to select the most related information for an entity from its text description according to different relations. Unlike DKRL and Jointly(Desp), SSP focuses on characterizing the stronger correlations between entity descriptions and structured triples by projecting triple embedding onto a semantic subspace such as a hyper-plane, as discussed above. Another common drawbacks among the presented approaches with text literals is they focus mostly on descriptions, which is long natural language text and thus, other types of text literals, such as, names, labels, titles, etc. are not widely considered. Moreover, another way to compare these approaches is by looking at their model complexity. Table 3 presents the complexity of these models in terms of their number of parameters.

Table 3
Complexity of the models with text literals in terms of the number of parameters. Θ is the number of parameters in the base model, H is the entity embedding size, N_d is the number of data relations, L is the number of attribute-value pairs, N_r is the number of relations, N_w is the number of words, H' is the word embedding size, $N_0^{(1)}$ is the dimension of input vectors at the first layer, $N_1^{(1)}$ is the dimension of output vectors at layer 1, K is window size, $N_0^{(2)}$ is the dimension of input vectors at the second layer, $N_1^{(2)}$ is the dimension of output vectors at second layer, N_{e_1} and N_{e_2} denote the number of entities in two different languages of a multilingual KG, N_{r_1} and N_{r_2} denote the number of relations in two different languages of a multilingual KG, N is the total number of entities and relations, and M is the total number of entities, relations and words.

Model	#Parameter
Extended Rescal	$\Theta + HL$
Jointly(Desp)	$\Theta + N_w H'$
DKRL	$\Theta + N_w H' + N_0^1 K N_1^{(1)} + N_0^{(2)} K N_1^{(2)}$
Jointly(ALSTM)	$\Theta + (2H+4)H$
SSP	$\Theta + (N_e + N_w)H$
KDCoE	$(N_{e_1} + N_{e_2} + N_{r_1} + N_{r_2} + H)H$ $+(5H' + 3N_w)H'$
KGlove with literals	$(N+1)N + N_w + M$

4.2. Models with Numeric Literals

In this section, the analysis of the presented KG embedding models which use numeric literals, namely, MT-KGNN [35], KBLRN [51], LiteralE [33], and TransEA [19] are presented followed by a summary. Moreover, in order to show the differences between the models based on complexity, the number of parameters of each model is presented in Table 4.

MT-KGNN is an approach for both relational learning and non-discrete attribute prediction on knowledge graphs in order to learn embeddings for entities, object properties, and data properties. It is composed of two networks, namely, the Relational Network (RelNet) and the Attribute Network (AttrNet). RelNet is a binary (pointwise) classifier for triple prediction whereas AttrNet is a regression task for attribute value prediction. Given n, m, and l as entity, relation, and literal embedding dimensions respectively, the model passes as an input $[e_i, r_k, e_j, t]$ to RelNet and $[a_i, v_i, a_j, v_j]$ to AttrNet, where e_i , $e_j \in R^n$, $r_k \in R^m$, t is the classification target which is 0 or 1, $a_i, a_j \in R^l$, and v_i and v_j

are normalized continuous values in the interval $[0, 1]$. Note that the inputs to AttrNet, i.e., $[a_i, v_i, a_j, v_j]$, are taken from attributive triples with non-discrete literal values. An embedding lookup layer is used to retrieve the corresponding vector representations given these inputs as one-hot encoded indices.

In RelNet, a concatenated triple is passed through a nonlinear transform and then a sigmoid function is applied to get a linear transform:

$$g_{rel}(e_i, r_k, e_j) = \sigma(\overrightarrow{w}^T f(W_d^T [\overrightarrow{e_i}; \overrightarrow{e_j}; \overrightarrow{r_k}]) + b_{rel}), \tag{16}$$

where $w \in R^{h \times 1}$ and $W_d \in R^{3n \times h}$ are parameters of the network. σ, f, and b_{rel} are the sigmoid function, the hyperbolic tangent function tanh, and a scalar bias respectively. RelNet is trained by minimizing the following cross entropy loss function:

$$L_{rel} = -\sum_{i=1}^{N} t_i \log g_{rel}(\xi_i) + (1 - t_i) \log(1 - g_{rel}(\xi_i))), \tag{17}$$

where ξ_i denotes triple i in batch of size N and t_i takes the value 0 or 1. In case of AttrNet, two regression tasks are performed, one for the head data properties and another for those of the tail. The following scoring functions are defined for these two tasks:

$$g_h(a_i) = \sigma(\overrightarrow{u}^T f(B^T [\overrightarrow{a_i}; \overrightarrow{e_i}]) + b_{z_1}), \tag{18}$$

and

$$g_t(a_j) = \sigma(\overrightarrow{y}^T f(C^T [\overrightarrow{a_j}; \overrightarrow{e_j}]) + b_{z_2}), \tag{19}$$

where $u, y \in R^{h_a \times 1}$ and $B, C \in R^{2n \times h_a}$ are parameters of AttrNet. h_a is the size of the hidden layer and b_{z_1}, b_{z_2} are scalar biases. Each AttrNet is trained by optimizing Mean Squared Error (MSE) loss function:

$$MSE(s, s^*) = \frac{1}{N} \sum_{i=1}^{N} (s_i - s_i^*)^2. \tag{20}$$

where s and s^* are predicted labels (scores computed by the model) and ground truth labels respectively.
The overall loss of the AttrNet is computed by adding the MSE of the head AttrNet and that of the tail AttrNet as follows:

$$L_{attr} = MSE(g_h(a_i), (a_i)^*) + MSE(g_t(a_j), (a_j)^*), \tag{21}$$

where $(a_i)^*, (a_j)^*$ are the ground truth labels. Finally, the two networks are trained in a multi-task fashion using a shared embedding space.
KBLRN works by combining relational (R), latent (L), and numerical (N) features together. The model is designed mainly for the purpose of KG completion. It uses a probabilistic PoE (Product of Experts) method to combine these feature types and train them jointly end to end. Each relational feature is formulated as a logical formula, by adopting the rule mining approach AMIE+ [70], to be evaluated in the KB to compute the feature's value. The latent features are the ones that are usually generated using an embedding approach such as DistMult. Numerical features are used with the assumption that, for some relation types, the differences between the head and tail can be seen as characteristics for the relation itself. Given a triple $d = (h, r, t)$, for each (relation type r, and feature type $F \in \{L, R, N\}$) pair, individual experts are defined based on linear models and DistMult embedding method as follows:

$$f_{(r,L)}(d \mid \theta_{(r,L)}) = exp((e_h * e_t) \cdot w^r), \tag{22}$$

$$f_{(r,R)}(d \mid \theta_{(r,R)}) = exp(r_{(h,t)} \cdot w^r_{rel}), \tag{23}$$

$$f_{(r,N)}(d \mid \theta_{(r,N)}) = exp(\phi(n_{(h,t)}) \cdot w^r_{num}), \tag{24}$$

and

$$f_{(r',F)}(d \mid \theta_{(r',F)}) = 1 \; for \; all \; r' \neq r \tag{25}$$

where $w_r, w^r_{rel}, w^r_{num}$ are the parameter vectors for the latent, relational, and numerical features corresponding to the relation r. Also, $*$ is the element-wise product, $\cdot$ is the dot product, and ϕ is the radial basis function (RBF) applied element-wise to the differences of values $n_{(h,t)}$ computed as follows:

$$\phi(n_{(h,t)}) = [exp(\frac{-\|n^{(1)}_{(h,t)} - c_1\|^2_2}{\sigma^2_1})\dots exp(\frac{-\|n^{(d_n)}_{(h,t)} - c_{d_n}\|^2_2}{\sigma^2_{d_n}})]. \quad (26)$$

Here, d_n corresponds to the relevant numerical features. A PoE's probability distribution for a triple $d = (h, r, t)$ is defined as follows:

$$p(d \mid \theta_1 \dots \theta_n) = \frac{\Pi_F f_{(r,F)}(d \mid \theta_{(r,F)})}{\sum_c \Pi_F f_{(r,F)}(c \mid \theta_{(r,F)})}, \quad (27)$$

where c denotes all possible triples. The parameters of the entity embedding model are shared by all the experts in order to create dependencies between them. In this approach, the PoE are trained with negative sampling and a cross entropy loss to give high probability to observed triples.

LiteralE incorporates literals into existing latent feature models designed for link prediction. In this approach, without loss of generality, the focus lies on incorporating numerical literals into three state-of-the-art embedding methods: DistMult, ComplEx, and ConvE. Given a base model, for instance Distmult, LiteralE modifies the scoring function f used in Distmult by replacing the vector representations of the entities e_i in f with literal enriched representations e_i^{lit}. In order to generate e_i^{lit}, LiteralE uses a learnable transformation function g which takes e_i and its corresponding literal vectors l_i as inputs and maps them to a new vector. The function g is defined, as shown below, based on the concept of GRU in order to make it flexible, learnable, and capable to decide, if it is beneficial to incorporate the literal information or not:

$$g : \mathbb{R}^H \times \mathbb{R}^{N_d} \to \mathbb{R}^H, \quad (28)$$

and

$$\mathbf{e}, \mathbf{l} \mapsto \mathbf{z} \odot \mathbf{h} + (1 - \mathbf{z}) \odot \mathbf{e}, \quad (29)$$

where

$$\mathbf{z} : \sigma(\mathbf{W}^T_{ze}\mathbf{e} + \mathbf{W}^T_{zl}\mathbf{l} + \mathbf{b}), \quad (30)$$

and

$$\mathbf{h} = h(\mathbf{W}^T_h[\mathbf{e}, \mathbf{l}]). \quad (31)$$

Note that $\mathbf{W}_{ze} \in \mathbb{R}^{H\times H}, \mathbf{W}_{zl} \in \mathbb{R}^{N_d\times H}, \mathbf{b} \in \mathbb{R}^H$, and $\mathbf{W}_h \in \mathbb{R}^{H+N_d\times H}$ are the parameters of g, σ is the sigmoid function, $\odot$ denotes the element-wise multiplication, and h is a component-wise nonlinearity. The scoring function $f(\mathbf{e}_i, \mathbf{e}_j, \mathbf{r}_k)$ has been replaced with $f(g(\mathbf{e}_i, \mathbf{l}_i), g(\mathbf{e}_j, \mathbf{l}_j), \mathbf{r}_k)$ and trained following the same procedure as in the base model.

TransEA has two component models; a directly adopted translation-based structure embedding model (i.e., TransE) and a newly proposed attribute embedding model. In the former, the scoring function of a given triple $< h, r, t >$, is defined as follows:

$$f_r(h, t) = -\|h + r - t\|_{1/2}, \quad (32)$$

where $||x||_{1/2}$ denotes either the L1 or L2 norm. The loss function of the structure embedding, for all the relational triples in the KG, is defined as:

$$L_R = \sum_{<h,r,t>\in T} \sum_{<h',r,t'>\in T'} max(\gamma + f_r(h, t) - f_r(h', t'), 0), \quad (33)$$

where T' denotes the set of negative triples constructed by corrupting either the head or the tail entity and $\gamma > 0$ is a margin hyperparameter.

For the attribute embedding, it uses all attributive triples containing numeric values as input and applies a linear regression model to learn embeddings of entities and attributes. Given an attributive triple $< e, a, v >$, the scoring function is defined as:

$$f_a(e, v) = -\|\mathbf{a}^T \cdot \mathbf{e} + b_a - v\|_{1/2}, \quad (34)$$

where **a** and **e** are vectors of attribute a and entity e, b_a is a bias for attribute a. On the other hand, given all the attributive triples with numeric values S, the loss function for the attributive embedding is computed as:

$$L_A = \sum_{<e,a,v>\in S} f_a(e, v), \quad (35)$$

The main loss function for TransEA (*i.e.*, $L = (1 - \alpha) \cdot L_R + \alpha \cdot L_A$) is defined by taking the sum of the respective loss functions of the component models with a hyperparameter to assign a weight for each of the models. Finally, the two models are jointly optimized in the training process by sharing the embeddings of entities.

Summary Despite their support for numerical literals, all the embedding methods discussed fail to interpret the semantics behind units/data typed literals. For instance, given the following two triples taken from DBpedia,

```
<http://dbpedia.org/resource/Anton_Baraniak,
    dbp:weight, "110.0"^^<http://dbpedia.org/
    datatype/kilogram>,
<http://dbpedia.org/resource/Katelin_Snyder,
    dbp:weight, "110.0"^^<http://dbpedia.org/
    datatype/pound>
```

the literal value "110.0" from the first triple and the literal value "110.0" from the second triple could be considered exactly the same if the semantics of the types kilogram and pound are ignored. Moreover, most of the models do not have a proper mechanism to handle multi-valued literals.

Regarding model complexity, the number of parameters used in each model is presented in Table 4 to show the complexity in terms of the parameters. It is noted that the complexity of the models depend on the size of the dataset and TransEA has lower complexity as compared to the other models.

Table 4

Complexity of the models with numerical literals in terms of the number of parameters. Θ is the number of parameters in the base model, H is the entity embedding size, N_d is the number of data relations, Λ is the size of the hidden layer in the Attrnet networks of MTKGNN, N_r is the number of relations, and M is attribute embedding size.

Model	#Parameters
LiteralE with g	$\Theta + 2H^2 + 2N_d\mathrm{H} + \mathrm{H}$
LiteralE with g_{lin}	$\Theta + (N_d\mathrm{H} + \mathrm{H})\mathrm{H}$
MTKGNN	$\Theta + N_d\mathrm{H} + 2(2\mathrm{H}\Lambda + \Lambda)$
KBLN	$\Theta + N_rN_d$
TransEA	$\Theta + N_dM$

4.3. Models with Image Literals

In this section, KG embedding models utilizing images of entities, namely, IKRL [20] and MTKGRL [60] are discussed. First, a detailed analysis of the models is presented followed by a summary. Moreover, in order to show the differences between the models based on complexity, the number of parameters of each model is presented in Table 5.

IKRL [20] learns embeddings for KGs by jointly training a structure-based representation with an image-based representation. The structure-based representation of an entity is learned by adapting a conventional embedding model like TransE. For the image-based representation, given the fact that an entity may have multiple image instances, an image encoder is applied to generate an embedding for each instance of a multi-valued image relation. The image encoder consists of a neural representation module and a projection module to extract discriminative features from images and to project these representations from image space to entity space respectively.

For the i-th image, its image-based representation p_i in entity space is computed as:

$$\mathbf{p}_i = \mathbf{M} \cdot f(img_i), \tag{36}$$

where $M \in \mathbb{R}^{d_i \times d_s}$ is the projection matrix with d_i and d_s representing the dimension of image features and the dimension of entities respectively. $f(img_i)$ is the i-th image feature representation in image space.

Attention-based multi-instance learning is used to integrate the representations learned for each image instance by automatically calculating the attention that should be given to each instance. The attention for the i-th image representation $p_i^{(k)}$ of the k-th entity is given as:

$$att(\mathbf{p}_i^{(k)}, \mathbf{e}_S^{(k)}) = \frac{exp(\mathbf{p}_i^{(k)} \cdot \mathbf{e}_S^{(k)})}{\sum_{j=1}^{n} exp(\mathbf{p}_j^{(k)} \cdot \mathbf{e}_S^{(k)})}, \tag{37}$$

where $\mathbf{e}_S^{(k)}$ denotes the structure-based representation of the k-th entity. The higher the attention the more similar the image-based representation is to its corresponding structure-based representation which indicates that it should be given more importance when aggregating the image-based representations. The aggregated image-based representation for the k-th entity is defined as follows:

$$\mathbf{e}_I^{(k)} = \sum_{i=1}^{n} \frac{att(\mathbf{p}_i^{(k)}, \mathbf{e}_S^{(k)}) \cdot \mathbf{p}_i^{(k)}}{\sum_{j=1}^{n} att(\mathbf{p}_j^{(k)}, \mathbf{e}_S^{(k)})}. \tag{38}$$

Given a triple, the overall energy function is defined by combining four energy functions (i.e., $E(h,r,t) = E_{SS} + E_{II} + E_{SI} + E_{IS}$. These energy functions are based on two kinds of entity representations (i.e, structure-based and image-based representations). The first energy function (i.e., $E_{SS} = \|h_S + r - t_S\|$) is same as

TransE and the second function (i.e., $E_{II} = ||h_I + r - t_I||$) uses their corresponding image-based representations for both head and tail entities. The third function (i.e., $E_{SI} = ||h_S + r - t_I||$) is based on the structure-based representation of the head entity and the image-based representation of the tail entity whereas the fourth function (i.e., $E_{IS} = ||h_I + r - t_S||$) is the exact opposite. These third and forth functions ensure that both structure-based representation and image-based representations are learned into the same vector space.

Given the energy function $E(h, r, t)$, a margin-based scoring function is defined as follows:

$$L = \sum_{(h,r,t)\in T} \sum_{(h',r',t')\in T'} max(\gamma + E(h, r, t) \\ -E(h', r', t'), 0), \tag{39}$$

where γ is a margin hyperparameter and T' is the negative sample set of T generated by replacing the head entity, tail entity or the relation for each triple in T. Note that triples which are already in T are removed from T'.

MTKGRL [60] is a KG embedding approach which combines structural (symbolic), visual, and linguistic KG representations. The structural representations are created by adopting TransE embedding technique whereas visual embeddings are obtained from the feature layers of deep networks for image classification on the images that are associated with entities. For linguistic representations, pre-trained word embedding technique, specifically the skipgram model, is used. However, the information source for the linguistic representation are not literals from the KG but an external source, i.e., the word embedding model, trained on Google 100B token news dataset. Due to this fact, the model MTKGRL is not considered as multi-modal KG embedding model in the context of this survey and thus, it is not categorized under 'Models with Multimodal Literals' (Sec 4.4).

MTKGRL defines an energy function for each kind of representation and also their combinations, i.e., structural energy, multimodal energies, and structural-multimodal energies. Structural energy is adopted from TransE, which is defined as $E_S = ||\boldsymbol{h_s} + \boldsymbol{r_s} - \boldsymbol{t_s}||$. The multimodal representations for the head and tail entities are computed as $\boldsymbol{h_m} = \boldsymbol{h_w} \oplus \boldsymbol{h_i}$ and $\boldsymbol{t_m} = \boldsymbol{t_w} \oplus \boldsymbol{t_i}$ respectively, where the operator $\oplus$ can be a concatenation operator or a mapping function.

The multimodal energy function under the translational assumption is given as:

$$E_{M1} = ||\mathbf{h_m} + \mathbf{r_s} - \mathbf{t_m}||. \tag{40}$$

E_{M1} can be extended by considering the structural embeddings in addition to the multimodal embeddings as follows:

$$E_{M2} = ||(\mathbf{h_m} + \mathbf{h_s})\mathbf{r_s} - (\mathbf{t_m} + \mathbf{t_s})||. \tag{41}$$

On the other hand, in order to allow the structural and multimodal embeddings to be learned in the same vector space, the following structural-multimodal energies are defined as shown below:

$$E_{SM} = ||\mathbf{h_s} + \mathbf{r_s} - \mathbf{t_m}|| \tag{42a}$$

$$E_{MS} = ||\mathbf{h_m} + \mathbf{r_s} - \mathbf{t_s}|| \tag{42b}$$

The overall energy function, shown in Equation 43, is defined by combining the aforementioned energy functions, i.e., E_S, E_{M1}, E_{M2}, E_{SM}, E_{MS}.

$$E(h, r, t) = E_S + E_{M1} + E_{M2} + E_{SM} \\ +E_{MS} \tag{43}$$

Finally, a margin-based ranking loss function is minimized in order to train the model.

Summary IKRL makes use of the images of entities for KG representation learning by combining structure-based representation with image-based representation. However, given a triple $< h, r, t >$, in order to achieve very good representations for the entities h and t, both entities are required to have images associated with them. The other issue with this model is that an image is considered as an attribute of only those entities it is associated with. For example, if there is an image of two entities e_1 and e_2 but the image is associated with only e_1, then it will be taken as one image instance of e_1 but not of e_2. However, it would be more beneficial to explicitly associate images with all the entities they represent before using them for learning KG embedding. Some of the main points which make MTKGRL differ from IKRL are: i) in addition to images, MTKGRL uses linguistic embeddings for entities, ii) MTKGRL introduces an additional energy function that considers both linguistic and visual representations of entities as discussed above. These differences allow MTKGRL to learn better representation for KGs as compared to IKRL.

Table 5
Complexity of the models with text literals in terms of the number of parameters. Θ is the number of parameters in the base model, H is the entity embedding size, H_i represents the dimension of image features, $\theta_{AlexNet}$ is the number of parameters in AlexNet [71], N_e represents the number of entities, and N_i is the number of images.

Model	#Parameter
IKRL	$\Theta + H_iH + \Theta_{AlexNet}$
MTKGRL	$\Theta + N_eH + N_iH_i$

4.4. Models with Multi-modal Literals

This section presents an analysis of the embedding models making use of at least two types of literals providing complementary information. First, the category with numeric and text literals is discussed followed by the category with numeric, text, and image. Moreover, in order to show the differences between the models based on complexity, the number of parameters of each model is presented in Table 6.

4.4.1. Models with Numeric and Text Literals

LiteralE with blocking [38] proposes to improve the effectiveness of the data linking task by combining LiteralE with a CER blocking [72] strategy. Unlike LiteralE, given an attributive triple $< h, d, v >$, in addition to the object literal value v it also takes literals from URI infixes of the head entity h and the data relation d. The CER blocking is based on a two-pass indexing scheme. In the first pass, Levenshtein distance metric is used to process literal objects and URI infixes whereas in the second pass semantic similarity computation with WordNet [73] is applied to process object/data relations. All the extracted literals are tokenized into word lists so as to create inverted indices. The same training procedure as in LiteralE is used to train this model. For every given triple $< h, r, t >$, the scoring function f from LiteralE is adopted to compute scores for all the triples $< h, r, t' >$ in the knowledge graph. A sigmoid function, $p = \sigma(f(.))$, is used to produce probabilities. Then, the model is trained by minimizing the binary cross-entropy loss of the produced probability function vector with respect to the vector of truth values for the triples.

EAKGAE [25] is an approach designed for entity alignment between KGs by learning a unified embedding space for the KGs. The entity alignment task has three main modules: Predicate alignment, Embedding learning, and Entity alignment. The predicate alignment module merges two KGs together by renaming similar predicates so as to create unified vector space for the relationship embeddings. The embedding learning module jointly learns entity embeddings of two KGs using structure embedding (by adapting TransE) and attribute character embedding. The adapted TransE is customized in a way that more focus can be given to triples with aligned predicates. This is obtained by adding a weight α to control the embedding learning over the triples. Thus, the following objective function J_{SE} is defined for the structure-based embedding:

$$J_{SE} = \sum_{t_r \in T_r} \sum_{t'_r \in T'_r} max(0, \gamma + \alpha(f(t_r) - f(t'_r))), \tag{44}$$

and

$$\alpha = \frac{count(r)}{|T|}, \tag{45}$$

where T_r and T'_r are the sets of valid triples and corrupted triples respectively, $count(r)$ is the number of occurrences of the relation r, and $|T|$ is the total number of triples in the merged KG.

On the other hand, the attributing character embedding is designed to learn embeddings for entities from the strings occurring in the attributes associated with the entities. The purpose is to enable the entity embeddings from two KGs to fall into the same vector space despite the fact that the attributes come from different KGs. The attribute character embedding is inspired by the concept of translation in TransE. Given a triple (h, r, a), the data property r is interpreted as a translation from the head entity h to the literal value a i.e. $h + r = f_a(a)$ where $f_a(a)$ is a compositional function. This function encodes the attribute values into a single vector mapping similar attribute values into similar representation. Three different compositional functions SUM, LSTM, and N-gram-based functions have been proposed. SUM is defined as a summation of all character embeddings of the attribute value. In LSTM, the final hidden state is taken as a vector representation of the attribute value. The N-gram-based function, which shows better performance than the others according to their experiments, uses the summation of n-gram combination of the attribute value.

The following objective function is defined for the attribute character embedding:

$$J_{CE} = \sum_{t_a \in T_a} \sum_{t'_a \in T'_a} max(0, [\gamma + \alpha(f(t_a) - f(t'_a))]), \tag{46}$$

$$T_a = < h, r, a > h \in G; f(t_a) = \|h + r - f_a(a)\|,$$

and

$$T'_a = \{< h', r, a > h' \in G\} \cup \{< h, r, a' > a' \in A\},$$

where, T_a and T'_a are the sets of valid attribute triples and corrupted attribute triples with A being the set of attributes in a given KG G. The corrupted triples are created by replacing the head entity with a random entity or the attribute with a random attribute value. Here, $f(t_a)$ is the plausibility score computed based on the embedding of the head entity h, the embedding of the relation r, and the vector representation of the attribute value obtained using one of the compositional functions $f_a(a)$.

The attribute character embedding h_{ce} is used to shift the structure embedding h_{se} into the same vector space by minimizing the following objective function:

$$J_{SIM} = \sum_{h \in G_1 \cup G_2} [1 - \|h_{se}\|_2 \cdot \|h_{ce}\|_2], \tag{47}$$

where, $\|x\|_2$ is the L_2-Norm of vector x. This way the similarity of entities from two KGs is captured by the structure embedding based on the entity relationships and by the attribute embedding based on the attribute values.

All the three functions are summed up to an overall objective function J (i.e., $J = J_{SE} + J_{CE} + J_{SIM}$) for jointly learning both structure and attribute embeddings. Finally, the alignment is done by defining a similarity equation with a specified threshold. Moreover, a transitivity rule has been applied to enrich triples in the KGs to get a better attribute embedding result.

Summary The common drawback with both methods (LiteralE with blocking and EAKGE) is that text and numeric literals are treated in the same way. They also do not consider literal data type semantics or multi-valued literals in their approach. Furthermore, since EAKGAE is using character-based attribute embedding, it fails to capture the semantics behind the co-occurrence of syllables.

4.4.2. Models with Numeric, Text, and Image Literals

MKBE [34] is a multi-modal KG embedding, in which the text, numeric and image literals are modelled together. The main objective of this approach is to utilize all the observed subjects, objects, and relations (object properties and data properties) in order to predict whether any fact holds. It extends DistMult, which creates embedding for entities and relations, by adding neural encoders for different data types. Given a triple $< s, r, o >$, the head entity s and the relation r are encoded as independent embedding vectors using one-hot encoding through a dense layer. Similarly, if the object o is a categorical value, then it will be represented through a dense layer with a relu activation which has the same number of nodes as the embedding space dimension. On the other hand, if the object o is rather a numerical value, then a feed forward layer, after standardizing the input, is used in order to learn embeddings for o by projecting it to a higher-dimensional space. If o is a short text (such as names and titles), it is encoded using character-based stacked, bidirectional GRUs and the final output of the top layer will be taken as the representation of o. On the contrary, if o is a long text such as entity descriptions, CNN over word embeddings will be used to get the embeddings for o. The object o can also be an image, and in such a case, the last hidden layer of VGG pretrained network on ImageNet [74], followed by compact bilinear pooling, is used to obtain the embedding of o. Given the vector representations of the entities, relations and attributes, the same scoring function from DistMult is used to determine the correctness probability of triples.

The binary cross-entropy loss, as defined below, is used to train the model:

$$\sum_{(s,r)} \sum_{o} t_o^{s,r} \log(p_o^{s,r}) + (1 - t_o^{s,r}) \log(1 - p_o^{s,r}), \tag{48}$$

where for a given subject relation pair (s, r), binary label vector $t^{s,r}$ over all entities is used to indicate whether $< s, r, o >$ is observed during training. $p_o^{s,r}$ denotes the model's probability of truth for any triple $< s, r, o >$ computed using a sigmoid function.

Moreover, using these learned embeddings and different neural decoders, a novel multimodal imputation model is introduced to generate missing multimodal values, such as numerical data, categorical data, text, and images, from information in the knowledge base.

Table 6

Complexity of the models with multimodal literals in terms of the number of parameters. Θ is the number of parameters in the base model, H is the entity embedding size, N_d is the number of data relations, N_{char} is the number of characters, and N_i is the number of images, Θ_{CNN} is the number of parameters in the CNN model used in [75], Θ_{ARAE} is the number of parameters in ARAE [76] where instead of using the random noise vector z, the generator is conditioned on the entity embeddings, Θ_{GAN} denotes the sum of the number of parameters in BE-GAN [77] and in pix2pix-GAN [78].

Model	#Parameter
LiteralE with blocking	$\Theta + (N_d H + H)H$
EAKGAE	$\Theta + (N_d + N_{char})H$
MKBE	$\Theta + (2(N_d + 3(N_{char} + H)) + N_i)H$ $+\Theta_{CNN} + \Theta_{ARAE} + \Theta_{GAN}$

In order to predict the missing numerical and categorical data such as dates, gender, and occupation, a simple feed-forward network on the entity embedding is used. For text, the adversarially regularized autoencoder (ARAE) has been used to train generators that decodes text from continuous codes, having the generator conditioned on the entity embeddings instead of random noise vector. Similarly, the combination of BE-GAN structure with pix2pix-GAN model is used to generate images, conditioning the generator on the entity embeddings.

Summary Despite the attempt made in incorporating text literals, numeric literals, and images into the KG embedding, the model (MKBE) fails to capture the semantics of the data types/units of (numeric) literal values. Besides, similar to IKRL, it takes an image I as an instance of a certain entity e only if, I is initially associated with e in the dataset considered (refer to Section 4.3 for more details).

5. Applications

This section discusses different applications of KG embeddings on which the previously described methods have been trained and/or evaluated.

Link prediction. In general terms, link prediction can be defined as a task of identifying missing information in complex networks [79, 80]. Specifically in the case of KGs, link prediction models aim at predicting new relations between entities leveraging the existing links for training. Along with predicting relations between the entities link prediction also focuses on the task of predicting either the head or the tail entity with respect to a relation. Then it decides if a new triple, which is not observed in the KG, is valid or not. Formally, let G be a KG with a set of entities $E = \{e_1, \ldots, e_n\}$ and a set of object relations $R = \{r_1, \ldots, r_m\}$, then link prediction can be defined by a mapping function $\psi : E \times E \times R \to R$ which assigns a score to every possible triple $(e_i, e_j, r_k) \in E \times E \times R$. A high score indicates that the triple is most likely to be true.

Link prediction is one of the most common tasks used for evaluating the performance of KG embeddings. Head prediction, tail prediction, and relation prediction are different kinds of sub-tasks related to link prediction. Head prediction aims at identifying a missing head entity where the relation and tail entity are given, and analogously for tail prediction and relation prediction. Most of the models discussed in Section 4 have been evaluated on some or all of these prediction tasks. Head and tail prediction are used to evaluate the models LiteralE [33], TransEA [19], KBLRN [51], KDCoE [24], EAKGAE [25], IKRL [20], MKBE [34], MTKGRL [60], Jointly(Desp) [21], Jointly [22], and SSP [23]. On the other hand, DKRL [13] has been evaluated on all kinds of link prediction tasks: head, tail, and relation predictions. In Extended RESCAL [37], two kinds of link prediction experiments have been conducted on the Yago 2 [81], i.e., i) tail prediction by fixing the relation type to `rdf:type`, and ii) general link prediction experiments for all relation types. Unfortunately, it is not possible to compare the obtained evaluation results of all these models because the experiments have been carried out on different datasets and also different link prediction procedures have been followed. Taking this into consideration, in this survey, experiments have been conducted on head and tail prediction tasks for these models (see Section 6).

Triple Classification. The goal of the triple classification task is the same as that of link prediction. A potential triple $< h, r, t >$ is classified as 0 (false) or 1 (true), i.e., a binary classification task. The embedding models MTKGNN [13], IKRL [20], MTKGRL [60], Jointly(Desp) [21], and Jointly [22] have been evaluated on this task. However, since they do not use a common evaluation dataset, it is not possible to compare the reported results directly.

Entity Classification. Given a KG G, with a set of entities E and types T and with an entity $e \in E$ and type $t \in T$, the task of entity classification is to determine if a potential entity type pair (e, t) which is not observed in G $((e, t) \notin G)$ is a missing fact or not. This task is an entity type prediction using a multi-label classification algorithm considering the entity types in G as given classes. In DKRL [13], Extended RESCAL [37], and SSP [23], Entity classification has been used for model evaluation.

Entity Alignment. Given two KGs G_1 and G_2, the goal of the entity alignment task is to identify those entity pairs (e_1, e_2) where e_1 is an entity in G_1 and e_2 is an entity in G_2 which denote the same real world entities, and hence the integration of G_1 and G_2 can be possible through these unified entities, i.e., entity pairs. Different embedding-based models have been proposed recently for the entity alignment task. Among the models that are included in this survey, EAKGAE [25] and KDCoE [24] have been proposed for the entity alignment task. Specifically, KDCoE [24] uses a cross-lingual entity alignment task which determines similar entities in different languages. Despite the fact that both these models use the same task for evaluation, the entity alignment task, their experimental results cannot be compared since they are based on different datasets.

Other Applications. Attribute-value prediction, nearest-neighbor analysis, data linking, document classification, and relational fact extraction are other application scenarios used for the evaluation of the models under discussion. Attribute-value prediction is the process of predicting the values of (discrete or non-discrete) attributes in a KG. For example, a missing value of a person's weight can be identified using the attribute value prediction task which is commonly seen as a KG completion task. In MTKGNN [35], attribute-value prediction is applied using an attribute-specific Linear Regression classifier for evaluation. The same task has been employed in MKBE [34] for model evaluation by imputing different multi-modal attribute values.

Nearest Neighbor Analysis is a task of detecting the nearest neighbors of some given entities in the latent space learned by an embedding model. This task has been performed in LiteralE [33] to compare DistMult+LiteralE with the base model DistMult. On the other hand, data linking and document classification tasks have been used in LiteralE with blocking [38] and KGlove with literals [36] respectively (refer to [38] and [36] for more details). Relational fact extraction is a task of extracting facts/triples from plain text and has been used as a model evaluation task in Jointly(Desp) [21]. Table 16 summarizes all the applications on which the KG embedding models with literals have been evaluated.

6. Experiments on Link Prediction

This section provides an empirical evaluation of the methods discussed in the previous section under a unified environmental settings and discusses the results based on the performance of the approaches applied to the task of link prediction. In this work, link prediction is chosen because most of the KG embedding models with literals are trained and evaluated on it. One of the major issues encountered while conducting these experiments is that the source code of some of these models is not openly available and is not easily reproducible. Such methods were excluded from the experimentation. In the subsequent sections, the datasets and the experiments with text, numeric, images and multi-modal literals are presented.

Based on the results of the experiments, a clear comparison is presented between the models with literals on link prediction. In addition, these models are also compared with the standard KG embedding approaches that they extend. Note that these models may inherit the problems that already exist in their corresponding base models - the standard KG embedding models that they extend). For instance, the models that extend DistMult such as DistMult-LiteralE$_g$ inherit the problem of DistMult, which is not being capable to properly capture anti-symmetric relations due to the way its scoring function is defined.

6.1. Datasets

The performance of the aforementioned models was measured using two of the most commonly used datasets for link prediction, i.e., FB15K [11] and FB15K-237 [82] are considered. FB15K is a subset of Freebase [83] which mostly contains triples describing the facts about movies, actors, awards, sports and sport teams. It contains a randomly split training, validation, and test sets. The issue with this dataset is that the test set contains a large number of triples which are obtained by simply inverting triples in the training set. This enables a simple embedding model which is symmetric with respect to the head and tail entity to obtain an excellent performance. In order to avoid this, the dataset FB15K-237 has been created by removing the inverse relations from FB15K. The statistics of these datasets is given in Table 7.

Table 7
The number of entities, object relations, data relations, relational triples, train sets, valid sets, and test sets of the FB15K and the FB15K-237 datasets.

	Datasets	
	FB15K	**FB15K-237**
Entities	14951	14541
Object Relations	1345	237
Relational triples	592213	310116
Train sets	483142	272115
Valid sets	50000	17535
Test sets	59071	20466

6.2. *Experiments with Text Literals*

As discussed in Section 4.1, the embedding models Extended RESCAL, DKRL, KDCoE, and KGloVe with literals utilize text literals. However, all of these models except DKRL are not considered for experimentation due to the following issues:

- The implementation of the model KGloVe with literals is not publicly available and it is not easily reproducible.
- KDCoE is designed specifically for cross-lingual entity alignment task which makes it difficult to apply it for link prediction.
- In case of Extended RESCAL, practically this method is computationally expensive and thus not considered as a feasible embedding model to incorporate literals.
  Moreover, none of the models with literals which are discussed in this paper consider Extended RESCAL in their experiments.

In order to conduct experiments with text literals, 15239 English entity descriptions of the entities common in both datasets FB15K and FB15K-237 shown in Table 7 are taken from LiteralE [33]. The focus lies on the common entity descriptions, i.e., for those entities existing in FB15K but not in FB15K-237 no description is used, because there has already been experiments done using the whole entity descriptions for FB15K dataset in the original paper. This way it would be possible to analyse the effect of the size of the dataset (the entity descriptions) on the performance of the embedding models. The average number of words (tokens) in the descriptions is 143 whereas the maximum and minimum are 804 and 2.

Dataset Pre-processing: For pre-processing of the text (the entity descriptions), spacy.io[1] has been used. This includes tokenization, named entity recognition and conversion of numbers to text, i.e., 16 has been converted to ‘sixteen’. After the pre-processing step, all the entities along with the corresponding triples having no or short description of less than 3 words are removed. Also, the triples containing these entities are removed as mentioned by the authors in the paper. Moreover, only one description is chosen randomly for the entities with multiple text descriptions.

Experimental Setup: The hyperparameters used for DKRL are as follows: learning rate 0.001, embedding size 100, loss margin 1, batch size 100 and epochs 1000. For TransE, learning rate 0.01, embedding size 50, margin 1, and epochs 1000 are used. The experiments with DKRL were performed on Ubuntu 16.04.5 LTS system with 503GiB RAM and 2.60GHz speed. On the other hand,the experiments with TransE are performed with TITAN X (Pascal) GPU.

Runtime: Note that the codes used in the experiments for both models DKRL and TransE are not implemented in the same environment, i.e., for DKRL, the code that is released by the authors of the paper is used and for TransE the code provided by the authors of TransEA [19] is used. Therefore, it is not fair to compare the runtime results of these two models directly. However, in order to provide some insights into the computational complexity of the models, their runtime results on the FB15K dataset are given as follows. DKRL takes 142 seconds to train 1 epoch using 16 threads whereas the runtime of TransE for a single iteration with batch size 4831 is 3.271 millisecond. This is computed by taking the average of 1000 iterations.

Evaluation Procedure and Results: The performance of the model is evaluated based on the link prediction task. For each triple in the test set, a set of corrupted triples is generated with respect to the head or the tail entity. A triple is said to be corrupted with respect to its head entity if that head entity is replaced with any other entity from the KG, and analogously for a triple corrupted with respect to its tail entity. The set of corrupted triples can also contain true triples that exist in the training, validation or test set. Since it is not a mistake to give these true triples better score than the actual test triple, they are removed from the set of corrupted triples and this is referred to as filtered set-

[1]https://spacy.io/usage

ting [11]. In order to check if the model assigns a better score to the actual test triple than the corrupted triples which are obtained by corrupting the test triple, it is evaluated using the metrics MR (Mean Rank), MRR (Mean Reciprocal Rank), and Hits@N. First, for every test triple, all of its corrupted triples with respect to head are ranked based on their scores which are computed by the model. Then, the rank of the actual (true) test triples are taken in order to compute the metrics MR, MRR, and Hits@N. MR is the mean of the ranks of all test triples - the lower the better and MRR is their average inverse rank - the higher the better. Hits@N is the percentage of ranks lower than or equal to N - the higher the better. The same procedure is repeated to evaluate the model against the corrupted triples with respect to tail.

The results of link prediction on FB15K and FB15K-237 datasets are shown in Table 8 for the models TransE, DKRL with Bernoulli distribution (DKRL$_{Bern}$), and DKRL with Uniform distribution (DKRL$_{unif}$). The Bernoulli distribution for sampling as defined in [14] is a probability distribution, $\frac{tph}{tph+hpt}$, where tph is the average number of tail entities per head entity and hpt is the average number of head entities per tail entity. Given a golden triple $< h, r, t >$, with the aforementioned probability, the triple is corrupted by replacing the head, and with probability $\frac{hpt}{tph+hpt}$, the triple is corrupted by replacing the tail. The results are reported separately for the head entity and tail entity along with the overall results obtained by taking the mean of the head and tail predictions. The best scores are the ones which are highlighted in bold text. The result of the TransE model is presented in order to allow a clear comparison with DKRL because, as shown in Table 2, DKRL extends TransE. This comparison would help to further analyse the advantages of using text literals for KG embedding.

Note that in the original paper, the result of DKRL on FB15K is slightly better than TransE. However, in our experiments, as the results in Table 8 indicate, on the FB15K dataset TransE achieves better result than both versions of DKRL on all metrics except MRR and MR. The reason for this is that, as mentioned above, the set of entity descriptions used in our experiments are common for both datasets FB15K and FB15K-237, i.e., there is less entity descriptions in our experiment than there is in the original paper for FB15K. This indicates that the size of the dataset (the entity description has impact on the performance of the model). On the other hand, on the dataset FB15K-237 TransE is outperformed by DKRL$_{Unif}$ with respect to MR and by DKRL$_{Bern}$ with respect to the rest of the metrics.

Furthermore, the result shows that DKRL model with Bernoulli distribution (DKRL$_{Bern}$) has better performance than the model with Uniform distribution (DKRL$_{unif}$) for both the datasets. DKRL$_{Bern}$ works best for the prediction of head, relation, and tail with respect to MRR, Hits@1, and Hits@3 whereas the $DKRL_{Unif}$ method works better according to MR for both the datasets. DKRL$_{Bern}$ works slightly better than $DKRL_{Unif}$ for FB15K-237 dataset. It is to be noted that DKRL has better improvement over TransE on FB15K-237 as compared to FB15K dataset because the former one does not contain symmetric relations, i.e., incorporating textual data to a clean dataset, such as FB15K-237, allows capturing more semantics.

6.3. Experiment with Numeric Literals

MT-KGNN, KBLRN, LiteralE, and TransEA are the KG embedding models which make use of numeric literals (see Section 4.2). KBLN, the submodel of KBLRN, which excludes the relational information provided by graph feature methods is used in the experiment instead of the main model KBLRN. This is the case because KBLN is directly comparable with the other three models (i.e., MT-KGNN, LiteralE, and TransEA) whereas KBLRN is not. The code[2] for the TransEA model is the original implementation from TransEA [19] where as the source codes[3] for the models MT-KGNN, KBLN, and LiteralE are taken from the implementation in LiteralE [33]. As described in Section. 4.2, the structure-based embedding component of MT-KGNN is based on a neural network and it is referred to as RelNet. However, in the version implemented in LiteralE [33], they have replaced RelNet with DistMult as a baseline in order to have a directly comparable MTKGNN-like method to their proposed approach. Thus, in this survey, the MT-KGNN-like model has been used instead of the original MT-KGNN model.

Moreover, the model LiteralE has different varieties depending on the baseline model and the transformation function used. As discussed in Section 4, in LiteralE there are two transformation functions: g (GRU based function) and lin (a simple linear function), and there are three baseline models - DistMult, ConvE and ComplEx. Thus, in this experi-

[2]https://github.com/kk0spence/TransEA
[3]https://github.com/SmartDataAnalytics/LiteralE

Table 8

Experiment results using DKRL model on FB15K and FB15K-237 datasets.

		MR	MRR	Hits@1	Hits@3	Hits@10
FB15K						
TransE	Head	142	0.219	**0.241**	**0.447**	**0.622**
	Tail	109	0.249	**0.339**	**0.514**	**0.690**
	All	125	0.234	**0.290**	**0.480**	**0.656**
DKRL$_{Bern}$	Head	162	**0.289**	0.179	0.336	0.502
	Tail	122	**0.356**	0.24	0.408	0.577
	All	142	**0.322**	0.209	0.372	0.539
DKRL$_{Unif}$	Head	**96**	**0.289**	0.172	0.335	0.52
	Tail	**75**	0.333	0.211	0.383	0.576
	All	**85**	0.311	0.191	0.359	0.548
FB15K-237						
		MR	MRR	Hits@1	Hits@3	Hits@10
TransE	Head	468	0.094	0.081	0.163	0.287
	Tail	255	0.190	0.233	0.373	0.517
	All	361	0.142	0.157	0.268	0.402
DKRL$_{Bern}$	Head	145	**0.294**	**0.184**	**0.337**	**0.507**
	Tail	98	**0.359**	**0.244**	**0.410**	**0.585**
	All	122	**0.327**	**0.214**	**0.374**	**0.546**
DKRL$_{Unif}$	Head	**104**	0.275	0.166	0.312	0.494
	Tail	**77**	0.322	0.209	0.363	0.552
	All	**91**	0.298	0.187	0.337	0.523

ment, six varieties of the LiteralE model are considered: DistMult-Literale$_g$, ComplEx-Literale$_g$, ConvE-Literale$_{lin}$, DistMult-Literale$_{lin}$, ComplEx-Literale$_{lin}$, and ConvE-Literale$_{lin}$. The datasets, the experimental setup, and the evaluation results are discussed in the subsequent sections.

Attributive Triples: In order to conduct the experiments with numeric literals, both the datasets FB15K and FB15K-237 given in Table 7 are extended with a set of 23521 attributive triples, containing only numeric literals, with 118 data relations. These triples are created based on the attributive triples from TransEA [19]. In TransEA, the authors have provided a set of attributive triples where the object values are numeric. However, it is not possible to directly use this data as the literal values are normalized in the interval [0-1] as required by the model but the other models in this experiment, like LiteralE, use the original unnormalized literal values instead. Therefore, it was necessary to query Freebase to replace the normalized object literal value for each (subject, data relation) pair from the TransEA attributive triples data. Moreover, only those data relations which occur in at least 5 triples are taken into consideration.

Experimental Setup: For both datasets, the hyperparameters for TransEA are: epoch 3000, dimension 100, batches 100, margin 2, and learning rate 0.3 and for TransE they are described in Section 6.2. For the other models, same as in LiteralE, the hyperparameters used for both datasets are: learning rate 0.001, batch size 128, embedding size 100 (for DistMult, ComplEx and their extensions with literals) and 200 (for KBLN, and MTKGNN, ConvE, and ConvE's extensions), embedding dropout probability 0.2, label smoothing 0.1, and epochs 1000 for ConvE and 100 for the rest. TITAN X (Pascal) GPU has been used for the models LiteralE, KBLN, and MTKGNN.

Runtime: As in the experiments with text literals, not all the models in the experiments with numeric literals are implemented in the same environment, i.e., for TransEA the code that is released by the authors of the paper is used and for the other models the code that is provided by the authors of LiteralE are used. Therefore, direct comparison of the runtime of TransEA and the other models would not be possible. However, the runtime of each of the models is computed on FB15K dataset so as to give insights into the models computational complexity. The running time of TransEA is 3.271 ms per a single iteration with batch size of 4831.

Table 9
Runtime of models considered in the experiments with numeric literals. The resutls are per single iteration and reported in milliseconds.

Models	Time(ms)
DistMult-LiteralE$_{g_{lin}}$	31.575
DistMult-LiteralE$_g$	37.138
ComplEx-LiteralE$_{g_{lin}}$	39.269
ComplEx-LiteralE$_g$	52.346
ConvE-LiteralE$_{g_{lin}}$	43.386
ConvE-LiteralE$_g$	50.439
KBLN	86.825
DistMult	29.679
ComplEx	33.526
ConvE	40.970

For the other models their runtime for a single iteration of batch size 128 is shown in Table 9. Note that the runtime results reported here are the average over runtime values of 1000 iterations.

Evaluation Procedure and Results: The same evaluation metrics which are discussed in Section 6.2 has been used to evaluate the performance of the models with numeric literals on the link prediction task. As shown in Table 10, according to the overall result, the model KBLN has considerably better performance than the other models in all metrics except MR. The results from the ComplEx-LiteralE$_g$ model show that it is capable to produce a highly competitive performance having the second best results with respect to the same metrics. This is the case due to the fact that this model is able to handle the inverse relations in FB15K by applying the complex conjugate of an entity embedding when the entity is used as a tail and its normal embedding when it is the head. Moreover, the model KBLN also achieves better result when compared to the standard models without literals presented in Table 11 with respect to all metrics except MR.

Another possible analysis to make is to compare the results of the standard models presented in Table 11 with the results of their extensions shown in the 'both head and tail Prediction' part of Table 10. For instance, ComplEx-LiteralE$_g$ achieves better performance than its base model ComplEx according to all metrics which indicates that using numeric literals with ComplEx by applying the approach in LiteralE is beneficial. However, this is not the case with DistMult and ConvE. One reason for this can be the fact that the number of attributive triples used in our experiment is not as big as in the original paper of LiteralE, i.e, increasing the number of numeric literals may improve the result as already seen in the original paper of literalE.

On the other hand, referring to the overall result on FB15K-237 dataset as shown in Table 12, the model DistMult-LiteralE$_g$ outperforms the other models according to all metrics. This entails that applying LiteralE to DistMult on FB15K-237 provides better performance than applying it to other baseline models. Note that the reason for DistMult-LiteralE$_g$ model to achieve the best result on FB15K-237 dataset, as comapred to FB15K, may be due to the fact that this dataset does not have any symmetric relation, i.e., DistMult already has difficulties in modeling asymmetric relations on FB15k and adding literals might introduce noise but in case of FB15K-237, incorporating literals improves DistMult because symmetric relations are removed. Regarding the two transformation functions g and g_{lin}, the function g leads to better results than g_{lin} according to the results on both dataset.

6.4. Experiment with Images

Note that it is not possible to compare the whole of MKBE [34] with any other model as it is the only embedding model which utilizes the three types of literals together: text, numeric, and images. Therefore, its sub model S+I which uses only images has been compared with the embedding model IKRL [20]. Since this comparison has already been done by the authors of MKBE [34], the result shown in Table 14 is directly taken from their paper. They have compared the models DistMult+S+I, ConvE+S+I, and IKRL where S stands for structure and I for Image. Both DistMult+S+I and ConvE+S+I are sub models of MKBE which use only relational triples and Images. The result indicates that ConvE+S+I outperforms the other two models in all metrics on the YAGO-10 dataset (refer to MKBE [34] for more details).

6.5. Experiment with Multi-modal Literals

As discussed in Section 4, the existing multi-modal embeddings are categorized into two types: i) models with text literal, numeric literal and image literals and ii) models with text and numeric literals. However, since MKBE is the only model in the first category only its submodel $S + I$ could be compared with IKRL (see Section 6.4). Regarding the models with text and numeric literals, i.e., LiteralE with blocking and EAKGAE, they are not included in the experiment as well. The issue with EAKGAE is the same as that of KD-

Table 10

Link prediction results on FB15K dataset using filtered setting.

Head Prediction					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{g_{lin}}$	121	0.495	0.383	0.559	0.697
ComplEx-LiteralE$_{g_{lin}}$	71	0.76	0.697	0.801	0.876
ConvE-LiteralE$_{g_{lin}}$	52	0.612	0.51	0.678	0.795
DistMult-LiteralE$_g$	72	0.581	0.479	0.642	0.762
ComplEx-LiteralE$_g$	63	0.768	**0.707**	0.809	0.878
ConvE-LiteralE$_g$	**49**	0.72	0.65	0.762	0.849
KBLN	77	**0.775**	0.705	**0.827**	**0.892**
MTKGNN	73	0.702	0.617	0.758	0.855
TransEA	103	0.285	0.367	0.609	0.728
Tail Prediction					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{g_{lin}}$	145	0.447	0.337	0.507	0.645
ComplEx-LiteralE$_{g_{lin}}$	101	0.704	0.64	0.743	0.821
ConvE-LiteralE$_{g_{lin}}$	**74**	0.567	0.465	0.63	0.746
DistMult-LiteralE$_g$	94	0.528	0.425	0.589	0.712
ComplEx-LiteralE$_g$	93	0.711	0.65	0.746	0.821
ConvE-LiteralE$_g$	79	0.657	0.586	0.698	0.783
KBLN	90	**0.727**	**0.656**	**0.776**	**0.848**
MTKGNN	91	0.65	0.562	0.708	0.806
TransEA	75	0.314	0.417	0.671	0.805
Both Head and Tail Prediction					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{g_{lin}}$	133	0.471	0.36	0.533	0.671
ComplEx-LiteralE$_{g_{lin}}$	86	0.732	0.668	0.772	0.848
ConvE-LiteralE$_{g_{lin}}$	**63**	0.589	0.487	0.654	0.77
DistMult-LiteralE$_g$	83	0.554	0.452	0.615	0.737
ComplEx-LiteralE$_g$	78	0.739	0.678	0.777	0.849
ConvE-LiteralE$_g$	64	0.688	0.618	0.73	0.816
KBLN	83	**0.751**	**0.68**	**0.801**	**0.87**
MTKGNN	82	0.676	0.589	0.733	0.83
TransEA	74	0.299	0.392	0.64	0.766

Table 11

Link prediction results with models without literals on FB15K using filtered setting.

FB15K					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult	119	0.67	0.589	0.723	0.817
ComplEx	127	**0.692**	**0.614**	0.742	0.833
ConvE	**50**	0.689	0.593	**0.757**	**0.852**
TransE	125	0.234	0.290	0.480	0.656

CoE, i.e., it is trained on entity alignment task where as the reason for not having LiteralE with blocking is that its code is not publicly available. On the contrary, LiteralE (a model with numerics) has also been adopted to incorporate text literals in the experiments conducted by the authors. Similarly, in our experiment, the LiteralE approach has been tried out with the combination of text and numeric literals, i.e., the model DistMult-LiteralE$_g$-text in Table 15. Then, the result has been compared with LiteralE with just numeric literals (DistMult-LiteralE$_g$) and DKRL (a model using only text literals) so as to investigate the benefits of utilizing information represented by different types of literals.

DistMult-LiteralE$_g$-text is a model which applies the LiteralE approach to DistMult by using both numeric

Table 12

Link prediction results on FB15K-237 dataset using filtered setting.

Head Prediction					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{g_{lin}}$	245	0.377	0.279	0.422	0.568
ComplEx-LiteralE$_{g_{lin}}$	371	0.36	0.271	0.4	0.538
ConvE-LiteralE$_{g_{lin}}$	**208**	0.388	0.296	0.427	0.572
DistMult-LiteralE$_g$	209	**0.413**	**0.320**	**0.456**	**0.591**
ComplEx-LiteralE$_g$	315	0.366	0.277	0.404	0.543
ConvE-LiteralE$_g$	236	0.317	0.229	0.345	0.501
KBLN	381	0.386	0.295	0.426	0.564
MTKGNN	437	0.383	0.295	0.423	0.559
TransEA	389	0.111	0.094	0.197	0.342
Tail Prediction					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{g_{lin}}$	426	0.195	0.119	0.214	0.349
ComplEx-LiteralE$_{g_{lin}}$	575	0.17	0.104	0.185	0.306
ConvE-LiteralE$_{g_{lin}}$	362	0.187	0.112	0.204	0.338
DistMult-LiteralE$_g$	359	**0.215**	0.137	0.234	0.371
ComplEx-LiteralE$_g$	493	0.175	0.106	0.19	0.312
ConvE-LiteralE$_g$	459	0.131	0.07	0.137	0.256
KBLN	501	0.207	0.128	0.23	0.362
MTKGNN	580	0.191	0.12	0.208	0.338
TransEA	**203**	0.206	**0.25**	**0.409**	0.57
Both Head and Tail Prediction					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{g_{lin}}$	335	0.286	0.199	0.318	0.458
ComplEx-LiteralE$_{g_{lin}}$	473	0.265	0.187	0.292	0.422
ConvE-LiteralE$_{g_{lin}}$	285	0.287	0.204	0.315	0.455
DistMult-LiteralE$_g$	**284**	**0.314**	**0.228**	**0.345**	**0.481**
ComplEx-LiteralE$_g$	404	0.27	0.191	0.297	0.427
ConvE-LiteralE$_g$	347	0.224	0.149	0.241	0.378
KBLN	441	0.296	0.211	0.328	0.463
MTKGNN	508	0.287	0.207	0.315	0.448
TransEA	296	0.158	0.172	0.303	0.456

Table 13

Link prediction results with models without literals on FB15K-237 dataset using filtered setting.

Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult	630	0.280	0.201	0.309	0.438
ComplEx	623	0.288	0.207	0.318	0.448
ConvE	**273**	**0.310**	**0.222**	**0.343**	**0.484**
TransE	361	0.142	0.157	0.268	0.402

Table 14

MRR results on link prediction task on YAGO-10 taken from MKBE [34].

YAGO-10				
Models	MRR	Hits@1	Hits@3	Hits@10
DistMult+S+I	0.342	0.235	0.352	0.618
ConvE+S+I	**0.566**	**0.471**	**0.597**	**0.72**
IKRL	0.509	0.423	0.556	0.663

and text literals. Note that DistMult is chosen here as a baseline due to the reason that the best result in the experiments with numerics on the FB15K-237 dataset is achieved using this model as discussed in Section 6.3. The datasets listed in Table 7 are also used for this experiment along with additional text attributive triples which are descriptions of entities. DistMult-LiteralE$_g$-text has also been compared with its numeric only equivalent DistMult-LiteralE$_g$ and DKRL$_{Bern}$.

The experimental results obtained on the datasets FB15K and FB15K-237 are shown in Table 15. As the result indicates, combining text and numeric literals on FB15K dataset with DistMult-LiteralE$_g$-text approach does not produce better results as compared to the other models DistMult-LiteralE$_g$ and DKRL$_{Bern}$. As mentioned before, this dataset contains a set of inverse relations which may lead to having a triple whose inverse has a different label. Given the fact that DistMult fails to model such asymmetric relations, incorporating more literals with DistMult may introduce much noise than improving the performance. On the other hand, for FB15K-237 dataset, according to all the measures except MR, DistMult-LiteralE$_g$-text model works better for the head entity prediction compared to the other two models. For tail entity prediction, DKRL$_{Bern}$ works better with respect to all measures for the same dataset.

7. Discussion and Conclusion

Given the recent massive attention towards the use of KGs in various applications, different KG embedding techniques have been proposed to enable efficient use of KGs. In some of these techniques, an attempt has been made to utilize the information represented in literals present in KGs for a better quality embedding of the elements of the KGs, i.e., entities and relations. In this paper, a comprehensive survey of those KG embedding models with literals has been presented. The survey provides a detailed analysis and categorization of these models based on the proposed methodology along with their application scenarios and limitations. Moreover, various experiments on link prediction task on these models have been conducted so as to compare the models' performances. [84] is a very recent work, not included in this survey, which re-evaluates KG completion models.

In this paper, two major research questions are formulated and presented in Section 3. The answers to these questions are given as follows:

– **RQ1** – *How can structured (triples with object relations) and unstructured information (attributive triples) in the KGs be combined into the representation learning?*
  In order to use both data sources, i.e., triples with object relations and triples with data relations (attributive triples) together for representation learning, in broader terms, the following two techniques are considered in the models discussed in this paper:
  * Handling literals separately: defining one task per data source like in TransEA or using a separate encoder for literals as in DKRL. The two tasks are trained simultaneously to make sure that for every entity the information available in both data sources are used to learn its embedding. The embeddings of the entities learned based on each data source can be unified in the vector space or not depending on how the model works. For instance, Jointly(Disp) learns unified representation for entities where as DKRL generates two representations per entity and do not force them to be unified.
  * Incorporating literals directly into entity embeddings: as in LiteralE, one way is to extend a certain latent feature method by directly enriching the embeddings with information from literals via a learnable parameter and use the same scoring function from the latent feature method.

– **RQ2** – *How can the heterogeneity of the types of literals present in the KGs be captured and combined into representation learning?*
  The following are some possible ways to combine different kind of literals, i.e., text, numeric, etc. together for representation learning.
  * Encoding each type of literal separately: in order to capture the semantics of literals, different encoders can be used for different types of literals, for example, CNN for textual descriptions. Then, as shown in MKBE, each attributive triple can be treated same as structured triples and use a single scoring function for training.
  * Incorporating information present in every kind of literal directly into the entity embedding: as in LiteralE, for a given entity, first the literals associated with it are encoded as vectors - using one vector per type of literal.

Table 15

Link prediction results on FB15K and FB15K-237 datasets using filtered set.

	FB15K					
	Models	MR	MRR	Hits@1	Hits@3	Hits@10
Head	DistMult-LiteralE$_g$	**72**	**0.581**	**0.479**	**0.642**	**0.762**
	DKRL$_{Bern}$	162	0.289	0.179	0.336	0.502
	DistMult-LiteralE$_g$-text	93	0.516	0.405	0.582	0.711
Tail	DistMult-LiteralE$_g$	**94**	**0.528**	**0.425**	**0.589**	**0.712**
	DKRL$_{Bern}$	122	0.356	0.24	0.408	0.577
	DistMult-LiteralE$_g$-text	119	0.463	0.351	0.532	0.66
All	DistMult-LiteralE$_g$	**83**	**0.554**	**0.452**	**0.615**	**0.737**
	DKRL$_{Bern}$	142	0.322	0.209	0.372	0.539
	DistMult-LiteralE$_g$-text	106	0.489	0.378	0.557	0.685
	FB15K-237					
	Models	MR	MRR	Hits@1	Hits@3	Hits@10
Head	DistMult-LiteralE$_g$	209	0.413	0.320	0.456	0.591
	DKRL$_{Bern}$	**145**	0.294	0.184	0.337	0.507
	DistMult-LiteralE$_g$-text	207	**0.416**	**0.323**	**0.462**	**0.594**
Tail	DistMult-LiteralE$_g$	359	0.215	0.137	0.234	0.371
	DKRL$_{Bern}$	**98**	**0.359**	**0.244**	**0.410**	**0.585**
	DistMult-LiteralE$_g$-text	354	0.223	0.142	0.246	0.385
All	DistMult-LiteralE$_g$	284	0.314	0.228	0.345	0.481
	DKRL$_{Bern}$	**122**	**0.327**	0.214	**0.374**	**0.546**
	DistMult-LiteralE$_g$-text	280	0.319	**0.232**	0.354	0.489

Then, a mapping function is used to map all these vectors (including the structure-based vector representation of the entity) into a single vector.

As mentioned in Section 4 or seen from the result of the experiments in Section 6, these embedding models have different drawbacks such as:

- The effect that data types/units have on the semantics of literals has not been considered by any of the models.
- Most of the embedding models which make use of numerical literals, such as LiteralE, TransEA, MT-KGNN, and KBLN consider only the year part of date typed literals and ignore the month and day values. This hinders the ability to properly capture the information represented in such kind of literals. For example, given the following three date typed literal values:

  ```
  "1999-10-29"^^xsd:date,
  "1999-04-14"^^xsd:date,
  "1999-10-30"^^xsd:date,
  ```

  a model which utilizes only the year part of these values considers all of them to be exactly the same despite the fact that the first date value is more close to the third value than it is to the second value.
- Most of the models also do not have a proper mechanism to handle multi-valued literals.
- The performance of most of the models is dependent on the dataset used for training and testing which shows that these models are not robust. For example, referring to Table 15, the results of the model DistMult-LiteralE$_g$-text indicate that combining text and numeric literals yields better performance on FB15K-237 but not on FB15K due to the technique used in the model and the nature of the datasets (see Section 6.5).
- Not all the models are effective in combining different types of literals. For example, the performance of DistMult-LiteralE$_g$-text (numeric + text literals), which combines text and numeric literals, on the dataset FB15K is lower as compared to DistMult-LiteralE$_g$ (only numeric literals).
- Only few approaches have been proposed for multi-modal KG embeddings and none of them take into consideration literals with URIs connected to items such as audio, video, or pdf files.

The above described shortcomings of the existing models clearly indicate that thorough investigation is needed on how to address different types of literals that obtain different inherent semantics. For instance, a possible perspective that arises by this detailed analysis is that there is a need to properly handle the data typed literals such as the values of the data relation *weight* given in *kilogram* and *pound*. One possible solution to target this issue could be to normalize these literal values to a standardized measures and to treat different measures like weights and lengths separately in the representation learning process.

One cannot expect that by leaving out available information present in the original KG, its latent representation as being only an approximation of the original KG, will perform equally well on tasks that depend on its semantic information content. Overall, the inclusion of datatyped literals with a proper representation of their semantics into the representation learning process will increase the model's semantic content and might thereby lead to quality improvement.

References

[1] A. Bordes, S. Chopra and J. Weston, Question Answering with Subgraph Embeddings, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 615–620. doi:10.3115/v1/D14-1067. https://www.aclweb.org/anthology/D14-1067.

[2] F. Zhang, N.J. Yuan, D. Lian, X. Xie and W.-Y. Ma, Collaborative Knowledge Base Embedding for Recommender Systems, in: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, ACM, New York, NY, USA, 2016, pp. 353–362. ISBN 978-1-4503-4232-2. doi:10.1145/2939672.2939673.

[3] J. Weston, A. Bordes, O. Yakhnenko and N. Usunier, Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction, in: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.*, Association for Computational Linguistics, Seattle, Washington, USA, 2013, pp. 1366–1371. https://www.aclweb.org/anthology/D13-1136.

[4] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer et al., DBpedia–A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia, *Semantic Web* **6** (2015), 167–195.

[5] D. Vrandečić and M. Krötzsch, Wikidata: A Free Collaborative Knowledge Base, *Communications of the ACM* **57** (2014), 78–85. http://cacm.acm.org/magazines/2014/10/178785-wikidata/fulltext.

[6] F. Mahdisoltani, J. Biega and F.M. Suchanek, Yago3: A Knowledge Base from Multilingual Wikipedias, in: *CIDR*, 2013.

[7] A. Bordes, J. Weston, R. Collobert and Y. Bengio, Learning Structured Embeddings of Knowledge Bases, in: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, AAAI Press, 2011, pp. 301–306. http://dl.acm.org/citation.cfm?id=2900423.2900470.

[8] M.R. Garey and D.S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA, 1990. ISBN 0716710455.

[9] Q. Wang, Z. Mao, B. Wang and L. Guo, Knowledge Graph Embedding: A Survey of Approaches and Applications, *TKDE* **29**(12) (2017), 2724–2743.

[10] B. Kotnis and V. Nastase, Analysis of the impact of negative sampling on link prediction in knowledge graphs, *arXiv preprint arXiv:1708.06816* (2017).

[11] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston and O. Yakhnenko, Translating Embeddings for Modeling Multi-Relational Data, in: *NIPS*, 2013.

[12] T. Dettmers, P. Minervini, P. Stenetorp and S. Riedel, Convolutional 2d knowledge graph embeddings, in: *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 1811–1818. https://arxiv.org/abs/1707.01476.

[13] R. Xie, Z. Liu, J. Jia, H. Luan and M. Sun, Representation Learning of Knowledge Graphs with Entity Descriptions, in: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 2016, pp. 2659–2665.

[14] Z. Wang, J. Zhang, J. Feng and Z. Chen, Knowledge Graph Embedding by Translating on Hyperplanes, in: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, 2014, pp. 1112–1119. https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[15] Y. Lin, Z. Liu, M. Sun, Y. Liu and X. Zhu, Learning Entity and Relation Embeddings for Knowledge Graph Completion, in: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, 2015, pp. 2181–2187. https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

[16] G. Ji, S. He, L. Xu, K. Liu and J. Zhao, Knowledge Graph Embedding via Dynamic Mapping Matrix, in: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Association for Computational Linguistics, Beijing, China, 2015, pp. 687–696. doi:10.3115/v1/P15-1067.

[17] G. Ji, K. Liu, S. He and J. Zhao, Knowledge Graph Completion with Adaptive Sparse Transfer Matrix, in: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, AAAI Press, Phoenix, Arizona, 2016, pp. 985–991. https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11982.

[18] Y. Jia, Y. Wang, H. Lin, X. Jin and X. Cheng, Locally Adaptive Translation for Knowledge Graph Embedding, in: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, AAAI Press, Phoenix, Arizona, 2016, pp. 992–998. http://dl.acm.org/citation.cfm?id=3015812.3015960.

[19] Y. Wu and Z. Wang, Knowledge Graph Embedding with Numeric Attributes of Entities, in: *Proceedings of The Third Workshop on Representation Learning for NLP*, Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 132–136. doi:10.18653/v1/W18-3017. https://www.aclweb.org/anthology/W18-3017.

[20] R. Xie, Z. Liu, T.-S. Chua, H.-B. Luan and M. Sun, Image-embodied Knowledge Representation Learning, in: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, Melbourne, Australia, 2017, pp. 3140–3146. doi:10.24963/ijcai.2017/438.

[21] H. Zhong, J. Zhang, Z. Wang, H. Wan and Z. Chen, Aligning knowledge and text embeddings by entity descriptions, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 267–272. https://aclweb.org/anthology/D/D15/D15-1031.

[22] J. Xu, X. Qiu, K. Chen and X. Huang, Knowledge Graph Representation with Jointly Structural and Textual Encoding, in: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 1318–1324. doi:10.24963/ijcai.2017/183.

[23] H. Xiao, M. Huang, L. Meng and X. Zhu, SSP: semantic space projection for knowledge graph embedding with text descriptions, in: *Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, San Francisco, California USA, 2017. https://www.aaai.org/Conferences/AAAI/2017/PreliminaryPapers/14-XiaoH-14306.pdf.

[24] M. Chen, Y. Tian, K.-W. Chang, S. Skiena and C. Zaniolo, Co-training Embeddings of Knowledge Graphs and Entity Descriptions for Cross-lingual Entity Alignment, in: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, IJCAI'18, International Joint Conferences on Artificial Intelligence Organization, Stockholm, Sweden, 2018, pp. 3998–4004. doi:10.24963/ijcai.2018/556.

[25] B.D. Trsedya, J. Qi and R. Zhang*, Entity Alignment between Knowledge Graphs Using Attribute Embeddings, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI'19, 2019. doi:https://doi.org/10.1609/aaai.v33i01.3301297.

[26] M. Nickel, V. Tresp and H.-P. Kriegel, A Three-Way Model for Collective Learning on Multi-Relational Data., in: *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, Omnipress, Bellevue, Washington, USA, 2011, pp. 809–816.

[27] B. Yang, W. Yih, X. He, J. Gao and L. Deng, Embedding Entities and Relations for Learning and Inference in Knowledge Bases, in: *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. http://arxiv.org/abs/1412.6575.

[28] M. Nickel, L. Rosasco and T. Poggio, Holographic Embeddings of Knowledge Graphs, in: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, AAAI Press, 2016, pp. 1955–1961. http://dl.acm.org/citation.cfm?id=3016100.3016172.

[29] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier and G. Bouchard, Complex Embeddings for Simple Link Prediction, in: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, JMLR.org, 2016, pp. 2071–2080. http://dl.acm.org/citation.cfm?id=3045390.3045609.

[30] A. Bordes, X. Glorot, J. Weston and Y. Bengio, A semantic matching energy function for learning with multi-relational data, *Machine Learning* **94**(2) (2014), 233–259. doi:10.1007/s10994-013-5363-6.

[31] R. Socher, D. Chen, C.D. Manning and A. Ng, Reasoning With Neural Tensor Networks for Knowledge Base Completion, in: *Advances in Neural Information Processing Systems 26*, C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K.Q. Weinberger, eds, Curran Associates, Inc., 2013, pp. 926–934.

[32] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun and W. Zhang, Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion, in: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, ACM, New York, NY, USA, 2014, pp. 601–610. ISBN 978-1-4503-2956-9. doi:10.1145/2623330.2623623.

[33] A. Kristiadi, M.A. Khan, D. Lukovnikov, J. Lehmann and A. Fischer, Incorporating Literals into Knowledge Graph Embeddings, in: *Proceedings of International Semantic Web Conference*, ISWC'2019, 2019. doi:https://doi.org/10.1007/978-3-030-30793-6$_2$0.

[34] P. Pezeshkpour, L. Chen and S. Singh, Embedding Multimodal Relational Data for Knowledge Base Completion, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018, pp. 3208–3218.

[35] Y. Tay, A.T. Luu, M.C. Phan and S.C. Hui, Multi-task Neural Network for Non-discrete Attribute Prediction in Knowledge Graphs, in: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM'17, Association for Computing Machinery, 2017. doi:10.1145/3132847.3132937.

[36] M. Cochez, M. Garofalo, J. Lenßen and M.A. Pellegrino, A First Experiment on Including Text Literals in KGloVe, in: *Joint Proceedings of ISWC 2018 Workshops SemDeep-4 and NLIWOD-4*, Aachen, Germany, 2018, pp. 103–106.

[37] M. Nickel, V. Tresp and H.-P. Kriegel, Factorizing Yago: Scalable Machine Learning for Linked Data, in: *Proceedings of the 21st international conference on World Wide Web*, ACM, 2012.

[38] G. de Assis Costa and J.M.P. de Oliveira, Towards Exploring Literals to Enrich Data Linking in Knowledge Graphs, in: *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*.

[39] S. Guo, Q. Wang, B. Wang, L. Wang and L. Guo, SSE: Semantically Smooth Embedding for Knowledge Graphs, *IEEE Transactions on Knowledge and Data Engineering* **PP** (2016), 1–1. doi:10.1109/TKDE.2016.2638425.

[40] R. Xie, Z. Liu and M. Sun, Representation Learning of Knowledge Graphs with Hierarchical Types, in: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, New York, USA, 2016, pp. 2965–2971.

[41] D. Krompaβ, S. Baier and V. Tresp, Type-Constrained Representation Learning in Knowledge Graphs, in: *Proceedings of the 14th International Conference on The Semantic Web - ISWC 2015 - Volume 9366*, Springer-Verlag, Berlin, Heidelberg, 2015, pp. 640–655. ISBN 978-3-319-25006-9.

[42] Q. Wang, B. Wang and L. Guo, Knowledge Base Completion Using Embeddings and Rules, in: *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, AAAI Press, 2015, pp. 1859–1865. ISBN 978-1-57735-738-4. http://dl.acm.org/citation.cfm?id=2832415.2832507.

[43] K.-W. Chang, W.-t. Yih, B. Yang and C. Meek, Typed Tensor Decomposition of Knowledge Bases for Relation Extraction, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1568–1579. doi:10.3115/v1/D14-1165. https://www.aclweb.org/anthology/D14-1165.

[44] Z. Hu, P. Huang, Y. Deng, Y. Gao and E. Xing, Entity Hierarchy Embedding, in: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Association for Computational Linguistics, Beijing, China, 2015, pp. 1292–1300. doi:10.3115/v1/P15-1125. https://www.aclweb.org/anthology/P15-1125.

[45] Y. Lin, Z. Liu and M. Sun, Modeling Relation Paths for Representation Learning of Knowledge Bases, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 705–714. doi:10.18653/v1/D15-1082. https://www.aclweb.org/anthology/D15-1082.

[46] K. Guu, J. Miller and P. Liang, Traversing Knowledge Graphs in Vector Space, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 318–327. doi:10.18653/v1/D15-1038. https://www.aclweb.org/anthology/D15-1038.

[47] A. García-Durán, A. Bordes and N. Usunier, Composing Relationships with Translations, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 286–290. doi:10.18653/v1/D15-1034. https://www.aclweb.org/anthology/D15-1034.

[48] A. Neelakantan, B. Roth and A. Mccallum, Compositional Vector Space Models for Knowledge Base Completion, in: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1, Association for Computational Linguistics, Beijing, China, 2015, pp. 156–166. doi:10.3115/v1/P15-1016. https://www.aclweb.org/anthology/P15-1016.

[49] R. Das, A. Neelakantan, D. Belanger and A. Mccallum, Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks, in: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 2017, pp. 132–141. doi:10.18653/v1/E17-1013.

[50] Y. Luo, Q. Wang, B. Wang and L. Guo, Context-Dependent Knowledge Graph Embedding, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1656–1661. doi:10.18653/v1/D15-1191. https://www.aclweb.org/anthology/D15-1191.

[51] A. García-Durán and M. Niepert, KBlrn: End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features, in: *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

[52] Z. Wei, J. Zhao, K. Liu, Z. Qi, Z. Sun and G. Tian, Large-scale Knowledge Base Completion: Inferring via Grounding Network Sampling over Selected Instances, in: *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, CIKM'15, Association for Computing Machinery, Melbourne, Australia, 2015, pp. 1331–1340. doi:10.1145/2806416.2806513.

[53] S. Guo, Q. Wang, L. Wang, B. Wang and L. Guo, Jointly Embedding Knowledge Graphs and Logical Rules, in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Austin, Texas, 2016, pp. 192–202. doi:10.18653/v1/D16-1019.

[54] T. Rocktäschel, S. Singh and S. Riedel, Injecting Logical Background Knowledge into Embeddings for Relation Extraction, in: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Denver, Colorado, 2015, pp. 1119–1129. doi:10.3115/v1/N15-1118. https://www.aclweb.org/anthology/N15-1118.

[55] T. Jiang, T. Liu, T. Ge, L. Sha, S. Li, B. Chang and Z. Sui, Encoding Temporal Information for Time-Aware Link Prediction, in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Austin, Texas, 2016, pp. 2350–2354. doi:10.18653/v1/D16-1260. https://www.aclweb.org/anthology/D16-1260.

[56] C. Esteban, V. Tresp, Y. Yang, S. Baier and D. Krompaß, Predicting the co-evolution of event and Knowledge Graphs, in: *2016 19th International Conference on Information Fusion (FUSION)*, 2016, pp. 98–105.

[57] R. Trivedi, H. Dai, Y. Wang and L. Song, Know-evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs, in: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, JMLR.org, 2017, pp. 3462–3471. http://dl.acm.org/citation.cfm?id=3305890.3306039.

[58] J. Feng, M. Huang, Y. Yang and X. Zhu, GAKE: Graph Aware Knowledge Embedding, in: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, The COLING 2016 Organizing Committee, Osaka, Japan, 2016, pp. 641–651. https://www.aclweb.org/anthology/C16-1062.

[59] X. Jiang, V. Tresp, Y. Huang and M. Nickel, Link Prediction in Multi-relational Graphs Using Additive Models, in: *Proceedings of the 2012 International Conference on Semantic Technologies Meet Recommender Systems &#38; Big Data - Volume 919*, SeRSy'12, CEUR-WS.org, Aachen, Germany, Germany, 2012, pp. 1–12. http://dl.acm.org/citation.cfm?id=2887638.2887639.

[60] H. Mousselly-Sergieh, T. Botschen, I. Gurevych and S. Roth, A multimodal translation-based approach for knowledge graph representation learning, in: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, 2018, pp. 225–234.

[61] P. Goyal and E. Ferrara, Graph Embedding Techniques, Applications, and Performance: A Survey, *Knowledge-Based Systems* (2018).

[62] H. Cai, V.W. Zheng and K.C.-C. Chang, A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications, *IEEE Transactions on Knowledge and Data Engineering* **30**(9) (2018), 1616–1637.

[63] Y. Lin, Z. Liu and M. Sun, Knowledge Representation Learning with Entities, Attributes and Relations, in: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, New York, USA, 2016, pp. 2866âĂŞ2872–.

[64] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic Web* **8**(3) (2017), 489–508. doi:10.3233/SW-160218.

[65] D. Ruffinelli, S. Broscheit and R. Gemulla, You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings, in: *International Conference on Learning Representations*, 2020. https://openreview.net/forum?id=BkxSmlBFvr.

[66] G.A. Gesese, R. Biswas and H. Sack, A Comprehensive Survey of Knowledge Graph Embeddings with Literals: Techniques and Applications, in: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019.*, 2019, pp. 31–40. http://ceur-ws.org/Vol-2377/paper_4.pdf.

[67] Z. Wang, J. Zhang, J. Feng and Z. Chen, Knowledge graph and text jointly embedding, in: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1591–1601.

[68] M. Cochez, P. Ristoski, S.P. Ponzetto and H. Paulheim, Global RDF Vector Space Embeddings, in: *International Semantic Web Conference*, Springer, 2017.

[69] J. Pennington, R. Socher and C. Manning, Glove: Gobal Vectors for Word Representation, in: *booktitle = "Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)"*, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532–1543. doi:10.3115/v1/D14-1162. https://www.aclweb.org/anthology/D14-1162.

[70] L. Galárraga, C. Teflioudi, K. Hose and F.M. Suchanek, Fast Rule Mining in Ontological Knowledge Bases with AMIE+, *The International Journal on Very Large Data Bases (VLDB)* (2015). doi:10.1007/s00778-015-0394-1.

[71] A. Krizhevsky, I. Sutskever and G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[72] G. de Assis Costa and J.M.P. de Oliveira, A Blocking Scheme for Entity Resolution in the Semantic Web, in: *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 1138–1145.

[73] G.A. Miller, WordNet: a lexical database for English, *Communications of the ACM* **38**(11) (1995), 39–41.

[74] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. http://arxiv.org/abs/1409.1556.

[75] M. Francis-Landau, G. Durrett and D. Klein, Capturing semantic similarity for entity linking with convolutional neural networks, in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, San Diego, California, 2016, pp. 1256–1261. doi:10.18653/v1/N16-1150. https://www.aclweb.org/anthology/N16-1150.

[76] J. Zhao, Y. Kim, K. Zhang, A.M. Rush and Y. LeCun, Adversarially regularized autoencoders, in: *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80, PMLR, Stockholmsmässan, Stockholm Sweden, 2018, pp. 5902–5911. http://proceedings.mlr.press/v80/zhao18b.html.

[77] D. Berthelot, T. Schumm and L. Metz, Began: Boundary equilibrium generative adversarial networks, *arXiv preprint arXiv:1703.10717* (2017).

[78] P. Isola, J.-Y. Zhu, T. Zhou and A.A. Efros, Image-to-image translation with conditional adversarial networks, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[79] D. Liben-nowell and J. Kleinberg, The Link Prediction Problem for Social Networks, *Journal of the American Society for Information Science and Technology* **58** (2003). doi:10.1002/asi.20591.

[80] L. Lü and T. Zhou, Link prediction in complex networks: A survey, *Physica A: statistical mechanics and its applications* **390**(6) (2011), 1150–1170.

[81] F.M. Suchanek, G. Kasneci and G. Weikum, Yago: A Core of Semantic Knowledge, in: *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, ACM, New York, NY, USA, 2007, pp. 697–706. ISBN 978-1-59593-654-7. doi:10.1145/1242572.1242667.

[82] K. Toutanova and D. Chen, Observed versus latent features for knowledge base and text inference, in: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, Association for Computational Linguistics, Beijing, China, 2015, pp. 57–66. doi:10.18653/v1/W15-4007. https://www.aclweb.org/anthology/W15-4007.

[83] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor, Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge, in: *ACM SIGMOD international conference on Management of data*, 2008.

[84] Z. Sun, S. Vashishth, S. Sanyal, P. Talukdar and Y. Yang, A re-evaluation of knowledge graph completion methods, *arXiv preprint arXiv:1911.03903* (2019).

Appendix A. Summary of Applications

Table 16: Summary of different applications on which the KG embedding techniques with literals, in their original papers, have been trained and/or evaluated

	Link Prediction	Triple Classification	Entity Classification	Entity Alignment	Attribute value prediction	Nearest neighbour analysis	Data linking	Document classification	Relational Fact Extraction
Extended RESCAL	✓		✓						
LiteralE	✓					✓			
TransEA	✓								
KBLRN	✓								
DKRL	✓		✓						
KDCoE	✓			✓					
KGlove with literals								✓	
IKRL	✓	✓							
EAKGE	✓			✓					
MKBE	✓				✓				
MT-KGNN		✓			✓				
LiteralE with blocking							✓		
Jointly(Desp)	✓	✓							✓
Jointly	✓	✓							
SSP	✓		✓						
MTKGRL	✓	✓							