FeaBI: A Feature Selection-Based Framework for Interpreting KG Embeddings

Youmna Ismaeil[1,2](✉), Daria Stepanova[1], Trung-Kien Tran[1], and Hendrik Blockeel[2]

[1] Bosch Center for Artificial Intelligence, Renningen, Germany
{youmna.ismaeil,daria.stepanova,trung-kien.tran}@de.bosch.com
[2] KU Leuven, Leuven, Belgium
{youmna.ismaeil,hendrik.blockeel}@kuleuven.be

Abstract. Knowledge Graph (KG) embedding methods represent KG entities as vectors in an embedding space, and they have been successfully used for a variety of tasks, including link prediction and entity classification. While some of the recent embedding methods outperform traditional approaches on these tasks, their main drawback is the lack of interpretability. Several methods for explaining predictions made by KG embeddings have been proposed in the literature. However, none of them targeted the problem of constructing model explanations for embeddings, i.e., interpretable KG representations that behave similarly to embeddings on certain tasks. We address this problem and propose a novel method for generating interpretable vectors for entity embeddings. To achieve this, we employ embedded feature selection techniques to extract from the KG, on which the embedding model was trained, propositional features that are important for a given KG embedding model. Our approach sheds light on the information in the KG captured by embeddings and provides valuable insights that can be used to further enhance the embedding models. Additionally, we demonstrate the usefulness of our method for explaining embedding-based entity similarity.

1 Introduction

Motivation. Knowledge Graphs (KG) describe facts about a certain domain of interest by representing them using entities interconnected via relations. Existing KGs such as YAGO [36], Freebase [5], and DBpedia [1] contain millions of facts about people, places, organizations, etc. over hundreds of relations. For instance, an example of a KG presenting information about companies, people, products, and relations among them is presented on the left side of Fig. 1.

Deep learning techniques, such as Knowledge Graph embeddings (see [41] for an overview) are increasingly being applied to solve various machine learning tasks (e.g., link prediction [6,39,43] or entity classification [18,31]) that use KGs as input data. However, these techniques typically learn a latent representation for the entities of interest, which is often not comprehensible to humans. Thus, deep learning techniques are usually considered to be black boxes.

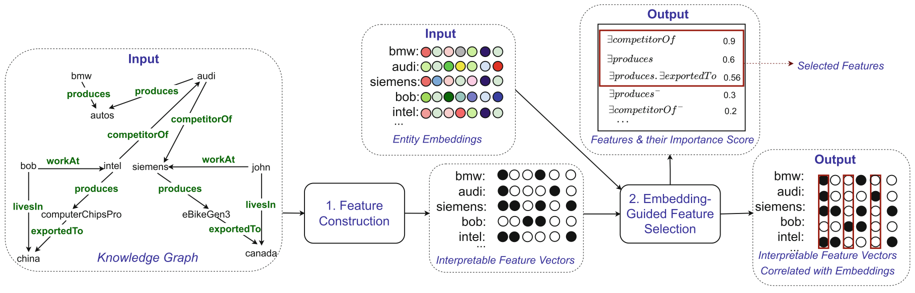

Fig. 1. Embedding-guided feature vector generation for KG entities

State-of-the-Art and Its Limitations. There has been a surge of interest in understanding how deep learning models work. Two distinct types of explanations have emerged: prediction explanations and model explanations [9]. While prediction explanations justify a particular prediction generated by a given black-box model, different definitions for model explanations exist in the literature (see, e.g., [22] for an extensive discussion). In this work, with model explanations we refer to the identification of parts of the input data that contributed most to the construction of the model, also known as feature importance-based model explanations [4]. Several works have focused on constructing prediction explanations for embeddings [3,7,14,32]. E.g., [32] generates explanations for the link prediction task by perturbing the training data, while [7] constructs prediction explanations for link prediction and triple classification using entity co-occurrence statistics. However, none of the above methods target the problem of computing model explanations, which are complementary to prediction explanations and can be exploited for model analysis. Some methods [35] construct interpretable embeddings, but these are not designed to explain or analyze other deep learning-based models. In this paper, we address the respective issue.

Our Approach and Contributions. We present a novel method (see Fig. 1 for overview), which computes explainable vectors for entity embeddings and generates KG embedding model explanations in the form of KG features important for the given embedding model.

Our method proceeds as follows. First, we extract propositional features from the KG and express them in Description Logic [2]. Then, we use these features to construct Boolean vectors (a.k.a. feature vectors) for each entity relying on its neighborhood. For example, in the KG from Fig. 1, the Boolean feature vector representation of the entity *siemens* contains the information that *siemens* produces some products that are exported to *canada* and *audi* is among its competitors. Next, given a pre-computed embedding model that we want to explain (i.e., a function mapping entities to vectors in some d-dimensional space), we train a regression random forest on the task of reconstructing embedding-based entity representations using features defined in the first step of our method. The regression random forest model ranks features based on their importance for the reconstruction task. We consider the highest-scoring features as the KG embedding model explanations.

Intuitively, the obtained feature vector and embedding entity representations are two ways of representing entities in the KG. Thus, if there is a regression model that can accurately reconstruct KG embeddings from feature vector-based representations, then the feature vectors and the embeddings contain comparable information. We use the list of the computed most important features as an explanation for the KG embedding model. Finally, we reduce the Boolean feature vector of each entity such that it contains only the features that exist in the list of most important features obtained in the previous step. For instance, in the KG from Fig. 1, the resulting Boolean feature vector representation for *siemens* will contain only selected features.

Our main contributions are summarized as follows:

- We present an automated method for generating model explanations for KG embeddings in the form of KG features most relevant for the given embedding. These identified important KG features are used to construct interpretable feature vector representations (entity-level explanations) for each KG entity, approximating their respective embedding vectors.
- We empirically demonstrate that the generated interpretable feature vectors behave similarly to the respective entity embeddings on the entity and relation classification tasks.
- We show how the interpretable entity representations generated by our method can facilitate the analysis of the behavior of embedding models.
- We provide evidence that our interpretable entity representations can be leveraged to explain similarities between entity embeddings.

2 Preliminaries

Knowledge Graphs. Knowledge Graphs (KGs) represent interlinked collections of factual information, encoded as a set of $\langle$*subject predicate object*$\rangle$ triples, *e.g.*, $\langle$*bmw produces autos*$\rangle$ which can also be represented as ground binary predicates in predicate logic format, e.g., $produces(bmw, autos)$. A signature of a KG $\mathcal{G}$ is $\Sigma_{\mathcal{G}} = \langle \mathbf{R}, \mathbf{E} \rangle$, where $\mathbf{R}$ is a set of binary predicates, *i.e.*, relations and $\mathbf{E}$ is a set of constants, *i.e.*, entities, in the knowledge graph $\mathcal{G}$. Concepts can be built over KGs following the Description Logic (DL) [2] syntax. In this work we mainly rely on concepts described in Table 1.

KG Embeddings. Deep learning methods (in particular, KG embeddings) have been proposed to perform different machine learning tasks on top of KGs, such as link prediction or entity classification. KG embeddings aim at representing all entities and relations in a continuous vector space, usually as vectors or matrices called *embeddings*. Embeddings can be used to estimate the likelihood of a triple being true via a scoring function: $f : \mathbf{E} \times \mathbf{R} \times \mathbf{E} \rightarrow \mathbb{R}$, or to classify entities [18,31,44,46]. In this work, we only make use of entity-based embedding representations leaving the exploitation of relation embeddings for future work.

Table 1. Description Logic concepts where r is an (atomic) relation or its inverse, R is any relation or its inverse, e is an entity, $\top$ is the universal (top) concept.

DL concept	Informal description
$\exists r.\top$	Existence of some relation r
$\exists r.\{e\}$	Existence of some relation r to the entity e
$\exists R.\{e\}$	Existence of any relation to the entity e
$\exists r_1.\exists r_2 ... \exists r_k.\top$	Existence of a chain of relations $r_1, r_2, \ldots, r_k$
$\leq kR.\top$, $\geq kR.\top$	Less (greater) than or equal to k overall number of relations

Thus, without loss of generality, we assume that an embedding model is given a function: $\mathcal{E} : \mathbf{E} \mapsto \mathbb{R}^d$, which maps entities from $\mathbf{E}$ to d-dimensional real vectors.

One of the earliest and most popular embeddings is *e.g.*, TransE [6], which embeds entities and relations as vectors and assumes $\mathbf{v_s}+\mathbf{v_r} \approx \mathbf{v_o}$ for true triples, where $\mathbf{v_s}, \mathbf{v_r}, \mathbf{v_o}$ are vector embeddings for subject s, relation r and object o, resp. The likelihood that the above assumption holds should be higher for triples in the KG than for those outside. The learning process is done by minimizing the error induced by the respective assumption given the considered loss function.

Several KG embeddings that are based on graph neural networks have also been proposed in the literature (e.g., [40]) and, to the best of our knowledge, currently, they demonstrate the state-of-the-art performance on link prediction and entity classification tasks [12,47]. One of the prominent representatives of this group of embeddings is CompGCN [40], which leverages a variety of entity-relation composition operations from KG embedding techniques as well as several existing multi-relational graph convolutional neural network methods.

3 Embedding-Guided Feature Construction

KG embedding models achieve promising results on different popular tasks; however, they are not interpretable. To address this issue, our work aims at providing insights into the effectiveness of pre-computed KG embedding models in capturing information in the KG by identifying the most important features in the training data for the embedding models and generating interpretable feature vector representations for KG entities. This can help in improving KG embedding models traditionally considered as black boxes.

More formally, given a KG $\mathcal{G}$ and an embedding model $\mathcal{E} : \mathbf{E} \mapsto \mathbb{R}^d$, which maps KG entities to numerical vectors, we aim at investigating the following questions: (a) Can we find a set of features F and an encoding $\mathcal{F}$ of entities using these features such that a function $\mathcal{W}$ exists that maps feature-based encoding $\mathcal{F}(e)$ of any entity $e \in \mathbf{E}$ to $\mathcal{E}(e)$? (b) If we use $\mathcal{F}(\mathbf{E})$ instead of $\mathcal{E}(\mathbf{E})$ as the input for downstream tasks, do we get similar results for these tasks?

In this section, we propose a method for constructing $\mathcal{F}$ according to criterion (a). Then in Sect. 4, we evaluate its practical usefulness according to (b) using entity and relation classification as downstream tasks.

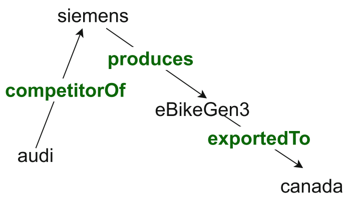

Fig. 2. Fragment of a KG from Fig. 1

	$\exists produces$	$\exists competitorOf$	$\exists exportedTo$	$\exists produces^-$	$\exists exportedTo^-$	$\exists competitor^-$	$\exists R.\{eBikeGen3\}$	$\exists R.\{audi\}$	$\exists R.\{canada\}$	$\exists R.\{siemens\}$	$\exists competitorOf.siemens$	$\exists produces.eBikeGen3$	$\exists exportedTo.canada$	$\exists exported^-.\{eBikeGen3\}$	$\exists produces^-.\{siemens\}$	$\exists competitorOf^-.\{audi\}$	$\exists produces.\exists exportedTo$
siemens	1	0	0	0	0	1	1	1	0	0	0	1	0	0	0	1	1

Fig. 3. Feature vector for the entity *siemens* for KG from Fig. 2

The overview of our method for constructing an interpretable model based on a given KG embedding is presented in Fig. 1. We take as input a knowledge graph and a pre-computed embedding model (e.g., TransE [6], CompGCN [40], or any other model). We then construct interpretable feature vectors for entities as described in Sect. 3.1. The obtained features are used for the embedding-guided feature selection to identify only those features that are important for approximating the given embedding model (see Sect. 3.2). The resulting features are then used to encode each KG entity and serve as embedding explanations, which are further evaluated in Sect. 4.

3.1 Feature Construction

In the first step of our method, we construct interpretable feature vector representations for KG entities relevant to a given embedding model. To achieve this, several proposionalization techniques [8,19,21,30,35] can be invoked, including methods that rely on rule learning [20,23]. In this work, we incorporate four types of features. In addition to relations and relations with entities features proposed in INK [35], we introduce two new features: graph structural statistics (capturing patterns important for graph-based embedding models) and surrounding entities (necessary for embedding models lacking relations and paths with labeled edges, such as Snore [24]). To ensure that the features used in our analysis originated from the triples, on which the embedding models were initially trained, we focus on the above simple feature types. While more complex features extracted from rules or ontologies could be utilized, we choose features (represented in DL) that are more likely to be learned by the embedding models, as they are used explicitly as part of the input during training.

Relations. The first feature type indicates whether an entity takes part in a given relation or not. For each relation r, there are features $\exists r.\top$ and $\exists r^-.\top$ that

	$\exists produces$	$\exists competitorOf$	$\exists produces.\exists exportedTo$
siemens	1	0	1

Fig. 4. Feature vector for the entity *siemens* from Fig. 3 after feature selection based on the important features in Fig. 1.

describe for each entity whether it has an outgoing or incoming relation with the respective label r. For example, for the entity *siemens* in Fig. 2 we have the following set of *relation* features: $\exists produces.\top$, $\exists competitorOf^{-}.\top$.

Entities. The second group of features is formed by collecting for each entity e, the entities e' to which e is connected via relations with some label. In the Description Logic syntax, this amounts to $\exists R.\{e'\}$ and $\exists R^{-}.\{e'\}$. E.g., *siemens* in Fig. 2 has $\exists R.\{eBikeGen3\}$ and $\exists R^{-}.\{audi\}$ among its *entity* features.

Relations with Entities. The third type of features indicates whether a given entity is connected to another entity via a certain relation. For each entity e we consider its outgoing or incoming relations along with the entities to which the given entity is connected. In the Description Logic syntax, this results in the features of the following form: $\exists r.\{e'\}$ and $\exists r^{-}.\{e'\}$. E.g., for *siemens* in Fig. 2 we have the following features $\exists produces.\{eBikeGen3\}$, $\exists competitorOf^{-}.\{audi\}$.

Paths. Another type of features corresponds to paths with a predefined length k $\exists r_1.\exists r_2.\dots.\exists r_k$. E.g., when considering the entity *siemens* and $k = 2$, a single path feature is generated: $\exists produces.\exists exportedTo$.

Statistics. Finally, we consider the number of outgoing and incoming relations r and r^{-} for each entity e. In the DL syntax, this amounts to the following constructors $\geq k.R \sqcap \leq k.R$ and $\geq k.R^{-} \sqcap \leq k.R^{-}$ respectively, for which (with some abuse of notation) we use the shortcuts $= k.R$ and $= k.R^{-}$ to save space. For instance, for *siemens* in Fig. 2 we have $= 1.R$, and $= 1.R^{-}$, as there is 1 outgoing and 1 incoming relation for this entity. Similar to Statistics, additional information about the entities (e.g., textual or numerical attributes) can be considered as an extra entry in the feature vector.

Building on the work in [35], we compute all features including paths for a predefined depth k. We then collect all features of the described types into the set $F = \{f_1, \dots, f_p\}$, where p is the total number of features extracted from the KG $\mathcal{G}$. Based on the knowledge graph and the respective set of features F, we form an interpretable Boolean vector $\mathbf{fv}_e = [f_0^e, f_1^e, \dots, f_p^e]$ for each entity e, such that $|\mathbf{fv}_e| = |F| = p$ and $f_i^e = 1$ iff the feature f_i holds for the entity e in the KG $\mathcal{G}$, and $f_i^e = 0$ otherwise. E.g., the resulting representation for the entity *siemens* for a small KG fragment from Fig. 2 is presented in Fig. 3.

3.2 Feature Selection

The pre-constructed set of features generated as described in Sect. 3.1 encodes information about the entities based on their neighborhood. However, it is unclear which of these features are relevant to the embedding model. To identify the most relevant features in the KG for embedding reconstruction, we adopt an embedded feature selection technique inspired by [37]. This approach allows us to use the embeddings to guide us in identifying and scoring the crucial features in the KG needed to reconstruct the embeddings. For the step of embedded feature selection, various models (e.g., decision trees, MLP) providing feature scores can be in principle utilized. In our case, we use random regression forest due to its simplicity, interpretability and resistance to overfitting [33]. The model, denoted as M, is trained to reconstruct the corresponding embeddings from their feature-based entity representations.

The random regression forest constructs a separate regression tree for predicting each embedding dimension using a subset of the features in the input. The respective trees are then combined to make predictions for all the embedding dimensions at once by averaging the predictions made by each regression tree for each dimension. During training, the random forest regression model scores the input features based on their importance (a.k.a. feature importance score) for the reconstruction task. The features with the highest importance score are used to explain the info captured by the KG embedding.

Formally, let $F = \{f_1, \ldots, f_p\}$ be a set of features computed in the previous step, e.g., $\exists \mathit{competitorOf}, \exists \mathit{produces}$, etc. As mentioned in Sect. 3.1, each entity $e \in \mathbf{E}$ is represented as a Boolean vector of the form $\mathbf{fv}_e = [f_0^e, \ldots, f_p^e]$, where f_i^e reflects the presence or absence of the respective feature f_i for the entity e. For the training input to the random forest regression model for each entity $e \in \mathbf{E}$, we have its Boolean feature vector $\mathbf{fv}_e = [f_0^e, f_1^e, \ldots, f_p^e]$ with the label being the embedding of e, i.e., $\mathbf{v}_e \in \mathbb{R}^d$ computed by the target embedding model $\mathcal{E}$ which we aim at explaining. The random forest regression model [33] is used to find the mapping between the feature vectors and embeddings. Additionally, the model outputs the importance score for each feature, which is computed relying on the mean squared error on the respective predictive task of reconstructing entity embeddings from feature vector-based entity representations. Once the model M is trained, the scores assigned to the features are extracted, and only the features with scores that are greater than a certain threshold θ are deemed essential. In our experiments, we set the threshold for the feature selection parameter to be the mean of the feature importance scores, as commonly done in practice.

The features selected by our feature selection algorithm reflect the parts of the KG that the embedding model focuses on most. Therefore, we refer to them as model explanations. Next, we utilize the selected features to obtain instance-level explanations for each entity embedding. This is achieved by keeping only the selected features within each entity representation. For example, if we have a set of selected features for a KG embedding model (Fig. 1), we obtain the corresponding feature vector-based representation for entity *siemens* (Fig. 4).

Table 2. Knowledge graph data statistics. The train and test splits are used for training the respective embedding models

KG elements	FB15K-237	DBpedia50K
Entities	14,541	49,900
Relations	237	654
Train triples	272,115	32,388
Validation triples	17,535	399
Test triples	20,466	10,969

4 Experiments

We evaluate the quality and usefulness of the interpretable feature-based entity representations computed by our method. To evaluate the quality, we compare the behavior of the original embedding-based entity representations and our interpretable entity representations on the two tasks: 1) the entity classification following [17], and 2) the relation classification, i.e., given a pair of entities, predict a relation that holds between the entities. Additionally, to demonstrate the usefulness of our method, we consider the entity similarity task and show how our feature-based entity representations can be exploited to compute explanations for entities being close to each other in the embedding space.

4.1 Experimental Setup

Datasets. We experiment with two widely used knowledge graphs namely, Freebase15k-237 (a.k.a. FB15k-237) [38] and DBpedia50K [34] (see Table 2).

Embedding Models. For embeddings, we consider (1) TransE [6] as one of the first and most popular models; (2) CompGCN [40] as one of the latest GNN-based models which achieves state-of-the-art results on the entity classification tasks; (3) NodePiece [12] as a shallow path-based embedding model; and (4) Snore [24] and INK [35] as embeddings that are interpretable by construction.

Evaluation Tasks. We consider entity classification and relation classification as the tasks on which we evaluate whether our feature vectors encompass the same information as the corresponding embedding-based entity representations.
• **Feature Selection Evaluation.** To evaluate our feature selection module, we apply the proposed framework to approximate Boolean INK [35] embeddings, for which interpretable features are known. More specifically, we compute the F1 score reflecting the similarity between the most important features generated by our method and the features used for constructing INK embeddings.

• **Entity Classification.** Entity classification is concerned with the prediction of a label from a given set of labels for KG entities. The goal of this evaluation is to verify whether different classifiers when trained on feature vector-based entity representations and embedding-based entity representations, align with each other by making the same correct and incorrect predictions for the entity classification task. For the FB15K-237 KG, we consider the labels for entities from [17], and for the DBPedia50K KG, we extract the entity types as labels. Similar to [28], we have selected the following classifiers: (1) Random forest [33]; (2) K-nearest neighbor classifier [10]; (3) Multi-layer perceptron (MLP) [15]. For all the models used in this work, we used the default parameter setup defined by the scikit-learn library [27] for the respective model.

We train the respective classifiers on the train set of labeled KG entity embeddings and the train set of our feature vector representations approximating the target embedding. We then use the respective classifiers to predict the labels for entities in the test set, pick the classifier that shows the best performance for the embedding-based entity classification, and use its predictions to compute the alignment score (weighted average F1 score) between the predictions made by the embedding-based and feature-vector based classifiers. The alignment score evaluates the consistency between predictions using embeddings and those using feature-vectors. Accounting for false positives and negatives, the F1 score quantifies the overall agreement. A perfect alignment score is 1, reflecting that the behavior of both embeddings and feature vectors is identical on a specific task.

Since, to the best of our knowledge, no previous works have targeted the problem of computing model explanations for KG embedding models, as a baseline, we use the feature vectors constructed by our method but without the feature selection step. If the feature selection step removes a large percentage of features, but this does not impact the weighted average F1 score, one can conclude that the removed features are indeed not important for the embedding model on the respective task.

• **Relation Classification.** In the relation classification task, we randomly select a set $\mathbf{R}'$ of 50 relations from the KG, resulting in 39,273 (resp. 24,832) entity pairs $E = \{(e_1, e_2) \,|\, \langle e_1, r, e_2\rangle, r \in R\}$ for FB15k-237 (resp. for DBpedia50k). We split the set E into the train (70 % of entity pairs) and the test (30 %) sets.

We proceed with evaluating the alignment between the embedding-based entity representations and the feature-vector-based entity representations in the same way as for the entity classification task. The only difference is that the task considered in this experiment is rather concerned with classifying pairs of entities from the test set into a class r from the set of 50 classes in $\mathbf{R}'$. Intuitively, the meaning of a pair being classified to a specific class r is that the relation r holds between the respective entities.

Applications. Our feature vector-based entity representations can be exploited for analyzing and comparing different embedding models. To this end, we report and examine the types of features in the model explanations computed by our method for different embeddings and datasets as well as present examples of

Table 3. Mean squared error of the random forest regression model used to learn important features for the regeneration of the embeddings.

	Mean squared error (MSE)	
Embedding model	DBPedia50K	FB15K-237
Random	55.70	54.76
INK	0.0001	0.0003
TransE	0.026	0.0257
CompGCN	0.007	0.005
NodePiece	0.016	0.035
Snore	0.138	0.076

selected features along with computed entity representations. Another application of our method is explainable entity similarity. Given the embeddings learned by an embedding model and their corresponding interpretable feature vectors **fv** computed by our method, one can generate explanations for the similarity between any pair of entities e_i and its neighbor e_j in the embedding space as the intersection between their respective feature vectors, i.e., $\mathbf{fv}_{e_i} \cap \mathbf{fv}_{e_j}$.

4.2 Experimental Results

Evaluation of Feature Selection. We report the mean squared error (MSE) of the random forest regression model, trained to regenerate the embeddings from feature vectors in Table 3. As a baseline, we consider a simple model that outputs a vector of random numbers of size 50 (average size of the embeddings used) when given a feature vector as input. The respective baseline is referred to as Random in Table 3. The low MSE values for all models apart from the baseline witness that the random forest regression model is capable of identifying a meaningful relationship between the input feature vectors and the corresponding KG embeddings. For the random embeddings, the model failed to find a connection between the input feature vectors and the randomly generated embeddings, leading to high MSE.

Additionally, we also computed the alignment F1 score between the features selected by our feature selection algorithm and the original features present in the INK embeddings. For FB15k-237, the F1 score is 0.77 and for DBpedia50K it is 0.82, indicating that the majority of the selected features were indeed in the input embeddings. These results further confirm the effectiveness of the feature selection method in accurately retrieving the content of embeddings. More results on custom embeddings are in Table 16 in Sec. 7.3 in the Appendix.

Entity and Relation Classification. In Table 4, we report the weighted average F1 score for the alignment between the classification results computed by the respective embedding models and the feature vector-based entity representations before (columns 4 and 6) and after (columns 5 and 7) the feature selection step.

Table 4. The alignment F1 score between embedding-based classifiers and our feature-vector-based classifiers (before and after the feature selection step) on the entity and relation classification tasks. The third column represents the percentage of the selected features out of the total number of features.

			Entity-classification (F1)		Relation classification (F1)	
Interpreted model	Dataset	Selected feature %	Before selection	After selection	Before selection	After selection
INK [35]	FB15K-237	75.5	0.93	0.80	0.90	0.90
	DBpedia50k	50.5	0.73	0.78	0.97	0.96
TransE [6]	FB15k-237	19.7	0.84	0.84	0.73	0.78
	DBpedia50k	13.9	0.61	0.64	0.53	0.54
CompGCN [40]	FB15k-237	24.4	0.64	0.64	0.57	0.61
	DBpedia50k	31.8	0.52	0.69	0.63	0.65
NodePiece [12]	FB15k-237	20.0	0.68	0.68	0.47	0.51
	DBpedia50k	22.7	0.73	0.73	0.75	0.75
Snore [24]	FB15k-237	11.9	0.66	0.69	0.23	0.23
	DBpedia50k	2.5	0.59	0.59	0.50	0.51

We also report the percentage of all features left after the feature selection step (column 3). One can observe that the alignment scores reach 0.84 for entity classification and 0.78 for relation classification tasks, respectively. The alignment score for Snore on the relation classification task was low, which is attributed to the fact that its embeddings do not consider relations as mentioned in [24]. As a consequence, the performance of Snore on the relation classification task is rather poor, with the F1 score of 0.16. Our feature selection-based method also manages to reproduce this, which is witnessed by the results in Fig. 6, where the top 10 features selected as the most important ones for Snore on DBPedia50K correspond to entities. Moreover, we found that this result is invariant to the value of θ, as the performance was poor even before feature selection. Furthermore, we observed that the alignment with INK is the highest, which can be attributed to the interpretable nature of the INK embeddings and the intersection of features used in our method with those used in INK. Based on the percentage of selected features, the results show that NodePiece uses less features than CompGCN but more than TransE. This is due to the facts that NodePiece retains small amount of information about explicit nodes without sacrificing its performance on down stream tasks.

Table 4 shows that the feature selection discards up to 97.5% of the features without negative impact on the alignment F1 score. This indicates that the embeddings do not utilize all of the information in the neighborhood of entities.

KG Embedding Analysis. In Fig. 5, we present detailed information regarding the contribution of each feature type to the total number of features before and after the feature selection step for the models TransE, CompGCN, NodePiece, and Snore on FB15k-237 and DBpedia50k. Figure 5 shows that even though inward relations with entities constitute more than 50% of all features, they are mostly ignored by our selection mechanism (at most 16% were selected for TransE and 17% for CompGCN across all considered datasets). This might indicate that the embedding models learn to represent each entity in terms of

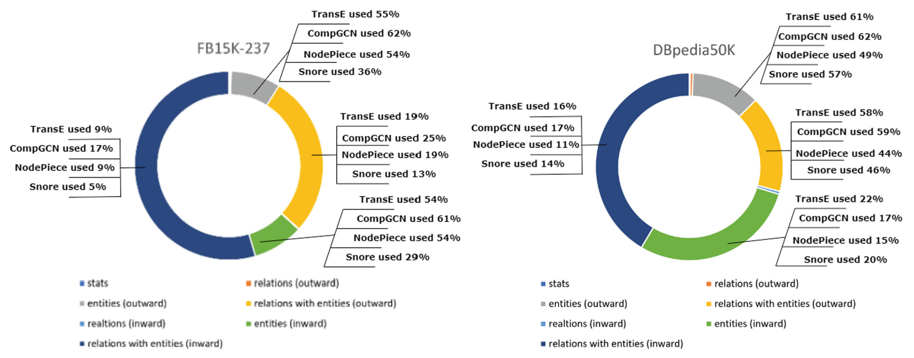

Fig. 5. The percentage of each feature type out of the total number of features for FB15k-237 (left) and DBPedia50k (right). Labels reflect the percentage of features per feature type selected by our method for the considered models.

FB15K-237		**DBpedia50K**	
TransE	**CompGCN**	**Snore**	**NodePiece**
$\exists profession$	$= 0.R^-$	$\exists genre$	$= 1.R$
$\exists film^-$	$= 1.R^-$	$\exists R.\{germany\}$	$\exists team$
$\exists award^-$	$= 2.R^-$	$\exists R.\{london\}$	$\exists musicalArtist^-$
$\exists genre$	$= 3.R^-$	$= 0.R$	$\exists birthplace$
$\exists artists^-$	$= 2.R$	$\exists R.\{drumKit\}$	$= 1.R^-$
$\exists institution^-$	$= 1.R$	$\exists recordLabel$	$\exists staring$
$\exists role^-$	$= 4.R$	$\exists R.\{iran\}$	$\exists genre$
$= 0.R^-$	$= 0.R$	$\exists R.\{insect\}$	$\exists team^-$
$\exists team^-$	$\exists placeOfBirth$	$\exists R.\{southKorea\}$	$\exists producer$
$= 1.R^-$	$= 3.R$	$\exists kingdom.\{plant\}$	$\exists writer$

Fig. 6. The top 10 most important features identified by our method for FB15K-237 and DBpedia50k, sorted in descending order of their importance scores.

outward (paths of) relations and entities rather than inward ones. This behavior was consistent over all four knowledge graph embedding models used.

We can observe that the portion of features filtered out by our selection method within each feature type is significant, reaching more than 60% for some feature types. Our feature selection technique filtered out more features for Snore and NodePiece models than for other models. The selected feature types are consistent with the methods for constructing the respective embeddings [12,24]. For the CompGCN model the majority of the features were retained. In Fig. 6, we also present the top 10 features selected by our framework for FB15k-237 on TransE and CompGCN. One can notice that CompGCN focuses more on the information about the number of inward and outward relations and entities with large numbers of incoming and outgoing relations which aligns with the limitations of GNNs in [42]. A similar behavior has been observed for DBpedia50k.

The model explanations can also be used to analyze the behavior of the embedding models throughout training phases (see Sec. 7.4 in the Appendix).

Table 5. Feature vector of entity "*América de Cali Club*" from DBpedia50k KG computed by our method for different models.

Model	Feature vector representation
Original (no feature selection)	$\exists team^{-}, \exists team^{-}.\{AndrsAndrade\}, \exists team^{-}.\{ChristianMontao\}, \exists team^{-}.\{DanielTlger\}, \exists team^{-}.\{DavidFerreira\}, \exists team^{-}.\{DiegoGmez\}, \exists team^{-}.\{EduardoFerreira\}, \exists team^{-}.\{ErnestoFaras\}, \exists team^{-}.\{GiovanniHernndez\}, \exists team^{-}.\{HugoSosa\}, \exists team^{-}.\{JairReinoso\}, \exists team^{-}.\{JavierArizala\}, \exists team^{-}.\{JohnCrdoba\}, \exists team^{-}.\{JohnHaroldLozano\}, \exists team^{-}.\{JuanCamiloAngulo\}, \exists team^{-}.\{LeandroCastellanos\}, \exists team^{-}.\{LuisBarbat\}, \exists team^{-}.\{LuisEduardoZapata\}, \exists team^{-}.\{LuisMarcoleta\}, \exists team^{-}.\{RobertoCabaas\}, \exists team^{-}.\{WilsonMorelo\}, = 20.R^{-}, = 0.R, \exists R^{-}.\{AndrsAndrade\}, \exists R^{-}.\{ChristianMontao\}, \exists R^{-}.\{DanielTlger\}, \exists R^{-}.\{DavidFerreira\}, \exists R^{-}.\{DiegoGmez\}, \exists R^{-}.\{EduardoFerreira\}, \exists R^{-}.\{ErnestoFaras\}, \exists R^{-}.\{GiovanniHernndez\}, \exists R^{-}.\{HugoSosa\}, \exists R^{-}.\{JairReinoso\}, \exists R^{-}.\{JavierArizala\}, \exists R^{-}.\{JohnCrdoba\}, \exists R^{-}.\{JohnHaroldLozano\}, \exists R^{-}.\{JuanCamiloAngulo\}, \exists R^{-}.\{LeandroCastellanos\}, \exists R^{-}.\{LuisBarbat\}, \exists R^{-}.\{LuisEduardoZapata\}, \exists R^{-}.\{LuisMarcoleta\}, \exists R^{-}.\{RobertoCabaas\}, \exists R^{-}.\{WilsonMorelo\}$
TransE	$\exists team^{-}, = 0.R, \exists team^{-}.\{DanielTlger\}, \exists R.\{JohnCrdoba\}$
CompGCN	$\exists team^{-}.\{JavierArizala\}, \exists team^{-}.\{LuisEduardoZapata\}, = 20.R, = 0.R^{-}, \exists R^{-}.\{JavierArizala\}, \exists R^{-}.\{LuisEduardoZapata\}, \exists R^{-}.\{LuisMarcoleta\}$
NodePiece	$\exists team^{-}.\{EduardoFerreira\}, \exists team^{-}, = 20.R^{-}, = 0.R, \exists R^{-}.\{EduardoFerreira\},$
Snore	$\exists team^{-}, = 20.R^{-}, = 0.R$

Example Feature Vectors. In Table 5 and 6, we present example feature-based representations computed by our method for entities "*América de Cali Club*" and "*Fox Channel (Asia)*" from DBPedia50K KG. For these entities the CompGCN embedding model tends to capture more inward relations with entities than other models. For TransE the majority of the selected features are relations with entities with different relations, which aligns well with its limitation to capture one-to-many relations [41].

Although CompGCN better grasps inward relations with entities compared to other models, it fails to capture all the entities with dense neighborhood, i.e., entities having numerous inward relations with the same label. We observed this behavior for the entity "*América de Cali Club*", which has 20 inward relations with entities labeled with the same relation type (see Table 5). The same holds for the outward relations with entities for "*Fox Channel (Asia)*" in Table 6. More examples are in the appendix. After conducting the feature selection process, we observed an average of around 29K-19K selected features for FB15K-236 and DBPedia50k, respectively. Despite the high average, individual entities typically possess a limited number of features in their representations, 6-54 features per entity on average for FB15K-236 and DBPedia50k, respectively. Hence, feature representations are easily comprehensible by humans, making them suitable for manual examination (see Table 9 and 10 in the appendix).

Explainable Entity Similarity. Our explanations for embedding models can be utilized to explain similarity between entities. To exemplify this application, in Table 7 we report the closest and the 100th closest entity to "*Christina*

Table 6. Feature vector of the entity "*Fox Channel (Asia)*" from DBpedia50k KG computed by our method.

Model	Feature vector representation
Original (no feature selection)	$\exists broadcastArea, \exists broadcastArea.\{SouthKorea)\}, \exists language,$ $\exists language.\{EnglishLanguage\}, \exists sisterStation, \exists sisterStation.\{FoxFilipino\},$ $\exists sisterStation.\{NatGeoPeople\}, \exists sisterStation.\{NatGeoWild\},$ $\exists sisterStation.\{StarSports\}, \exists sisterStation.\{StarWorld\}, =7.R, =0.R^{-},$ $\exists R.\{SouthKorea)\}, \exists R.\{EnglishLanguage\}, \exists R.\{FoxFilipino\},$ $\exists R.\{NatGeoPeople\}, \exists R.\{NatGeoWild\}, \exists R.\{StarSports\}, \exists R.\{StarWorld\}$
TransE	$\exists broadcastArea, \exists broadcastArea.\{SouthKorea)\}, \exists language,$ $\exists language.\{EnglishLanguage\}, \exists sisterStation, =0.R^{-}, \exists R.\{SouthKorea)\},$ $\exists R.\{EnglishLanguage\}, \exists R.\{NatGeoPeople\}$
CompGCN	$\exists broadcastArea, \exists language, \exists language.\{EnglishLanguage\},$ $\exists sisterStation, \exists sisterStation.\{FoxFilipino\}, \exists sisterStation.\{NatGeoPeople\},$ $\exists sisterStation.\{NatGeoWild\}, =7.R, =0.R^{-}, \exists R.\{SouthKorea)\},$ $\exists R.\{EnglishLanguage\}, \exists R.\{FoxFilipino\}, \exists R.\{NatGeoPeople\}, \exists R.\{NatGeoWild\}$

Aguilera" based on the cosine similarity of the respective vectors for TransE model on the FB15K-237 dataset along with the features computed by our method that the respective entities share. The results reveal that "*Christina Aguilera*" shares more features with its closest neighbor "*Katy Perry*" than with "*Ice Cube*", which explains why the respective entities are closer in the embedding space. We provide further results for the entity similarity application in appendix. These results show insights into the embedding space learned by the embedding models, allowing for a better understanding of possible reasons for the respective positions of entities in the embedding space.

5 Related Work

Explanations of KG Embedding Models. In recent years, a variety of KG embeddings have been proposed, e.g., [6,12,26,31,40] (see [41] for an overview). While some of the methods achieve state-of-the-art performance on certain tasks [12,40,47], the models often remain to be black boxes. This has raised interest in explaining KG embeddings and led to works targeting outcome explanations for embeddings on the link prediction task [3,7,11,13,25,32]. In [32] a Kelpie framework has been introduced, which explains predictions made by a given embedding by identifying the combinations of training facts that have enabled the respective predictions. The works [3,13] exploit rule learning techniques for identifying triples which are logical explanations for a particular prediction. Generation of post-hoc explanation for triples inferred by a (factorization-based) embedding model has been considered in [25]. This method first augments the underlying KG by introducing weighted edges between entities relying on their embedding-based similarity and then computes human-understandable explanations in the form of paths. All of the above methods

Table 7. The list of explanations for the similarity of "*Christina Aguilera*" to its nearest neighbours. The entity is from FB15K-237 and its 1st and 100th nearest neighbors are based on TransE embeddings.

n^{th} Neighbor	Neighbor	Similarity Explanation
1^{st}	Katy Perry	$\exists artists^{-}.\{dancepop\}, \exists artists^{-}.\{popmusic\},$ $\exists award.\{GrammyAwardforBestFemalePopVocalPerformance\},$ $\exists award.\{MTVVideoMusicAwardoftheYear\}$ $\exists award.\{MTVVideoMusicAwardforBestNewArtist\},$ $\exists award.\{(MTVVideoMusicAwardForBestFemaleVideo\},$ $\exists netWorthCurrency.\{USdollar\}, \exists gender.\{femaleorganism\},$ $\exists profession.\{actor\}, \exists vacationer^{-}, \exists participant^{-},$ $\exists awardWinner^{-}, \exists person^{-}, \exists artist^{-}, \exists award,$ $\exists specialPerformanceType, \exists awardNominee, \exists profession,$ $\exists origin, \exists film, \exists participant, \exists netWorthCurrency,$ $\exists religion, \exists nationality, \exists R^{-}.\{poprock\},$ $\exists R^{-}.\{popmusic\}, \exists R^{-}.\{dancepop\}, = 43.R^{-},$ $\exists R.\{MTVVideoMusicAwardforVideooftheYear\},$ $\exists R.\{MTVVideoMusicAwardforBestFemaleVideo\},$ $\exists R.\{MTVVideoMusicAwardforBestNewArtist\},$ $\exists R.\{MTVVideoMusicAwardforBestPopVideo\},$ $\exists R.\{MTVVideoMusicAwardforBestArtDirection\},$ $\exists R.\{GrammyAwardforBestFemalePopVocalPerformance\},$ $\exists R.\{femaleorganism\}, \exists R.\{actor\}, \exists R.\{USdollar\}, \exists R.\{friend\},$
100^{th}	Ice Cube	$\exists profession.\{recordproducer\}, \exists languages.\{English\},$ $\exists netWorthCurrency.\{USdollar\}, \exists artists^{-},$ $\exists awardNominee^{-}, \exists awardWinner^{-}, \exists profession,$ $\exists film, \exists gender, \exists languages, \exists award, \exists netWorthCurrency,$ $\exists nationality, \exists religion, \exists location, \exists awardNominee,$ $\exists awardWinner, \exists R.\{USdollar\}, \exists R.\{recordproducer\}, \exists R.\{English\}$

focus on explaining the *outcome* of an embedding model rather than generating interpretable feature-based representations of entities that would mimic the behavior of embeddings on downstream tasks as we do. The recent work [12] is aligned with our idea of representing entities using a fixed-size entity vocabulary, but the respective model targets a different task of making embedding models space efficient, and in contrast to our entity representations, the representations generated by NodePiece are not interpretable.

In contrast, the works [24,35] generate KG embeddings that are interpretable by construction. Their feature-driven entity encodings are similar to ours, but these methods do not approximate existing black box models as we do.

Interpretable Propositionalization-Based Embeddings. Propositionalization, the task of constructing table-based representations, has been proposed for relational data [19]. It has also been studied in the context of knowledge graphs [21,29–31,35]. While these existing propositionalization methods are interpretable by nature, they were developed and used as an alternative to black box KG embedding models, rather than for explaining existing KGE embedding models. Thus, these methods complement our work, and any interpretable propositionalization method can be incorporated during the feature construction step of our approach. To the best of our knowledge, none of the existing propositionalization methods targeted the problem of approximating the behavior of a given embedding model, which is our main focus.

6 Conclusion

Knowledge Graph embedding models are widely used for various tasks, but their numeric vector representations are not interpretable. Even with KG embedding models that create embeddings from interpretable features like NodePiece [12] and Snore [24], an extra embedding layer that is not reversible is used to turn the feature vectors into numerical vectors that are not interpretable. To address this issue, we have presented a feature selection-based approach to explain the behavior of knowledge graph embedding models. The results demonstrate that the proposed approach is effective in identifying the most important features for a given embedding model by finding the alignment between feature-based entity representations and the embedding-based entity representations. The findings reveal that KG embeddings do not capture all the information in the neighborhood of entities and that the feature selection process can discard up to 86% of the features without sacrificing the F1 alignment score. Furthermore, our method can be used to explain similarities of entities in the embedding space. We believe that our work offers interesting perspectives for debugging embedding models and makes a further step towards revealing relations between embeddings and propositionalization methods following [21].

While the presented framework provides insights into the behavior of embeddings, it still could be further extended by accounting for embeddings of relations along with entity embeddings, more complex features like longer paths or trees as well as textual and numerical attributes. Despite the advantages of propositionalization for interpretable feature vector construction, it also has limitations. The size of the feature vector naturally scales with the knowledge graph size. Additionally, for larger KGs, the search space for the feature selection algorithm increases, which may lead to scalability issues. Thus, for ongoing and future work, we plan to integrate embeddings of relations into our method and exploit the information coming from the embeddings already at the feature generation stage. This can be achieved by relying on frequent pattern mining methods or rule learning approaches [20,23], especially those that already account for embeddings during rule construction [16,45]. We also plan to identify ways for concise representation of feature vectors to adapt for large KGs.

Last but not least, as another ongoing and future work direction we are analyzing the performance of our method on other interpretable embeddings.

Supplemental Material Statement: Source code, datasets and a version of the paper with appendix are available for reproducing the results.[1]

Acknowledgements. We thank anonymous reviews for their useful feedback, and Dr. Blaž Škrlj for helpful comments on the initial version of this work.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: ISWC/ASWC, pp. 722–735 (2007)
2. Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D., et al.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
3. Betz, P., Meilicke, C., Stuckenschmidt, H.: Adversarial explanations for knowledge graph embeddings. In: Raedt, L.D. (ed.) IJCAI 2022, pp. 2820–2826 (2022)
4. Bhatt, U., Xiang, A., Shubham Sharma, t.: Explainable machine learning in deployment. In: FAT* 2020, pp. 648–657 (2020)
5. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD 2008, pp. 1247–1250 (2008)
6. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NeurIPs, pp. 2787–2795 (2013)
7. Chandrahas, Sengupta, T., Pragadeesh, C., Talukdar, P.P.: Inducing interpretability in knowledge graph embeddings. In: Bhattacharyya, P., Sharma, D.M., Sangal, R. (eds.) ICON 2020, pp. 70–75 (2020)
8. Cheng, W., Kasneci, G., Graepel, T., Stern, D.H., Herbrich, R.: Automated feature generation from structured knowledge. In: CIKM, 2011, pp. 1395–1404 (2011)
9. Costabello, L., et al.: On explainable AI: from theory to motivation, applications and limitations. In: A Tutorial at AAAI 2019 (2019)
10. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
11. Galárraga, L.: Effects of locality and rule language on explanations for knowledge graph embeddings. CoRR abs/2302.06967 (2023)
12. Galkin, M., Denis, E.G., Wu, J., Hamilton, W.L.: NodePiece: compositional and parameter-efficient representations of large knowledge graphs. In: ICLR 2022 (2022)
13. Gusmão, A.C., Correia, A.H.C., Bona, G.D., Cozman, F.G.: Interpreting embedding models of knowledge bases: a pedagogical approach. CoRR abs/1806.09504 (2018)
14. Halliwell, N., Gandon, F., Lécué, F.: User scored evaluation of non-unique explanations for relational graph convolutional network link prediction on knowledge graphs. In: K-CAP, pp. 57–64 (2021)
15. Haykin, S.: Neural Networks: A Comprehensive Foundation (1994)
16. Ho, V.T., Stepanova, D., Gad-Elrab, M.H., Kharlamov, E., Weikum, G.: Rule Learning from Knowledge Graphs Guided by Embedding Models. In: Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M.C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.-A., Simperl, E. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 72–90. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_5

[1] https://shorturl.at/lGV49.

17. Jain, N., Kalo, J.C., Balke, W.T., Krestel, R.: Do embeddings actually capture knowledge graph semantics? In: ESWC (2021)
18. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR 2017 (2017)
19. Krogel, M., Rawles, S.A., Zelezný, F., Flach, P.A., Lavrac, N., Wrobel, S.: In: ILP 2003, vol. 2835, pp. 197–214 (2003)
20. Lajus, J., Galárraga, L., Suchanek, F.: Fast and exact rule mining with AMIE 3. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 36–52. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_3
21. Lavrač, N., Škrlj, B., Robnik-Šikonja, M.: Propositionalization and embeddings: two sides of the same coin. Mach. Learn. **109**(7), 1465–1507 (2020)
22. Lawler, I., Sullivan, E.: Model explanation versus model-induced explanation. Found. Sci. **26**, 1049–1074 (2021)
23. Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H.: Anytime bottom-up rule learning for knowledge graph completion. In: IJCAI 2019, pp. 3137–3143 (2019)
24. Mežnar, S., Lavrač, N., Škrlj, B.: Snore: scalable unsupervised learning of symbolic node representations. IEEE Access **8**, 212568–212588 (2020)
25. Nandwani, Y., Gupta, A., Agrawal, A., Chauhan, M.S., Singla, P., Mausam: OXKBC: outcome explanation for factorization based knowledge base completion. In: AKBC 2020 (2020)
26. Nguyen, D.Q., Nguyen, T.D., Nguyen, D.Q., Phung, D.Q.: A novel embedding model for knowledge base completion based on convolutional neural network. In: NAACL (2018)
27. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
28. Pellegrino, M.A., Altabba, A., Garofalo, M., Ristoski, P., Cochez, M.: GEval: a modular and extensible evaluation framework for graph embedding techniques. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 565–582. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_33
29. Portisch, J., Heist, N., Paulheim, H.: Knowledge graph embedding for data mining vs. knowledge graph embedding for link prediction - two sides of the same coin? Semant. Web **13**(3), 399–422 (2022)
30. Ristoski, P., Paulheim, H.: A comparison of propositionalization strategies for creating features from linked open data. In: Tiddi, I., d'Aquin, M., Jay, N. (eds.) LOD Workshop at ECML PKDD 2014 (2014)
31. Ristoski, P., Rosati, J., Noia, T.D., Leone, R.D., Paulheim, H.: RDF2Vec: RDF graph embeddings and their applications. Semant. Web **10**(4), 721–752 (2019)
32. Rossi, A., Firmani, D., Merialdo, P., Teofili, T.: Explaining link prediction systems based on knowledge graph embeddings. In: SIGMOD 2022, pp. 2062–2075 (2022)
33. Segal, M., Xiao, Y.: Multivariate random forests. WIREs Data Min. Knowl. Disc. **1**, 80–87 (2011)
34. Shi, B., Weninger, T.: Open-world knowledge graph completion. ArXiv abs/1711.03438 (2018)
35. Steenwinckel, B., Vandewiele, G., Weyns, M., Agozzino, T., Turck, F.D., Ongenae, F.: INK: knowledge graph embeddings for node classification. Data Min. Knowl. Discov. **36**(2), 620–667 (2022)
36. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A core of semantic knowledge. In: WWW 2007 (2007)

37. Tang, J., et al.: Computational advances of tumor marker selection and sample classification in cancer proteomics. Comput. Struct. Biotechnol. J. **18**, 2012–2025 (2020)
38. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality (2015)
39. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML, pp. 2071–2080 (2016)
40. Vashishth, S., Sanyal, S., Nitin, V., Talukdar, P.P.: Composition-based multi-relational graph convolutional networks. In: ICLR 2020 (2020)
41. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
42. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? CoRR abs/1810.00826 (2018)
43. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR (2015)
44. Yogatama, D., Gillick, D., Lazic, N.: Embedding methods for fine grained entity type classification. In: ACL 2015, pp. 291–296 (2015)
45. Zhang, W., et al.: Iteratively learning embeddings and rules for knowledge graph reasoning. In: WWW 2019, pp. 2366–2377 (2019)
46. Zhao, Y., Zhang, A., Xie, R., Liu, K., Wang, X.: Connecting embeddings for knowledge graph entity typing. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J.R. (eds.) ACL 2020, pp. 6419–6428 (2020)
47. Zhu, Z., Zhang, Z., Xhonneux, L.A.C., Tang, J.: Neural bellman-ford networks: a general graph neural network framework for link prediction. In: NeurIPS 2021, pp. 29476–29490 (2021)