Entity Type Prediction Leveraging Graph Walks and Entity Descriptions

Russa Biswas[1,2*][0000−0002−7421−3389], Jan Portisch[3,4⋆][0000−0001−5420−0663],
Heiko Paulheim[4][0000−0003−4386−8195], Harald Sack[1,2][0000−0001−7069−9804], and
Mehwish Alam[1,2][0000−0002−7867−6612]

[1] FIZ Karlsruhe – Leibniz Institute for Infromation Infrastructure, Germany
firstname.lastname@fiz-karlsruhe.de
[2] Karlsruhe Institute of Technology, Institute AIFB, Germany
[3] SAP SE, Walldorf, Germany
jan.portisch@sap.com
[4] Data and Web Science Group, University of Mannheim, Germany
{jan,heiko}@informatik.uni-mannheim.de

Abstract. The entity type information in Knowledge Graphs (KGs) such as DBpedia, Freebase, etc. is often incomplete due to automated generation or human curation. Entity typing is the task of assigning or inferring the semantic type of an entity in a KG. This paper presents *GRAND*, a novel approach for entity typing leveraging different graph walk strategies in RDF2vec together with textual entity descriptions. RDF2vec first generates graph walks and then uses a language model to obtain embeddings for each node in the graph. This study shows that the walk generation strategy and the embedding model have a significant effect on the performance of the entity typing task. The proposed approach outperforms the baseline approaches on the benchmark datasets DBpedia and FIGER for entity typing in KGs for both fine-grained and coarse-grained classes. The results show that the combination of order-aware RDF2vec variants together with the contextual embeddings of the textual entity descriptions achieve the best results.
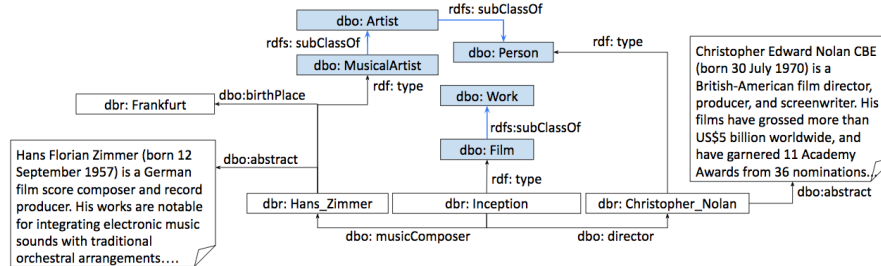
Keywords: Entity Type Prediction · RDF2vec · Knowledge Graph Embedding · Graph Walks · Language Models

1 Introduction

Many efforts have been made towards the automated generation of Knowledge Graphs (KGs) from heterogeneous resources such as text or images. One such effort is the creation of cross-domain KGs such as DBpedia [1], Wikidata [32], Freebase [4], etc. which are either extracted automatically from structured data, generated using heuristics, or are human-curated. This leads to incomplete information in the KGs which can occur on factual level (e.g., missing entities and/or relations between the entities) or on schema level (e.g., the missing entity type

---
* The authors contributed equally to this paper.

Fig. 1. Excerpt from DBpedia

information). For instance, DBpedia version 2016-10 consists of 48 subclasses of *dbo:Person*; however, only 36.6% of the total number of entities belonging to *dbo:Person* are assigned to its subclasses. Moreover, 307,164 entities in the entire DBpedia 2016-10 version are assigned to *owl:Thing*.

To address the KG incompleteness on the factual level, a lot of models [5, 6, 28], etc. have been proposed. These models focus mainly on predicting the missing entities and relations in the KGs but not the entity types. However, the entity type information in KGs plays a vital role in various Natural Language Processing based applications such as question answering [31], relation extraction [10], recommendation, or system [33]. Following these lines, this paper focuses on the problem of entity typing which is the task of assigning or inferring the semantic type of an entity in a KG. Figure 1 shows an excerpt from DBpedia where the class *dbo:MusicalArtist* is a subclass of *dbo:Artist* which is a subclass of *dbo:Person*. *dbo:Artist* and *dbo:MusicalArtist*, respectively, are the fine-grained entity types for *dbr:Hans_Zimmer* and *dbo:Artist* is the missing type information. *dbo:Person* is the coarse-grained type.

Recent years have witnessed a few studies on entity typing approaches in KGs using heuristics [20] and machine learning based classification models [17, 37, 11, 12, 3]. These models predict entity types using different KG features such as the anchor text mentions in the textual entity descriptions, relations between the entities, entity names, and Wikipedia categories. They learn the representation of the entities from their KG structure by using translational models [15], GCN-based models [12], neighborhood based attention models [41] followed by the correlation between the entities and its types. These models exploit the neighborhood information only by the entities directly connected, i.e., the triple information of the entities. However, the large amount of contextual information of the entities captured in the graph walks remains unexplored. The work presented in this paper emphasizes on modeling the KG by taking advantage of the semantics of graph walks to predict the entity types with the help of different kinds of walk generation strategies, such as classic random walks, entity walks, and property walks. The paths generated by these graph walk strategies are used within the RDF2vec model [27] to generate different entity representations. Additionally, the textual entity descriptions in the KGs contain rich semantic

information which is beneficial in predicting the missing entity types. For instance, as depicted in Figure 1, the textual entity descriptions of the entities clearly mentions that *dbr: Christopher_Nolan* is a *director*, *dbr: Hans_Zimmer* is a *music composer*, and *dbr: Inception* is a *film*. Some of the existing baseline models such as MuLR [38] use non-contextual Neural Language Models (NLMs), whereas the other uses GCN model [12] on the words extracted from the entity descriptions. Therefore, to capture the contextual information of the textual entity description contextual NLM, is used to generate entity representations.

This paper presents a framework named **GRAND** (**G**raph Walks for **R**DF2vec **an**d E**n**tity **D**escriptions), which exploits different variants of the RDF2vec model based on different graph walk strategies together with textual entity descriptions to predict the missing entity types in a KG. In this work, the entity typing problem is modelled as a classification problem. A flat and a hierarchical classification model are deployed on the top of the feature vectors generated from the aforementioned entity representations to predict the missing entity types. The empirical results based on the extensive experiments on two benchmark datasets FIGER [37] and DBpedia630k [39] show that the proposed approach is robust and outperforms the state-of-the-art (SOTA) models. Further experiments show that *GRAND* performs considerably well on unseen entities. The main contributions of this work are:

- A framework which leverages different graph walk strategies based RDF2vec models and a contextual NLM for textual entity descriptions is proposed to predict the missing entity types.
- A generalized classification framework consisting of three different modules namely multi-class, multi-label, and hierarchical classification is introduced to predict the missing entity types on different levels of granularity. It can be easily deployed for predicting entity types on entity representations from any KGs.
- Extensive experiments are conducted on the benchmark datasets to study the impact of several combinations of entity representations generated from the RDF2Vec variants and the NLM. An analysis on the weights in the classification has been conducted for analyzing which entity representations are suitable in which entity typing situations. Furthermore, the impact of dimensionality reduction of the entity representations on the local and global level using Principle Component Analysis (PCA) is studied.

The rest of the paper is organized as follows: Section 2 gives an overview of the baseline approaches. Section 3 describes the proposed methodology, followed by experiments and results in Section 4. Finally, Section 5 provides the conclusion and an outlook of future work.

2   Related Work

This section discusses existing literature on entity typing and categorizes them based on their underlying methodology such as heuristics-based methods or machine learning based methods.

SDType [20] is a statistical heuristic model that exploits links between instances using weighted voting. The model is based on the assumption that certain relations occur only with particular types. SDType often does not perform well if two or more classes share the same sets of properties and also if specific relations are missing for the entities.

One of the recent models, Cat2Type [3], takes into account the semantics underlying the textual information in the Wikipedia categories using language models such as BERT. In order to consider the structural information of Wikipedia categories, a category-category network is generated which is then fed to Node2Vec for obtaining the category embeddings. The embeddings of both structural and textual information are combined for classifying entities into their types. In [2], different word embedding models, trained on triples, are leveraged together with a classification model to predict the entity types. Therefore, contextual information is not captured. In CUTE [36], a hierarchical classification model has been proposed which helps in cross-lingual entity typing by exploiting category, property, and property-value pairs. Another model has been proposed in [17] which performs type prediction using the Scalable Local Classifier per Node (SLCN) algorithm based on a set of incoming and outgoing relations. However, the entities with few relations are likely to be misclassified. MuLR [38] learns multi-level representations of entities via character, word, and entity embeddings followed by the hierarchical multi-label classification. Another model, namely FIGMENT [37], uses a global model and a context model. The global model predicts entity types based on the entity mentions from the corpus and the entity names. The context model calculates a score for each context of an entity and assigns it to a type. Therefore, it requires a large annotated corpus which is a drawback of the model. In APE [11], a partially labeled attribute entity-entity network is constructed containing structural, attribute, and type information for entities followed by deep neural networks to learn the entity embeddings. MRGCN [35] is a multi-modal message-passing network that learns end-to-end from the structure of KGs as well as from multimodal node features. In HMGCN [12], the authors propose a GCN-based model to predict the entity types considering the relations, textual entity descriptions, and the Wikipedia categories. ConnectE [40] and AttET [41] models find correlation between neighborhood entities to predict the missing types. However, unlike GRAND, these two models do not look for information far away from the source entity. They work on the principal of L2 distance in their embedding space to detect the types and therefore not compared with the proposed model. Also, in order to employ the hidden layer as latent features for entity representation, restricted Boltzman machines (RBMs) are used to learn a target distribution across the usage of relations of entities [34].

3   Entity Type Prediction: GRAND framework

An overview of the GRAND framework is illustrated in Figure 2. Component (A) represents the RDF2vec variants that use the different strategies for generating
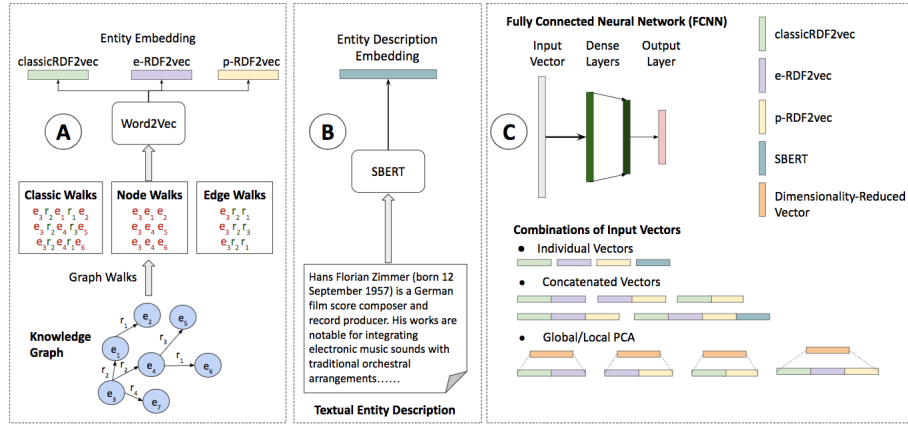
Fig. 2. Architecture of the GRAND framework

graph walks, i.e., classic walks, node walks, and property walks. Component (B) generates the representations of the entities from the textual entity description by using SBERT. Finally, component (C) shows combinations of the variants of entity representations used for flat as well as hierarchical classification. The rest of the section contains the explanation of the component details.

Preliminaries. We define a knowledge graph $\mathcal{G}$ as a labeled directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ for a set of relations $\mathcal{R}$. Vertices are subsequently also referred to as *entities* and edges as *predicates*.

3.1   Entity Representation

RDF2vec [27] is one of the first approaches to adopt statistical language modeling techniques to KGs. The key idea of RDF2vec is a two-step approach: first, random walks over the graph are executed, thereby collecting sequences of entities and relations. To employ language modeling techniques, these sequences are then considered as sentences where each entity and relation in the sequence are treated as words. In RDF2vec, those sentences are then processed by word2vec [19, 18], where both variants of word2vec, i.e., continuous bag of words (CBOW) and skip-gram (SG), are possible.

One limitation of the word2vec algorithm is that it is not aware of the word order. For instance, for a window size of 4, the sentences "John ate a pizza" and "pizza ate a John" are equivalent. This is also the case with *RDF2vec*: For instance, the statements `<Severus> <loves> <Lily>` and `<Lily> <loves> <Severus>`, are considered equivalent even though `<loves>` is not a symmetric property. To overcome this limitation, an order-aware version of RDF2vec has been proposed [24] which has shown improved performance on multiple machine learning datasets. This order-aware variant of RDF2vec uses a *structured word2vec* model [16] which incorporates the positional information of the words

in a sentence. The main advantage of the order-aware RDF2vec model over the classical RDF2vec model is that it respects the positional information of the entities and relations in the random walks, thereby learning embeddings which are better in terms of type separation.

Another type of RDF2vec extension is to explore different strategies for performing graph walks. These strategies have been explored using either variants of random walks (e.g., community hops [14], walklets [21], or hierarchical walks [29]), or by combining different random walk strategies, as the *ontowalk2vec* approach, which combines RDF2vec and node2vec walks [8]. In this paper, the aforementioned order-aware as well as different RDF2vec graph walk strategies [25] are leveraged to predict the missing types of the entities.

Graph Walk Generation Strategies. RDF2vec combines the notion of similarity and relatedness. This can be easily observed when printing the most related concepts for "Berlin" on DBpedia via KGvec2go [22], i.e., many people who are *related* to the city are identified as politicians. However, those are not really *similar* – they do not share properties with Berlin (which is a city rather than a living being). This leads to further exploration of RDF2vec for entity typing.

In this paper, six different RDF2vec configurations are presented and evaluated – stand alone as well as combinations. For the task of entity typing, three different walk generation strategies are applied: (1) classic walks, (2) entity walks, and (3) predicate walks. Each strategy is explained below in more detail.

Classic Walks. The originally presented RDF2vec variant generates multiple random walks for each node in the graph. A random walk of length n (where n is an even number) is of the form

$$w = (w_{-\frac{n}{2}}, w_{-\frac{n}{2}+1}, ..., w_0, ..., w_{\frac{n}{2}-1}, w_{\frac{n}{2}}) \tag{1}$$

where $w_i \in \mathcal{V}$ if i is even, and $w_i \in \mathcal{R}$ if i is odd. For better readability, we stylize $w_i \in \mathcal{V}$ as e_i and $w_i \in \mathcal{R}$ as p_i:

$$w = (e_{-\frac{n}{2}}, p_{-\frac{n}{2}+1}, ..., e_0, ..., p_{\frac{n}{2}-1}, e_{\frac{n}{2}}) \tag{2}$$

Entity Walks (e-RDF2vec). An entity walk contains only entities without any other properties. Such an approach is also known as *e-RDF2vec*, given by

$$w_e = (e_{-\frac{n}{2}}, e_{-\frac{n}{2}+2}, ..., e_0, ..., e_{\frac{n}{2}-2}, e_{\frac{n}{2}}) \tag{3}$$

For an entity walk, all elements are entities, i.e., $w_{n_i} \in \mathcal{V}$.

Predicate Walks (p-RDF2vec). A predicate walk contains only one entity together with object properties known as *p-RDF2vec* and is defined as:

$$w_p = (p_{-\frac{n}{2}+1}, p_{-\frac{n}{2}+3}, ..., e_0, ..., p_{\frac{n}{2}-3}, p_{\frac{n}{2}-1}) \tag{4}$$

The different walk strategies are visualized in component (**A**) in Figure 1.

Generating Entity Embeddings using RDF2Vec variants. An embedding model is trained for each set of walks using word2vec [19, 18] and position-aware word2vec [16] (suffix *oa* in the following) which yields six sets of embeddings:

(1) Classic RDF2vec, (2) e-RDF2vec, (3) p-RDF2vec, (4) Classic RDF2vec$_{oa}$, (5) e-RDF2vec$_{oa}$, and (6) p-RDF2vec$_{oa}$. The proposed model, GRAND, is evaluated by using the configurations presented in 3.1 on their own as well as in a fused way. Concerning the fusion of vectors, three modes are employed: (1) Vector concatenation, (2) Local PCA (LPCA), and (3) Global PCA (GPCA). PCA is a technique for reducing the dimensionality of the vectors with minimal loss in encoded information. It is used for identification of a smaller number of uncorrelated variables known as principal components. The difference between (2) and (3) is that in the case of the LPCA, a principal component analysis is only performed for the subset of vectors that appear in the datasets (see Section 4) whereas for the GPCA, one all vectors generated from the KG using RDF2vec variants are considered. Each of these configurations can be used as vector within GRAND (see component (C) in Figure 2).

The main advantages of using different RDF2vec variants are: **(i)** With a growing length of walks and training window, they can take advantage of large entity context ranges by effectively treating every entity as being connected to all the others in the graph – this is in contrast to the baseline models which are based on local aggregation, i.e. they learn the representation of each entity based on its adjacent entities in the KG [12, 41]. **(ii)** The graph walk strategies are effective, robust, and equitable, i.e., all relations and nodes are given equal importance in generating the embeddings. **(iii)** The walk strategies put emphasis on certain semantic aspects – namely *relatedness* and *similarity* [25]. **(iv)** RDF2vec is a very scalable embedding algorithm. **(v)** The experimental results from [42] show RDF2vec works better on the separability task compared to the other embedding models. The separability task aims at measuring if embeddings from different classes can be linearly separable and the evaluation is done on 10,000 pairs of classes from DBpedia. **(vi)** Any classification algorithm can be deployed on top of entity embeddings to predict the missing types.

3.2   Entity Description Representation

The textual descriptions of an entity provide rich semantic information. Sentence-BERT (SBERT) [26] fine-tunes the BERT [7] model using the siamese and triplet networks to update the weights such that the resulting sentence embeddings are semantically meaningful and semantically similar sentences are closely positioned in the embedding space. For one epoch, a 3-way softmax classifier objective function is used for the fine-tuning of the BERT model. In the training phase of SBERT, two input sentences are passed through the BERT model followed by a pooling layer namely, MEAN-strategy, and MAX-strategy. A fixed-size representation for the input sentences are generated by this pooling layer. Next, they are concatenated with the element-wise difference and multiplied with a trainable weight. The cross-entropy loss is used for optimization. In order to encode the semantics, the twin network is fine-tuned on Semantic Textual Similarity data. SBERT model follows a two-step process in which it is first trained on Wikipedia via BERT and then fine-tuned on Natural Language Inference (NLI) data. NLI

is a collection of 1,000,000 sentence pairs created by combining The Stanford Natural Language Inference (SNLI) and Multi-Genre NLI (MG.NLI) datasets.

In this work, the same approach is followed to extract the embedding of the textual entity descriptions as mentioned in the evaluation of the quality of sentence embeddings in [26]. Given be a textual entity description D_{e_i} denoted by a sequence of words $\{W_1, W_2, ..., W_n\}$, where W_j is the jth word in the entity description, and e_i is the corresponding entity. The entity description D_{e_i} is considered as a single sequence of words which is provided as an input to the SBERT model to get the embedding of the textual entity description $\mathbf{E_{D_i}}$. The pre-trained SBERT model used in GRAND is the SBERT-SNLI-STS-base model which is fine tuned on SNLI and STS datasets which outperforms the baseline models as shown in[26]. The MEAN pooling strategy is used in the pooling layer.

The main advantages of using pre-trained SBERT model are: **(i)** Since the pre-trained SBERT model is fine-tuned with two different datasets, the entity description embeddings obtained lose domain-specific knowledge and bias, and learn task-agnostic properties of the language. **(ii)** Unlike static word embedding models, such as word2vec, the contextual embedding model SBERT encodes semantics of the words differently based on different contexts. Therefore, the entity description embeddings capture the contextual information for the task of entity typing unlike the baseline models [12, 38] **(iii)** They are computationally inexpensive as the model is pre-trained on huge amount of text and can be easily fine-tuned based on the information available. **(iv)** A representation of the entities can be obtained from the textual entity description for long-tailed entities in the KG, i.e., entities with no or few properties. **(v)** A task-specific classification model can be deployed on top of the entity description embeddings for entity typing task as illustrated in the proposed GRAND framework.

3.3   Entity Type Prediction

GRAND consists of three different classification modules: **(1)** Multi-class, **(2)** Multi-label, and **(3)** Hierarchical, that are discussion below.

Entity Representation. The aforementioned approaches generate entity embeddings from various RDF2vec variants and from the contextual embedding model SBERT, which are provided as input to the classification modules. The input entity vectors are generated by concatenating the different vectors generated by the embedding models as depicted in component (C) in Figure 2.

Classifiers. For ***multi-class classification***, a Fully Connected Neural Network (FCNN) consisting of two dense layers with ReLU as an activation function is deployed on the top of the entity representation. A softmax classifier with a cross-entropy loss function is used in the last layer to calculate the probability of the entities belonging to different classes. Formally it is given by,

$$f(s)_i = \frac{e^{s_i}}{\sum_j^{C_T} e^{s_j}}, \quad \text{and} \quad CE_{loss} = -\sum_i^{C_T} t_i log(f(s)_i), \tag{5}$$

where s_j are the scores inferred for each class in C_T given in Equation 5. t_i and s_i are the ground truth and the score for each class in C, respectively.

In **multi-label classification**, an entity can belong to more than one class or type. Therefore, a certain entity e_i belonging to one class c_i has no impact on the decision of it belonging to another class c_j, where $c_i, c_j \in C_T$. A FCNN with RELU as an activation function is used for the two dense layers. A sigmoid function with binary cross-entropy loss is used in the last layer which sets up a binary classification problem for each class in C_T and is given by,

$$CE_{loss} = -t_i log(f(s_i)) - (1 - t_i) log(1 - f(s_i)), \tag{6}$$

where s_i and t_i are the score and ground truth for i^{th} class in C_T.

Hierarchical Classification can be broadly categorized into local and global classification. The local information in local classifier can be utilized in different ways leading to different types of local classifiers such as Local classifier Per Node (LPN), a Local classifier Per Parent Node (LPPN) and a Local classifier Per Level (LPL) [13]. The proposed framework GRAND uses LPL which consists of training a flat classifier for each level of the class hierarchy. A multi-class classifier is trained at each level of the class hierarchy is used to discriminate among the classes at that level. The two main advantages of the LPL model are: **(i)** It is computationally efficient compared to LPN for large KGs consisting of large number of classes as LPN model would have equal number of classifiers. The number of classifiers in LPL are restricted to the number of levels in the class hierarchy. **(ii)** Since a single classifier is trained at each level, it reduces the horizontal class prediction inconsistencies. In GRAND, a two-layered FCNN with ReLU activation function and cross-entropy loss has been deployed at each level of the class hierarchy. However, one of the drawbacks of LPL is that an entity can be classified as class 1 at one level and then it can be again classified as class 2.1 on the second level. Here, class 2.1 is not a subclass of 1 and the entity should be classified to a subclass of 1. In order to tackle such inconsistencies, in this work, the entity which is misclassified as 2.1 in level 2 will be typed as 1 as its entity type as it was correctly identified in level 1.

4   Experiments and Results

This section provides details on the benchmark datasets, experimental setup, analysis of the results obtained, and the ablation study.

Datasets. The two benchmark datasets FIGER [37] and DBpedia630k [39] are used to evaluate the performance of the GRAND framework against the baseline models. DBpedia630k consists of 630,000 entities and 14 non-overlapping classes and FIGER consists of 201,933 entities with 102 classes from Freebase. The entities of the extended DBpedia630k dataset are split equally into three parts DB-1, DB-2, and DB-3, each containing 210,000 entities. Each DBpedia split is divided into a train, test and validation set with 50%, 30%, and 20% of the total entities respectively [12] as well as to 48 classes in the class hierarchy. There are no shared entities between the train, test, and validation sets for all the DBpedia630k splits and in FIGER. FIGER has been extended with triples from

Table 1. Statistics of the datasets

Parameters	DB-1	DB-2	DB-3	FIGER
#Entities	210,000	210,000	210,000	201,933
#Entities train	105,000	105,000	105,000	101,266
#Entities test	63,000	63,000	63,000	60,447
#Entities validation	42,000	42,000	42,000	40,220

DBpedia as explained in [12, 3]. The statistics is provided in Table 1. The code, and data are publicly available[5].

Experimental Setup. The experiments are conducted on six sets of embeddings: (1) Classic RDF2vec, (2) e-RDF2vec, (3) p-RDF2vec, (4) Classic RDF-2vec$_{oa}$, (5) e-RDF2vec$_{oa}$, and (6) p-RDF2vec$_{oa}$. The walks are generated with a depth of 8 and 500 walks per entity. Classic and OA embeddings are trained using SG with 200 dimensions and 5 epochs. For training the order aware variants (4-6), walks from the corresponding non-order aware variants (1-3) are reused. The training was performed using the jRDF2vec framework[6] [23]. All the classifiers are used with the batch size 64, 100 epochs, and adam optmizer. The vectors are publicly available.[7]

Results. In order to evaluate the proposed approach against the baseline models, Micro-averaged F_1 (Mi-F_1) and Macro-averaged F_1 (Ma-F_1) metrics are used along with the accuracy. Different variants of *RDF2vec* have been evaluated which serve as an ablation study. The baselines used for the experiments are: CUTE [36], MuLR [38], FIGMENT [37], APE [11], HMGCN [12], and CAT2Type [3]. The results of the proposed framework on two benchmark datasets and their comparison with the baseline models are depicted in Table 2. The results of GRAND as depicted in Table 2 can be obtained as follows: **(i)** *Coarse-grained setting:* For DBpedia splits, the original dataset consisting of 14 non-overlapping classes is used. For FIGER, the number of coarse-grained classes is 30 and they are non-overlapping as well. Since, none of the entities belong to more than one class, *multi-class* classification settings have been used here. **(ii)** *Fine-grained setting:* The original DBpedia630k dataset is expanded with the DBpedia hierarchy to 37 fine-grained classes and these are non-overlapping classes. Therefore, a *multi-class* classification model is used here as well. On the other hand, the FIGER dataset consists of overlapping fine-grained classes, i.e., one entity can belong to multiple classes. Therefore, a *multi-label* classification is used for fine-grained FIGER dataset. **(iii)** For *Hierarchical Classification*, a classifier on each level of the hierarchy is deployed. For DBpedia splits, it is a multi-class classification model and for FIGER it is a multi-label classification model at each level of the hierarchy. The baseline models which use a non-hierarchical classification such as CAT2Type [3] also use a multi-class classification for DBpedia splits and a multi-label one for FIGER dataset. The SOTA model for hierarchical classification HMGCN [12] uses multi-label classification model. The results show that GRAND outperforms the SOTA model CAT2Type with an improvement of 0.8% on Ma-F_1 and 0.7% on Mi-F_1 for DB-1, 0.7%

Table 2. Results of GRAND on benchmark datasets. The best result of each mode is printed in bold, the runner-up is underlined.

	Model	DB-1		DB2		DB3		FIGER	
		Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1
Baselines	CUTE [36]	0.679	0.702	0.681	0.713	0.685	0.717	0.743	0.782
	MuLR [38]	0.748	0.771	0.757	0.784	0.752	0.775	0.776	0.812
	FIGMENT [37]	0.740	0.766	0.738	0.765	0.745	0.769	0.785	0.819
	APE [11]	0.758	0.784	0.761	0.785	0.760	0.782	0.722	0.756
	HMGCN-no hier [12]	0.785	0.812	0.794	0.820	0.791	0.817	0.789	0.827
	CAT2Type-BERT [3]	0.983	0.984	0.983	0.983	0.985	0.985	0.764	0.881
GRAND Coarse-grained	classic-RDF2vec$_{oa}$ ⊕ s-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ SBERT	**0.991**	**0.991**	**0.990**	**0.990**	**0.989**	**0.989**	**0.801**	**0.893**
	SBERT - only	0.972	0.972	0.97	0.97	0.97	0.97	0.648	0.844
Baselines Fine-grained	CAT2Type-BERT [3]	0.402	0.732	0.369	0.721	0.847	0.915	0.703	0.835
	CAT2Type-node2vec [3]	0.391	0.694	0.365	0.677	0.807	0.878	0.701	0.833
GRAND Fine-grained	classic-RDF2vec$_{oa}$ ⊕ s-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ SBERT	**0.745**	**0.870**	**0.723**	**0.851**	**0.880**	**0.931**	**0.706**	**0.881**
Baseline Hierarchical	HMGCN-hier [12]	**0.794**	0.816	**0.796**	0.824	**0.798**	0.819	**0.798**	0.836
GRAND Hierarchical	classic-RDF2vec$_{oa}$ ⊕ s-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	<u>0.731</u>	**0.882**	<u>0.729</u>	**0.881**	0.726	0.877	0.701	<u>0.880</u>
	classic-RDF2vec$_{oa}$ ⊕ s-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ SBERT	<u>0.731</u>	0.875	0.718	0.869	**0.935**	**0.946**	<u>0.712</u>	**0.883**

and 0.4% on both the metrics for DB-2 and DB-3 respectively for the coarse-grained classes. The original dataset with 14 classes which do not contain the hierarchy is used for this coarse-grained non-hierarchical variant. Furthermore, for hierarchical classification, the proposed model significantly outperforms the SOTA HMGCN-hier model with an increment of 6.6% for DB-1, 5.7% for DB-2, and 12.7% for DB-3 on the $Mi\text{-}F_1$ measure. For FIGER, the coarse-grained approach is a multi-class classification whereas the fine-grained approach is a multi-label classification. GRAND achieves the best results for FIGER on the coarse-grained approach which outperforms the baseline models. Moreover, with the multi-label fine-grained settings it achieves comparable results with the non-hierarchical baseline model CAT2Type and significantly outperforms the other non-hierarchical model HMGCN. One advantage of GRAND over CAT2Type is that it can be applied to any KGs and is not restricted to KGs containing information on Wikipedia Categories. Table 3 and Table 4 show the experimental results of the proposed approach for the coarse-grained and fine-grained classes respectively with different variants of RDF2vec and their combinations. Experiments using *Single* strategy show that *all* order-aware RDF2vec embeddings significantly outperform their classic counterparts. Hence, the fusion strategies only focus on position-aware embeddings reducing the combinatorial complexity.

Impact of RDF2vec variants on Coarse-Grained Entity Typing. Table 3 shows the results of the experiment for coarse-grained entity typing. On the DB1 Split of the dataset, the best results for GRAND are obtained where the models are combined, i.e., $classic\text{-}RDF2vec_{oa} \oplus p\text{-}RDF2vec_{oa} \oplus e\text{-}RDF2vec_{oa}$ (*concat*) outperforms $HMGCN$ for $Ma\text{-}F_1$ by 0.1744 and for $Mi\text{-}F_1$ by 0.148 and achieves

comparable results with CAT2Type. However, e-RDF2vec configurations perform the weakest on their own but introduces additional value when combined with other approaches as depicted in the concat model. The best performing configuration includes the entity embeddings. Given the data, it appears that the PCA discards too much valuable information for DBpedia splits but not for FIGER. Overall, it can be observed that the performance differences between p-RDF2vec and classic-RDF2vec are minor. Nonetheless, the embeddings encode different information which is visible when combining the embeddings. Therefore, it can be concluded that the contextual information of the entities in form of path captures the characteristics features of the entities. Similar observation has been made for both DB2, DB3 split and FIGER. A detailed analysis of the impact of different vector components is provided in Section 4.

Impact of RDF2vec variants on Fine-Grained Entity Typing. GRAND is compared with the two best variants of CAT2Type namely BERT and node2vec as shown in Table 2 and results show that the proposed model significantly outperforms the CAT2Type model for all DBpedia splits and FIGER. In general, it is observed for uneven class distribution the evaluation metric $Ma\text{-}F_1$ achieves lower values compared to $Mi\text{-}F_1$. However, the $Ma\text{-}F_1$ results of GRAND for DB1 and DB2 splits are much better than that of CAT2Type. It strengthens the fact that the representation of entities obtained using strategic graph walks and contextual embedding of entity descriptions contain more information about entities compared to the embeddings used in CAT2Type.

Impact of RDF2vec on Hierarchical classification. Table 5 shows the results of the hierarchical classification of the GRAND framework on different levels of the class hierarchy. The performance is computed for only *classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ e-RDF2vec$_{oa}$* since it is the highest performing model based on experiments discussed in previous sections. The results show higher performances on level 1 since the number of classes is lesser i.e., 5, as compared to other levels. GRAND outperforms the baseline model *HMGCN-withHier* for $Mi\text{-}F_1$ metric as depicted in Table 2.

Impact of Textual Entity Descriptions To analyze the impact of entity descriptions, a multi-class classification was performed on the entity embeddings generated from the SBERT model. As shown in Table 2, GRAND with only SBERT performs better than all the baseline models except CAT2Type. Therefore, it can be concluded that contextual embeddings using SBERT provide the necessary relevant information as compared to the triple-based baseline models.

Analysis of Vector Component Weight. In the experiments, it can be seen that the concatenation of embeddings achieves the best result. Therefore, it is further evaluated (1) which components are the most and the least important for the predictions and (2) whether there is a difference in the weights given the coarse-grained and the fine-grained prediction tasks.

Experimental Setup. In order to analyze the weights each vector component receives in the neural network, a FCNN with one layer was trained on the combination of all ordered aware RDF2vec (depicted in 1st 2 rows in coarse-grained and 1st 2 rows in fine-grained in Table 6) and also with SBERT. It is noted that

Table 3. Evaluation of Single Classifier Results on the Coarse-Grained Dataset. The best result of each mode is printed in bold, the runner-up is underlined. The overall best configuration for each dataset is bold and underlined.

Dataset	Mode	Model	DB-1 ACC	Ma-F_1	Mi-F_1	DB-2 ACC	Ma-F_1	Mi-F_1	DB-3 ACC	Ma-F_1	Mi-F_1	FIGER ACC	Ma-F_1	Mi-F_1
Coarse-Grained	Single	classic-RDF2vec	0.9163	0.9150	0.9163	0.9062	0.9043	0.9062	0.9123	0.9109	0.9123	*0.931*	**0.431**	0.778
		classic-RDF2vec$_{oa}$	**0.9448**	**0.9439**	**0.9448**	**0.9346**	**0.9330**	**0.9346**	**0.9457**	**0.9449**	**0.9457**	**0.933**	0.419	**0.781**
		e-RDF2vec	0.7352	0.7318	0.7352	0.7250	0.7308	0.7250	0.7357	0.7304	0.7357	0.927	0.421	0.771
		e-RDF2vec$_{oa}$	0.7665	0.7651	0.7665	0.7625	0.7453	0.7625	0.7694	0.7650	0.7694	0.927	0.422	0.771
		p-RDF2vec	0.8949	0.8946	0.8949	0.8999	0.8914	0.8999	0.8882	0.887	0.8882	0.922	*0.426*	0.778
		p-RDF2vec$_{oa}$	*0.9412*	0.9404	0.9412	0.9332	0.9303	0.9332	0.9430	0.9421	0.9430	0.928	0.422	0.779
	Concat	e-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	0.9518	0.9512	0.9518	0.9482	0.9412	0.9482	0.9502	0.9495	0.9502	0.912	0.414	0.77
		e-RDF2vec$_{oa}$ ⊕ classic-RDF2vec$_{oa}$	0.9450	0.9444	0.9450	0.9450	0.9144	0.9450	0.9452	0.9482	0.9452	0.908	0.418	*0.772*
		classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	*0.9564*	*0.9555*	*0.9563*	*0.9560*	**0.9546**	*0.9560*	**0.9582**	*0.9513*	**0.9592**	0.92	**0.429**	**0.774**
		classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ e-RDF2vec$_{oa}$	**0.9600**	**0.9594**	**0.9600**	**0.9667**	0.9544	**0.9667**	0.9572	**0.9564**	0.9574	**0.924**	*0.424*	0.772
	Local PCA	e-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	0.8855	0.8845	0.8855	0.8757	0.8770	0.8757	0.8918	0.8905	0.8918	*0.921*	*0.422*	0.769
		e-RDF2vec$_{oa}$ ⊕ classic-RDF2vec$_{oa}$	0.9323	0.9314	0.9324	0.9314	0.9122	0.9314	0.9015	0.9000	0.9015	0.919	0.419	*0.770*
		classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	**0.9471**	**0.9466**	**0.9472**	*0.9442*	**0.9300**	*0.9442*	*0.9378*	*0.9217*	*0.9378*	0.92	*0.421*	0.724
		classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ e-RDF2vec$_{oa}$	*0.9405*	*0.9395*	*0.9405*	**0.9551**	*0.9195*	**0.9551**	**0.9413**	**0.9402**	**0.9413**	**0.925**	**0.428**	**0.778**
	Global PCA	e-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	0.9325	0.9316	0.9325	0.9412	*0.9330*	0.9412	0.9321	0.9310	0.9321	0.923	**0.428**	*0.778*
		e-RDF2vec$_{oa}$ ⊕ classic-RDF2vec$_{oa}$	0.9413	0.9405	0.9414	0.9322	0.9311	0.9322	0.9416	0.9405	0.9416	0.925	*0.428*	0.776
		classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	**0.9499**	**0.9490**	**0.9499**	*0.9356*	0.9212	*0.9356*	**0.9490**	**0.9482**	**0.9490**	*0.927*	0.427	0.767
		classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ e-RDF2vec$_{oa}$	*0.9476*	*0.9468*	*0.9476*	**0.9568**	**0.9412**	**0.9568**	*0.9489*	*0.9481*	*0.9489*	**0.929**	**0.433**	**0.779**

Table 4. Evaluation of Single Classifier Results on the Fine-Grained Dataset. The best result of each mode is printed in bold, the runner-up is underlined. The overall best configuration for each dataset is bold and underlined.

Dataset Mode	Model	DB-1 ACC	DB-1 Ma-F_1	DB-1 Mi-F_1	DB-2 ACC	DB-2 Ma-F_1	DB-2 Mi-F_1	DB-3 ACC	DB-3 Ma-F_1	DB-3 Mi-F_1	FIGER ACC	FIGER Ma-F_1	FIGER Mi-F_1
Single	classic-RDF2vec	0.6716	0.374	0.672	0.6635	0.363	0.663	0.8402	0.736	0.840	0.987	0.467	0.774
	classic-RDF2vec$_{oa}$	0.704	**0.386**	0.704	0.701	0.356	0.701	**0.871**	**0.774**	**0.871**	**0.991**	**0.469**	**0.778**
	e-RDF2vec	0.564	0.297	0.5643	0.5231	0.3164	0.5231	0.6709	0.5632	0.6709	0.946	0.445	0.721
	e-RDF2vec$_{oa}$	0.5831	0.3064	0.5831	0.5542	0.3174	0.5442	0.6926	0.5747	0.6926	0.951	0.452	0.722
	p-RDF2vec	0.6500	0.3549	0.6499	0.6504	0.3449	0.6504	0.7848	0.6513	0.7848	0.949	0.467	0.77
	p-RDF2vec$_{oa}$	**0.706**	**0.384**	**0.706**	**0.702**	**0.381**	**0.7022**	0.847	0.732	0.8471	0.951	0.459	0.772
Concat	e-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	0.699	0.378	0.6996	0.698	0.388	0.698	0.877	0.784	0.877	0.949	0.454	0.774
	classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	0.698	0.374	0.6978	0.701	0.384	0.7011	0.881	0.7811	0.881	0.96	0.512	0.781
	e-RDF2vec$_{oa}$ ⊕ classic-RDF2vec$_{oa}$	**0.707**	0.386	**0.707**	0.719	**0.396**	0.719	0.887	0.781	0.881	0.955	0.501	0.778
	classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ e-RDF2vec$_{oa}$	0.653	0.358	0.6538	0.648	0.385	0.648	0.806	0.695	0.8060	0.948	0.457	0.778
Local PCA	e-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	0.703	**0.393**	0.720	**0.7204**	0.3912	0.720	**0.890**	**0.801**	**0.8908**	**0.961**	**0.519**	**0.783**
	classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	0.6865	0.3683	0.6865	0.6952	0.3682	0.6952	0.8746	0.7770	0.8746	0.951	0.501	0.779
	e-RDF2vec$_{oa}$ ⊕ classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	**0.7006**	**0.3902**	**0.7006**	0.7116	**0.3907**	0.7116	0.8774	0.7801	0.8774	0.950	0.504	0.771
	classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ e-RDF2vec$_{oa}$	0.6936	0.3839	0.6936	**0.7122**	0.3438	**0.7123**	0.864	0.764	0.864	**0.958**	0.514	0.781
Global PCA	e-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	0.6845	0.3716	0.6844	0.66125	0.3189	0.6612	0.855	0.7525	0.8547	0.942	0.449	0.772
	e-RDF2vec$_{oa}$ ⊕ classic-RDF2vec$_{oa}$	0.6908	**0.3879**	0.6908	0.67143	0.3119	0.67143	0.8677	0.7686	0.8677	0.945	0.449	0.769
	classic-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	0.6981	0.3778	0.6981	0.6881	0.3241	0.6881	**0.8754**	0.7771	**0.8754**	0.956	0.457	0.771
	p-RDF2vec$_{oa}$ ⊕ e-RDF2vec$_{oa}$	**0.7005**	0.3768	**0.7004**	**0.7014**	**0.3228**	**0.7014**	0.8709	**0.7780**	0.8709	**0.961**	**0.498**	**0.784**

Table 5. Results of the GRAND-LPL classification model at each level

Level	#classes	DB1		DB2		DB3	
		$Ma\text{-}F_1$	$Mi\text{-}F_1$	$Ma\text{-}F_1$	$Mi\text{-}F_1$	$Ma\text{-}F_1$	$Mi\text{-}F_1$
1	5	**0.961**	**0.962**	**0.960**	**0.960**	**0.959**	**0.959**
2	11	0.744	0.925	0.747	0.929	0.744	0.924
3	12	0.857	0.934	0.851	0.926	0.859	0.935
4	17	0.361	0.705	0.358	0.702	0.359	0.674

Table 6. Relative network weights of each vector component group for DB-1 split.

Dataset	Epoch	SBERT	Classic RDF2vec$_{oa}$	p-RDF2vec$_{oa}$	e-RDF2vec$_{oa}$
Coarse-Grained	1	-	35.5%	44.4%	20.0%
	10	-	32.9%	49.9%	17.1%
	1	58.04%	14.6%	16.28%	11.08%
	10	47.9%	18.5%	22.8%	10.8%
Fine-Grained	1	-	35.4%	42.1%	22.5%
	10	-	33.6%	46.4%	20.0%
	1	56.7%	15.36%	16.84%	11.1%
	10	51.19%	16.83%	19.5%	12.48%

the overall goal of this setup is to analyze how much weight each of the four vector groups receive. Therefore, the sum of absolute weights in the network given to each vector is calculated for the first, and the tenth epoch.

Results. The relative weights can be found in Table 6. It is observed that the highest overall impact is independent of the dataset, achieved using the p-RDF2vec embeddings. This is followed by the classic RDF2vec embeddings. The least impact is achieved by the e-RDF2vec embeddings. Interestingly, a weight-shift occurs when switching from the coarse-grained entity typing to fine-grained entity typing, i.e., it is visible that the classic and the entity embeddings are more important for fine-grained predictions. The results suggest that p-RDF2vec is helpful for coarse-grained type prediction – an intuitive finding given that p-RDF2vec encodes structural similarity. However, the more fine-grained the task gets, the more important are the *actual* neighbor vertices.

5   Summary & Future Directions

This paper proposes a novel entity type prediction framework, named **GRAND** based on RDF2vec variants and textual entity descriptions. The variants are constructed by different walk generation strategies and a new order-aware variant of word2vec. GRAND is evaluated on DBpedia630k and FIGER datasets. The results show that GRAND considerably outperforms all the baseline models. Also, given the weight analysis, further experimentation on more fine-granular type systems – such as in YAGO [30] or CaLiGraph [9] is to be conducted.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: Dbpedia: A nucleus for a web of open data. In: Aberer, K., Choi, K., Noy, N.F., Allemang, D., Lee, K., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007, Lecture Notes in Computer Science, vol. 4825, pp. 722–735. Springer (2007). https://doi.org/10.1007/978-3-540-76298-0_52, https://doi.org/10.1007/978-3-540-76298-0_52
2. Biswas, R., Sofronova, R., Alam, M., Sack, H.: Entity type prediction in knowledge graphs using embeddings. arXiv pp. arXiv–2004 (2020)
3. Biswas, R., Sofronova, R., Sack, H., Alam, M.: Cat2type: Wikipedia category embeddings for entity typing in knowledge graphs. In: Gentile, A.L., Gonçalves, R. (eds.) K-CAP '21: Knowledge Capture Conference, Virtual Event, USA, December 2-3, 2021. pp. 81–88. ACM (2021). https://doi.org/10.1145/3460210.3493575, https://doi.org/10.1145/3460210.3493575
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM SIGMOD international conference on Management of data (2008)
5. Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning structured embeddings of knowledge bases. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)
6. Dettmers, T., Pasquale, M., Pontus, S., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Proceedings of the 32th AAAI Conference on Artificial Intelligence (2018)
7. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (2019)
8. Gkotse, B.: Ontology-based Generation of Personalised Data Management Systems: an Application to Experimental Particle Physics. Ph.D. thesis, Université Paris sciences et lettres (2020)
9. Heist, N., Paulheim, H.: Entity extraction from wikipedia list pages. In: Harth, A., Kirrane, S., Ngomo, A.N., Paulheim, H., Rula, A., Gentile, A.L., Haase, P., Cochez, M. (eds.) The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12123, pp. 327–342. Springer (2020). https://doi.org/10.1007/978-3-030-49461-2_19, https://doi.org/10.1007/978-3-030-49461-2_19
10. Jain, P., Kumar, P., Chakrabarti, S., et al.: Type-sensitive knowledge base inference without explicit type supervision. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 75–80 (2018)
11. Jin, H., Hou, L., Li, J., Dong, T.: Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases. In: 27th International Conference on Computational Linguistics (2018)
12. Jin, H., Hou, L., Li, J., Dong, T.: Fine-grained entity typing via hierarchical multi graph convolutional networks. In: Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (2019)

13. Jr., C.N.S., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Min. Knowl. Discov. **22**(1-2), 31–72 (2011). https://doi.org/10.1007/s10618-010-0175-9, https://doi.org/10.1007/s10618-010-0175-9

14. Keikha, M.M., Rahgozar, M., Asadpour, M.: Community aware random walk for network embedding. Knowledge-Based Systems **148**, 47–54 (2018)

15. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Twenty-ninth AAAI conference on artificial intelligence (2015)

16. Ling, W., Dyer, C., Black, A.W., Trancoso, I.: Two/too simple adaptations of word2vec for syntax problems. In: NAACL HLT 2015. pp. 1299–1304. ACL (2015)

17. Melo, A., Paulheim, H., Völker, J.: Type Prediction in RDF Knowledge Bases Using Hierarchical Multilabel Classification. In: WIMS (2016)

18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv:1310.4546 (2013)

20. Paulheim, H., Bizer, C.: Type Inference on Noisy RDF Data. In: ISWC (2013)

21. Perozzi, B., Kulkarni, V., Chen, H., Skiena, S.: Don't walk, skip! online learning of multi-scale network embeddings. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017. pp. 258–265 (2017)

22. Portisch, J., Hladik, M., Paulheim, H.: Kgvec2go - knowledge graph embeddings as a service. In: Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S. (eds.) Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020. pp. 5641–5647. European Language Resources Association (2020), https://aclanthology.org/2020.lrec-1.692/

23. Portisch, J., Hladik, M., Paulheim, H.: Rdf2vec light - A lightweight approachfor knowledge graph embeddings. In: Taylor, K.L., Gonçalves, R.S., Lécué, F., Yan, J. (eds.) Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC). CEUR Workshop Proceedings, vol. 2721, pp. 79–84. CEUR-WS.org (2020), http://ceur-ws.org/Vol-2721/paper520.pdf

24. Portisch, J., Paulheim, H.: Putting rdf2vec in order. In: Seneviratne, O., Pesquita, C., Sequeda, J., Etcheverry, L. (eds.) Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24-28, 2021. CEUR Workshop Proceedings, vol. 2980. CEUR-WS.org (2021), http://ceur-ws.org/Vol-2980/paper352.pdf

25. Portisch, J., Paulheim, H.: Walk this way! entity walks and property walks for rdf2vec. CoRR **abs/2204.02777** (2022)

26. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP (2019)

27. Ristoski, P., Rosati, J., Noia, T.D., Leone, R.D., Paulheim, H.: Rdf2vec: RDF graph embeddings and their applications. Semantic Web **10**(4), 721–752 (2019). https://doi.org/10.3233/SW-180317, https://doi.org/10.3233/SW-180317

28. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Proceedings of the European semantic web conference (2018)

29. Schlötterer, J., Wehking, M., Rizi, F.S., Granitzer, M.: Investigating extensions to random walk based graph embedding. In: 2019 IEEE International Conference on Cognitive Computing (ICCC). pp. 81–89. IEEE (2019)

30. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J. (eds.) Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007. pp. 697–706. ACM (2007). https://doi.org/10.1145/1242572.1242667, https://doi.org/10.1145/1242572.1242667

31. Tong, P., Zhang, Q., Yao, J.: Leveraging domain context for question answering over knowledge graph. Data Science and Engineering (2019)

32. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM (2014)

33. Wang, X., Wang, D., Xu, C., He, X., Cao, Y., Chua, T.S.: Explainable reasoning over knowledge graphs for recommendation. In: Proceedings of the AAAI conference on artificial intelligence (2019)

34. Weller, T., Acosta, M.: Predicting instance type assertions in knowledge graphs using stochastic neural networks. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp. 2111–2118 (2021)

35. Wilcke, W., Bloem, P., de Boer, V., van't Veer, R., van Harmelen, F.: End-to-end entity classification on multimodal knowledge graphs. arXiv (2020)

36. Xu, B., Zhang, Y., Liang, J., Xiao, Y., Hwang, S., Wang, W.: Cross-lingual type inference. In: Database Systems for Advanced Applications - 21st International Conference, DASFAA (2016)

37. Yaghoobzadeh, Y., Adel, H., Schütze, H.: Corpus-level fine-grained entity typing. J. Artif. Intell. Res. (2018)

38. Yaghoobzadeh, Y., Schütze, H.: Multi-level representations for fine-grained typing of knowledge base entities. In: 15th Conference of the European Chapter of the Association for Computational Linguistics (2017)

39. Zhang, X., Zhao, J.J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems (2015)

40. Zhao, Y., Zhang, A., Xie, R., Liu, K., Wang, X.: Connecting embeddings for knowledge graph entity typing. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 6419–6428 (2020)

41. Zhuo, J., Zhu, Q., Yue, Y., Zhao, Y., Han, W.: A neighborhood-attention fine-grained entity typing for knowledge graph completion. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining. pp. 1525–1533 (2022)

42. Zouaq, A., Martel, F.: What is the schema of your knowledge graph? leveraging knowledge graph embeddings and clustering for expressive taxonomy learning. In: Proceedings of the international workshop on semantic big data. pp. 1–6 (2020)