The RDF2vec Family of Knowledge Graph Embedding Methods

An Experimental Evaluation of RDF2vec Variants and their Capabilities

Jan PORTISCH [a,1], Heiko PAULHEIM [b]
[a] *SAP SE, Germany*
[b] *Data and Web Science Group, University of Mannheim, Germany*

Abstract. Knowledge graph embeddings represent a group of machine learning techniques which project entities and relations of a knowledge graph to continuous vector spaces. RDF2vec is a scalable embedding approach rooted in the combination of random walks with a language model. It has been successfully used in various applications. Recently, multiple variants to the RDF2vec approach have been proposed, introducing variations both on the walk generation and on the language modeling side. The combination of those different approaches has lead to an increasing family of RDF2vec variants.

In this paper, we evaluate a total of twelve RDF2vec variants on a comprehensive set of benchmark models, and compare them to seven existing knowledge graph embedding methods from the family of link prediction approaches. Besides the established GEval benchmark introducing various downstream machine learning tasks on the DBpedia knowledge graph, we also use the new DLCC (Description Logic Class Constructors) benchmark consisting of two gold standards, one based on DBpedia, and one based on synthetically generated graphs. The latter allows for analyzing which ontological patterns in a knowledge graph can actually be learned by different embedding.

With this evaluation, we observe that certain tailored RDF2vec variants can lead to improved performance on different downstream tasks, given the nature of the underlying problem, and that they, in particular, have a different behavior in modeling similarity and relatedness. The findings can be used to provide guidance in selecting a particular RDF2vec method for a given task.

Keywords. RDF2vec, knowledge graph embedding, representation learning, embedding evaluation

1. Introduction

RDF2vec [1] is an approach for embedding entities of a knowledge graph in a continuous vector space. It extracts sequences of entities from knowledge graphs,

[1]Corresponding Author. E-mail: jan.portisch@sap.com

which are then fed into a word2vec encoder [2,3]. Such embeddings have been shown to be useful in downstream tasks which require numeric representations of entities and rely on a distance metric between entities that captures entity similarity and/or relatedness [4]. Examples of RDF2vec applications include knowledge graph matching [5,6,7], general machine learning involving named entities [8], entity type prediction [9,10], relation prediction [4], named entity classification [11,12], or information retrieval [13,14].

Since its inception, multiple extensions have been proposed for RDF2vec. In this paper, ~~two recent extensions are further scrutinized:~~ we analyze two recent RDF2vec ~~walk variations~~ extensions in more detail. They concern variations in the walk generation (named e-RDF2vec and p-RDF2vec) ~~and~~ as well as training word2vec in an order-aware ~~RDF2vec~~ fashion (named RDF2vec$_{oa}$). These extensions have been evaluated on their own on task-based datasets before [15,16]. Preliminary evaluations revealed that the flavor that is chosen influences the weight which is put on different (semantic) features – for example, e-RDF2vec spaces are considered to be more focused on relatedness while there is indication that p-RDF2vec spaces cover fine-grained similarity better. This paper presents the first comprehensive evaluation of all combinations of classic, e-RDF2vec, and p-RDF2vec, in their order aware and non-order aware variants.

Moreover, not all of the evaluations in previous papers have been fully conclusive. This poses the question: "What is *actually* learned?" It is not easy to answer this question since task-based evaluation are subjective in nature and blend different *semantic requirements*. This paper strives to achieve a deeper understanding of what knowledge graph embedding methods, such as RDF2vec, are actually capable of representing. To that end, we perform an in-depth comparison of the different variants, as well as a comparison of RDF2vec-based approaches to non RDF2vec-based ones.

While we also perform task-based evaluations with multiple variants of RDF2vec, the evaluation goes beyond single task-based discussions and tries to tackle the question more fundamentally. We use multiple description logic (DL) class constructors [17], which are used to create two benchmarks: One benchmark is based on DBpedia and one benchmark is synthetic in nature. We furthermore formulate hypotheses which of classes can be learned using which embedding method. The two benchmarks – and particularly the comparison of results between them – allow us to evaluate our hypotheses and to determine which DL class constructors are learned by which approach. Furthermore, we analyze whether the DL class constructor is actually learned or whether the approach is merely exploiting cross signals which can be found in the knowledge graphs. In our evaluation, we include not only twelve different RDF2vec configurations but also seven different state of the art embedding models.

This paper makes two main contributions: (1) An in-depth evaluation of multiple RDF2vec configurations including their combinations is performed. (2) In addition, an in-depth evaluation of existing state of the art models on completely novel tasks is run to expose their strengths and weaknesses. To our knowledge, our work is the first attempt to understand what knowledge graph embedding methods can actually represent, both with respect to RDF2vec variants as well as to other embedding methods, and, at the same time, the most comprehensive

evaluation for knowledge graph embeddings in general and RDF2vec variants in particular.

While some results of this paper have already been published [15,16,17], the following contributions are novel:

1. We discuss theoretical hypotheses about the representational power of different RDF2vec based variants and test them with systematic benchmarks.
2. We demonstrate that information on the nature of the task for which embeddings are to be used can help to make an informed decision on an embedding model.
3. We provide a full comparison of twelve RDF2vec variants and seven additional baseline models.

The rest of this article is structured as follows: The following section introduces related work in the field of knowledge graph embeddings and embedding evaluation gold standards. We then discuss RDF2vec extensions in section 3. Subsequently, we introduce a frequently used gold standard for evaluating knowledge graph embeddings through machine learning applications in section 4. In section 5, we introduce a broad set of description logic class constructors whereby we are interested in how far each constructor can be learned by an embedding approach. Together with the constructors, we hypothesize which RDF2vec variant may be able to cover which constructor and why. After constructors and hypotheses are introduced, a set of test cases is required to evaluate the embeddings and to validate our assumptions. Therefore, section 6 introduces a framework which we developed to derive two gold standards, named DLCC (Description Logic Class Constructors). In section 7 we present the obtained results, discuss them, and check the previously posed hypotheses. Lastly, this paper is concluded in section 8 by a summary together with an outlook on future work.

All relevant artifacts (embedding models, gold standards, developed frameworks) are publicly available.[2]

2. Related Work

Knowledge Graph Embeddings A knowledge graph $\mathcal{G}$ is 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*. Such a graph is also referred to as *directed heterogeneous graph* [18,19]. A knowledge graph embedding (KGE) is a projection Π for all vertices $v \in \mathcal{V}$ and optionally $r \in \mathcal{R}$ into a multi-dimensional space of dimension Δ. Hence $\Pi = \{e_i \in \mathbb{R}^\Delta\}$ where $i \in \{1, 2, ..., |\mathcal{V}|\}$ or $i \in \{1, 2, ..., |\mathcal{V}| + |\mathcal{R}|\}$.[3]

Numerous approaches for knowledge graph embeddings were presented in the past and multiple surveys on knowledge graph embeddings were published [4,18, 19,22,23]. Cai et al. [18] distinguish five different techniques for graph embedding:

[2]Instructions on how to reproduce the results in this paper are available online at `http://rdf2vec.org/swj_paper/`

[3]In this paper, the focus lies on deterministic point vector embedding approaches. The notation assumes a real vector space, this is not the case for ComplEx [20] and RotatE [21].

(1) matrix factorization, (2) deep learning, (3) edge reconstruction, (4) graph kernel, and (5) generative model.[4]

A well-known matrix factorization approach is RESCAL [24]. The approach models a graph as a three-way tensor and subsequently applies tensor decomposition. DistMult [20] is a scalability improvement over RESCAL at the cost that relationships are assumed to be symmetric. ComplEx [20] extends DistMult by using complex vector spaces rather than real ones.[5] In this paper, we use all models of the above as benchmark models.

RDF2vec [8] (and all its variants ~~[16,15]~~[15,16]) fall into the category of random walk-based deep learning: Multiple walks are performed within a graph, typically for each node, and the set of walks is then interpreted as sentences by the word2vec language embedding algorithm [2,3]. Conceptually, RDF2vec is similar to node2vec [25] and DeepWalk [26], with the difference that the latter approaches were presented in the context of homogeneous graphs, i.e., graphs with merely one edge type.

TransE [27] is a well-known edge-reconstruction approach which minimizes the margin-based ranking loss. Given a triple in the form *(**h**ead, **r**elation, **t**ail)*, TransE trains embeddings $\mathbf{h}$, $\mathbf{r}$, $\mathbf{t}$, such that $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$. As an extension, TransR [28] learns two embedding spaces, one for entities and one for relations, so that it better captures compositional rules and non-one-to-one cardinalities of relationships. RotatE [21] regards relations as rotations of vertices in complex space.[6] All edge-reconstruction approaches discussed above are used as benchmark models in this paper.

Since graph kernels are designed for embedding a whole graph, this category is not relevant for the article at hand. An example of generative models would be the Latent Dirichlet Allocation applied on graphs. Embedding approaches from this category, however, are not commonly used for knowledge graph embedding applications and are not further discussed in this article.

Knowledge Graph Embedding Evaluation In the area of link prediction (or knowledge base completion), the two well-known evaluation datasets FB15k and WN18 [27] are both based on real datasets: FB15k is based on the Freebase ~~dataset~~knowledge graph [29], and WN18 is based on WordNet [30]. They were presented in the context of link prediction: Given a triple in the form *(**h**ead, **r**elation, **t**ail)*, two prediction tasks *(head, relation, ?)* and *(?, relation, tail)* are created. ~~The evaluation is performed by calculating the mean rank/HITS@10 for a list of proposals.~~ Since it has been remarked that those datasets contain too many simple inferences due to inverse relations, the more challenging variants FB15k-237 [31] and WN18RR [32] have been proposed. More recently, evalua-

[4] Within these categories, even finer categories are presented. In this paper, we will only discuss the main classes and point to subclasses if relevant. For a complete overview of the classification system, we refer the reader to the original publication [18]. While the paper is about graph embedding in general, not knowledge graph embedding in particular, the authors list knowledge graphs as one kind of graphs under consideration for their categorization. Moreover, they do not restrict any category to a particular kind of graph. Therefore, we use this categorization as a categorization for KGE approaches.

[5] Hence, for ComplEx: $\Pi = \{e_i \in \mathbb{C}^{\Delta}\}$ where $i = 1, 2, ...|\mathcal{V}| + |\mathcal{R}|$

[6] Hence, for RotatE: $\Pi = \{e_i \in \mathbb{C}^{\Delta}\}$ where $i = 1, 2, ...|\mathcal{V}| + |\mathcal{R}|$

tion sets based on larger knowledge graphs, such as YAGO3-10 [32] and DBpedia50k/DBpedia500k [33] have been introduced. Typical measures for evaluating link prediction are mean reciprocal rank (MRR) and HITS@k.

Alshagari et al. [34] present a framework for ontological concepts covering three aspects: (i) categorization, (ii) hierarchy, and (iii) logic validation. The framework can be used for language models and for knowledge graph embeddings. The work presented in this paper differs in that it goes beyond explicit DBpedia types. The evaluation of this paper is, therefore, of analytical rather than descriptive nature. Moreover, the task sets of DLCC are significantly larger and more comprehensive.

Ristoski et al. [35] provide a collection of benchmarking datasets for machine learning including classification, clustering, and regression tasks. Later, the GEval framework [36,37] was introduced to provide a standardized evaluation protocol for this dataset. The evaluation datasets are based on DBpedia. Internally, the embeddings are processed by different downstream classification, regression, or clustering algorithms, using typical machine learning metrics like accuracy or root mean squared error (RMSE) for evaluation. The evaluation framework presented in this paper is similar to GEval in that it also evaluates multiple classifiers given a concept vector input.

Melo and Paulheim [38] provide a method for synthesizing benchmark datasets for link and entity type prediction, which are used in conjunction with a fixed ontology. Their goal is to mimic the characteristic of existing knowledge graphs in terms of distributions and patterns. However, it does not come with any specific prediction objective.

Bloem et al. [39] introduce *kgbench*, a node classification benchmark for knowledge graphs, which is based on real-world datasets and comes with tasks in different sizes and predefined train/test splits. Unlike DLCC, kgbench is based on real-world datasets. Therefore, it is suitable to evaluate and compare the quality of different embedding approaches on real-world tasks but does not provide any insights into what these embedding approaches are capable of representing.

In this paper, we introduce a new benchmark for node classification, i.e., Description Logic Class Constructors (DLCC), first introduced in [17], which allows for an isolated consideration of different types of node classification problems in knowledge graphs and therefore can provide insights in which problems can be tackled by a particular embedding method and which cannot.

For the experiments in this paper, we use both the established GEval benchmark as well as the rather new DLCC benchmark, in order to have an encompassing comparison of RDF2vec variants and benchmark models, with respect to both realistic problems using the widely used DBpedia knowledge graph, as well as on synthetic problems allowing to analyze the representational capabilities of the RDF2vec variants in detail.

3. RDF2vec and its Variants

RDF2vec has two main steps (see Fig. 1): First, sequences are extracted from a knowledge graph using random walks. In a second step, these sequences are

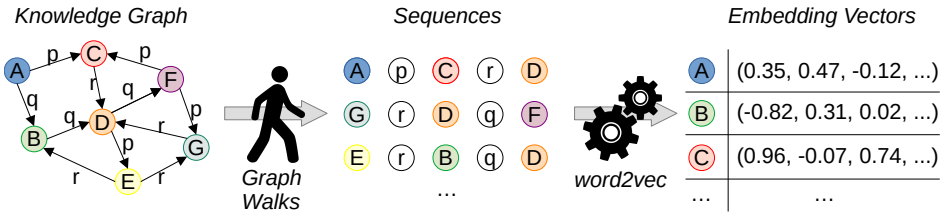

Figure 1. Overall workflow of RDF2vec [40]

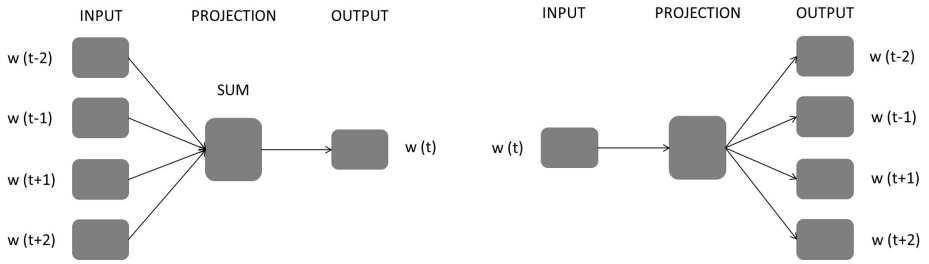

Figure 2. The two basic architectures of word2vec [8]

processed by the word embedding algorithm ~~RDF2vec~~word2vec. The algorithm considers entities and predicates from the graph as "words", so that it produces embedding vectors for entities and predicates.

Word2vec itself has two principle variants (see Fig. 2: context bag of words (CBOW) tries to predict a word from its context, while skip-gram (SG) tries to predict the context from a word. In both cases, a hidden projection layer is used to produce word embeddings [2].

Combining RDF2vec with more recent and advanced word embedding methods, such as FastText [41] and BERT [42], has yielded inconclusive results so far [43]. A potential reason for this is that the ratio of a corpus size extracted by random walks from a graph to the vocabulary size is far smaller than for large text corpora, on which models like BERT are trained.[7] Therefore, most implementations of RDF2vec stick to the more light weight and efficient word2vec.

Over time, RDF2vec was extended multiple times. Generally, three kinds of extensions can be distinguished: (1) Changes in the walk generation algorithm, (2) changes in the embedding algorithm, and (3) other changes. The extensions are presented in the following paragraphs. Out of those extensions, we picked the most promising and interesting candidates and present them in more detail in the subsequent subsections 3.1 and 3.2.

[7] The pre-trained BERT model described in [42] is trained for 30k tokens on a corpus of 3.3B words, which makes a ratio of 110k words per token. On the other hand, extracting 500 length 4 random walks for each entity in a knowledge graph will result in a ratio of only 2.5k "words" per entity, which is two orders of magnitude smaller.

Walk Generation Extensions One of the first extensions to the random walk generation algorithm was biased graph walks [44]. In this extension, multiple edge weighting mechanisms are proposed and evaluated to influence the walk generation. Using the predicate frequency strategy, for instance, increases the likelihood that the random walks will include predicates that are very common. While improvements in some test cases with some configurations are observable compared to the classic strategy, the overall results are inconclusive in that there is not a single best configuration for all tasks and that it is hard to determine which configuration should be used in which situation. It is also important to note that biasing walks increases the overall runtime of the RDF2vec approach since a large number of weights has to be calculated and considered during the walk configuration. While those experiments use graph-internal metrics for weighting edges, later experiments indicate that graph-external metrics for edge importance (in that case: derived from user clickstreams in Wikipedia) can be advantegeous for the resulting embeddings [45]. Other variants of walk generation include the incorporation of community hops or walklets [13], but the evidence here is mixed as well.

Most recently, entity walks and property walks were presented [15]. Those change the walk generation algorithm in terms of what graph elements are included. They are described in more depth in subsection 3.1. The approaches are neutral in terms of additional embedding runtime, entity walks are even significantly faster since the vocabulary is smaller during training.

Embedding Algorithm Extensions The classic RDF2vec configuration is based on word2vec. RDF2vec$_{oa}$ [16] uses an order-aware variant [46] of the original word2vec algorithm. That approach has shown to be *consistently* better than the classic RDF2vec configuration in various publications [4,16].

Other Extensions RDF2vec always generates embedding vectors for an entire knowledge graph. This process can be very expensive for large knowledge graphs and may be even unfeasible for very large knowledge graphs. At the same time, most tasks do not require an embedding for every concept in a knowledge graph. In many cases, the set of required embeddings can be determined ex ante – e.g. *entities of type city* when the task is to regress the score for the quality of living. In such instances, RDF2vec Light [47] can be used. The approach applies the walk generation algorithm only to the predefined entities and thereby reduces the required time for walk generation and training significantly. Experiments showed that the performance is comparable to the more expensive classic variant – particularly in ~~instances~~ cases where the set of entities is homogeneous and their degree is not too large.

3.1. Walk Generation Methods

In this paper, three different walk ~~algorithms~~ generation methods are evaluated: Classic walks, entity walks (e-walks), and predicate walks (p-walks). These configurations have been picked since they have previously been shown to be able to separate the paradigmatic relations of *similarity* and *relatedness* [15].[8]

[8]Similarity describes in how far two concepts are similar to each other "by virtue of their similarity" [48]. Similarity and relatedness are often not clearly separated from each other (for

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)[9] is of the form

$$w = (w_0, w_1, ..., w_{n-1}, w_n) \tag{1}$$

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:

$$w = (e_0, p_1, ..., p_{n-1}, e_n) \tag{2}$$

Entity Walks (e-RDF2vec) An entity walk contains only entities without any other properties. Such an approach is also known as *e-RDF2vec*. It has the form:

$$w_e = (e_0, e_1, ..., e_{n-1}, e_n) \tag{3}$$

For an entity walk, all elements are entities, i.e., $w_{n_i} \in \mathcal{V}$.[10]

Predicate Walks (p-RDF2vec) A predicate walk contains only one entity together with object properties. Such an approach is also known as *p-RDF2vec*. It has the form:

$$w_p = (e_0, p_1, p_2, ..., p_{n-1}, p_n) \tag{4}$$

For a predicate walk, all elements but e_0 are properties, i.e., $e_0 \in \mathcal{V}$, $p_i \in \mathcal{R}$ for all i. The entity does not necessarily need to appear in the beginning of the walk, but can occur in any position.

All three walk strategies are visualized in Figure 3.

3.2. Embedding Models~~of this Publication~~

In this paper, the two original configurations (SG and CBOW) are evaluated. In addition, the order-aware variants are evaluated which are in the following denoted with the suffix "OA". This yields four language model configurations: (1) SG, (2) CBOW, (3) SG$_{oa}$, and (4) CBOW$_{oa}$.

---

instance in [49]). Nevertheless, there are significant differences. Dissimilar entities can even be semantically related by antonomy relationships [48]. Hill et al. distinguish the two relations by giving examples: While the concepts *coffee* and *cup* are certainly related, they are not similar; however, a mug and a cup can – in language as in the real world – almost be used interchangeably and are, therefore, similar [50].

[9] It is important to point out that not all implementations of RDF2vec share the same terminology. The two-hop sequence above would be referred to as a "walk of length 2" (i.e., counting only nodes) by some implementations, while others would consider it a "walk of length 4" (i.e., counting nodes and edges). In this paper, we follow the latter terminology.

[10] Note that in the above example, a walk of length n would comprise n entities. In the graph, the entity e_n would be twice as far away from e_0 as the entity e_n in a classic walk. In other words: when transforming a classic walk of length n into an entity walk by removing all uneven nodes, the corresponding entity walk would be of length $\frac{n}{2}$.

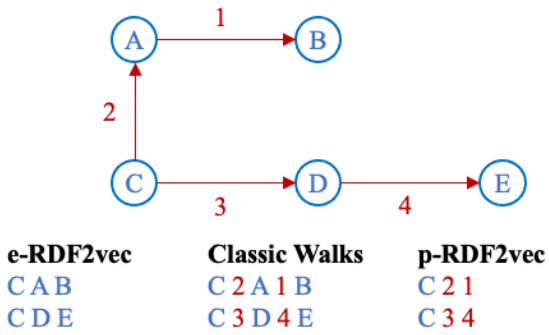

Figure 3. Different walk types visualized, showing walks starting from node C.

3.3. RDF2vec Configurations of this Publication

The walk generation processes and the embedding models are independent components of RDF2vec which can be freely combined. In this paper, we evaluate the following walk generation algorithms:

1. classic walks
2. entity walks
3. predicate walks

We combine these with the following language models:

1. classic word2vec (CBOW and SG)
2. order-aware word2vec (CBOW_{oa} and SG_{oa})

This leads to the following combinations:

1. RDF2vec (original: classic word2vec with classic walks)
2. RDF2vec_{oa} (order aware word2vec with classic walks)
3. p-RDF2vec (predicate walks with word2vec)
4. p-RDF2vec_{oa} (predicate walks with order-aware word2vec)
5. e-RDF2vec (entity walks with classic word2vec)
6. e-RDF2vec_{oa} (entity walks with order-aware word2vec)

Since all of the above combinations can be used with the SG and the CBOW flavor of word2vec, this paper evaluates 12 variants of RDF2vec in total.

While section 2 lists more extensions of RDF2vec, we restricted ourselves to those models listed above. In the scope of this paper, we are mainly investigating the question of which RDF2vec variant is suitable for which problem at hand. In contrast, some of the other extensions mentioned above, like RDF2vec ~~light~~Light, rather target computational performance improvement. ~~On the other hand, experiments~~ Experiments in [47] suggest that the representational power of RDF2vec and RDF2vec ~~light~~ Light are comparable.

For other extensions, like the use of graph external edge or node weights as in [45], external signals are required, which may be created for specific graphs like DBpedia, but not for others. Moreover, we expect that introducing weighted walks may change the quantitative results by putting more emphasis on certain parts of

the graph than on others, but not the representational power of RDF2vec, since, with a large enough number of walks, the embedding algorithm will eventually observe all graph structures, regardless of the weights.

4. Machine Learning Gold Standard

For a comprehensive understanding of the configurations presented in subsection 3.3, an evaluation is performed using the machine learning task set for knowledge graph embeddings published by Ristoski et al. [35]. It is comprised of six tasks using 20 datasets in total:

- Five classification tasks, evaluated by accuracy (ACC). Those tasks use the same ground truth as the regression tasks (see below). The numeric prediction target is discretized into high/medium/low (for the Cities, AAUP, and Forbes dataset) or high/low (for the Albums and Movies datasets). All five tasks are single-label classification tasks.
- Five regression tasks, evaluated by root mean squared error (RMSE). Those datasets are constructed by acquiring an external target variable for instances in knowledge graphs which is not contained in the knowledge graph per se. Specifically, the ground truth variables for the datasets are: a quality of living indicator for the Cities dataset, obtained from Mercer; average salary of university professors per university, obtained from the AAUP; profitability of companies, obtained from Forbes; average ratings of albums and movies, obtained from Facebook.
- Four clustering tasks (with ground truth clusters), evaluated by accuracy (ACC). The clusters are obtained by retrieving entities of different ontology classes from the knowledge graph. The clustering problems range from distinguishing coarser clusters (e.g., cities vs. countries) to finer ones (e.g., basketball teams vs. football teams).
- A document similarity task (where the similarity is assessed by computing the similarity between entities identified in the documents), evaluated by the harmonic mean of Pearson and Spearman correlation coefficients. The dataset is based on the LP50 dataset [51]. It consists of 50 documents, each of which has been annotated with DBpedia entities using DBpedia spotlight [52]. The task is to predict the similarity of each pair of documents.
- An entity relatedness task (where semantic similarity is used as a proxy for semantic relatedness), evaluated by Kendall's Tau. The dataset is based on the KORE dataset [53]. The dataset consists of 20 seed entities from the YAGO knowledge graph, and 20 related entities each. Those 20 related entities per seed entity have been ranked by humans to capture the strength of relatedness. The task is to rank the entities per seed by relatedness.
- Four semantic analogy tasks (e.g., *Athens is to Greece as Oslo is to X*), which are based on the original datasets on which word2vec was evaluated [3]. The original datasets were created by manual annotation. In our evaluation, we aim at predicting the fourth element (D) in an analogy $A : B = C : D$ by considering the closest n vectors to $B - A + C$. If the element is contained the top n predictions, we consider the answer to be

Task	Dataset	# entities	Target variable
Classification	Cities	212	3 classes (67/106/39)
	AAUP	960	3 classes (236/527/197)
	Forbes	1,585	3 classes (738/781/66)
	Albums	1,600	2 classes (800/800)
	Movies	2,000	2 classes (1,000/1,000)
Regression	Cities	212	numeric [23, 106]
	AAUP	960	numeric [277, 1009]
	Forbes	1,585	numeric [0.0, 416.6]
	Albums	1,600	numeric [15, 97]
	Movies	2,000	numeric [1, 100]
Clustering	Cities and Countries (2k)	4,344	2 clusters (2,000/2,344)
	Cities and Countries	11,182	2 clusters (8,838/2,344)
	Cities, Countries, Albums, Movies, AAUP, Forbes	6,357	5 clusters (2,000/960/1,600/212/1,585)
	Teams	4,206	2 clusters (4,185/21)
Document Similarity	Pairs of 50 documents with entities	1,225	numeric similarity score [1.0,5.0]
Entity Relatedness	20x20 entity pairs	400	ranking of entities
Semantic Analogies	(All) capitals and countries	4,523	entity prediction
	Capitals and countries	505	entity prediction
	Cities and States	2,467	entity prediction
	Countries and Currencies	866	entity prediction

Table 1. Overview of the Evaluation Datasets

correct, i.e., the evaluation metric is top-n accuracy. In the default setting of the evaluation framework used, n is set to 2.

Table 1 shows a summary of the characteristics of the datasets used in the evaluation. It can be observed that they cover a wide range of tasks, topics, sizes, and other characteristics (e.g., balance). In this paper, the evaluation protocol as proposed in [35,37] is followed: All entities are linked to a knowledge graph. Different feature extraction methods – in this case pure knowledge graph embedding approaches – can then be compared using a fixed set of learning methods. The evaluation is performed using the GEval framework[11].

5. DL Class Constructors and Hypotheses

In section 4, a gold standard was introduced. That gold standard is task-oriented, i.e., it gives an indication of which embedding configuration is suitable for a specific task – however, the gold standard is not suitable to perform a deeper analysis such as *what* is or can be learned.

The DLCC gold standard aims to close that gap by focusing on specific ontological constructs as targets for entity classification. The underlying idea is that

[11] `https://github.com/mariaangelapellegrino/Evaluation-Framework`

if a classifier is able to separate classes created by specific ontological constructs, with entities represented by means of an embedding E, then this embedding can represent the respective ontological construct. The aim of DLCC thus is to provide a benchmark for analyzing which kinds of constructs in a knowledge graph can be recognized by different embedding methods. The construction of that benchmark is described in section 6.

In order to analyze the representational capabilities of embedding methods, we define class labels using different DL class constructors and argue which variants of RDF2vec are capable of learning them. For each constructor, we formulate hypotheses of which variants of RDF2vec can learn the classes. More precisely, we reject the hypothesis that an embedding can learn a class if a classifier trained on positive examples (members of a class) and negative examples (non-members of a class) does not perform significantly better than random guessing.

The selection of constructors has been mainly motivated by earlier works on propositionalization of RDF for processing in data mining pipelines [54,55], which was a common approach before the emergence of knowledge graph embeddings. [56]

Ingoing and Outgoing Relations All entities that have a particular outgoing or ingoing relation (e.g., *everything that has a location* or *everything that is a location of something*).

$$\exists r.\top \tag{5}$$

$$\exists r^{-1}.\top \tag{6}$$

$$\exists r.\top \sqcup \exists r^{-1}.\top \tag{7}$$

where r is bound to a particular relation.[12]

Hypothesis 1a (5) and (6) can be learned by RDF2vec$_{oa}$ and p-RDF2vec$_{oa}$. Non-oa variants cannot properly learn them because they cannot distinguish the two. e-RDF2vec variants cannot properly learn them because they cannot distinguish particular properties.

Hypothesis 1b (7) can be learned by RDF2vec, RDF2vec$_{oa}$, p-RDF2vec, and p-RDF2vec$_{oa}$.

Use case An exemplary use case would be entity classification. If a relation has a particular domain or range, an embedding vector capturing that information could be used to infer the corresponding class. Using such structural information for entity classification is quite common [9,57,58].

Relations to Particular Individuals All entities that have a relation (in any direction) to a particular individual (e.g., *everything that is related to Mannheim*).

$$\exists R.\{e\} \sqcup \exists R^{-1}.\{e\} \tag{8}$$

[12] We use r to denote a particular relation, whereas R denotes *any* relation.

where R is *not* bound to a particular relation. Those relations can also span two (or more[13]) hops:

$$\exists R_1.(\exists R_2.\{e\}) \sqcup \exists R_1^{-1}.(\exists R_2^{-1}.\{e\}) \tag{9}$$

Hypothesis 2a (8) can be learned by ~~*RDF2vec, RDF2vec$_{oa}$, e − RDF2vec, and e − RDF2vec$_{oa}$*~~ RDF2vec, RDF2vec$_{oa}$, e-RDF2vec, and e-RDF2vec$_{oa}$. Sub-hypothesis: It is possible that the non-oa variants learn it a bit better. However, the non-oa variants will not be able to tell closely related entities (one hop away) from less related ones (more than two hops away).[14]

Hypothesis 2b (9) can be learned by ~~*RDF2vec, RDF2vec$_{oa}$, e − RDF2vec, and e − RDF2vec$_{oa}$*~~ RDF2vec, RDF2vec$_{oa}$, e-RDF2vec, and e-RDF2vec$_{oa}$, as long as the walk length allows for capturing those relations. Sub-hypothesis: It is possible that the non-oa variants learn it a bit better.

Use case An exemplary use case would be capturing entity relatedness. Two entities sharing many connections to a third entity are typically related. This can also be useful in query expansion for information retrieval [59]. The distinction between closely and vaguely related entities (sharing an entity one or two hops away) may be crucial if queries should not be expanded too much. Also in collective entity disambiguation in texts [60], this notion of relatedness can be useful: one would assume that co-mentioned entities are related, but not necessarily want to restrict the kinds of relation among them.

Particular Relations to Particular Individuals All entities that have a particular relation to a particular individual (e.g., *movies directed by Steven Spielberg*).

$$\exists r.\{e\} \tag{10}$$

Hypothesis 3 (10) can only be learned properly by ~~*RDF2vec$_{oa}$*~~ RDF2vec$_{oa}$. Non-oa variants cannot distinguish between the two.[15]

Use case An exemplary use case would be capturing entity similarity. For example, two movies which have the same director and some overlapping cast can be considered similar. This can be used, e.g., in recommender systems [61] or other predictive modeling tasks.

Qualified Restrictions All entities that have a particular relation to an individual of a given type (e.g., *all people married to soccer players*).

$$\exists r.T \tag{11}$$

$$\exists r^{-1}.T \tag{12}$$

If types are included in the graph, then `rdf:type` becomes yet another restriction, and we can reformulate (11) to

[13] For reasons of scalability, we restrict the provided gold standard to two hops.

[14] Depending on the entity at hand, the second set might grow very large. For example, in DBpedia, half of the entities are reachable from *New York City* within two hops.

[15] For example: distinguishing people influenced by Leibniz vs. people who influenced Leibniz.

$$\exists r.(\exists \texttt{rdf:type}.T) \tag{13}$$

Therefore, it behaves equally to a chained variant of (10), and, given a long enough walk length, should have similar constraints. However, if the related entity has strong domain and range signals, it may be learned just by observing the ingoing and outgoing relations of that entity. In that case, ~~*p — RDF2vec*$_{oa}$~~ p-RDF2vec$_{oa}$ could also be capable of learning that class to a certain extent.

Hypothesis 4a (11) can only be learned properly by ~~*RDF2vec*$_{oa}$~~RDF2vec$_{oa}$, and, to a certain extent, by ~~*p — RDF2vec*$_{oa}$~~p-RDF2vec$_{oa}$.

The second case (12) is trickier. Here, the relation to the entity at hand and the type information of the related entity can only appear in two *different* walks, but never together (at least if the inverse relation is not explicitly contained in the graph). Hence, we assume:

Hypothesis 4b (12) cannot be learned by any ~~*RDF2vec*~~ RDF2vec variant.

Use case Qualified restrictions are often useful for fine-grained entity classification and thereby capture some aspects of entity similarity. For example, for distinguishing a basketball and a baseball team, it is not sufficient that both have a coach and players, but that those are of the class `BasketballPlayer` or `BaseballPlayer`. If the similarity aspects become rather fine-grained, they may also be used in predictive modeling tasks.

Cardinality Restrictions of Relations All entities that have at least or at most n relations of a particular kind (e.g., *people who have at least two citizenships*). Here we depict only the *at least* variant because the corresponding classification problem is the same as the *at most* variant (classifying $\geq 2r.\top$ vs. ~~$\neg \geq 2r.\top$~~ $\neg \geq 2r.\top$ is identical to classifying $\leq 1r.\top$ vs. $\neg \leq 1r.\top$).[16]

$$\geq 2r.\top \tag{14}$$

$$\geq 2r^{-1}.\top \tag{15}$$

Since RDF2vec is based on single walks, it cannot *directly* learn cardinalities. However, if a relation appears with a higher cardinality, it is occurring in the walks including the corresponding instance more often, making it a stronger signal for the word2vec algorithm.

Hypothesis 5 (14) and (15) can be learned to a certain extent by ~~*RDF2vec*$_{oa}$ and *p — RDF2vec*$_{oa}$~~RDF2vec$_{oa}$ and p-RDF2vec$_{oa}$. Non-oa variants cannot distinguish the two cases.[17]

Use case Cardinalities often capture entity similarity aspects not expressed in other restrictions. For example, when comparing two authors in a knowledge graph of publications, both will have published papers (which makes them indistinguishable when only looking at qualified restrictions), but there is still a

[16] The fact that most knowledge graphs follow the open-world assumption is ignored here.

[17] For example: distinguishing someone who has been influenced by more than two people vs. someone who has influenced more than two people.

Hypothesis	Test Case	DL Expression
H1a	tc01	$r.\top$
H1a'	tc02	$r^{-1}.\top$
H1b	tc03	$\exists r.\top \sqcup \exists r^{-1}.\top$
H2a	tc04	$\exists R.\{e\} \sqcup \exists R^{-1}.\{e\}$
H2b	tc05	$\exists R_1.(\exists R_2.\{e\}) \sqcup \exists R_1^{-1}.(\exists R_2^{-1}\{e\})$
H3	tc06	$r.\{e\}$
H4a	tc07	$\exists r.T$
H4b	tc08	$\exists r^{-1}.T$
H5	tc09	$\geq 2r.\top$
H5'	tc10	$\geq 2r^{-1}.\top$
H6a	tc11	$\geq 2r.T$
H6b	tc12	$\geq 2r^{-1}.T$

Table 2. Overview of Hypotheses and Test Cases

difference if one has published two and the other has published two hundred papers. Therefore, this distinction is useful in cases where strengths of relations, measured in their cardinality, play a role. One example are recommender engines for scientific papers [62], where highly ranked papers would be given preference over lowly ranked ones.

Qualified Cardinality Restrictions Qualified cardinality restrictions combine qualified restrictions with cardinalities (for example, all people who have published at least three bestsellers).

$$\geq 2r.T \tag{16}$$

$$\geq 2r^{-1}.T \tag{17}$$

Since this is a combination of qualified restrictions and cardinality restrictions, we hypothesize that it can be captured by RDF2vec variants that can handle both of them:

Hypothesis 6a (16) can be learned to a certain extent by ~~*RDF2vec*$_{oa}$~~RDF2vec$_{oa}$.

Hypothesis 6b (17) cannot be learned by any variant of ~~*RDF2vec*~~RDF2vec.

Use case Just like qualified restrictions and cardinality restrictions, these restrictions capture finer-grained aspects of entity similarity and are thus useable both for fine-grained entity classification and for predictive modeling tasks. A few examples of classification patterns were given in [63], where explanations on the cities classification task in the GEval benchmark were analyzed, and explanations like *Cities which are the hometown of many bands have a high quality of living* were observed, which would full into this category.

Table 2 summarizes the test cases that we have discussed above. While for most of them, we can formulate a hypothesis on whether or not they can be represented with a particular RDF2vec variant, we have no particular hypothesis for CBOW vs. SG.

6. DLCC Gold Standard

For the twelve test cases in Table 2, we create positive examples (i.e., those which fall into the respective class) and those which do not (under closed-world semantics). For example, for tc01, we would generate a set of positive instances for which $\exists r.\top$ holds and a set of negative instances for which $\nexists r.\top$ holds. We then evaluate how well these two classes can be separated, given the embedding vectors of the positive and negative instances. For that, we split the examples into a training and testing partition, we train binary classifiers on the training ~~subset of the examples and~~ partition, and we evaluate their performance on the test ~~subset~~partition.

The approach is visualized in Figure 4: A gold standard generator generates a set of positive and negative URIs, as well as a fixed train/test split. The approach presented allows for generating custom gold standards – however, a pre-calculated gold standard is also provided. This pre-calculated gold standard can be used to guarantee reproducibility. We publish pre-calculated gold standards at Zenodo which are versioned to allow for future improvements while allowing for comparable experiments. In this paper, we use version `v1` of the gold standard.

A user provides embeddings in a simple textual format, together with the ground truth labels for the training and the testing partition as input to the evaluator. The evaluator trains multiple classifiers and evaluates them on the selected gold standard using the provided vectors as classification input. The program then calculates multiple statistics in the form of CSV files that can be further analyzed in a spreadsheet program or through data analysis frameworks such as pandas[18]. These analyses help the user to understand how well the provided vectors are performing on a particular DL class constructor.

There are two benchmarks: A DBpedia benchmark ~~(see subsection 6.3)~~ and a synthetic benchmark~~(see subsection 6.4)~~. The benchmarks are publicly available and significant efforts were made to comply with the FAIR [64] principles.[19] ~~Before discussing the two benchmarks in detail, we will quickly~~ In the remainder of this section, we introduce the two software components, namely the gold standard generator (see subsection 6.1) and the evaluation component (see subsection 6.2), and the two benchmarks (subsections 6.3 and 6.4).

6.1. Gold Standard Generator

The gold standard generator is publicly available[20]. It is implemented as a Java maven project. The generator can generate either a DBpedia benchmark (see subsection 6.3) or a synthetic one (see subsection 6.4). Any DBpedia version can be used, the user merely needs to provide a SPARQL endpoint. A comprehensive set of unit tests ensures a high code quality. The generator automatically generates a fixed train-test split for the evaluation framework or any other downstream application. The split is configurable; for the pre-generated gold standards, an

[18] `https://pandas.pydata.org/`

[19] Dataset DOI: `10.5281/zenodo.6509715`; uploaded and indexed via zenodo; published with a permissive license; re-usable; metadata is provided.

[20] `https://github.com/janothan/DL-TC-Generator`

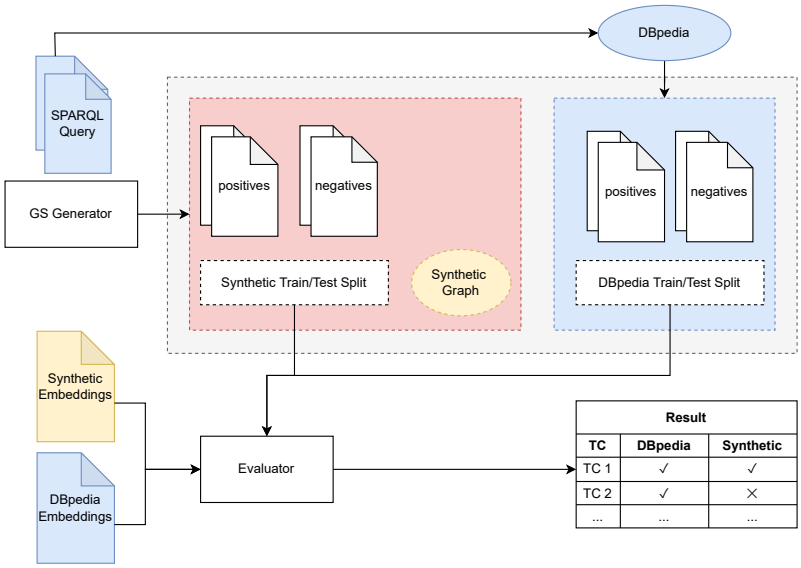

Figure 4. Overview of the DLCC Approach [17]

80-20 split is used. The resulting gold standard is balanced – i.e. the number of positives equals the number of negatives – and the train and test partitions are stratified. Hence, any classifier which achieves an accuracy significantly above 50% is capable of learning the test case's problem type from the vectors to some extent.

It is important to note that the generator only needs to be run by users who want to build their own gold standards. For analyzing the capabilities of a particular knowledge graph embedding approach, it is sufficient to merely download[21] the pre-calculated gold standard files online. We recommend using the pre-calculated and versioned gold standards to ensure comparability across publications.

6.2. Evaluation Framework

The evaluator is publicly available[22] together with usage examples. It is implemented in Python and can be easily used in a Jupyter notebook. A comprehensive set of unit tests ensures a high code quality.

The standard user can directly download the gold standard and use the evaluation framework. To test class separability, the evaluation framework currently runs six machine learning classifiers which are commonly used together with embedding methods for node classification[23] (1) decision trees, (2) naïve Bayes, (3) KNN, (4) SVM, (5) random forest, and (6) a multilayer perceptron network. The framework uses the default configurations of the sklearn library[24].

[21] DOI: `10.5281/zenodo.6509715`; GitHub link for the latest version. `https://github.com/janothan/DL-TC-Generator/tree/master/results`

[22] `https://github.com/janothan/dl-evaluation-framework`

[23] The evaluation framework is not restricted to the set of classifiers listed here. New classifiers can be easily added if desired.

[24] `https://scikit-learn.org/stable/index.html`

After training and evaluation, the framework ~~persists~~ outputs multiple CSV files per test case as well as higher-level aggregate CSV files. Examples of such CSV files are a file listing the accuracy per classifier and per test case or a file listing the accuracy of the best classifier per test case. In the case of DBpedia test cases where multiple domains are available per test case, the results can be analyzed on the level of each domain separately, or in an aggregated manner on the level of the test case.

6.3. DBpedia Benchmark

We use the DBpedia knowledge graph to create test cases.[25] We created SPARQL queries for each test case (see Table 2) to generate positives, negatives, and hard negatives. While an ordinary negative example is simply any entity that does not fulfill the necessary conditions for a positive example[26], a hard negative is an entity that fulfills *some*, but not *all* those conditions. For example, for qualified relations, a positive example would be a person playing in a team which is a basketball team. A simple negative example would be any person not playing in a basketball team, whereas a hard negative example would be any person playing in a team which is not a basketball team.

Query examples for every test case in the people domain are provided in Tables 8, 9 and 10 in the appendix. The framework uses slightly more ~~involved~~ complex queries to vary the size of the result set and to better randomize results.

In total, we used six different domains: people (P), books (B), cities (C), music albums (A), movies (M), and species (S). This setup yields more than 200 hand-written SPARQL queries which are used to obtain positives, negatives, and hard negatives; they are available online[27] and can be easily extended~~e.g.~~ , e.g., to add an additional domain. For each test case, we created differently sized (50, 500, 5000) balanced test sets.[28]

6.4. Synthetic Benchmark

The previous benchmark is realistic and well suited to compare approaches on differently typed DL class constructors.

However, the following aspects have to be considered: (1) DBpedia is a large knowledge graph, not every embedding approach can be used to learn an embedding for it (or not every researcher has the computational means to do so, respectively). (2) Depending on the DL class constructor and the domain, not enough examples can be found on DBpedia. (3) It cannot be precluded that patterns correlate, therefore, the fact that an embedding approach can learn a particular class can only be an indicator that it *might* learn the underlying constructor pattern, but the results are not conclusive, since the performance may also hint at the

[25] We used DBpedia version 2021-09. The generator can be configured to use any DBpedia SPARQL endpoint if desired.

[26] Since negative examples are generated at random, they are very likely not to fulfill any of those conditions.

[27] `https://github.com/janothan/DL-TC-Generator/tree/master/src/main/resources/queries`

[28] The desired size ~~classes~~ of test sets can be configured in the framework.

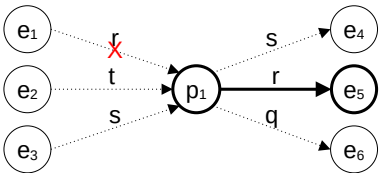

Figure 5. Illustration of the instance generation, using the class constructor $\exists r.T$. First, the pattern is instantiated for the positive example p_1 with the edge (p_1, r, e_5). Then, random edges are inserted (dashed lines). The edge (e_1, r, p_1) is removed, because it would turn e_1 into an additional positive example. [17]

approach learning a cooccurring pattern. Correlating properties, type biases for entities, etc. may lead to surprising results in some domains.

Therefore, we complement the DBpedia-based gold standard with a synthetic benchmark. The idea is to generate a graph that contains the DL class constructors (positive and negative) of interest. The graph can be constructed to resemble the DBpedia graph statistically but can be significantly smaller (and contain a sufficient number of positives and negatives), and, by construction, side effects and correlations which exist in DBpedia can be mitigated to a large extent. However, the generator also allows for using other schema characteristics as well, which paves the way to broadly investigate the behavior of knowledge graph embedding methods for other cases as well. Unlike other synthetic data generators, like LUBM [65], we create both a schema (T-Box) and instances (~~an~~ A-Box), while LUBM merely creates instances given a *fixed* schema.

The configurable parameters are `numClasses`, `numProperties`, `numInstances`, `branchingFactor`, `maxTriplesPerNode`, and `numNodesInterest` (all parameters are integers). The overall process is depicted in Algorithm 1: First, a class tree with `numClasses` classes is constructed in a way that each class has at most `branchingFactor` children. Then, `numProperties` properties are generated. Each property is assigned to a range and domain from the class tree whereby the first property has the root node as domain and range type so that every node can be involved in at least one triple statement. A skew can be introduced so that domain and range refer ~~with a higher probability~~ to a more general class than to a specific one with a higher probability. Lastly, we generate instances and assign them to a class as type which is depicted in Algorithm 1.

Once the ontology is created, `numNodesInterest` positives and negatives are generated (adhering to domain/range restrictions). Each class constructor is first initialized explicitly for the positive examples. Then, for each entity e in the graph (i.e., positive and negative examples), $rand(n) \in [1, maxTriplesPerNode]$ random triples are generated which have e as a subject and adhere to the domain and range definitions~~, whereby it is checked~~. Additionally, we check that no additional positives are created and no negatives are turned into positives accidentally (see Figure 5).

For version `v1` of the gold standard, `numClasses=760`, `numProperties=1,355`, `numInstances=10,000`, `branchingFactor=5`, `maxTriplesPerNode=11`, and `numNodesInterest=1,000` were chosen. The parameters were chosen to form graphs which are smaller than DBpedia but resemble the DBpedia graph statistically, so that the results can be meaningfully compared to those on the non-synthetic

part of DLCC. ~~Therefore,~~ We used the statistical properties of the DBpedia ontology calculated by Heist et al. [66] ~~It is important to point out, however, that~~ However, this choice of parameters is not at all obligatory, and other parameters can be chosen to resemble other ontologies and/or build synthetic test cases with particular characteristics of interest.

7. Evaluation

7.1. Training Details

RDF2vec We trained 12 RDF2vec embeddings using the configurations listed in subsection 3.3. For the DBpedia benchmarks, we use version 2021-09. We generated 500 walks per entity, with a depth of 4, a window size of 5, 5 epochs, and a dimension of 200. We used the same parameters for the synthetic gold standard with the exception of $dimension = 100$ and $walks = 100$ to account for the smaller gold standard size. The embeddings were trained using the jRDF2vec[29] framework [47]. The embedding files are publicly available[30] via KGvec2go [67] and can also be used for other downstream tasks.

Benchmark Models We trained DBpedia embeddings using seven benchmark models:

- TransE [27] with L1 norm
- TransE [27] with L2 norm
- TransR [28]
- ComplEx [20]
- DistMult [20]
- RESCAL [24]
- RotatE [21]

The above-mentioned benchmark models were trained using the DGL-KE framework[31] [68], using the respective default parameters, with 200 dimensions for DBpedia and 100 for the synthetic datasets, as for RDF2vec. The models are publicly available and can also be used for other downstream tasks.[32]

7.2. Results on the ML Gold Standard

The results for the ML gold standard introduced in section 4 are provided in Tables 3 (classification and clustering), 4 (regression and semantic analogies), and 5 (entity relatedness and document similarity). For each task with multiple test sets (i.e., classification, regression, clustering, and semantic analogies), we performed a Friedman test to test whether the results achieved with the different embedding methods are ~~signficantly~~ significantly different. The test showed signifi-

[29] https://github.com/dwslab/jRDF2Vec
[30] http://data.dws.informatik.uni-mannheim.de/kgvec2go/dbpedia/2021-09/
[31] https://github.com/awslabs/dgl-ke
[32] http://data.dws.informatik.uni-mannheim.de/kgvec2go/dbpedia/2021-09/non-rdf2vec/

Algorithm 1 Ontology Creation

```
procedure generateClassTree(numClasses, branchingFactor)
    clsURIs ← generateURIs(numClasses)
    root ← randomDraw(clsURIs)
    i ← 0
    workList ← newList( )
    result ← newTree( )
    currentURI ← root
    for clsURI in clsURIs do
        if clsURI = root then
            CONTINUE
        end if
        if i = branchingFactor then
            currentURI ← workList.removeFirst()
            i ← 0
        end if
        result.addLeaf(currentURI, clsURI)
        i ← i + 1
        workList.add(clsURI)
    end for
    return result
end procedure

procedure generateProperties(numProperties, classTree)
    properties ← generateURIs(numProperties)
    for property in properties do
        property.addDomain( drawDomainRange(classTree, 0.25) )
        property.addRange( drawDomainRange(classTree, 0.25) )
    end for
    return properties
end procedure

procedure drawDomainRange(classTree, p)
    result ← classTree.randomClass()
    while Random.nextDouble > p ∧ ¬(classTree.getChildren(result) == ∅)
do
        result ← randomDraw(classTree.getChildren(result))
    end while
end procedure

procedure generateInstances(numInstances, classTree)
    instances ← generateURIs(numInstances)
    for instance in instances do
        instance.type(classTree.randomClass())
    end for
    return instances
end procedure
```

cance for the tasks of classification (Q=61.38, p=0.000001), regression (Q=46.18, p=0.000279), and semantic analogy (Q=56.84, p=0.000007), but not for clustering. For those cases where the Friedman test shows significance, we report ~~signficance~~ significance on individual comparisons of approaches according to a one-sided t-test.

Classification On the classification task, it can be observed that the order-aware RDF2vec variants lead – with few exceptions – to generally better or the same results[33]. It is further observable that the SG configuration outperforms the CBOW configuration.[34] Within the RDF2vec family, the classic and the e-walks variant achieve the best results.[35] Concerning the benchmark models, the overall best results are achieved using TransE with L2[36]; RDF2vec SG configurations are close to the best scores.

Clustering Concerning the benchmark models, the overall best results are achieved using TransE with L2. Concerning the RDF2vec configurations, the results are rather inconclusive. As mentioned above, the results for clustering are not significant according to the Friedman test.

Regression Again, on the regression tasks, improvements can be observed for the order-aware variants which outperform non-order-aware variants, although not significant. Again, TransE with L2 regularization achieves the best results in most cases[37] with RDF2vec SG_{oa} being the runner-up.[38]

Semantic Analogies On the semantic analogies task, the classic RDF2vec variant with SG configuration performs best[39]. Improvements by the order-aware variants cannot be observed on this task[40]. Among the baseline models, RESCAL[41] and RotatE[42] perform comparatively badly on this task.

---

[33]The order-aware variant significantly ($p < 0.05$) outperforms the non-order-aware variant for p-RDF2vec SG and p-RDF2vec CBOW

[34]The SG variant significantly ($p < 0.05$) outperforms the corresponding CBOW variant for: RDF2vec, RDF2vec$_{oa}$, p-RDF2vec, and e-RDF2vec$_{oa}$.

[35]RDF2vec SG significantly ($p < 0.05$) outperforms, RDF2vec CBOW$_{oa}$, p-RDF2vec CBOW, and e-RDF2vec e-RDF2vec CBOW. e-RDF2vec SG$_{oa}$ significantly ($p < 0.05$) outperforms RDF2vec CBOW, RDF2vec CBOW$_{oa}$, p-RDF2vec SG, and p-RDF2vec CBOW.

[36]TransE-L2 significantly ($p < 0.05$) outperforms all RDF2vec variants but RDF2vec SG and RDF2vec SG$_{oa}$, and all other benchmark models.

[37]TransE-L2 significantly ($p < 0.05$) outperforms all variants of RDF2vec except RDF2vec SG$_{oa}$, p-RDF2vec SG, p-RDF2vec CBOW, as well as RotatE and RESCAL

[38]RDF2vec SG$_{oa}$ significantly ($p < 0.05$) outperforms RDF2vec CBOW, RDF2vec CBOW$_{oa}$, p-RDF2vec CBOW$_{oa}$, e-RDF2vec SG, and e-RDF2vec CBOW.

[39]RDF2vec SG significantly ($p < 0.05$ outperforms all other RDF2vec variants, as well as all baseline models except TransE-L1 and TransR.

[40]Only the differences for the order-aware and non-order-aware variants of p-RDF2vec SG and p-RDF2vec CBOW are significant ($p < 0.05$), but the absolute scores are very low compared to other approaches.

[41]RESCAL is significantly ($p < 0.05$) outperformed by RDF2vec SG, RDF2vec SG$_{oa}$, RDF2vec CBOW, RDF2vec CBOW$_{oa}$, e-RDF2vec SG, e-RDF2vec SG$_{oa}$, e-RDF2vec CBOW$_{oa}$, as well TransE-L1, TransE-L2, and TransR.

[42]RotatE is significantly ($p < 0.05$) outperformed by RDF2vec SG, RDF2vec SG$_{oa}$, RDF2vec CBOW, RDF2vec CBOW$_{oa}$, e-RDF2vec SG, e-RDF2vec SG$_{oa}$, as well TransE-L1, TransE-L2, and TransR.

Entity Relatedness and Document Similarity On the entity relatedness task, the e-RDF2vec variants perform comparatively well with e-RDF2vec SG being the best model. This is intuitive since the e-RDF2vec variant can be expected to pick up the notion of entity relatedness best. On the document similarity task, it can be observed that the p-RDF2vec variant outperforms the other RDF2vec configurations. Again, this finding is intuitive since the configuration is expected to pick up fine-grained entity similarity best – for example, for distinguishing politics from sports texts, it is not sufficient to know that both mention persons, but it is required to distinguish athletes from politicians.

7.3. Results on DLCC

As outlined in subsection 6.1, the DLCC benchmarks are balanced. That means that a performance significantly above 50% indicates that the model learns the constructor to some extent. It is important to highlight that Tables 6 and 7 state the best results out of six classifiers (see subsection 6.2). In order to determine whether the stated result for an embedding configuration for a particular test case is significant, we performed an approximated one-sided binomial significance test with $\alpha = 0.05$. Since multiple classifiers were trained for each test case, we applied the conservative Bonferroni correction [69] of α to account for the multiple testing problem. The hypothesis underlying each significance test is that in the embedding space spanned by a given approach, positive and negative examples can be separated by a classifier. Therefore, we test whether the classification results yield an accuracy significantly greater than 0.5, since all classification problems are fully balanced. The null hypothesis is that the classes cannot be separated, i.e., the classification accuracy does not significantly exceed 0.5.

DBpedia Benchmark The results on the DLCC DBpedia benchmark (class size 5,000) are reported in Table 6. For each model, six classifiers were trained resulting in more than 2,000 classification results. At first sight, it is quickly observable that all models can learn all tasks comparatively well; all results are statistically significant. It is, furthermore, visible that the hard test cases are indeed harder.

On the DBpedia gold standard, it can be seen that ~~s-RDF2vec~~ p-RDF2vec is rather suitable for similarity-based constructors (tc1, tc2, tc3, tc6) while e-RDF2vec is doing better on relatedness-oriented constructors (tc04, tc05).

Moreover, we can observe that it seems easier to predict patterns involving outgoing edges than those involving ingoing edges (cf. tc02 vs. tc01, tc08 vs. tc07, tc10 vs. tc09, tc12 vs. tc11). Even though the tasks are very related, this can be explained by the learning process which often emphasizes outgoing directions: In RDF2vec, random walks are performed in forward direction; similarly, TransE is directed in its training process. On the DBpedia benchmark, it is observable that the TransE-L2 configuration performs, overall, best scoring first place in 9 out of 20 ~~instances~~cases.

Figure 6 depicts the simplicity per domain of the DBpedia gold standard in a box-and-whisker plot. The simplicity was determined by using the accuracy of the best classifier of each embedding model without hard test cases (since not every domain has an equal amount of hard test cases), i.e., the difficulty for a test case t and an embedding model e is

Table 3. ML Results for Classification and Clustering

Approach	Classification (Accuracy)					Clustering (Accuracy)			
	AAUP	Cities	Forbes	Metacritic Albums	Metacritic Movies	Cities and Countries (2k)	Cities and Countries	Cities, Albums Movies, AAUP, Forbes	Teams
RDF2vec SG	0.706	0.818	**0.623**	0.586	0.726	0.789	0.587	0.829	0.909
RDF2vec SG_{oa}	0.713	0.803	0.605	0.585	0.716	0.9	0.76	0.854	0.931
RDF2vec CBOW	0.643	0.725	0.575	0.536	0.549	0.52	0.783	0.547	0.94
RDF2vec $CBOW_{oa}$	0.69	0.723	0.6	0.532	0.626	0.917	0.72	0.652	0.925
p-RDF2vec SG	0.564	0.606	0.581	0.634	0.61	0.605	0.687	0.598	**0.941**
p-RDF2vec SG_{oa}	0.623	0.677	0.61	0.632	0.66	0.52	0.782	0.798	0.938
p-RDF2vec CBOW	0.551	0.501	0.56	0.569	0.535	0.637	0.787	0.663	0.94
p-RDF2vec $CBOW_{oa}$	0.612	0.707	0.578	0.667	0.663	0.733	0.728	0.748	0.58
e-RDF2vec SG	0.696	0.77	0.608	0.596	0.724	0.726	0.749	0.759	0.889
e-RDF2vec SG_{oa}	**0.717**	0.743	0.605	0.583	0.732	0.726	0.766	0.828	0.926
e-RDF2vec CBOW	0.703	0.75	0.612	0.564	0.686	0.668	0.82	0.557	0.916
e-RDF2vec $CBOW_{oa}$	0.69	0.702	0.6	0.584	0.676	0.66	0.745	0.719	0.931
TransE-L1	0.639	0.716	0.572	0.624	0.645	0.933	0.93	0.901	0.835
TransE-L2	0.668	**0.827**	0.61	**0.668**	**0.76**	**0.94**	**0.939**	**0.906**	0.893
TransR	0.637	0.775	0.576	0.619	0.715	0.929	0.917	0.753	0.816
RotatE	0.628	0.653	0.542	0.582	0.573	0.821	0.641	0.76	0.688
RESCAL	0.653	0.755	0.596	0.622	0.689	0.933	0.927	0.894	0.835
DistMult	0.637	0.689	0.577	0.634	0.678	0.868	0.896	0.859	0.814
ComplEx	0.628	0.756	0.585	0.632	0.7	0.897	0.909	0.859	0.815

Table 4. ML Results for Regression and Semantic Analogies

Approach	Regression (Root Mean Squared Error)					Semantic Analogies (Accuracy)			
	AAUP	Cities	Forbes	Metacritic Albums	Metacritic Movies	capital country entities	all capital country entities	currency entities	city state entities
RDF2vec SG	65.985	15.375	36.545	15.288	20.215	**0.957**	0.905	**0.574**	**0.609**
RDF2vec SG_{oa}	**63.814**	12.782	36.05	15.903	20.42	0.864	0.857	0.535	0.578
RDF2vec CBOW	77.25	18.963	39.204	15.812	24.238	0.81	0.594	0.338	0.507
RDF2vec $CBOW_{oa}$	66.473	19.287	37.067	15.705	23.362	0.789	0.758	0.447	0.442
p-RDF2vec SG	80.275	20.322	37.146	15.178	23.235	0.008	0.014	0.006	0.009
p-RDF2vec SG_{oa}	72.61	17.214	36.374	14.869	22.402	0.091	0.073	0.076	0.048
p-RDF2vec CBOW	96.248	24.743	37.947	15.0	23.979	0.0	0.002	0.002	0.0
p-RDF2vec $CBOW_{oa}$	77.895	20.334	38.952	16.679	22.071	0.036	0.052	0.085	0.036
e-RDF2vec SG	67.337	17.017	38.589	15.573	20.436	0.794	0.657	0.309	0.459
e-RDF2vec SG_{oa}	65.429	16.913	38.558	15.785	20.258	0.747	0.591	0.193	0.484
e-RDF2vec CBOW	70.482	17.29	39.867	15.574	23.348	0.66	0.359	0.198	0.25
e-RDF2vec $CBOW_{oa}$	69.292	20.798	36.313	14.64	22.518	0.397	0.592	0.297	0.361
TransE-L1	82.007	16.485	37.465	14.652	22.796	0.901	0.909	0.09	0.345
TransE-L2	64.386	**12.301**	36.454	**13.689**	**19.765**	0.874	0.884	0.39	0.321
TransR	85.084	13.436	38.067	14.581	20.624	0.923	**0.925**	0.136	0.398
RotatE	83.21	20.869	38.713	14.949	23.9	0.676	0.515	0.0	0.237
RESCAL	68.589	16.383	35.875	14.608	21.562	0.395	0.372	0.0	0.161
DistMult	73.205	17.65	36.737	14.213	21.292	0.779	0.856	0.001	0.295
ComplEx	75.846	15.33	**35.689**	14.236	21.041	0.609	0.829	0.004	0.29

Table 5. ML Results for Entity Relatedness and Document Similarity

Approach	Entity Relatedness (Kendall Tau)	Document Similarity (Harmonic Mean)
RDF2vec SG	0.747	0.237
RDF2vec SG_{oa}	0.716	0.23
RDF2vec CBOW	0.611	0.283
RDF2vec $CBOW_{oa}$	0.547	0.209
p-RDF2vec SG	0.432	0.193
p-RDF2vec SG_{oa}	0.768	0.382
p-RDF2vec CBOW	0.568	0.296
p-RDF2vec $CBOW_{oa}$	0.737	0.256
e-RDF2vec SG	**0.832**	0.275
e-RDF2vec SG_{oa}	0.8	0.25
e-RDF2vec CBOW	0.726	0.17
e-RDF2vec $CBOW_{oa}$	0.779	0.111
TransE-L1	0.632	0.388
TransE-L2	0.537	0.398
TransR	0.589	**0.484**
RotatE	0.432	0.467
RESCAL	0.558	0.358
DistMult	0.432	0.406
ComplEx	0.589	0.387

$$simplicity(t, e) = \max_{c \in classifiers} acc(c, e, t), \tag{18}$$

where $acc(c, e, t)$ is the accuracy of classifier c on test case t using the embedding e as a feature representation. The distribution of the simplicity values across all tasks and embedding models can be used to quantify the simplicity of the task – the closer the values are to 1, the easier the task. If a single metric is sought, the median across all simplicity values can be used. We observe that all domain test cases are similarly hard to solve whereby the albums, people, and species domain are a bit simpler to solve than the books and cities domain. Overall, however, we observe that the majority of problems in the DBpedia gold standard is not too hard to solve, since almost all median simplicity values are above 0.9.

Synthetic Benchmark The results on the synthetic benchmark (class size 1,000) are reported in Table 7. Again, for each model, six classifiers were trained whereby only the best performing classifiers' results are discussed. RDF2vec configurations are performing very well on this gold standard being the best performing embedding model in 10 out of 12 cases. In terms of the best RDF2vec configuration, the classic CBOW variant achieves the best results in five cases.

The intuition that ~~s-RDF2vec~~ p-RDF2vec is doing better on similarity-based constructors while e-RDF2vec is doing better on relatedness-oriented constructors can again be observed: This time e-RDF2vec is not able to learn tc02 and tc03 which is intuitive since the approach does not learn the notion of predicate types.

Table 6. Results on the DBpedia Gold Standard (Accuracy). The best results are printed in bold. All results are significantly larger than the random baseline.

TC	SG	SG_{oa}	CBOW	$CBOW_{oa}$	p-SG	$p\text{-}SG_{oa}$	p-CBOW	$p\text{-}CBOW_{oa}$	e-SG	$e\text{-}SG_{oa}$	e-CBOW	$e\text{-}CBOW_{oa}$	TransE-L1	TransE-L2	TransR	DistMult	ComplEx	RESCAL	RotatE
tc01	0.915	0.937	0.778	0.870	0.907	0.933	0.780	0.924	0.845	0.860	0.840	0.840	0.842	0.947	0.858	0.874	0.862	**0.966**	0.768
tc01 hard	0.681	0.891	0.637	0.891	0.627	0.903	0.576	0.894	0.644	0.651	0.659	0.659	0.799	**0.916**	0.744	0.646	0.651	0.830	0.618
tc02	0.953	0.961	0.865	0.956	0.930	0.972	0.901	**0.974**	0.883	0.895	0.906	0.906	0.852	0.970	0.832	0.859	0.853	0.908	0.737
tc02 hard	0.637	0.780	0.618	0.774	0.628	0.828	0.583	0.838	0.623	0.628	0.607	0.607	0.780	**0.849**	0.693	0.622	0.608	0.729	0.649
tc03	0.949	**0.958**	0.846	0.905	0.913	0.956	0.800	0.938	0.883	0.900	0.886	0.886	0.821	0.933	0.856	0.894	0.874	0.943	0.780
tc04	0.960	0.968	0.705	0.872	0.877	0.908	0.659	0.873	0.965	0.969	0.915	0.915	0.934	0.986	0.973	0.984	**0.990**	0.990	0.862
tc04 hard	0.963	0.984	0.674	**0.992**	0.725	0.828	0.583	0.782	0.938	0.990	0.983	0.983	0.814	0.912	0.855	0.917	0.935	0.918	0.789
tc05	0.986	0.992	0.772	0.906	0.869	0.899	0.719	0.870	0.990	**0.995**	0.931	0.931	0.867	0.948	0.881	0.907	0.905	0.908	0.802
tc06	0.957	0.963	0.698	0.850	0.876	0.903	0.641	0.857	0.960	0.969	0.928	0.928	0.929	0.985	0.976	0.985	**0.991**	0.990	0.866
tc06 hard	0.863	0.936	0.604	0.908	0.708	0.770	0.559	0.745	0.699	0.708	0.650	0.650	0.823	0.779	**0.964**	0.882	0.933	0.964	0.819
tc07	0.938	0.955	0.742	0.785	0.895	0.924	0.726	0.863	0.946	0.946	0.859	0.859	0.930	**0.987**	0.978	0.929	0.966	0.945	0.847
tc08	0.961	0.966	0.891	0.896	0.911	**0.968**	0.841	0.951	0.904	0.914	0.925	0.925	0.898	0.964	0.870	0.856	0.888	0.875	0.831
tc09	0.902	0.901	0.773	0.858	0.819	0.858	0.726	0.832	0.874	0.884	0.840	0.840	0.884	**0.938**	0.879	0.877	0.883	0.929	0.780
tc09 hard	0.785	0.793	0.659	0.751	0.698	0.741	0.600	0.712	0.777	0.782	0.744	0.744	0.749	**0.848**	0.758	0.774	0.776	0.820	0.676
tc10	0.947	0.958	0.918	0.905	0.924	0.975	0.852	0.969	0.911	0.912	0.925	0.925	0.957	**0.984**	0.898	0.918	0.931	0.927	0.878
tc10 hard	0.740	0.737	0.716	0.711	0.610	0.679	0.569	0.652	0.715	0.718	0.729	0.729	**0.775**	0.774	0.656	0.743	0.739	0.713	0.665
tc11	0.932	0.897	0.865	0.780	0.884	**0.991**	0.808	0.954	0.928	0.972	0.921	0.921	0.917	0.960	0.930	0.889	0.946	0.954	0.838
tc11 hard	0.725	0.737	0.687	0.676	0.684	0.707	0.631	0.707	0.763	0.734	0.641	0.641	0.712	**0.806**	0.753	0.666	0.723	0.726	0.638
tc12	0.955	0.938	0.888	0.909	0.900	0.971	0.830	0.965	0.893	0.905	0.904	0.904	0.961	**0.984**	0.879	0.912	0.894	0.927	0.834
tc12 hard	0.714	0.717	0.712	0.699	0.628	0.637	0.545	0.628	0.690	0.713	0.715	0.715	0.762	**0.765**	0.659	0.714	0.710	0.701	0.652

Table 7. Results on the Synthetic Gold Standard (Accuracy). The best result for each test case is printed in bold, statistically insignificant scores (w.r.t. a random baseline) are stated in italics. Listed are the results of the best classifier for each task and model.

TC	SG	SG$_{oa}$	CBOW	CBOW$_{oa}$	p-SG	p-SG$_{oa}$	p-CBOW	p-CBOW$_{oa}$	e-SG	e-SG$_{oa}$	e-CBOW	e-CBOW$_{oa}$	TransE-L1	TransE-L2	TransR	DistMult	ComplEx	RESCAL	RotatE
tc01	0.882	0.867	0.566	0.877	0.870	0.842	0.802	0.847	0.774	0.757	0.752	0.727	0.767	0.752	0.712	0.837	0.789	**0.895**	0.769
tc02	0.742	0.737	0.769	0.732	**0.822**	0.734	0.769	0.754	*0.536*	*0.529*	*0.536*	*0.529*	0.677	0.677	*0.531*	0.584	*0.549*	0.689	*0.546*
tc03	0.797	0.812	**0.927**	0.774	0.794	0.709	0.784	0.742	*0.526*	*0.526*	*0.561*	*0.519*	*0.531*	0.581	*0.554*	*0.556*	*0.536*	0.634	*0.541*
tc04	**1.000**	0.998	0.990	0.998	0.568	0.588	0.608	0.628	**1.000**	0.995	**1.000**	0.998	0.790	0.898	0.685	0.588	*0.553*	*0.528*	0.728
tc05	**0.892**	0.819	0.889	0.819	0.631	0.648	0.681	0.648	0.832	0.819	0.882	0.791	0.691	0.774	0.631	0.658	0.726	0.608	0.646
tc06	0.978	0.963	0.898	0.965	0.800	0.828	0.748	0.820	0.970	0.968	0.905	0.965	0.898	0.978	0.888	**1.000**	**1.000**	**1.000**	0.955
tc07	0.583	0.583	0.575	*0.555*	*0.553*	*0.553*	*0.535*	*0.540*	*0.543*	*0.525*	*0.498*	*0.518*	*0.540*	0.615	**0.673**	0.565	*0.518*	*0.550*	*0.508*
tc08	0.563	0.585	*0.555*	0.583	0.635	**0.638**	0.568	0.618	*0.525*	*0.533*	*0.553*	*0.540*	0.585	0.613	*0.540*	*0.535*	*0.523*	*0.533*	*0.535*
tc09	0.610	0.628	**0.648**	0.605	0.563	*0.550*	0.605	0.590	*0.550*	*0.535*	*0.508*	*0.528*	0.588	*0.543*	*0.525*	*0.525*	*0.545*	0.638	*0.538*
tc10	0.638	0.623	**0.665**	0.600	*0.548*	*0.560*	0.633	0.565	0.593	0.565	0.568	*0.515*	0.588	0.573	*0.518*	*0.525*	*0.510*	0.580	*0.533*
tc11	0.633	0.580	**0.668**	0.575	0.573	*0.555*	0.580	*0.553*	*0.550*	*0.545*	*0.540*	*0.545*	0.583	0.590	0.573	*0.518*	0.590	0.625	*0.538*
tc12	0.644	0.614	**0.657**	0.638	0.563	0.565	0.590	0.640	*0.541*	0.568	*0.560*	*0.524*	0.618	*0.550*	*0.513*	*0.553*	*0.540*	0.578	*0.533*

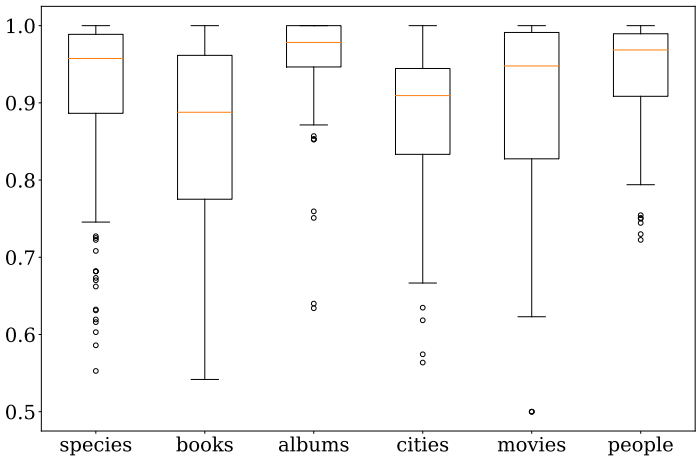

Figure 6. Simplicity of the DBpedia Gold Standard (Size Class 5000)

On tc04 and tc05, on the other hand, the e-RDF2vec approach performs very well (much better than ~~s-RDF2vec~~p-RDF2vec).

The best benchmark model is RESCAL. RotatE produces ~~more often~~ insignificant results than significant results more often – the model outperforms pure guessing in only a third of the cases.

The overall most complicating test case is tc07. Similarly, more than half of the models are not significantly able to learn tc08. This is remarkable since the constructors can be almost perfectly predicted on the corresponding DBpedia gold standards. Hence, we can reason that handling qualified restrictions is a very intricate task. The second hardest group of tasks is those involving cardinalities (tc10-tc12).

DBpedia Benchmark vs. Synthetic Benchmark The comparison of the DBpedia and the synthetic benchmark is particularly intriguing. We can see that the synthetic benchmark is much harder to solve since the results are drastically lower in most cases. While there are no insignificant results on the DBpedia gold standard, there are many for the synthetic one – particularly when it comes to the benchmark models. ~~Any~~ Many class constructors that are easily learnable on the DBpedia gold standard are hard on the synthetic one. Moreover, the previously reported superiority of RDF2vec$_{oa}$ over standard RDF2vec [4,16] cannot be observed on the synthetic data.

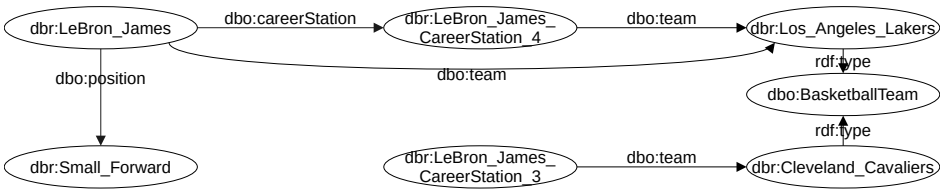

Figure 7. Excerpt of DBpedia

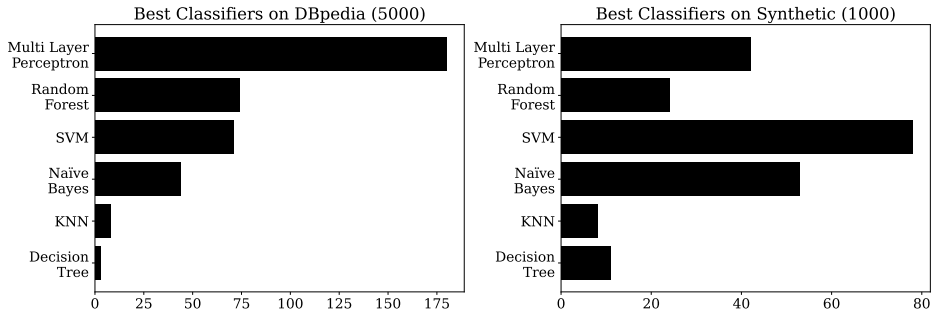

Figure 8. Best DLCC Classifiers on DBpedia and Synthetic. It is important to note that the total number of test cases varies between the two gold standards – therefore, two separate plots were drawn.

Figure 7 shows an excerpt of DBpedia, which we will use to illustrate these deviations. The instance `dbr:LeBron_James` is a positive example for task tc07 in Table 9. At the same time, 95.6% of all entities in DBpedia fulfilling the positive query for positive examples also fall in the class $\exists$`dbo:position`.$\top$ (which is a tc01 problem), but only 13.6% of all entities fulfilling the query for trivial negatives. Hence, on a balanced dataset, this class can be learned with an accuracy of 0.91 by any approach than can learn classes of type tc01. As a comparison to the synthetic dataset shows, the results on the DBpedia test set for tc07 actually overestimate the capability of many embedding approaches to learn classes constructed with a tc07 class constructor. Such correlations are quite frequent in DBpedia but vastly absent in the synthetic dataset.

The example can also explain the advantage of RDF2vec$_{oa}$ on DBpedia. Unlike standard RDF2vec, this approach would distinguish the appearance of `dbo:team` as a direct edge of `dbr:LeBron_James` as well as an indirect edge connected to `dbr:LeBron_James_CareerStation_`N, where the former denotes the current team, whereas the latter also denote all previous teams. Those subtle semantic differences of different usages of the same property in different contexts also do not exist in the synthetic gold standard. Hence, the order-aware variant of RDF2vec does not have an advantage here. In the cases where a DLCC can be learned on the DBpedia dataset, but not on the synthetic dataset, we have to assume that the downstream learning algorithm cannot learn the DLCC per se, but some other pattern which appears in correlation with the DLCC at hand, since such correlations exist in the DBpedia dataset, but not in the synthetic dataset.

~~Figure~~Finally, figure 8 shows the aggregated number of the best classifiers for each embedding on each test case. It is visible that on DBpedia, MLPs work best followed by random forests and SVMs. On the synthetic gold standard, SVMs work best most of the time followed by naïve Bayes and MLPs. The differences can partly be explained by the different size classes of the training sets (MLPs and random forests typically work better on more data).

7.4. Discussion of the Hypotheses

In this section, the hypotheses stated in section 5 are verified and discussed. We treat the hypotheses as non-exclusive. That is, we accept the hypotheses if there is significance that the stated configurations can indeed learn the ~~description logic~~ corresponding class constructor; in cases where we hypothesize that the constructor can be learned by neither configuration, we reject the hypothesis if a single approach can learn the constructor. However, we do not want to mislead the reader: We underestimated which other configurations are also capable of learning constructors. We, therefore, encourage the reader to not just check which hypotheses are accepted but to also follow the reasoning. Hence, we use the hypotheses as structured discussion points for a deeper analysis.

Hypothesis 1 The hypothesis can be accepted. It has to be acknowledged though that – with the exception of e-RDF2vec – all RDF2vec configurations perform rather well.

Hypothesis 1a/1a' In fact, out of all RDF2vec configurations, RDF2vec$_{oa}$ and p-RDF2vec$_{oa}$ are performing best on tc01 and tc02 for DBpedia. On the synthetic gold standard, this can similarly be observed albeit the improvement of ~~OA~~ order aware variants does not account for all RDF2vec variants. The previously discussed directionality bias in the training likely leads to better results on tc01 compared to tc02.

Hypothesis 1b Particularly on tc03 (synthetic), it is visible that e-RDF2vec cannot really learn the constructor: None of the configurations performs significantly better than random guessing. As expected, once the directionality restriction is lifted, the results generally improve.

Hypothesis 2 The hypothesis can be accepted. Again, however, it has to be noted that even the p-RDF2vec configuration performs well on tc04 and tc05. While performing worse than the other configurations, p-RDF2vec is still able to a small extent to learn the constructor as witnessed by the results on the synthetic gold standard. The sub-hypotheses, stating that non-order-aware variants perform better than order-aware variants, can be rejected. On DBpedia, significant increases can be observed when using the order-aware variant. Although there are multiple cases of non-oa variants slightly outperforming order-aware variants on the synthetic gold standard, there is, overall, also not enough evidence to accept this hypothesis.

Hypothesis 3 The hypothesis can be accepted. Particularly on the hard tc06 test case, the classic RDF2vec configuration with the order-aware training component performs best. It has to be admitted though, that on the synthetic gold standard the e-RDF2vec variant performs very well. A reason for this may be the fact that domain/range restrictions can also be found in the synthetic gold standard which allows to reason on a likely predicate given an object entity.

Hypothesis 4 The hypothesis can only be partially accepted.

Hypothesis 4a The RDF2vec$_{oa}$ configuration is indeed the best performing configuration on tc07 for both gold standards. A look at the synthetic gold standard reveals that p-RDF2vec cannot learn this constructor.

Hypothesis 4b While we assumed that this constructor cannot be learned by any configuration, there is indication that at least to a small extent, classic and p-RDF2vec can learn to recognize the constructor. In both cases, the p-RDF2vec$_{oa}$ configuration achieves the overall best result. The improvement of the order aware component can be explained since only this component can detect the inverse usage of the relationship.

Hypothesis 5 The hypothesis can be accepted. On DBpedia, p-RDF2vec and classic RDF2vec can learn cardinality restrictions. On the synthetic gold standard, this is only true for RDF2vec classic and CBOW p-RDF2vec configurations. From the rather low score (in the 60ies in terms of accuracy), it can be seen that learning cardinality is rather hard.

Hypothesis 6 This hypothesis can only partially be accepted since multiple configurations are capable of learning tc12. What can be concluded when comparing hypothesis 6 to hypothesis 5 is that the addition of the type restriction makes the test cases harder to solve: This can be seen when comparing the scores for tc09 versus tc11 and tc10 versus tc12. e-RDF2vec can surprisingly learn the constructors on DBpedia (even well) – but a look at the synthetic gold standard reveals that it can neither learn tc11 nor tc12 when correlations are mostly removed. This finding is intuitive since e-RDF2vec is unaware of the actual predicates within a graph (it is merely aware of their existence).

~~Results on the DBpedia Gold Standard (Accuracy). The best results are printed in bold. All results are significantly larger than the random baseline. TC SGSG$_{oa}$CBOWCBOW$_{oa}$p-SGp-SG$_{oa}$p-CBOWp-CBOW$_{oa}$e-SGe-SG$_{oa}$e-CBOW$_{oa}$e-CBOW$_{oa}$TransE-L1TransE 0.915 0.937 0.778 0.870 0.907 0.933 0.780 0.924 0.845 0.860 0.840 0.840 0.842 0.947 0.858 0.874 0.862 **0.966** 0.768tc01 hard 0.681 0.891 0.637 0.891 0.627 0.903 0.576 0.894 0.644 0.651 0.659 0.659 0.799 **0.916** 0.744 0.646 0.651 0.830 0.618tc02 0.953 0.961 0.865 0.956 0.930 0.972 0.901 **0.974** 0.883 0.895 0.906 0.906 0.852 0.970 0.832 0.859 0.853 0.908 0.737tc02 hard 0.637 0.780 0.618 0.774 0.628 0.828 0.583 0.838 0.623 0.628 0.607 0.607 0.780 **0.849** 0.693 0.622 0.608 0.729 0.649tc03 0.949 **0.958** 0.846 0.905 0.913 0.956 0.800 0.938 0.883 0.900 0.886 0.886 0.821 0.933 0.856 0.894 0.874 0.943 0.780tc04 0.960 0.968 0.705 0.872 0.877 0.908 0.659 0.873 0.965 0.969 0.915 0.915 0.934 0.986 0.973 0.984 **0.990** 0.990 0.862tc04 hard 0.963 0.984 0.674 **0.992** 0.725 0.828 0.583 0.782 0.938 0.990 0.983 0.983 0.814 0.912 0.855 0.917 0.935 0.918 0.789tc05 0.986 0.992 0.772 0.906 0.869 0.899 0.719 0.870 0.990 **0.995** 0.931 0.931 0.867 0.948 0.881 0.907 0.905 0.908 0.802tc06 0.957 0.963 0.698 0.850 0.876 0.903 0.641 0.857 0.960 0.969 0.928 0.928 0.929 0.985 0.976 0.985 **0.991** 0.990 0.866tc06 hard 0.863 0.936 0.604 0.908 0.708 0.770 0.559 0.745 0.699 0.708 0.650 0.650 0.823 0.779 **0.964** 0.882 0.933 0.964 0.819tc07 0.938 0.955 0.742 0.785 0.895 0.924 0.726 0.863 0.946 0.946 0.859 0.859 0.930 **0.987** 0.978 0.929 0.966 0.945 0.847tc08 0.961 0.966 0.891 0.896 0.911 **0.968** 0.841 0.951 0.904 0.914 0.925 0.925 0.898 0.964 0.870 0.856 0.888 0.875 0.831tc09 0.902 0.901 0.773 0.858 0.819 0.858 0.726 0.832 0.874 0.884 0.840 0.840 0.884 **0.938** 0.879 0.877 0.883 0.929 0.780tc09 hard 0.785 0.793 0.659 0.751 0.698 0.741 0.600 0.712 0.777 0.782 0.744 0.744 0.749 **0.848** 0.758 0.774 0.776 0.820 0.676tc10 0.947 0.958 0.918 0.905 0.924 0.975 0.852 0.969 0.911 0.912 0.925 0.925 0.957 **0.984** 0.898 0.918 0.931 0.927 0.878tc10 hard 0.740 0.737 0.716 0.711 0.610 0.679 0.569 0.652 0.715 0.718 0.729 0.729 **0.775**~~

0.774 0.656 0.743 0.739 0.713 0.665tc11 0.932 0.897 0.865 0.780 0.884 **0.991** 0.808 0.954 0.928 0.972 0.921 0.921 0.917 0.960 0.930 0.889 0.946 0.954 0.838tc11 hard 0.725 0.737 0.687 0.676 0.684 0.707 0.631 0.707 0.763 0.734 0.641 0.641 0.712 **0.806** 0.753 0.666 0.723 0.726 0.638tc12 0.955 0.938 0.888 0.909 0.900 0.971 0.830 0.965 0.893 0.905 0.904 0.904 0.961 **0.984** 0.879 0.912 0.894 0.927 0.834tc12 hard 0.714 0.717 0.712 0.699 0.628 0.637 0.545 0.628 0.690 0.713 0.715 0.715 0.762 **0.765** 0.659 0.714 0.710 0.701 0.652

Results on the Synthetic Gold Standard (Accuracy). The best result for each test case is printed in bold, statistically insignificant scores (w.r.t. a random baseline) are stated in italics. Listed are the results of the best classifier for each task and model. TC SGSG$_{oa}$CBOWCBOW$_{oa}$p-SGp-SG$_{oa}$p-CBOWp-CBOW$_{oa}$e-SGe-SG$_{oa}$e-CBOWe-CBOW$_{o}$ 0.882 0.867 0.566 0.877 0.870 0.842 0.802 0.847 0.774 0.757 0.752 0.727 0.767 0.752 0.712 0.837 0.789 **0.895** 0.769tc02 0.742 0.737 0.769 0.732 **0.822** 0.734 0.769 0.754 *0.536* *0.529* *0.536* *0.529* 0.677 0.677 *0.531* 0.584 *0.549* 0.689 *0.546*tc03 0.797 0.812 **0.927** 0.774 0.794 0.709 0.784 0.742 *0.526* *0.526* *0.561* *0.519* *0.531* 0.581 *0.554* *0.556* *0.536* 0.634 *0.541*tc04 **1.000** 0.998 0.990 0.998 0.568 0.588 0.608 0.628 **1.000** 0.995 **1.000** 0.998 0.790 0.898 0.685 0.588 *0.553* *0.528* 0.728tc05 **0.892** 0.819 0.889 0.819 0.631 0.648 0.681 0.648 0.832 0.819 0.882 0.791 0.691 0.774 0.631 0.658 0.726 0.608 0.646tc06 0.978 0.963 0.898 0.965 0.800 0.828 0.748 0.820 0.970 0.968 0.905 0.965 0.898 0.978 0.888 **1.000** **1.000** **1.000** 0.955tc07 0.583 0.583 0.575 *0.555* *0.553* *0.553* *0.535* *0.540* *0.543* *0.525* *0.498* *0.518* *0.540* 0.615 **0.673** 0.565 *0.518* *0.550* *0.508*tc08 0.563 0.585 *0.555* 0.583 0.635 **0.638** 0.568 0.618 *0.525* *0.533* *0.553* *0.540* 0.585 0.613 *0.540* *0.535* *0.523* *0.533* *0.535*tc09 0.610 0.628 **0.648** 0.605 0.563 *0.550* 0.605 0.590 *0.550* *0.535* *0.508* *0.528* 0.588 *0.543* *0.525* *0.525* *0.545* 0.638 *0.538*tc10 0.638 0.623 **0.665** 0.600 *0.548* *0.560* 0.633 0.565 0.593 0.565 0.568 *0.515* 0.588 0.573 *0.518* *0.525* *0.510* 0.580 *0.533*tc11 0.633 0.580 **0.668** 0.575 0.573 *0.555* 0.580 *0.553* *0.550* *0.545* *0.540* *0.545* 0.583 0.590 0.573 *0.518* 0.590 0.625 *0.538*tc12 0.644 0.614 **0.657** 0.638 0.563 0.565 0.590 0.640 *0.541* 0.568 *0.560* *0.524* 0.618 *0.550* *0.513* *0.553* *0.540* 0.578 *0.533*

8. Conclusion

In this paper, we presented an extensive evaluation of 12 RDF2vec variants and benchmark models using default benchmarks and DLCC, a newly introduced benchmark for description logic constructorsthe established GEval and the newly introduced DLCC benchmark.

The resource DLCC is used to analyze embedding approaches in terms of which kinds of classes they are able to represent. DLCC It comes with an evaluation framework to easily evaluate embeddings using a reproducible protocol. All DLCC components, i.e. the gold standard, the generation framework, and the evaluation framework, are publicly available. Significant efforts were made to comply with the FAIR [64] principles.[43]

By analyzing the performance of different RDF2vec variants on a pattern-by-pattern-basis, the findings of this paper can provide some guidance on which em-

[43]Dataset DOI: `10.5281/zenodo.6509715`; uploaded and indexed via zenodo; published with a permissive license; re-usable; metadata is provided.

bedding method to use for which downstream task. For example, for identifying related items (e.g., for knowledge-based recommender systems [61] or collective entity disambiguation [60]), approaches performing well on tc04 and tc05, like e-RDF2vec, are preferable, while for entity classification based on structural features [70], approaches performing well on tc01-tc03, tc07, and tc08, i.e., mostly the p-RDF2vec variants, are preferable. With such considerations, users of RDF2vec can make more informed decisions on which variant to choose, as an alternative to blindly trying all available variants.

~~We~~ Furthermore, we have shown that many patterns using DL class constructors on DBpedia are actually learned by recognizing patterns with other constructors correlating with the pattern to be learned, thus yielding misleading results. This effect is less prominent in the synthetic gold standard. We showed that certain DL class constructors, ~~particularly~~ especially qualified restrictions and cardinality constraints, are particularly hard to learn. Such insights open an interesting way to new developments in knowledge graph embeddings, since they point to conceptual shortcomings of methods instead of using pure leaderboard-based methods for assessing embedding methods.

In the future, we plan to extend the systematic evaluation by adding more gold standard datasets. The synthetic dataset generator also allows for more interesting experiments: We can systematically analyze the scalability of existing approaches, or study how variations in the synthetic gold standard (e.g., larger and smaller ontologies) influence the outcome.

9. Acknowledgments

The publication of this article was funded by the Ministry of Science, Research and the Arts Baden-Württemberg and the University of Mannheim.

References

[1] P. Ristoski and H. Paulheim, RDF2Vec: RDF Graph Embeddings for Data Mining, in: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, P. Groth, E. Simperl, A.J.G. Gray, M. Sabou, M. Krötzsch, F. Lécué, F. Flöck and Y. Gil, eds, Lecture Notes in Computer Science, Vol. 9981, 2016, pp. 498–514. doi:10.1007/978-3-319-46523-4_30.

[2] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado and J. Dean, Distributed Representations of Words and Phrases and their Compositionality, in: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C.J.C. Burges, L. Bottou, Z. Ghahramani and K.Q. Weinberger, eds, 2013, pp. 3111–3119. `https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html`.

[3] T. Mikolov, K. Chen, G. Corrado and J. Dean, Efficient Estimation of Word Representations in Vector Space, in: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, eds, 2013. `http://arxiv.org/abs/1301.3781`.

[4] J. Portisch, N. Heist and H. Paulheim, Knowledge graph embedding for data mining vs. knowledge graph embedding for link prediction - two sides of the same coin?, *Semantic Web* **13**(3) (2022), 399–422. doi:10.3233/SW-212892.

[5] J. Portisch and H. Paulheim, ALOD2Vec matcher results for OAEI 2021, in: *Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 25, 2021*, P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, O. Hassanzadeh and C. Trojahn, eds, CEUR Workshop Proceedings, Vol. 3063, CEUR-WS.org, 2021, pp. 117–123. http://ceur-ws.org/Vol-3063/oaei21_paper2.pdf.

[6] J. Portisch, M. Hladik and H. Paulheim, Background Knowledge in Schema Matching: Strategy vs. Data, in: *The Semantic Web - ISWC 2021 - 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24-28, 2021, Proceedings*, A. Hotho, E. Blomqvist, S. Dietze, A. Fokoue, Y. Ding, P.M. Barnaghi, A. Haller, M. Dragoni and H. Alani, eds, Lecture Notes in Computer Science, Vol. 12922, Springer, 2021, pp. 287–303. doi:10.1007/978-3-030-88361-4_17.

[7] M. Monych, J. Portisch, M. Hladik and H. Paulheim, DESKMatcher, in: *Proceedings of the 15th International Workshop on Ontology Matching co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 2, 2020*, P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, O. Hassanzadeh and C. Trojahn, eds, CEUR Workshop Proceedings, Vol. 2788, CEUR-WS.org, 2020, pp. 181–186. http://ceur-ws.org/Vol-2788/oaei20_paper7.pdf.

[8] P. Ristoski, J. Rosati, T.D. Noia, R.D. Leone and H. Paulheim, RDF2Vec: RDF graph embeddings and their applications, *Semantic Web* **10**(4) (2019), 721–752. doi:10.3233/SW-180317.

[9] R. Sofronova, R. Biswas, M. Alam and H. Sack, Entity Typing Based on RDF2Vec Using Supervised and Unsupervised Methods, in: *The Semantic Web: ESWC 2020 Satellite Events - ESWC 2020 Satellite Events, Heraklion, Crete, Greece, May 31 - June 4, 2020, Revised Selected Papers*, A. Harth, V. Presutti, R. Troncy, M. Acosta, A. Polleres, J.D. Fernández, J.X. Parreira, O. Hartig, K. Hose and M. Cochez, eds, Lecture Notes in Computer Science, Vol. 12124, Springer, 2020, pp. 203–207. doi:10.1007/978-3-030-62327-2_35.

[10] M. Kejriwal and P.A. Szekely, Supervised typing of big graphs using semantic embeddings, in: *Proceedings of The International Workshop on Semantic Big Data, SBD@SIGMOD 2017, Chicago, IL, USA, May 19, 2017*, S. Groppe and L. Gruenwald, eds, ACM, 2017, pp. 3:1–3:6. doi:10.1145/3066911.3066918.

[11] N. Engleitner, W. Kreiner, N. Schwarz, T. Kopetzky and L. Ehrlinger, Knowledge Graph Embeddings for News Article Tag Recommendation, in: *Joint Proceedings of the Semantics co-located events: Poster&Demo track and Workshop on Ontology-Driven Conceptual Modelling of Digital Twins co-located with Semantics 2021, Amsterdam and Online, September 6-9, 2021*, I. Tiddi, M. Maleshkova, T. Pellegrini and V. de Boer, eds, CEUR Workshop Proceedings, Vol. 2941, CEUR-WS.org, 2021. http://ceur-ws.org/Vol-2941/paper4.pdf.

[12] J. Portisch, M. Hladik and H. Paulheim, FinMatcher at FinSim-2: Hypernym Detection in the Financial Services Domain using Knowledge Graphs, in: *Companion of The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang and L. Zia, eds, ACM / IW3C2, 2021, pp. 293–297. doi:10.1145/3442442.3451382.

[13] B. Steenwinckel, G. Vandewiele, I. Rausch, P. Heyvaert, R. Taelman, P. Colpaert, P. Simoens, A. Dimou, F.D. Turck and F. Ongenae, Facilitating the Analysis of COVID-19 Literature Through a Knowledge Graph, in: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II*, J.Z. Pan, V.A.M. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne and L. Kagal, eds, Lecture Notes in Computer Science, Vol. 12507, Springer, 2020, pp. 344–357. doi:10.1007/978-3-030-62466-8_22.

[14] J. Loesch, L. Meeckers, I. van Lier, A. de Boer, M. Dumontier and R. Celebi, Automated Identification of Food Substitutions Using Knowledge Graph Embeddings, in: *13th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences, SWAT4HCLS 2022, Virtual Event, Leiden, The Netherlands, January 10th to 14th, 2022*, K. Wolstencroft, A. Splendiani, M.S. Marshall, C. Baker, A. Waag-

meester, M. Roos, R.A. Vos, R. Fijten and L.J. Castro, eds, CEUR Workshop Proceedings, Vol. 3127, CEUR-WS.org, 2022, pp. 19–28. http://ceur-ws.org/Vol-3127/paper-3.pdf.

[15] J. Portisch and H. Paulheim, Walk this Way! Entity Walks and Property Walks for RDF2vec, *CoRR* **abs/2204.02777** (2022). doi:10.48550/arXiv.2204.02777.

[16] J. Portisch and H. Paulheim, Putting RDF2vec in Order, in: *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*, O. Seneviratne, C. Pesquita, J. Sequeda and L. Etcheverry, eds, CEUR Workshop Proceedings, Vol. 2980, CEUR-WS.org, 2021. http://ceur-ws.org/Vol-2980/paper352.pdf.

[17] J. Portisch and H. Paulheim, The DLCC Node Classification Benchmark for Analyzing Knowledge Graph Embeddings, in: *International Semantic Web Conference*, Springer, 2022, pp. 592–609.

[18] H. Cai, V.W. Zheng and K.C. Chang, A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications, *IEEE Trans. Knowl. Data Eng.* **30**(9) (2018), 1616–1637. doi:10.1109/TKDE.2018.2807452.

[19] M. Xu, Understanding Graph Embedding Methods and Their Applications, *SIAM Rev.* **63**(4) (2021), 825–853. doi:10.1137/20M1386062.

[20] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier and G. Bouchard, Complex Embeddings for Simple Link Prediction, in: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, M. Balcan and K.Q. Weinberger, eds, JMLR Workshop and Conference Proceedings, Vol. 48, JMLR.org, 2016, pp. 2071–2080. http://proceedings.mlr.press/v48/trouillon16.html.

[21] Z. Sun, Z. Deng, J. Nie and J. Tang, RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space, in: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019. https://openreview.net/forum?id=HkgEQnRqYQ.

[22] Y. Dai, S. Wang, N.N. Xiong and W. Guo, A survey on knowledge graph embedding: Approaches, applications and benchmarks, *Electronics* **9**(5) (2020), 750. doi:10.3390/electronics9050750.

[23] Q. Wang, Z. Mao, B. Wang and L. Guo, Knowledge Graph Embedding: A Survey of Approaches and Applications, *IEEE Trans. Knowl. Data Eng.* **29**(12) (2017), 2724–2743. doi:10.1109/TKDE.2017.2754499.

[24] M. Nickel, V. Tresp and H. Kriegel, A Three-Way Model for Collective Learning on Multi-Relational Data, in: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, L. Getoor and T. Scheffer, eds, Omnipress, 2011, pp. 809–816. https://icml.cc/2011/papers/438_icmlpaper.pdf.

[25] A. Grover and J. Leskovec, node2vec: Scalable Feature Learning for Networks, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A.J. Smola, C.C. Aggarwal, D. Shen and R. Rastogi, eds, ACM, 2016, pp. 855–864. doi:10.1145/2939672.2939754.

[26] B. Perozzi, R. Al-Rfou and S. Skiena, DeepWalk: online learning of social representations, in: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, S.A. Macskassy, C. Perlich, J. Leskovec, W. Wang and R. Ghani, eds, ACM, 2014, pp. 701–710. doi:10.1145/2623330.2623732.

[27] A. Bordes, N. Usunier, A. García-Durán, J. Weston and O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, in: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C.J.C. Burges, L. Bottou, Z. Ghahramani and K.Q. Weinberger, eds, 2013, pp. 2787–2795. https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

[28] Y. Lin, Z. Liu, M. Sun, Y. Liu and X. Zhu, Learning Entity and Relation Embeddings for

Knowledge Graph Completion, in: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, B. Bonet and S. Koenig, eds, AAAI Press, 2015, pp. 2181–2187. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

[29] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.

[30] C. Fellbaum (ed.), *WordNet: An Electronic Lexical Database*, Language, Speech, and Communication, MIT Press, Cambridge, Massachusetts, 1998. ISBN ISBN 978-0-262-06197-1. doi:10.7551/mitpress/7287.001.0001.

[31] K. Toutanova and D. Chen, Observed versus latent features for knowledge base and text inference, in: *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, 2015, pp. 57–66. doi:10.18653/v1/W15-4007.

[32] T. Dettmers, P. Minervini, P. Stenetorp and S. Riedel, Convolutional 2D Knowledge Graph Embeddings, in: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, S.A. McIlraith and K.Q. Weinberger, eds, AAAI Press, 2018, pp. 1811–1818. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17366.

[33] B. Shi and T. Weninger, Open-World Knowledge Graph Completion, in: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, S.A. McIlraith and K.Q. Weinberger, eds, AAAI Press, 2018, pp. 1957–1964. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16055.

[34] F. Alshargi, S. Shekarpour, T. Soru and A.P. Sheth, Metrics for Evaluating Quality of Embeddings for Ontological Concepts, in: *Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019) Stanford University, Palo Alto, California, USA, March 25-27, 2019., Stanford University, Palo Alto, California, USA, March 25-27, 2019*, A. Martin, K. Hinkelmann, A. Gerber, D. Lenat, F. van Harmelen and P. Clark, eds, CEUR Workshop Proceedings, Vol. 2350, CEUR-WS.org, 2019. http://ceur-ws.org/Vol-2350/paper26.pdf.

[35] P. Ristoski, G.K.D. de Vries and H. Paulheim, A Collection of Benchmark Datasets for Systematic Evaluations of Machine Learning on the Semantic Web, in: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II*, P. Groth, E. Simperl, A.J.G. Gray, M. Sabou, M. Krötzsch, F. Lécué, F. Flöck and Y. Gil, eds, Lecture Notes in Computer Science, Vol. 9982, 2016, pp. 186–194. doi:10.1007/978-3-319-46547-0_20.

[36] M.A. Pellegrino, A. Altabba, M. Garofalo, P. Ristoski and M. Cochez, GEval: A Modular and Extensible Evaluation Framework for Graph Embedding Techniques, in: *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, A. Harth, S. Kirrane, A.N. Ngomo, H. Paulheim, A. Rula, A.L. Gentile, P. Haase and M. Cochez, eds, Lecture Notes in Computer Science, Vol. 12123, Springer, 2020, pp. 565–582. doi:10.1007/978-3-030-49461-2_33.

[37] M.A. Pellegrino, M. Cochez, M. Garofalo and P. Ristoski, A Configurable Evaluation Framework for Node Embedding Techniques, in: *The Semantic Web: ESWC 2019 Satellite Events - ESWC 2019 Satellite Events, Portorož, Slovenia, June 2-6, 2019, Revised Selected Papers*, P. Hitzler, S. Kirrane, O. Hartig, V. de Boer, M. Vidal, M. Maleshkova, S. Schlobach, K. Hammar, N. Lasierra, S. Stadtmüller, K. Hose and R. Verborgh, eds, Lecture Notes in Computer Science, Vol. 11762, Springer, 2019, pp. 156–160. doi:10.1007/978-3-030-32327-1_31.

[38] A. Melo and H. Paulheim, Synthesizing Knowledge Graphs for Link and Type Prediction Benchmarking, in: *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*, E. Blomqvist, D. Maynard, A. Gangemi, R. Hoekstra, P. Hitzler and O. Hartig, eds, Lecture Notes in Computer

Science, Vol. 10249, 2017, pp. 136–151. doi:10.1007/978-3-319-58068-5_9.
[39] P. Bloem, X. Wilcke, L. van Berkel and V. de Boer, kgbench: A Collection of Knowledge Graph Datasets for Evaluating Relational and Multimodal Machine Learning, in: *The Semantic Web - 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings*, R. Verborgh, K. Hose, H. Paulheim, P. Champin, M. Maleshkova, Ó. Corcho, P. Ristoski and M. Alam, eds, Lecture Notes in Computer Science, Vol. 12731, Springer, 2021, pp. 614–630. doi:10.1007/978-3-030-77385-4_37.
[40] H. Paulheim, J. Portisch and P. Ristoski, *Embedding Knowledge Graphs with RDF2vec*, Springer, 2023, to appear.
[41] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, Enriching word vectors with subword information, *Transactions of the association for computational linguistics* **5** (2017), 135–146.
[42] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* (2018).
[43] T. Agozzino, A Trip to Sesame Street: Evaluation of BERT and Other Recent Embedding echniques Within RDF2Vec, 2021, Master's thesis at Ghent University.
[44] M. Cochez, P. Ristoski, S.P. Ponzetto and H. Paulheim, Biased graph walks for RDF graph embeddings, in: *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, WIMS 2017, Amantea, Italy, June 19-22, 2017*, R. Akerkar, A. Cuzzocrea, J. Cao and M. Hacid, eds, ACM, 2017, pp. 21:1–21:12. doi:10.1145/3102254.3102279.
[45] A.A. Taweel and H. Paulheim, Towards Exploiting Implicit Human Feedback for Improving RDF2vec Embeddings, in: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2020) co-located with the 17th Extended Semantic Web Conference 2020 (ESWC 2020), Heraklion, Greece, June 02, 2020 - moved online*, M. Alam, D. Buscaldi, M. Cochez, F. Osborne, D.R. Recupero and H. Sack, eds, CEUR Workshop Proceedings, Vol. 2635, CEUR-WS.org, 2020. `http://ceur-ws.org/Vol-2635/paper1.pdf`.
[46] W. Ling, C. Dyer, A.W. Black and I. Trancoso, Two/Too Simple Adaptations of Word2Vec for Syntax Problems, in: *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, R. Mihalcea, J.Y. Chai and A. Sarkar, eds, The Association for Computational Linguistics, 2015, pp. 1299–1304. doi:10.3115/v1/n15-1142.
[47] J. Portisch, M. Hladik and H. Paulheim, RDF2Vec Light - A Lightweight Approachfor Knowledge Graph Embeddings, in: *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)*, K.L. Taylor, R.S. Gonçalves, F. Lécué and J. Yan, eds, CEUR Workshop Proceedings, Vol. 2721, CEUR-WS.org, 2020, pp. 79–84. `http://ceur-ws.org/Vol-2721/paper520.pdf`.
[48] A. Budanitsky and G. Hirst, Evaluating WordNet-based Measures of Lexical Semantic Relatedness, *Comput. Linguistics* **32**(1) (2006), 13–47. doi:10.1162/coli.2006.32.1.13.
[49] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman and E. Ruppin, Placing search in context: the concept revisited, *ACM Trans. Inf. Syst.* **20**(1) (2002), 116–131. doi:10.1145/503104.503110.
[50] F. Hill, R. Reichart and A. Korhonen, SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation, *Comput. Linguistics* **41**(4) (2015), 665–695. doi:10.1162/COLI_a_00237.
[51] M.D. Lee, B. Pincombe and M. Welsh, An Empirical Evaluation of Models of Text Document Similarity, *Proceedings of the Annual Meeting of the Cognitive Science Society* **7**(7) (2005), 1254–1529. `https://hdl.handle.net/2440/28910`.
[52] P.N. Mendes, M. Jakob, A. García-Silva and C. Bizer, DBpedia spotlight: shedding light on the web of documents, in: *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, C. Ghidini, A.N. Ngomo, S.N. Lindstaedt and T. Pellegrini, eds, ACM International Conference Proceeding Series, ACM, 2011, pp. 1–8. doi:10.1145/2063518.2063519.
[53] J. Hoffart, S. Seufert, D.B. Nguyen, M. Theobald and G. Weikum, KORE: keyphrase

overlap relatedness for entity disambiguation, in: *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, X. Chen, G. Lebanon, H. Wang and M.J. Zaki, eds, ACM, 2012, pp. 545–554. doi:10.1145/2396761.2396832.

[54] H. Paulheim and J. Fümkranz, Unsupervised generation of data mining features from linked open data, in: *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, 2012, pp. 1–12.

[55] P. Ristoski and H. Paulheim, A comparison of propositionalization strategies for creating features from linked open data, *Linked Data for Knowledge Discovery* **6** (2014).

[56] N. Lavrač, B. Škrlj and M. Robnik-Šikonja, Propositionalization and embeddings: two sides of the same coin, *Machine Learning* **109** (2020), 1465–1507.

[57] H. Paulheim and C. Bizer, Type inference on noisy RDF data, in: *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I 12*, Springer, 2013, pp. 510–525.

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

[59] M.A. Raza, R. Mokhtar, N. Ahmad, M. Pasha and U. Pasha, A taxonomy and survey of semantic approaches for query expansion, *IEEE Access* **7** (2019), 17823–17833.

[60] I.L. Oliveira, R. Fileto, R. Speck, L.P. Garcia, D. Moussallem and J. Lehmann, Towards holistic entity linking: Survey and directions, *Information Systems* **95** (2021), 101624.

[61] A. Iana, M. Alam and H. Paulheim, A survey on knowledge-aware news recommender systems, *Semantic Web* (2022).

[62] M. Färber and A. Jatowt, Citation recommendation: approaches and datasets, *International Journal on Digital Libraries* **21**(4) (2020), 375–405.

[63] H. Paulheim, Generating Possible Interpretations for Statistics from Linked Open Data., in: *ESWC*, 2012, pp. 560–574.

[64] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne et al., The FAIR Guiding Principles for scientific data management and stewardship, *Scientific data* **3**(1) (2016), 1–9. doi:10.1038/sdata.2016.18.

[65] Y. Guo, Z. Pan and J. Heflin, LUBM: A benchmark for OWL knowledge base systems, *Journal of Web Semantics* **3**(2–3) (2005), 158–182.

[66] N. Heist, S. Hertling, D. Ringler and H. Paulheim, Knowledge Graphs on the Web - An Overview, in: *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*, I. Tiddi, F. Lécué and P. Hitzler, eds, Studies on the Semantic Web, Vol. 47, IOS Press, 2020, pp. 3–22. doi:10.3233/SSW200009.

[67] J. Portisch, M. Hladik and H. Paulheim, KGvec2go - Knowledge Graph Embeddings as a Service, in: *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk and S. Piperidis, eds, European Language Resources Association, 2020, pp. 5641–5647. `https://aclanthology.org/2020.lrec-1.692/`.

[68] D. Zheng, X. Song, C. Ma, Z. Tan, Z. Ye, J. Dong, H. Xiong, Z. Zhang and G. Karypis, DGL-KE: Training Knowledge Graph Embeddings at Scale, in: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen and Y. Liu, eds, ACM, 2020, pp. 739–748. doi:10.1145/3397271.3401172.

[69] S. Salzberg, On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach, *Data Min. Knowl. Discov.* **1**(3) (1997), 317–328. doi:10.1023/A:1009752403260.

[70] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic web* **8**(3) (2017), 489–508.

A. Creation of DBpedia based Gold Standard

Tables 8, 9 and 10 show the queries which are used to create the gold standard for the class *Person* from DBpedia.

Hyp.	TC	Query Positive	Query Negative (normal)	Query Negative (hard)

H1a	tc01	`SELECT DISTINCT(?x) WHERE {`
		`  ?x a dbo:Person .`
		`  ?x dbo:child ?y . }`
			`SELECT DISTINCT(?x) WHERE {`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS {`
			`  ?x dbo:child ?z})}`
				`SELECT DISTINCT(?x) WHERE {`
				`  ?x a dbo:Person .`
				`  ?y dbo:child ?x.`
				`  FILTER(NOT EXISTS {`
				`  ?x dbo:child ?z})}`

H1a'	tc02	`SELECT DISTINCT(?x) WHERE {`
		`  ?x a dbo:Person .`
		`  ?y dbo:child ?x . }`
			`SELECT DISTINCT(?x) WHERE {`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS {`
			`  ?y dbo:child ?x})}`
				`SELECT DISTINCT(?x) WHERE {`
				`  ?x a dbo:Person .`
				`  ?x dbo:child ?y.`
				`  FILTER(NOT EXISTS {`
				`  ?z dbo:child ?x})}`

H1b	tc03	`SELECT DISTINCT(?x) WHERE {`
		`{ ?x a dbo:Person .`
		`  ?x dbo:child ?y} UNION`
		`{ ?x a dbo:Person .`
		`  ?y dbo:child ?x}}`
			`SELECT COUNT(?x) WHERE {`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS{`
			`  ?x dbo:child ?y}`
			`  AND NOT EXISTS {`
			`  ?z dbo:child ?x})}`
				–

H2a	tc04	`SELECT DISTINCT(?x) WHERE {`
		`{ ?x a dbo:Person .`
		`  ?x ?y dbr:New_York_City} UNION`
		`{ ?x a dbo:Person .`
		`  dbr:New_York_City ?y ?x}}`
			`SELECT DISTINCT(?x) WHERE {`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS{`
			`  ?x ?y dbr:New_York_City}`
			`  AND NOT EXISTS {`
			`  dbr:New_York_City ?y ?x})}`
				`SELECT DISTINCT(?x) WHERE {{`
				`  ?x a dbo:Person .`
				`  ?x ?y1 ?z .`
				`  ?z ?y2 dbr:New_York_City }`
				`  UNION {`
				`  ?x a dbo:Person .`
				`  ?z ?y1 ?x .`
				`  dbr:New_York_City ?y2 ?z }`
				`  FILTER(NOT EXISTS`
				`  {?x ?r dbr:New_York_City}`
				`  AND NOT EXISTS`
				`  {dbr:New_York_City ?s ?x})}`

H2b	tc05	`SELECT DISTINCT(?x) WHERE {{`
		`  ?x a dbo:Person .`
		`  ?x ?y1 ?z .`
		`  ?z ?y2 dbr:New_York_City }`
		`UNION {`
		`  ?x a dbo:Person .`
		`  ?z ?y1 ?x .`
		`  dbr:New_York_City ?y2 ?z }}`
			–	–

Table 8. Test cases for class `Person`, Hypotheses 1 and 2 / tc01 - tc05

Hyp.	TC	Query Positive	Query Negative (normal)	Query Negative (hard)

H3	tc06	`SELECT DISTINCT(?x) WHERE {`
		`  ?x a dbo:Person .`
		`  ?x dbo:birthPlace`
		`  dbr:New_York_City }`
			`SELECT DISTINCT(?x) WHERE {`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS{`
			`  ?x dbo:birthPlace`
			`  dbr:New_York_City })}`
				`SELECT DISTINCT(?x) ?r WHERE {{`
				`  ?x a dbo:Person .`
				`  ?x dbo:birthPlace ?y .`
				`  dbr:New_York_City ?r ?x .`
				`  FILTER(?y!=dbr:New_York_City)}`
				`  UNION {`
				`  ?x a dbo:Person .`
				`  ?x dbo:birthPlace ?y .`
				`  ?x ?r dbr:New_York_City .`
				`  FILTER(?y!=dbr:New_York_City)}}`

H4a	tc07	`SELECT DISTINCT(?x) WHERE {`
		`  ?x a dbo:Person .`
		`  ?x dbo:team ?y .`
		`  ?y a dbo:BasketballTeam }`
			`SELECT DISTINCT(?x) WHERE {`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS{`
			`  ?x dbo:team ?y .`
			`  ?y a dbo:BasketballTeam})}`
				`SELECT DISTINCT(?x) WHERE {`
				`  ?x a dbo:Person .`
				`  ?x dbo:team ?z1 .`
				`  ?x ?r ?z2 .`
				`  ?z2 a dbo:BaseballTeam`
				`  FILTER(NOT EXISTS{`
				`  ?x dbo:team ?y .`
				`  ?y a dbo:BasketballTeam })}`

H4b	tc08	`SELECT DISTINCT(?x) WHERE {`
		`  ?x a dbo:Person .`
		`  ?y dbo:starring ?x .`
		`  ?y a dbo:TelevisionShow .}`
			`SELECT DISTINCT(?x) WHERE {`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS{`
			`  ?y dbo:starring ?x .`
			`  ?y a dbo:TelevisionShow .})}`
				`SELECT DISTINCT(?x) WHERE {`
				`  ?x a dbo:Person .`
				`  ?z1 dbo:starring ?x .`
				`  ?z2 ?r ?x .`
				`  ?z2 a dbo:TelevisionShow`
				`  FILTER(NOT EXISTS{`
				`  ?y dbo:starring ?x .`
				`  ?y a dbo:TelevisionShow .})}`

Table 9. Test cases for class `Person`, Hypotheses 3 and 4

Hyp.	TC	Query Positive	Query Negative (normal)	Query Negative (hard)

H5	tc09	`SELECT DISTINCT(?x) WHERE {`
		`  ?x a dbo:Person .`
		`  ?x dbo:award ?y1.`
		`  ?x dbo:award ?y2.`
		`  FILTER(?y1!=?y2)}`
			`SELECT DISTINCT(?x) WHERE {`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS{`
			`  ?x dbo:award ?y1.`
			`  ?x dbo:award ?y2.`
			`  FILTER(?y1!=?y2)})}`
				`SELECT DISTINCT(?x) WHERE {`
				`  ?x a dbo:Person .`
				`  ?x dbo:award ?y .`
				`  FILTER(NOT EXISTS{`
				`  ?x dbo:award ?z.`
				`  FILTER(?y!=?z)})}`

H5'	tc10	`SELECT DISTINCT(?x) WHERE{`
		`  ?x a dbo:Person .`
		`  ?y1 dbo:director ?x .`
		`  ?y2 dbo:director ?x .`
		`  FILTER(?y1!=?y2)`
		`}`
			`SELECT DISTINCT(?x) WHERE{`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS{`
			`  ?y1 dbo:director ?x .`
			`  ?y2 dbo:director ?x .`
			`  FILTER(?y1!=?y2)})}`
				`SELECT DISTINCT(?x) WHERE {`
				`  ?x a dbo:Person .`
				`  ?y dbo:director ?x .`
				`  FILTER(NOT EXISTS{`
				`  ?z dbo:director ?x.`
				`  FILTER(?y!=?z)})}`

H6a	tc11	`SELECT DISTINCT(?x) WHERE {`
		`  ?x a dbo:Person .`
		`  ?x dbo:recordLabel ?y1 .`
		`  ?y1 a dbo:RecordLabel .`
		`  ?x dbo:recordLabel ?y2 .`
		`  ?y2 a dbo:RecordLabel .`
		`  FILTER(?y1!=?y2)}`
			`SELECT DISTINCT(?x) WHERE {`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS{`
			`  ?x dbo:recordLabel ?y1 .`
			`  ?y1 a dbo:RecordLabel .`
			`  ?x dbo:recordLabel ?y2 .`
			`  ?y2 a dbo:RecordLabel .`
			`  FILTER(?y1!=?y2)})}`
				`SELECT DISTINCT(?x) WHERE {`
				`  ?x a dbo:Person .`
				`  ?x dbo:recordLabel ?y1 .`
				`  ?y1 a dbo:RecordLabel .`
				`  FILTER(NOT EXISTS{`
				`  ?x dbo:recordLabel ?y2 .`
				`  ?y2 a dbo:RecordLabel .`
				`  FILTER(?y1!=?y2)})}`

H6b	tc12	`SELECT DISTINCT(?x) WHERE{`
		`  ?x a dbo:Person .`
		`  ?y1 dbo:director ?x .`
		`  ?y1 a dbo:Film .`
		`  ?y2 dbo:director ?x .`
		`  ?y2 a dbo:Film .`
		`  FILTER(?y1!=?y2)}`
			`SELECT DISTINCT(?x) WHERE{`
			`  ?x a dbo:Person .`
			`  FILTER(NOT EXISTS{`
			`  ?y1 dbo:director ?x .`
			`  ?y1 a dbo:Film .`
			`  ?y2 dbo:director ?x .`
			`  ?y2 a dbo:Film .`
			`  FILTER(?y1!=?y2)})}`
				`SELECT DISTINCT(?x) WHERE{`
				`  ?x a dbo:Person .`
				`  ?y1 dbo:director ?x .`
				`  ?y1 a dbo:Film .`
				`  FILTER(NOT EXISTS{`
				`  ?y2 dbo:director ?x .`
				`  ?y2 a dbo:Film .`
				`  FILTER(?y1!=?y2)})}`

Table 10. Test cases for class `Person`, Hypotheses 5 and 6