Author(s): Cochez, Michael; Ristoski, Petar; Ponzetto, Simone Paolo; Paulheim, Heiko

Title: Biased GraphWalks for RDF Graph Embeddings

Year: 2017

Version:

Please cite the original version:

Cochez, M., Ristoski, P., Ponzetto, S. P., & Paulheim, H. (2017). Biased GraphWalks for RDF Graph Embeddings. In WIMS '17 : Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics (Article 21). ACM. https://doi.org/10.1145/3102254.3102279

Biased Graph Walks for RDF Graph Embeddings

Michael Cochez
Fraunhofer
Institute for Applied Information Technology FIT
53757, Sankt Augustin, Germany
michael.cochez@fit.fraunhofer.de
RWTH University Aachen
Informatik 5
52062, Aachen, Germany
University of Jyvaskyla
Faculty of Information Technology
40014, Jyvaskyla, Finland

Petar Ristoski
University of Mannheim
Data and Web Science Group
68159, Mannheim, Germany
petar.ristoski@informatik.uni-mannheim.de

Simone Paolo Ponzetto
University of Mannheim
Data and Web Science Group
68159, Mannheim, Germany
simone@informatik.uni-mannheim.de

Heiko Paulheim
University of Mannheim
Data and Web Science Group
68159, Mannheim, Germany
heiko@informatik.uni-mannheim.de

ABSTRACT

Knowledge Graphs have been recognized as a valuable source for background information in many data mining, information retrieval, natural language processing, and knowledge extraction tasks. However, obtaining a suitable feature vector representation from RDF graphs is a challenging task. In this paper, we extend the RDF2Vec approach, which leverages language modeling techniques for unsupervised feature extraction from sequences of entities. We generate sequences by exploiting local information from graph substructures, harvested by graph walks, and learn latent numerical representations of entities in RDF graphs. We extend the way we compute feature vector representations by comparing twelve different edge weighting functions for performing biased walks on the RDF graph, in order to generate higher quality graph embeddings. We evaluate our approach using different machine learning, as well as entity and document modeling benchmark data sets, and show that the naive RDF2Vec approach can be improved by exploiting Biased Graph Walks.

CCS CONCEPTS

•**Information systems** → **Data mining;** *Semantic web description languages;* •**Computing methodologies** → *Unsupervised learning;* •**Theory of computation** → Graph algorithms analysis;

KEYWORDS

Graph Embeddings, Linked Open Data, Data Mining

ACM Reference format:
Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. 2017. Biased Graph Walks for RDF Graph Embeddings. In *Proceedings of 7th ACM International Conference on Web Intelligence, Mining and Semantics, Amantea, Italy, June 19–22, 2017 (WIMS '17),* 0 pages.
DOI: 10.1145/3102254.3102279

1 INTRODUCTION

Linked Open Data (LOD) [40], and in particular large-scale, cross-domain knowledge graphs such as DBpedia [17], have been recognized as a valuable source of background knowledge in many data mining tasks and knowledge discovery in general [37]. Augmenting a dataset with features taken from knowledge graphs can, in many cases, improve the results of a data mining problem at hand, while externalizing the cost of maintaining that background knowledge [28, 37].

Most data mining algorithms work with a propositional *feature vector* representation of the data, i.e., each instance is represented as a vector of features $\langle f_1, f_2, ..., f_n \rangle$, where the features are either binary (i.e., $f_i \in \{true, false\}$), numerical (i.e., $f_i \in \mathbb{R}$), or nominal (i.e., $f_i \in S$, where S is a finite set of symbols). Knowledge graphs, however, are graphs by nature, connecting resources with types and relations, backed by a schema or ontology.

Thus, for accessing knowledge graphs with existing data mining tools and algorithms, transformations have to be performed, which create propositional features from the graphs, i.e., a process called *propositionalization* [15]. Usually, binary features (e.g., `true` if a type or relation exists, `false` otherwise) or numerical features (e.g., counting the number of relations of a certain type) are used [30, 35]. Other variants, e.g., counting different graph sub-structures are possible [6], which are not suitable for large datasets or graphs, as they produce a large number of features.

WIMS '17, Amantea, Italy

978-1-4503-5225-3/17/06…$15.00
DOI: 10.1145/3102254.3102279

In [36], we have introduced *RDF2Vec*, a generic method for embedding entities in knowledge graphs into lower-dimensional vector spaces. The approach adapts neural language modeling techniques, specifically word2vec, which takes sequences of words to embed words into vector spaces [20, 21]. We have shown that it is possible to compute and reuse such embeddings for large-scale knowledge graphs.

For adapting word2vec for knowledge graphs, the first step is to extract meaningful sequences of entities from a knowledge graph, which capture the surrounding knowledge of each entity. Our results in [36] have shown that *random walks* are a feasible and, in contrast to other techniques such as kernels, also a well scalable approach for extracting sequences.

In this paper, we examine methods to direct the random walks in more meaningful ways, i.e., being able to capture more important information about each entity in the graph. We test a dozen weighting schemes which influence the walks and, thus, the resulting sequences. The experiments show that the choice of weights has a crucial influence on the utility of the resulting embeddings.

The rest of this paper is structured as follows. In Section 2, we give an overview of related work. In Section 3, we provide a short introduction to neural language models. In Section 4, we introduce our approach, followed by an evaluation in Section 5. We conclude with a summary and an outlook on future work.

2 RELATED WORK

In the recent past, a few approaches for generating data mining features from Linked Open Data have been proposed. Many of those approaches are supervised, i.e., they let the user formulate SPARQL queries, and a fully automatic feature generation is not possible. LiDDM [13] allows the users to declare SPARQL queries for retrieving features from LOD that can be used in different machine learning techniques. Similarly, Cheng et al. [3] propose an approach for feature generation which requires the user to specify SPARQL queries. A similar approach has been used in the RapidMiner[1] semweb plugin [14], which preprocesses RDF data in a way that it can be further processed directly in RapidMiner. Mynarz et al. [23] have considered using user specified SPARQL queries in combination with SPARQL aggregates.

FeGeLOD [30] and its successor, the *RapidMiner Linked Open Data Extension* [33], have been the first fully automatic unsupervised approach for enriching data with features that are derived from LOD. The approach uses six different unsupervised feature generation strategies, exploring specific or generic relations. It has been shown that such feature generation strategies can be used in many data mining tasks [31, 33].

A similar problem is handled by *Kernel functions*, which compute the distance between two data instances by counting common substructures in the graphs of the instances, i.e. walks, paths, and trees. In the past, many graph kernels have been proposed that are tailored towards specific applications [12], or towards specific semantic representations [9]. Only a few approaches are general enough to be applied on any given RDF data, regardless the data mining task. Lösch et al. [19] introduce two general RDF graph kernels, based on intersection graphs and intersection trees. Later, the intersection tree path kernel was simplified by de Vries et al. [5]. In another work, de Vries et al. [4, 6] introduce an approximation of the state-of-the-art Weisfeiler-Lehman graph kernel algorithm aimed at improving the computational time of the kernel when applied to RDF. Furthermore, the kernel implementation allows for explicit calculation of the instances' feature vectors, instead of pairwise similarities.

The RDF2Vec approach is closely related to the approaches DeepWalk [32] and Deep Graph Kernels [45]. DeepWalk uses language modeling approaches to learn social representations of vertices of graphs by modeling short random-walks on large social graphs, like BlogCatalog, Flickr, and YouTube. The Deep Graph Kernel approach extends the DeepWalk approach, by modeling graph substructures, like graphlets, instead of graph walks. In this paper, we pursue and deepen the idea of random and biased walks since those, unlike other transformation approaches, such as graph kernels, walks have proven to be scalable even to large RDF graphs. Node2vec [10] is another approach very similar to DeepWalk, which uses second order random walks to preserve the network neighborhood of the nodes.

Furthermore, multiple approaches for knowledge graph embeddings for the task of link prediction have been proposed [25]. These could also be considered as approaches for generating propositional features from graphs. RESCAL [26] is one of the earliest approaches, which is based on factorization of a three-way tensor. The approach is later extended into Neural Tensor Networks (NTN) [41] which can be used for the same purpose. One of the most successful approaches is the model based on translating embeddings, TransE [1]. This model builds entity and relation embeddings by regarding a relation as a translation from head entity to tail entity. This approach assumes some relationships between words could be computed by their vector difference in the embedding space. However, this approach cannot deal with reflexive, one-to-many, many-to-one, and many-to-many relations. This problem was resolved in the TransH model [44], which models a relation as a hyperplane together with a translation operation on it. More precisely, each relation is characterized by two vectors, the norm vector of the hyperplane, and the translation vector on the hyperplane. While both TransE and TransH, embed the relations and the entities in the same semantic space, the TransR model [18] builds entity and relation embeddings in separate entity space and multiple relation spaces. This approach is able to model entities that have multiple aspects, and various relations that focus on different aspects of entities.

3 PRELIMINARIES

Neural language models have been developed in the NLP field as an alternative to represent texts as a bag of words, and hence, a binary feature vector, where each vector index represents one word. While such approaches are simple and robust, they suffer from several drawbacks, e.g., high dimensionality and severe data sparsity, which limits the performance of such techniques. To overcome such limitations, neural language models have been proposed, inducing low-dimensional, distributed embeddings of words by means of neural networks. The goal of such approaches is to estimate the likelihood of a specific sequence of words appearing in a corpus,

[1] http://www.rapidminer.com/

explicitly modeling the assumption that closer words in the word sequence are statistically more dependent.

While some of the initially proposed approaches suffered from inefficient training of the neural network models, with the recent advancements in the field several efficient approaches has been proposed. One of the most popular and widely used is the word2vec neural language model [20, 21]. Word2vec is a particularly computationally-efficient two-layer neural net model for learning word embeddings from raw text. There are two different algorithms, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. We will introduce both briefly. A more elaborated discussion can be found from the original RDF2Vec paper [36].

3.1 Continuous Bag-of-Words Model

The CBOW model predicts target words from context words within a given window. The model architecture is shown in fig. 1. The input layer comprises all the surrounding words for which the input vectors are retrieved from the input weight matrix, averaged, and projected in the projection layer. Then, using the weights from the output weight matrix, a score for each word in the vocabulary is computed, which is the probability of the word being a target word. Formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, and a context window c, the objective of the CBOW model is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T} log\, p(w_t|w_{t-c}...w_{t+c}), \tag{1}$$

where the probability $p(w_t|w_{t-c}...w_{t+c})$ is calculated using the softmax function:

$$p(w_t|w_{t-c}...w_{t+c}) = \frac{exp(\bar{v}^T v'_{w_t})}{\sum_{w=1}^{|V|} exp(\bar{v}^T v'_w)}, \tag{2}$$

where v'_w is the output vector of the word w, V is the complete vocabulary of words, and $\bar{v}$ is the averaged input vector of all the context words:

$$\bar{v} = \frac{1}{2c}\sum_{-c\leq j\leq c, j\neq 0} v_{w_{t+j}} \tag{3}$$

3.2 Skip-Gram Model

The skip-gram model does the inverse of the CBOW model and tries to predict the context words from the target words (Fig. 2). More formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, and a context window c, the objective of the skip-gram model is to maximize the following average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} log\, p(w_{t+j}|w_t), \tag{4}$$

where the probability $p(w_{t+j}|w_t)$ is calculated using the softmax function:

$$p(w_o|w_i) = \frac{exp(v'^T_{wo} v_{wi})}{\sum_{w=1}^{|V|} exp(v'^T_w v_{wi})}, \tag{5}$$

where v_w and v'_w are the input and the output vector of the word w, and V is the complete vocabulary of words.

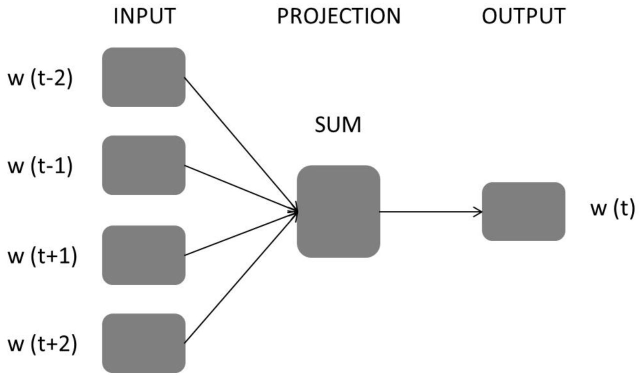

Figure 1: CBOW architecture

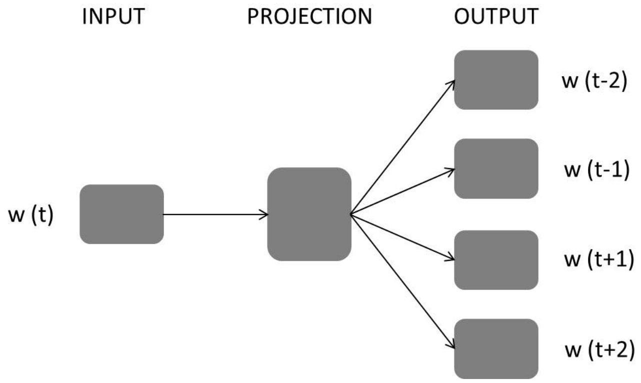

Figure 2: Skip-gram architecture

In both cases, calculating the softmax function is computationally inefficient, as the cost for computing is proportional to the size of the vocabulary. Therefore, two optimization techniques have been proposed, i.e., hierarchical softmax and negative sampling [21]. Empirical studies have shown that in most cases negative sampling leads to a better performance than hierarchical softmax, which depends on the selected negative samples, but it has higher runtime.

Once the training is finished, all words (or, in our case, entities from the knowledge base) are projected into a lower-dimensional feature space, and semantically similar words are positioned close to each other.

4 APPROACH

In our approach, we adapt neural language models for RDF graph embeddings. Such approaches take advantage of the word order in text documents, explicitly modeling the assumption that closer words in the word sequences are statistically more dependent. In the case of RDF graphs, we consider entities and relations between entities instead of word sequences. Thus, in order to apply such approaches on RDF graph data, we first have to transform the graph data into sequences of entities, which can be considered as sentences. Using those sequences, we can train the same neural language models to represent each entity in the RDF graph as a vector of numerical values in a latent feature space.

We use graph walks for converting graphs into a set of sequences of entities. An example of an entity sequence extracted using graph walks from DBpedia would be: $dbr : Trent\_Reznor \rightarrow dbo : associatedBand \rightarrow dbr : Nine\_Inch\_Nails \rightarrow dbo : genre \rightarrow dbr : Industrial\_Rock$. To perform these walks on RDF graphs, we represent the graph as a set of vertices (the entities in the RDF graph) and a set of directed edges (the relations between the entities).

The objective of the walk functions is for each vertex $v \in V$ to generate a set of sequences S_v, where the first token of each sequence $s \in S_v$ is the vertex v followed by a sequence of tokens, which might be the labels of edges, vertices, or any substructure extracted from the RDF graph, in an order that reflects the relations between the vertex v and the rest of the tokens, as well as among those tokens.

What we want to achieve is a biasing of these walks to make them more meaningful, i.e., being able to capture the most important information about the observed entities. Therefore, we augment the edges to not only have a label, but also a weight. We apply twelve different strategies for assigning these weights to the edges of the graph. These weights will then in turn bias the random walks on the graph. In particular, when a walk arrives in a vertex v with out edges $v_{o1}, \ldots v_{od}$, then the walk will follow edge v_{ol} with a probability computed by

$$\Pr[\text{follow edge } v_{ol}] = \frac{weight(v_{ol})}{\sum_{i=1}^{d} weight(v_{oi})}$$

In other words, the normalized edge weights are directly interpreted as the probability to follow a particular edge.

To obtain these edge weights, we make use of different statistics computed on the RDF data. The statistics computed are the following:

Predicate Frequency for each RDF predicate (i.e., label on the edges) in the dataset, we count the number of times the predicate occurs (only occurrences as a predicate are counted).

Object Frequency for each resource in the dataset, we count the number of times it occurs as the object of an RDF triple (i.e., is the target of an edge).

Predicate-Object frequency for each pair of a predicate and an object in the dataset, we count the number of times there is a statement with this predicate and object (i.e., we count pairs of edge labels and the label of its target node).

Besides these statistics, we also use PageRank [2] computed for the entities in the knowledge graph [42]. This PageRank is computed based on links between the Wikipedia articles representing the respective entities. When using the PageRank computed for DBpedia, not each node has a value assigned, as only entities which have a corresponding Wikipedia page are accounted for in the PageRank computation. Examples of nodes which do not have a PageRank include DBpedia types or categories, like http://dbpedia.org/ontology/Place and http://dbpedia.org/resource/Category:Central_Europe. Therefore, we assigned a fixed PageRank to all nodes which are not entities. We chose a value of 0.2, which is roughly the median PageRank [43], in the non-normalized page rank values we used.

Note that there are essentially two types of metrics, those assigned to nodes, and those assigned to edges. The predicate frequency and predicate-object frequency, as well as the inverses of these, can be directly used as weights for edges. Therefore, we call these weighting methods *edge-centric*. In the case of predicate frequency each predicate edge with that label is assigned the weight in question. In the case of predicate-object frequency, each predicate edge which ends in a given object gets assigned the predicate-object frequency. When computing the inverse metrics, not the absolute frequency is assigned, but its multiplicative inverse.

In contrast, the object frequency, and also the used PageRank metric, assign a numeric score to each node in the graph. Therefore, we call weighting approaches based on them *node-centric*. To obtain a weight for the edges, we either *push* the number down or *split* the number down to all in edges. By pushing down, we mean that the number assigned to a node is used as the weight of all in edges. By splitting down, we mean that the weight is divided by the number of in edges and then assigned to all edges. Then, these weights can be normalized as described above. If *split* is not mentioned explicitly in node centric weighting strategies, then it is a push down strategy.

In total, we inspected twelve different approaches for weighting edges using the metrics defined above.

Note that uniform weights are equivalent to using object frequency with splitting the weights. To see why this holds true, we have to follow the steps which will be taken. First, each node gets assigned the amount of times it is used as an object. This number is equal to the number of in edges to the node. Then, this number is split over the in edges, i.e., each in edge gets assigned the number 1. Finally, this weight is normalized, assigning to each out link a uniform weight. Hence, this strategy would result in the same walks as using unbiased random walks over the graph.

So, even if we add unbiased random walks to the list of weighting strategies, we retain 12 unique ones, each with their own characteristics. These strategies are:
Uniform approach:

(1) *Uniform Weight = Object Frequency Split Weight* – This is the most straight forward approach, also taken by the standard RDF2Vec models. At first glance, it also looks like the most neutral strategy. However, the input graph does not have a regular structure in the sense that some entities have a (much) higher in degree as others and hence they are more likely to be visited. Thus, more strongly connected entities will have a higher influence on the resulting embeddings.

Edge-centric approaches:

(2) *Predicate Frequency Weight* – With this strategy, edges with predicates which are commonly used in the dataset are more often followed. The effect of this is that many uncommon predicates are never followed in our experiments and, as a result of that, many entities are also never visited in the walks. On the other hand,there are a few entities which have a very high in degree, and which thus attract a lot of walks towards them.

(3) *Inverse Predicate Frequency Weight* – This strategy has at first sight a similar effect as the previous, but for other nodes. Those predicates which are rare will be followed often. However, predicates follow a long-tail distribution, and there are more predicates which are rare than common, thus, the diversity of predicates occurring in the walks is

higher. Moreover, despite having a low probability, also edges with a common predicate are followed once in a while as they occur so often in the dataset.

(4) *Predicate-Object Frequency Weight* – This is similar to the Predicate Frequency Weight, but differentiates between the objects as well. If we have for example an outgoing link with label `rdf:type` with object `owl:Thing`, then this link will be followed more often than, e.g., the same predicate with object `dbpedia:AdministrativeRegion`.

(5) *Inverse Predicate-Object Frequency Weight* – The inverse of the previous, with similar features to Inverse Predicate Frequency Weight. We measured that this approach results in walks in which nodes occur most uniformly.

Node-centric object freq. approaches (See also strategy 1):

(6) *Object Frequency Weight* – This weighting does essentially ignore the predicate altogether and just ensures that entities which have a high in degree get visited even more often.

(7) *Inverse Object Frequency Weight* – This approach also ignores the predicate, but makes the probability for nodes to be visited more equally distributed. Hence, according to our measurements entities occur nearly as uniformly in walks as for Inverse Predicate-Object Frequency Weight.

(8) *Inverse Object Frequency Split Weight* – The general statistics for these walks look surprisingly similar to the non inverted strategy.

Node-centric PageRank approaches:

(9) *PageRank Weight* – Similar to Object Frequency Weight, this strategy makes some nodes more important and hence there will be resources which are more frequent in the walks as others.

(10) *Inverse PageRank Weight* – One would expect that this approach would have a similar effect as Inverse Object Frequency Weight, however, our measurements show that the inversion does not cause more uniform occurrence of entities as strongly as that strategy.

(11) *PageRank Split Weight* – Both this approach and the next one are somewhat difficult to predict as they do not only depend on the structure on the graph. Our analysis of the walks show that nodes are fairly uniformly used in these walks. Furthermore, these strategies result in a high uniformity in the absolute frequency of predicates.

(12) *Inverse PageRank Split Weight* – The generated walks have similar statistics as PageRank Split Weight. The expectation is, however, that in this metric tends to include more unimportant nodes in the walks.

For each set of the twelve sets of sequences created using those metrics, we build one CBOW and one skip-gram model, as described in Section 3. Hence, we compare a total of 24 different embeddings.

5 EVALUATION

First, we evaluate the different weighting strategies on a number of classification and regression tasks, comparing the results of different feature extraction strategies combined with different learning algorithms. Second, we evaluate the weighting strategies on entity and document modeling tasks, i.e., entity relatedness and document similarity.

To build the neural language models, we generate 250 walks per entity with depths of 2,4,6, and 8 for each of the twelve edge weighting strategies. A depth of eight means four hops in the graph, as each hop adds two elements to the sequence (i.e., the predicate and the object). Since, the entity which is the source of the walk is also include in the path, the corresponding path lengths are 3,5,7, and 9. When the walk reaches a "dead end", i.e., a node without any outgoing edges, the walk ends in that node, even if the maximum depth is not reached.

We use the corpora of sequences to build both CBOW and Skip-Gram models with the following parameters: window size = 5; number of iterations = 5; negative sampling for optimization; negative samples = 25; dimensions = 200; with average input vector for CBOW. The parameters are selected based on recommendations from the literature. All the models, as well as the code, are publicly available[2].

5.1 Machine Learning Tasks

Linking entities in a machine learning task to those in the LOD cloud helps generating additional features, which may help improving the overall learning outcome [37]. For example, when learning a predictive model for the success of a movie, adding knowledge from the LOD cloud (such as the movie's budget, director, genre, etc.) can lead to a more accurate model. In this case, for each entity in the dataset, we use the correposing entity's embedded vector from the knowledge base as a feature vector.

5.1.1 *Datasets.*

We evaluate our approach on DBpedia [17]. We use the English version of the 2016-04 DBpedia dataset, which contains $4,678,230$ instances and $1,379$ mapping-based properties. In our evaluation we only consider object properties, and ignore datatype properties and literals.

We use the entity embeddings on five different datasets from different domains, for the tasks of classification and regression [34]. Those five datasets are used to provide classification/regression targets for the large RDF datasets (see Table 1).

- The *Cities* dataset contains a list of cities and their quality of living, as captured by Mercer[3]. We use the dataset both for regression and classification.
- The *Metacritic Movies* dataset is retrieved from Metacritic.com[4], which contains an average rating of all time reviews for a list of movies [38]. The initial dataset contained around $10,000$ movies, from which we selected $1,000$ movies from the top of the list, and $1,000$ movies from the bottom of the list. We use the dataset both for regression and classification.
- Similarly, the *Metacritic Albums* dataset is retrieved from Metacritic.com[5], which contains an average rating of all time reviews for a list of albums [39].
- The *AAUP* (American Association of University Professors) dataset contains a list of universities, including eight

[2] http://data.dws.informatik.uni-mannheim.de/rdf2vec/

[3] https://www.imercer.com/content/mobility/quality-of-living-city-rankings.html

[4] http://www.metacritic.com/browse/movies/score/metascore/all

[5] http://www.metacritic.com/browse/albums/score/metascore/all

Table 1: Classification and regression datasets overview. For each dataset, we depict the number of instances, the machine learning tasks in which the dataset is used (*C* stands for classification, and *R* stands for regression) and the source of the dataset.

Dataset	#Instances	ML Task	Original Source
Cities	212	R/C (c=3)	Mercer
Metacritic Albums	1600	R/C (c=2)	Metacritic
Metacritic Movies	2000	R/C (c=2)	Metacritic
AAUP	960	R/C (c=3)	JSE
Forbes	1585	R/C (c=3)	Forbes

target variables describing the salary of different staff at the universities[6]. We use the average salary as a target variable both for regression and classification, discretizing the target variable into "high", "medium" and "low", using equal frequency binning.

- The *Forbes* dataset contains a list of companies including several features of the companies, which was generated from the Forbes list of leading companies 2015[7]. The target is to predict the company's market value as a regression task. To use it for the task of classification we discretize the target variable into "high", "medium", and "low", using equal frequency binning.

5.1.2 Experimental Setup. As in [36], we compare our approach to several baselines. For generating the data mining features, we use three strategies that take into account the direct relations to other resources in the graph [30], and two strategies for features derived from graph sub-structures [6]:

- Features derived from specific relations. In the experiments we use the relations *rdf:type* (types), and *dcterms:subject* (categories).
- Features derived from generic relations, i.e., we generate a feature for each incoming (rel in) or outgoing relation (rel out) of an entity, ignoring the value or target entity of the relation.
- Features derived from generic relations-values, i.e, we generate feature for each incoming (rel-vals in) or outgoing relation (rel-vals out) of an entity including the value of the relation.
- Kernels that count substructures in the RDF graph around the instance node. These substructures are explicitly generated and represented as sparse feature vectors.
  - The Weisfeiler-Lehman (WL) graph kernel for RDF [6] counts full subtrees in the subgraph around the instance node. This kernel has two parameters, the subgraph depth d and the number of iterations h (which determines the depth of the subtrees). We use two pairs of settings, $d = 1, h = 2$ and $d = 2, h = 3$.
  - The Intersection Tree Path kernel for RDF [6] counts the walks in the subtree that spans from the instance node. Only the walks that go through the instance node are considered. We will therefore refer to it as the root Walk Count (WC) kernel. The root WC kernel has one parameter: the length of the paths l, for which we test 2 and 3.

Furthermore, we compare the results to the state-of-the art graph embeddings approaches: TransE, TransH and TransR. We use an existing implementation and build models on the the DBpedia data with the default parameters.[8]

We perform two learning tasks, i.e., classification and regression. For classification tasks, we use Naive Bayes, k-Nearest Neighbors (k=3), C4.5 decision tree, and Support Vector Machines. For the SVM classifier we optimize the parameter C in the range $\{10^{-3}, 10^{-2}, 0.1, 1, 10, 10^2, 10^3\}$. For regression, we use Linear Regression, M5Rules, and k-Nearest Neighbors (k=3). The results are calculated using stratified 10-fold cross validation.

The strategies for creating propositional features from Linked Open Data are implemented in the RapidMiner LOD extension[9] [31, 33]. The experiments, including the feature generation and the evaluation, were performed using the RapidMiner data analytics platform.[10] The RapidMiner processes and the complete results can be found online.[11]

For comparing the approaches, we follow the approach introduced by Demšar [7]. The approach proposes to first rank the strategies for each dataset in isolation, and then to compute a significance level for the difference of ranks using a Friedman test. While the Friedman test only determines whether there is a significant difference between *any* of the compared approaches, pairwise significance levels are computed with a post-hoc Nemenyi test [24]. The results of the post-hoc test allows for concluding if one approach significantly outperforms another one. For the Friedman test we select a significance level of $\alpha = 0.10$, and for the post-hoc Nemenyi test we use critical values $q = 0.05$. We carry out the test on each learning method separately.

5.1.3 Results. The results for the task of classification on the five different datasets using four different learning methods are given in Table 2. For each of the datasets and for each learning method, we select the best performing results of all the baselines, and report it under *Best baseline*. Using the Friedman test, the null hypothesis was rejected for the performances of the strategies when using Naive Bayes and KNN, meaning there is a significant performance difference between the strategies.

The results for the task of regression on the five different datasets using four different learning methods are given in Table 3. Using the Friedman test, the null hypothesis was rejected for the performances of the strategies when using Linear Regression, meaning there is a significant performance difference between the strategies.

From the results for both tasks we can conclude that the RDF2Vec approach outperforms the baseline approaches and also outperforms the state-of-the art graph embeddings models. Furthermore, *Inverse PageRank Weight* and *PageRank Split Weight* strategies perform well for different learning methods. Overall, the skip-gram models outperform the corresponding CBOW models for most of

[6] http://www.amstat.org/publications/jse/jse_data_archive.htm
[7] http://www.forbes.com/global2000/list/
[8] https://github.com/thunlp/KB2E/
[9] http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension
[10] https://rapidminer.com/
[11] http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/

Table 2: Classification average rank results. The best ranked results for each method are marked in bold. The learning models for which the strategies were shown to have significant difference based on the Friedman test with $\alpha < 0.05$ are marked with *. The single values marked with $*$ mean that are significantly worse than the best strategy at significance level $q = 0.05$

Method		NB*	KNN*	SVM	C4.5
Uniform Weight	CBOW	14.4	9.7	12.8	9.4
	SG	6.4	**3.3**	10.0	**6.6**
Edge-centric approaches					
Predicate Frequency Weight	CBOW	14.0	11.3	12.6	14.0
	SG	11.6	11.1	10.4	12.8
Inverse Predicate Frequency Weight	CBOW	24.6*	25.6*	22.5	19.8
	SG	23.0	19.4	15.8	18.2
Predicate Object Frequency Weight	CBOW	20.5	20.9	17.9	20.8
	SG	20.4	20.3	16.7	20.6
Inverse Predicate Object Frequency Weight	CBOW	19.0	16.8	15.3	15.4
	SG	17.2	15.6	10.6	12.2
Node-centric object freq. approaches					
Object Frequency Weight	CBOW	19.1	20.2	17.9	21.0
	SG	17.8	14.6	14.0	15.8
Inverse Object Frequency Weight	CBOW	7.0	10.6	10.2	7.6
	SG	19.6	19.4	15.7	21.0
Inverse Object Frequency Split Weight	CBOW	18.8	16.7	16.0	13.4
	SG	7.4	10.9	13.1	14.2
Node-centric PageRank approaches					
PageRank Weight	CBOW	25.2*	22.6	20.9	19.0
	SG	14.2	9.8	9.8	13.0
Inverse PageRank Weight	CBOW	8.2	14.8	12.4	10.6
	SG	4.8	10.0	9.8	9.0
PageRank Split Weight	CBOW	23.4	10.9	17.0	15.2
	SG	**4.4**	4.7	**6.7**	8.4
Inverse PageRank Split Weight	CBOW	13.4	11.3	17.9	15.6
	SG	7.4	8.9	11.6	10.6
Baseline and related approaches					
Best Baseline		12.0	15.0	19.0	7.8
TransE		10.0	16.7	16.8	16.6
TransH		9.8	15.8	16.3	17.2
TransR		12.4	19.1	16.3	20.2

the strategies. Unexpectedly, the *Uniform Weight* strategy also yields competitive results.

However, for the variety of tasks at hand, there is no universal approach, i.e., embedding model and a machine learning method, that consistently outperforms the others.

5.2 Entity and Document Modeling

Calculating entity relatedness and similarity are fundamental problems in numerous tasks in information retrieval, natural language processing, and Web-based knowledge extractions. While similarity only considers subsumption relations to assess how two objects are alike, relatedness takes into account a broader range of relations, i.e., the notion of relatedness is wider than that of similarity. For example, "Facebook" and "Google" are both entities of the class company, and they have high similarity and relatedness score. On the other hand, "Facebook" and "Mark Zuckerberg" are not similar at all, but are highly related. While "Google" and "Mark Zuckerberg" are not similar at all, and have low relatedness value.

As previously mentioned, in the RDF2Vec feature embedding space (see Section 4) semantically similar entities appear close to each other in the feature space. Therefore, the problem of calculating the similarity between two instances is a matter of calculating the distance between two instances in the given feature space. To do so, we use the standard cosine similarity measure, which is applied on the vectors of the entities. Formally, the similarity between two entities e_1 and e_2, with vectors V_1 and V_2, is calculated as the cosine similarity between the vectors V_1 and V_2:

$$sim(e_1, e_2) = \frac{V_1 \cdot V_2}{||V_1|| \cdot ||V_2||} \tag{6}$$

We use the entity similarity approach in the task of calculating semantic document similarity. We follow similar approach as the one presented in [27], where two documents are considered to be similar if many entities of the one document are similar to at least

Table 3: Regression average rank results. The best ranked results for each method are marked in bold. The learning models for which the strategies were shown to have significant difference based on the Friedman test with $\alpha < 0.05$ are marked with *. The single values marked with $*$ are significantly worse than the best strategy at significance level $q = 0.05$

Method		LR*	KNN	M5
Uniform Weight	CBOW	8.0	7.4	9.0
	SG	**4.4**	7.6	8.8
Edge-centric approaches				
Predicate Frequency Weight	CBOW	10.8	13.4	10.8
	SG	15.0	11.6	16.4
Inverse Predicate Frequency Weight	CBOW	22.0	16.8	21.6
	SG	13.0	15.4	17.2
Predicate Object Frequency Weight	CBOW	24.6*	22.4	24.2
	SG	24.8*	23.6	24.8
Inverse Predicate Object Frequency Weight	CBOW	12.6	14.0	13.4
	SG	6.2	10.6	8.2
Node-centric object freq. approaches				
Object Frequency Weight	CBOW	22.8	22.2	21.6
	SG	10.8	15.0	14.6
Inverse Object Frequency Weight	CBOW	6.8	10.0	9.4
	SG	26.0*	22.8	23.8
Inverse Object Frequency Split Weight	CBOW	21.0	20.2	19.0
	SG	13.2	15.6	13.2
Node-centric PageRank approaches				
PageRank Weight	CBOW	25.8*	18.0	25.6
	SG	7.0	15.4	7.8
Inverse PageRank Weight	CBOW	11.4	8.8	13.0
	SG	7.4	**6.8**	**6.2**
PageRank Split Weight	CBOW	17.6	12.2	17.8
	SG	8.6	10.2	8.4
Inverse PageRank Split Weight	CBOW	17.6	18.2	17.8
	SG	9.4	11.2	7.2
Baseline and related approaches				
Best Baseline		17.4	9.6	9.6
TransE		12.8	16.7	13.0
TransH		12.8	14.1	12.4
TransR		16.2	16.2	11.2

one entity in the other document. More precisely, we try to identify the most similar pairs of entities in both documents, ignoring the similarity of all the other 1-1 similarities values.

Given two documents d_1 and d_2, the similarity between the documents $sim(d_1, d_2)$ is calculated as follows:

(1) Extract the sets of entities E_1 and E_2 in the documents d_1 and d_2.
(2) Calculate the similarity score $sim(e_{1i}, e_{2j})$ for each pair of entities in document d_1 and d_2, where $e_{1i} \in E_1$ and $e_{2j} \in E_2$
(3) For each entity e_{1i} in d_1 identify the maximum similarity to an entity in d_2 $max\_sim(e_{1i}, e_{2j} \in E_2)$, and vice versa.
(4) Calculate the similarity score between the documents d_1 and d_2 as:

$$sim(d_1, d_2) = \frac{\sum_{i=1}^{|E_1|} max\_sim(e_{1i}, e_{2j} \in E_2) + \sum_{j=1}^{|E_2|} max\_sim(e_{2j}, e_{1i} \in E_1)}{|E_1| + |E_2|} \tag{7}$$

For entity similarity, we assume that two entities are related if they often appear in the same context. For example, "Facebook" and "Mark Zuckerberg", which are highly related, are often used in the same context in many sentences. To calculate the probability of two entities being in the same context, we make use of the RDF2Vec models and the set of sequences of entities generated as described in Section 4. Given an RDF2Vec model and a set of sequences of entities, we calculate the relatedness between two entities e_1 and e_2, as the probability $p(e_1|e_2)$ calculated using the softmax function. In the case of a CBOW model, the probability is calculated as:

$$p(e_1|e_2) = \frac{exp(v_{e_2}^T v'_{e_1})}{\sum_{e=1}^{V} exp(v_{e_2}^T v'_e)},$$

where v'_e is the output vector of the entity e, and V is the complete vocabulary of entities.

In the case of a skip-gram model, the probability is calculated as:

$$p(e_1|e_2) = \frac{exp(v_{e_1}'^T v_{e_2})}{\sum_{e=1}^{V} exp(v_e'^T v_{e_2})}, \tag{8}$$

where v_e and v_e' are the input and the output vector of the entity e, and V is the complete vocabulary of entities.

5.2.1 *Datasets.*

For both tasks of determining entity relatedness and document similarity, benchmark datasets exist. We use those datasets to compare the use of RDF2Vec models against state of the art approaches.

For evaluating the entity relatedness approach, we use the KORE dataset [11]. The dataset consists of 21 main entities, whose relatedness to the other 20 entities each has been manually assessed, leading to 420 rated entity pairs. We use the Spearman's rank correlation as an evaluation metric.

To evaluate the document similarity approach, we use the LP50 dataset [16], namely a collection of 50 news articles from the Australian Broadcasting Corporation (ABC), which were pairwise annotated with similarity rating on a Likert scale from 1 (very different) to 5 (very similar) by 8 to 12 different human annotators. To obtain the final similarity judgments, the scores are averaged for each pair the scores of all annotators. As a evaluation metrics we use Pearson's linear correlation coefficient and Spearman's rank correlation plus their harmonic mean.

5.2.2 *Results.*

In this section we evaluate the different weighting strategies and compare them to the state of the art graph embedding approaches TransE, TransH and TransR, on the entity relatedness and document similarity tasks.

The results for the entity relatedness task are shown in Table 4. We can observe that the translating embeddings models perform rather poor, becuase we use the simple cosine similarity between the entitites to calculate the similarity. The best results are achieved when using the PageRank Split Weight strategy, using skip-gram model.

The results for the document similarity task are shown in Table 5. Again, the RDF2Vec models outperform the translating embeddings models. As for the entity relatedness task, the best results are obtained using the PageRank Split Weight strategy.

5.3 Walk statistics

As we already partially discussed in section 4, each strategy results in different characteristics for the random walks. In particular, the walks can have different lengths, since each strategy has a different likelihood to steer the walks into "dead ends", i.e., nodes with no outgoing edges, more quickly. Thus, for the DBpedia dataset which we used, we further analyzed the length of the walks generated using the different strategies. The histograms for the walk lengths for each of the strategies are plotted together in fig. 3. We can observe that the uniform weight, predicate frequency weight, object frequency weight and predicate object frequency weight strategies result in a lot of short walks. These are exactly the strategies which give extra weight to edges going into specific popular nodes. It appears that these entities are then often also end points of the walk, i.e, there are no out edges from these nodes. We observe that the other strategies produce walks of maximum length most of the time.

Table 4: Entity relatedness Spearman's rank correlation results

Method		ρ
Uniform Weight	CBOW	0.384
	SG	0.564
Edge-centric approaches		
Predicate Frequency Weight	CBOW	-0.058
	SG	-0.123
Inverse Predicate Frequency Weight	CBOW	0.468
	SG	0.584
Predicate Object Frequency Weight	CBOW	0.076
	SG	-0.043
Inverse Predicate Object Frequency	CBOW	0.578
	SG	0.610
Node-centric object freq. approaches		
Object Frequency Weight	CBOW	-0.096
	SG	-0.102
Inverse Object Frequency Weight	CBOW	0.429
	SG	0.554
Inverse Object Frequency Split Weight	CBOW	0.447
	SG	0.489
Node-centric PageRank approaches		
PageRank Weight	CBOW	0.378
	SG	0.456
Inverse PageRank Weight	CBOW	0.411
	SG	0.426
PageRank Split Weight	CBOW	0.621
	SG	**0.634**
Inverse PageRank Split Weight	CBOW	0.462
	SG	0.487
Related approaches		
TransE		0.091
TransH		0.050
TransR		0.058

However, we cannot derive that longer or shorter paths are better, as the three best performing approaches, i.e., uniform, inverse page rank, and page rank split, have very different behaviors: the first favors shorter sequences, the second produces an equal distribution of sequences of different lengths, and the third favors longer sequences.

6 CONCLUSION

Vector space embeddings for RDF graphs have been proven a high utility and powerful approach for transforming RDF data and knowledge graphs to propositional forms. The RDF2Vec approach, first introduced in [36], leverages random walks for transforming RDF graphs to token sequences, which is a necessary approach to be able to apply standard vector space embeddings techniques like CBOW and Skip-Gram.

In this paper, we have examined the influence of edge weights and transition probabilities to guide the walks, i.e., to make them

Table 5: Document similarity results - Pearson's linear correlation coefficient (r) Spearman's rank correlation (ρ) and their harmonic mean μ

Method		r	ρ	μ
Uniform Weight	CBOW	0.562	0.480	0.518
	SG	0.608	0.448	0.516
Edge-centric approaches				
Predicate Frequency Weight	CBOW	0.547	0.454	0.496
	SG	0.355	0.284	0.316
Inverse Predicate Frequency Weight	CBOW	0.560	0.395	0.463
	SG	0.653	0.487	0.558
Predicate Object Frequency Weight	CBOW	0.339	0.302	0.319
	SG	0.238	0.183	0.207
Inverse Predicate Object Frequency Weight	CBOW	0.549	0.473	0.508
	SG	0.628	0.491	0.551
Node-centric object freq. approaches				
Object Frequency Weight	CBOW	0.372	0.317	0.342
	SG	0.255	0.190	0.218
Inverse Object Frequency Weight	CBOW	0.552	0.455	0.499
	SG	0.585	0.452	0.510
Inverse Object Frequency Split Weight	CBOW	0.501	0.405	0.448
	SG	0.469	0.335	0.391
Node-centric PageRank approaches				
PageRank Weight	CBOW	0.530	0.303	0.386
	SG	0.589	0.384	0.465
Inverse PageRank Weight	CBOW	0.588	0.491	0.535
	SG	0.467	0.390	0.425
PageRank Split Weight	CBOW	0.578	0.426	0.490
	SG	0.658	0.476	**0.552**
Inverse PageRank Split Weight	CBOW	0.525	0.419	0.466
	SG	0.369	0.292	0.326
Related approaches				
TransE		0.550	0.430	0.483
TransH		0.517	0.414	0.460
TransR		0.568	0.431	0.490

less uniformly random. We have shown that introducing biases to the walks can lead to significant improvements. In particular, the PageRank split and the inverse PageRank weighting schemes provide good results.

So far, we have based our evaluations on machine learning and document modeling tasks. For future work, we will also study the effect on other tasks in which knowledge graph embeddings have been applied successfully, such as content-based recommender systems [8], as well as link prediction, type prediction, or graph completion and error detection in knowledge graphs [29], as discussed in [22, 25].

In our experiments, we have also experienced that there is not a one-size-fits-all solution for the weighting schemes. Although there are some trends that can be observed, the performance of the weighting schemes is hard to predict in individual cases. Among others, we assume that one crucial factor is the popularity of entities: for example, for very popular entities, the PageRank heuristic is assumed to work well, because it extracts more sequences containing popular entities, while for tail entities, the inverse PageRank heuristic will lead to better results. Future evaluations should examine those effects more deeply.

Summarizing, RDF and knowledge graph embeddings are a useful approach for leveraging background knowledge in those sources. With this paper, we have explored one crucial factor which influences the utility of those embeddings. We believe that this work will be a helpful milestone towards exploiting the full power of RDF graph embeddings.

Acknowledgments. The work presented in this paper has been partially funded by the Junior-professor funding programme of the Ministry of Science, Research and the Arts of the state of Baden-Württemberg (project "Deep semantic models for high-end NLP application"), and by the German Research Foundation (DFG) under grant number PA 2373/1-1 (Mine@LOD). The implementation of our benchmarks was greatly aided by the work done on the Stanford Network Analysis Platform (SNAP).

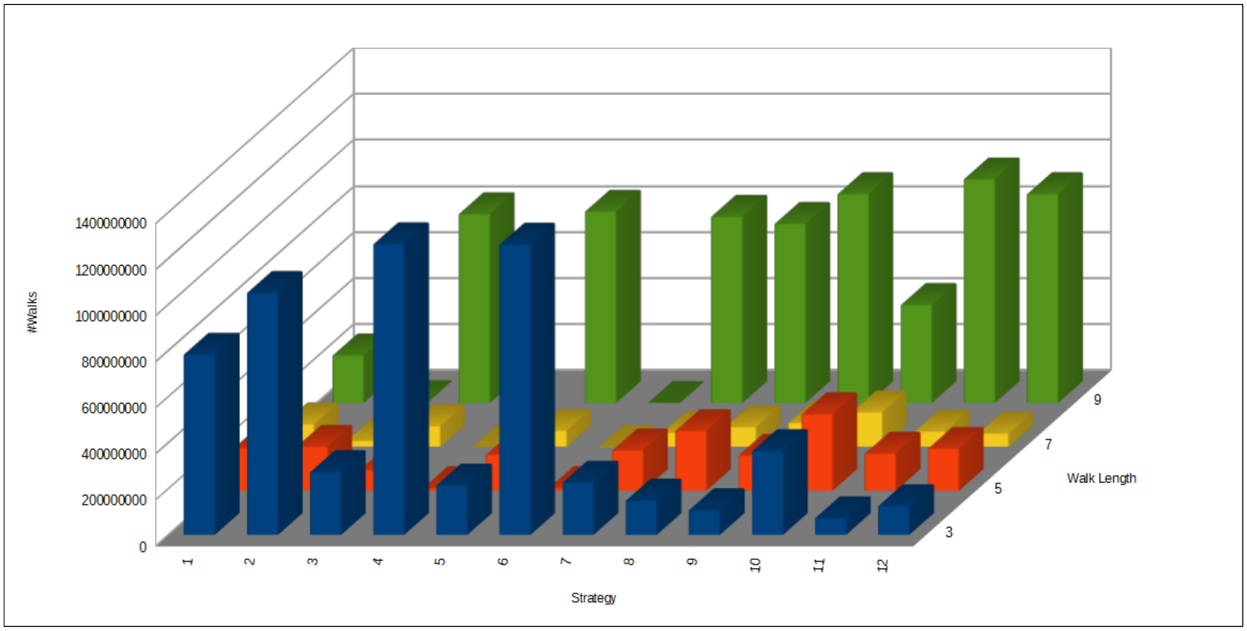

Figure 3: Distribution of walk lengths per strategy.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[2] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 30, 1 (1998), 107 – 117.

[3] Weiwei Cheng, Gjergji Kasneci, Thore Graepel, David Stern, and Ralf Herbrich. 2011. Automated Feature Generation from Structured Knowledge. In *CIKM*.

[4] Gerben Klaas Dirk de Vries. 2013. A Fast Approximation of the Weisfeiler-Lehman Graph Kernel for RDF Data.. In *ECML/PKDD (1)*.

[5] Gerben Klaas Dirk de Vries and Steven de Rooij. 2013. A fast and simple graph kernel for RDF. In *DMLOD*.

[6] Gerben Klaas Dirk de Vries and Steven de Rooij. 2015. Substructure counting graph kernels for machine learning from RDF data. *Web Semantics: Science, Services and Agents on the World Wide Web* 35 (2015), 71–84.

[7] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7 (2006), 1–30.

[8] Tommaso Di Noia and Vito Claudio Ostuni. 2015. Recommender Systems and Linked Open Data. In *Reasoning Web. Web Logic Rules*. Springer, 88–113.

[9] Nicola Fanizzi and Claudia d’Amato. 2006. A Declarative Kernel for ALC Concept Descriptions. In *Foundations of Intelligent Systems*. 322–331.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.

[11] Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 545–554.

[12] Yi Huang, Volker Tresp, Maximilian Nickel, and Hans-Peter Kriegel. 2014. A scalable approach for statistical learning in semantic graphs. *Semantic Web* (2014).

[13] Venkata Narasimha Pavan Kappara, Ryutaro Ichise, and O.P. Vyas. 2011. LiDDM: A Data Mining System for Linked Data. In *LDOW*.

[14] Mansoor A. Khan, Gunnar A. Grimnes, and Andreas Dengel. 2010. Two pre-processing operators for improved learning from SemanticWeb data. In *RCOMM*.

[15] Stefan Kramer, Nada Lavrac, and Peter Flach. 2001. Propositionalization Approaches to Relational Data Mining. In *Relational Data Mining*, Saso Dzeroski and Nada Lavrac (Eds.). Springer Berlin Heidelberg, 262–291. DOI:http://dx.doi.org/10.1007/978-3-662-04599-2_11

[16] M Lee, Brandon Pincombe, and Matthew Welsh. 2005. An empirical evaluation of models of text document similarity. Cognitive Science Society.

[17] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sren Auer, and Christian Bizer. 2013. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal* (2013).

[18] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion.. In *AAAI*. 2181–2187.

[19] Uta Lösch, Stephan Bloehdorn, and Achim Rettinger. 2012. Graph kernels for RDF data. In *The Semantic Web: Research and Applications*. Springer, 134–148.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[22] Pasquale Minervini, Nicola Fanizzi, Claudia d’Amato, and Floriana Esposito. 2015. Scalable Learning of Entity and Predicate Embeddings for Knowledge Graph Completion. In *International Conference on Machine Learning and Applications (ICMLA)*. 162–167.

[23] Jindřich Mynarz and Vojtěch Svátek. 2013. Towards a Benchmark for LOD-Enhanced Knowledge Discovery from Structured Data. In *The Second International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data*.

[24] PB Nemenyi. 1963. Distribution-free multiple comparisons.” vol. *PhD: Princeton University* (1963).

[25] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proc. IEEE* 104, 1 (2016), 11–33.

[26] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 809–816.

[27] Christian Paul, Achim Rettinger, Aditya Mogadala, Craig A Knoblock, and Pedro Szekely. 2016. Efficient Graph-Based Document Similarity. In *International Semantic Web Conference*. Springer, 334–349.

[28] Heiko Paulheim. 2013. Exploiting Linked Open Data as Background Knowledge in Data Mining. In *Workshop on Data Mining on Linked Open Data*.

[29] Heiko Paulheim. 2016. Knowlegde Graph Refinement: A Survey of Approaches and Evaluation Methods. *Semantic Web Journal* (2016). to appear.

[30] Heiko Paulheim and Johannes Fümkranz. 2012. Unsupervised generation of data mining features from linked open data. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*. ACM, 31.

[31] Heiko Paulheim, Petar Ristoski, Evgeny Mitichkin, and Christian Bizer. 2014. Data mining with background knowledge from the web. *RapidMiner World* (2014).

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international*

conference on Knowledge discovery and data mining. ACM, 701–710.

[33] Petar Ristoski, Christian Bizer, and Heiko Paulheim. 2015. Mining the web of linked data with rapidminer. *Web Semantics: Science, Services and Agents on the World Wide Web* 35 (2015), 142–151.

[34] Petar Ristoski, Gerben Klaas Dirk de Vries, and Heiko Paulheim. 2016. A Collection of Benchmark Datasets for Systematic Evaluations of Machine Learning on the Semantic Web. In *International Semantic Web Conference (To Appear)*. Springer.

[35] Petar Ristoski and Heiko Paulheim. 2014. A Comparison of Propositionalization Strategies for Creating Features from Linked Open Data. In *LD4KD*.

[36] Petar Ristoski and Heiko Paulheim. 2016. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*. Springer, 498–514.

[37] Petar Ristoski and Heiko Paulheim. 2016. Semantic Web in data mining and knowledge discovery: A comprehensive survey. *Web Semantics: Science, Services and Agents on the World Wide Web* (2016).

[38] Petar Ristoski, Heiko Paulheim, Vojtech Svátek, and Vaclav Zeman. 2015. The Linked Data Mining Challenge 2015.. In *KNOW@LOD*.

[39] Petar Ristoski, Heiko Paulheim, Vojtech Svátek, and Vaclav Zeman. 2016. The Linked Data Mining Challenge 2016. In *KNOWLOD*.

[40] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. 2014. Adoption of the linked data best practices in different topical domains. In *The Semantic Web–ISWC*.

[41] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*. 926–934.

[42] Andreas Thalhammer and Achim Rettinger. 2016. PageRank on Wikipedia: Towards General Importance Scores for Entities. In *The Semantic Web: ESWC 2016 Satellite Events*. Springer International Publishing, Crete, Greece, 227–240.

[43] Marieke van Erp, Pablo Mendes, Heiko Paulheim, Filip Ilievski, Julien Plu, Giuseppe Rizzo, and Jörg Waitelonis. 2016. Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job. In *10th International Conference on Language Resources and Evaluation (LREC)*.

[44] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In *AAAI*. Citeseer, 1112–1119.

[45] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1365–1374.