Undefined 0 (2016) 1–0
IOS Press

RDF2Vec: RDF Graph Embeddings and Their Applications

Petar Ristoski [a], Jessica Rosati [b,c], Tommaso Di Noia [b], Renato De Leone [c], Heiko Paulheim [a]
[a] *Data and Web Science Group, University of Mannheim, B6, 26, 68159 Mannheim*
E-mail: petar.ristoski@informatik.uni-mannheim.de, heiko@informatik.uni-mannheim.de
[b] *Polytechnic University of Bari, Via Orabona, 4, 70125 Bari*
E-mail: jessica.rosati@poliba.it, tommaso.dinoia@poliba.it
[c] *University of Camerino, Piazza Cavour 19/f, 62032 Camerino*
E-mail: jessica.rosati@unicam.it, renato.deleone@unicam.it

Abstract. Linked Open Data has been recognized as a valuable source for background information in many data mining and information retrieval tasks. However, most of the existing tools require features in propositional form, i.e., a vector of nominal or numerical features associated with an instance, while Linked Open Data sources are graphs by nature. In this paper, we present RDF2Vec, an approach that uses language modeling approaches for unsupervised feature extraction from sequences of words, and adapts them to RDF graphs. We generate sequences by leveraging local information from graph sub-structures, harvested by Weisfeiler-Lehman Subtree RDF Graph Kernels and graph walks, and learn latent numerical representations of entities in RDF graphs. We evaluate our approach on three different tasks: (i) standard machine learning tasks, (ii) entity and document modeling, and (iii) content-based recommender systems. The evaluation shows that the proposed entity embeddings outperform existing techniques, and that pre-computed feature vector representations of general knowledge graphs such as DBpedia and Wikidata can be easily reused for different tasks.

Keywords: Graph Embeddings, Linked Open Data, Data Mining, Document Semantic Similarity, Entity Relatedness, Recommender Systems

1. Introduction

Since its introduction, the Linked Open Data (LOD) [75] initiative has played a leading role in the rise of a new breed of open and interlinked knowledge bases freely accessible on the Web, each of them being part of a huge decentralized data space, the LOD cloud. This latter is implemented as an open, interlinked collection of datasets in machine-interpretable form, mainly built on top of *World Wide Web Consortium* (W3C) standards, such as RDF[1] and SPARQL[2]. Currently, the LOD cloud consists of about $1,000$ interlinked datasets covering multiple domains from life science to government data [75]. The LOD cloud has been recognized as a valuable source of background knowledge in data mining and knowledge discovery in general [70], as well as for information retrieval and recommender systems [14]. Augmenting a dataset with features taken from Linked Open Data can, in many cases, improve the results of the problem at hand, while externalizing the cost of maintaining that background knowledge [58].

Most data mining algorithms work with a propositional *feature vector* representation of the data, which means that 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). Linked Open Data, however, comes in the form of *graphs*, connecting resources with types and

[1] `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`, 2004.
[2] `http://www.w3.org/TR/rdf-sparql-query/`, 2008

relations, backed by a schema or ontology. In order to make LOD accessible to existing data mining tools, an initial *propositionalization* [34] of the corresponding graph is required. Even though the new set of propositional features do not encode all the knowledge available in the original ontological data, they can be effectively used to train a model via machine learning techniques and algorithms. 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 [60,68]. Other variants, e.g., counting different graph sub-structures, have also been proposed and used [87].

In language modeling, vector space word embeddings have been proposed in 2013 by Mikolov et al. [41,42]. They train neural networks for creating a low-dimensional, dense representation of words, which show two essential properties: (a) similar words are close in the vector space, and (b) relations between pairs of words can be represented as vectors as well, allowing for arithmetic operations in the vector space. In this work, we adapt those language modeling approaches for creating a latent representation of entities in RDF graphs. Since language modeling techniques work on sentences, we first convert the graph into a set of sequences of entities using two different approaches, i.e., graph walks and Weisfeiler-Lehman Subtree RDF graph kernels. In the second step, we use those sequences to train a neural language model, which estimates the likelihood of a sequence of entities appearing in a graph. Once the training is finished, each entity in the graph is represented as a vector of latent numerical features. We show that the properties of word embeddings also hold for RDF entity embeddings, and that they can be exploited for various tasks.

We use several RDF graphs to show that such latent representation of entities have high relevance for different data mining and information retrieval tasks. The generation of the entities' vectors is task and dataset independent, i.e., we show that once the vectors are generated, they can be used for machine learning tasks, like classification and regression, entity and document modeling, and to estimate the closeness of items for content-based or hybrid recommender systems in a top-N scenario. Furthermore, since all entities are represented in a low dimensional feature space, building the learning models and algorithms becomes more efficient. To foster the reuse of the created feature sets, we provide the vector representations of DBpedia and Wikidata entities as ready-to-use files for download.

This paper considerably extends [69], in which we introduced RDF2Vec for the first time. In particular, we demonstrate the versatility of RDF embeddings by extending the experiments to different kinds of tasks: we show that the vector embeddings not only can be used in machine learning tasks, but also for document modeling and recommender systems, without a need to retrain the embedding models. In addition, we extend the evaluation section for the machine learning tasks by comparing our proposed approach to some of the state-of-the-art graph embeddings, which have not been used for the specified tasks before. Furthermore, to the best of our knowledge, graph embeddings have not been used for document modeling, e.g., for entity relatedness and document similarity. Again, we conduct series of experiments to compare our approach to the state-of-the-art document modeling approaches, as well to the state-of-the-art graph embedding approaches.

Preliminary results for the recommender task have already been published in [72]. In this paper, we further extend the evaluation in recommendation scenarios by considering a new dataset (`Last.FM`) and by implementing a hybrid approach based on Factorization Machines. Both contributions highlight the effectiveness of RDF2Vec in building recommendation engines that overcame state of the art hybrid approaches in terms of accuracy also when dealing with very sparse datasets (as is the case of `Last.FM`).

The rest of this paper is structured as follows. In Section 2, we give an overview of related work. In Section 3, we introduce our approach. In Section 4 through Section 7, we describe the evaluation setup and evaluate our approach on three different sets of tasks, i.e., machine learning, document modeling, and recommender systems. We conclude with a summary and an outlook on future work.

2. Related Work

While the representation of RDF as vectors in an embedding space itself is a considerably new area of research, there is a larger body of related work in the three application areas discussed in this paper, i.e., the use of LOD in data mining, in document modeling, and in content-based recommender systems.

Generally, our work is closely related to the approaches DeepWalk [62] and Deep Graph Kernels [91]. 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 random walks. Node2vec [21] is another approach very similar to DeepWalk, which uses second order random walks to preserve the network neighborhood of the nodes. The approach we propose in this paper differs from these approaches in several aspects. First, we adapt the language modeling approaches on directed labeled RDF graphs, unlike the approaches mentioned above, which work on undirected graphs. Second, we show that task-independent entity vectors can be generated on large-scale knowledge graphs, which later can be reused on a variety of machine learning tasks on different datasets.

2.1. LOD in Machine Learning

In the recent past, a few approaches for generating data mining features from Linked Open Data have been proposed. Many of those approaches assume a manual design of the procedure for feature selection and, in most cases, this procedure results in the formulation of a SPARQL query by the user. LiDDM [30] 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. [9] proposes an approach for automated feature generation after the user has specified the type of features in the form of custom SPARQL queries.

A similar approach has been used in the RapidMiner[3] semweb plugin [31], which preprocesses RDF data in a way that can be further handled directly in RapidMiner. Mynarz et al. [47] have considered using user specified SPARQL queries in combination with SPARQL aggregates. *FeGeLOD* [60] and its successor, the *RapidMiner Linked Open Data Extension* [66], 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 [61,66].

When dealing with *Kernel Functions* for graph-based data, we face similar problems as in feature generation and selection. Usually, the basic idea behind their computation is to evaluate 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 [28,55], or towards specific semantic representations [16]. However, only a few approaches are general enough to be applied on any given RDF data, regardless the data mining task. Lösch et al. [39] introduce two general RDF graph kernels, based on intersection graphs and intersection trees. Later, the intersection tree path kernel was simplified by Vries et al. [86]. In another work, Vries et al. [85,87] introduce an approximation of the state-of-the-art Weisfeiler-Lehman graph kernel algorithm aimed at improving the computation 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.

Furthermore, multiple approaches for knowledge graph embeddings for the task of link prediction have been proposed [49], which could also be considered as approaches for generating propositional features from graphs. RESCAL [50] 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) [79] which can be used for the same purpose. One of the most successful approaches is the model based on translating embeddings, TransE [5]. This model builds entity and relation embeddings by regarding a relation as translation from head entity to tail entity. This approach assumes that 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 [88], 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 [38] 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] http://www.rapidminer.com/

2.2. Entity and Document Modeling

Both for entity and document ranking, as well as for the subtask of computing the similarity or relatedness of entities and documents, different methods using LOD have been proposed.

2.2.1. Entity Relatedness

Semantic relatedness of entities has been heavily researched over the past couple of decades. There are two main direction of studies. The first are approaches based on word distributions, which model entities as multi-dimensional vectors that are computed based on distributional semantics techniques [1,17,26]. The second are graph-based approaches relying on a graph structured knowledge base, or knowledge graph, which are the focus of this paper.

Schuhmacher et al. [76] proposed one of the first approaches for entity ranking using the DBpedia knowledge graph. They use several path and graph based approaches for weighting the relations between entities, which are later used to calculate the entity relatedness. A similar approach is developed by Hulpus et al. [29], which uses local graph measures, targeted to the specific pair, while the previous approach uses global measures. More precisely, the authors propose the exclusivity-based relatedness measure that gives higher weights to relations that are less used in the graph. In [15] the authors propose a hybrid approach that exploits both textual and RDF data to rank resources in DBpedia related to the IT domain.

2.2.2. Entity and Document Similarity

As for the entity relatedness approaches, there are two main directions of research in the field of semantic document similarity, i.e., approaches based on word distributions, and graph-based approaches. Some of the earliest approaches of the first category make use of standard techniques like bag-of-words models, but also more sophisticated approaches. Explicit Semantic Analysis (ESA) [17] represents text as a vector of relevant concepts. Each concept corresponds to a Wikipedia article mapped into a vector space using the TF-IDF measure on the article's text. Similarly, Salient Semantic Analysis (SSA) [24] uses hyperlinks within Wikipedia articles to other articles as vector features, instead of using the full body of text.

Nunes et al. [53] present a DBpedia based document similarity approach, in which they compute a document connectivity score based on document annotations, using measures from social network theory. Thiagarajan et al. [82] present a general framework showing how spreading activation can be used on semantic networks to determine similarity of groups of entities. They experiment with Wordnet and the Wikipedia Ontology as knowledge bases and determine similarity of generated user profiles based on a 1-1 annotation matching.

Schumacher et al. [76] use the same measure used for entity ranking (see above) to calculate semantic document similarity. Similarly, Paul et al. [57] present an approach for efficient semantic similarity computation that exploits hierarchical and transverse relations in the graph.

One approach that does not belong to these two main directions of research is the machine-learning approach by Huang et al. [27]. The approach proposes a measure that assesses similarity at both the lexical and semantic levels, and learns from human judgments how to combine them by using machine-learning techniques.

Our work is, to the best of our knowledge, the first to exploit the graph structure using neural language modeling for the purpose of entity relatedness and similarity.

2.3. Recommender Systems

Providing accurate suggestions, tailored to user's needs and interests, is the main target of Recommender Systems (RS) [65], information filtering techniques commonly used to suggest items that are likely to be of use to a user. These techniques have proven to be very effective to face the *information overload* problem, that is the huge amount of information available on the Web, which risks to overwhelm user's experience while retrieving items of interest. The numerous approaches facilitate the access to information in a personalized way, building a user profile and keeping it up-to-date.

RS address the information overload issue in two different ways, often combined into hybrid systems [7]: the *collaborative* approach [65] exploits information about the past behaviour and opinions of an existing user community to predict which item the current user will be more interested in, while the *content-based* approach [65] relies on the items "content", that is the description of items' characteristics. In a collaborative setting, a profile of the user is built by estimating her choice pattern through the behaviour of the overall user community. The content-based approach, instead, represents items by means of a set of features and defines a user profile as an assignment of

importance to such features, exploiting the past interaction with the system. To overcome the limitations of traditional approaches, which define the content based on partial metadata or on textual information optionally associated to an item, a process of "knowledge infusion" [77] has been performed for the last years, giving rise to the class of *semantics-aware* content-based recommender systems [19]. Many RS have incorporated ontological knowledge [40], unstructured or semi-structured knowledge sources (e.g., Wikipedia) [77], or the wealth of the LOD cloud, and recently the interest in unsupervised techniques where the human intervention is reduced or even withdrawn, has significantly increased.

LOD datasets, e.g., DBpedia [37], have been used in content-based recommender systems in [13] and [14]. The former performs a semantic expansion of the item content based on ontological information extracted from DBpedia and LinkedMDB [25], the first open semantic web database for movies, and tries to derive implicit relations between items. The latter involves both DBpedia and LinkedMDB and adapt the Vector Space Model to Linked Open Data: it represents the RDF graph as a 3-dimensional tensor where each slice is an ontological property (e.g. starring, director,...) and represents its adjacency matrix.

It has been proved that leveraging LOD datasets is also effective for hybrid recommender systems [7], that is in those approaches that boost the collaborative information with additional knowledge, such as the item content. In [12], the authors propose *SPRank*, a hybrid recommendation algorithm that extracts semantic path-based features from DBpedia and uses them to compute top-N recommendations in a learning to rank approach and in multiple domains, movies, books and musical artists. SPRank is compared with numerous collaborative approaches based on matrix factorization [33,64] and with other hybrid RS, such as *BPR-SSLIM* [52], and exhibits good performance especially in those contexts characterized by high sparsity, where the contribution of the content becomes essential. Another hybrid approach is proposed in [67], which builds on training individual base recommenders and using global popularity scores as generic recommenders. The results of the individual recommenders are combined using stacking regression and rank aggregation.

Most of these approaches can be referred to as *top-down* approaches [19], since they rely on the integration of external knowledge and cannot work without human intervention. On the other side, *bottom-up* approaches ground on the *distributional hypothesis* [23] for language modeling, according to which the meaning of words depends on the context in which they occur, in some textual content. The resulting strategy is therefore unsupervised, requiring a corpora of textual documents for training as large as possible. Approaches based on the distributional hypothesis, referred to as *discriminative models*, behave as word embeddings techniques where each term (and document) becomes a point in the vector space. They substitute the term-document matrix typical of Vector Space Model with a term-context matrix, on which they apply dimensionality reduction techniques such as Latent Semantic Indexing (LSI) [11] and the more scalable and incremental Random Indexing (RI) [73]. The latter has been involved in [45] and [46] to define the so called enhanced Vector Space Model (eVSM) for content-based RS, where user's profile is incrementally built summing the features vectors representing documents liked by the user and a negation operator is introduced to take into account also negative preferences, inspired by [90], that is according to the principles of Quantum Logic.

Word embedding techniques are not limited to LSI and RI. The word2vec strategy has been recently presented in [41] and [42], and to the best of our knowldge, has been applied to item recommendations in a few works [44,56]. In particular, [44] is an empirical evaluation of LSI, RI and word2vec to make content-based movie recommendation exploiting textual information from Wikipedia, while [56] deals with check-in venue (location) recommendations and adds a non-textual feature, the past check-ins of the user. They both draw the conclusion that word2vec techniques are promising for the recommendation task. Finally, there is a single example of *product embedding* [20], namely *prod2vec*, which operates on the artificial graph of purchases, treating a purchase sequence as a "sentence" and products within the sequence as words.

3. 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 a sequence 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 sentences, 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.

3.1. RDF Graph Sub-Structures Extraction

We propose two general approaches for converting graphs into a set of sequences of entities, i.e, graph walks and Weisfeiler-Lehman Subtree RDF Graph Kernels.

Definition 1 *An RDF graph is a labeled graph G = (V, E), where V is a set of vertices, and E is a set of directed edges, where each vertex $v \in V$ is identified by a unique identifier, and each edge $e \in E$ is labeled with a label from a finite set of edge labels.*

The objective of the conversion 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 edge labels, vertex identifiers, 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.

3.1.1. Graph Walks

In this approach, given a graph $G = (V, E)$, for each vertex $v \in V$, we generate all graph walks P_v of depth d rooted in vertex v. To generate the walks, we use the breadth-first algorithm. In the first iteration, the algorithm generates paths by exploring the direct outgoing edges of the root node v_r. The paths generated after the first iteration will have the following pattern $v_r \rightarrow e_i$, where $e_i \in E_{v_r}$, and E_{v_r} is the set of all outgoing edges from the root node v_r. In the second iteration, for each of the previously explored edges the algorithm visits the connected vertices. The paths generated after the second iteration will follow the following pattern $v_r \rightarrow e_i \rightarrow v_i$. The algorithm continues until d iterations are reached. The final set of sequences for the given graph G is the union of the sequences of all the vertices $P_G = \bigcup_{v \in V} P_v$. The algorithm is shown in Algorithm 1.

In the case of large RDF graphs, generating all possible walks for all vertices results in a large number of walks, which makes the training of the neural language model highly inefficient. To avoid this problem, we suggest for each vertex in the graph to generate only a subset, with size n, of all possible walks. To generate the walks, the outgoing edge to follow from the currently observed vertex v_c is selected based on the edge weight, i.e., the probability for selecting an edge e_i is $Pr[e_i] = \frac{weight(e_i)}{\sum_{j=1}^{|E_{v_c}|} weight(e_j)}$, where $e_i \in E_{v_c}$, and E_{v_c} is the set of all outgoing edges from the current node v_c. While there are many possibilities to set the weight of the edges, in this work we only consider equal weights, i.e., random selection of outgoing edges where an edge e_i is selected with probability $Pr[e_i] = \frac{1}{|E(v_c)|}$, where $e_i \in E_{v_c}$, and E_{v_c} is the set of all outgoing edges from the current node v_c. The algorithm is shown in Algorithm 2. Other weighting strategies can be integrated into the algorithm by exchanging the function *selectEdge* in line 11, e.g., weighting the edge based on the frequency, based on the fre-

Algorithm 1: Algorithm for generating RDF graph walks

```
   Data: G = (V, E): RDF Graph, d: walk depth
   Result: P_G: Set of sequences
1  P_G = ∅
2  foreach vertex v ∈ V do
3      Q = initialize queue
4      w = initialize walk
5      add v to w
6      add Entry(v, w) to Q
7      while Q is nonempty do
8          entry = deq(Q)
9          currentVertex = entry.key
10         currentWalk = entry.value
11         if currentWalk.length == d then
12             add currentWalk to P_G
13             continue
14         end
15         E_c = currentVertex.outEdges()
16         foreach vertex e ∈ E_c do
17             w = currentWalk
18             add e to w
19             if w.length == d then
20                 add w to P_G
21                 continue
22             end
23             v_e = e.endVertex()
24             add v_e to w
25             add Entry(v_e, w) to Q
26         end
27     end
28 end
```

Algorithm 2: Algorithm for generating weighted RDF graph walks

Data: $G = (V, E)$: RDF Graph, d: walk depth, n: number of walks
Result: P_G: Set of sequences

$P_G := \emptyset$
foreach *vertex* $v \in V$ **do**
  n_v = n
  while $n_v > 0$ **do**
    w = initialize walk
    add v to w
    $currentVertex = v$
    $d_v = d$
    while $d_v > 0$ **do**
      E_c = $currentVertex.outEdges()$
      e = selectEdge(E_c)
      $d_v = d_v$ - 1
      add e to w
      if $d_v > 0$ **then**
        v_e = $e.endVertex()$
        add v_e to w
        $currentVertex = v_e$
        $d_v = d_v$ - 1
      end
    end
    add w to P_G
    $n_v = n_v$ - 1
  end
end

quency of the edge's end node, or based on global weighting metrics, like PageRank [6].

3.1.2. Weisfeiler-Lehman Subtree RDF Graph Kernels

In this approach, we use the subtree RDF adaptation of the Weisfeiler-Lehman algorithm presented in [85,87]. The Weisfeiler-Lehman Subtree graph kernel is a state-of-the-art, efficient kernel for graph comparison [78]. The kernel computes the number of subtrees shared between two (or more) graphs by using the Weisfeiler-Lehman test of graph isomorphism. This algorithm creates labels representing subtrees in h iterations. The rewriting procedure of Weisfeiler-Lehman goes as follows: (i) the algorithm creates a multiset label for each vertex based on the labels of the neighbors of that vertex; (ii) this multiset is sorted and together with the original label concatenated into a string, which is the new label; (iii) for each unique string a new (shorter) label replaces the original vertex label; (iv) at the end of each iteration, each label represents a unique full subtree.

There are two main modifications of the original Weisfeiler-Lehman graph kernel algorithm in order to be applicable on RDF graphs [85,87]. First, the RDF graphs have directed edges, which is reflected in the fact that the neighborhood of a vertex v contains only the vertices reachable via outgoing edges. Second, as mentioned in the original algorithm, labels from two iterations can potentially be different while still representing the same subtree. To make sure that this does not happen, the authors in [85,87] have added tracking of the neighboring labels in the previous iteration, via the multiset of the previous iteration. If the multiset of the current iteration is identical to that of the previous iteration, the label of the previous iteration is reused.

The Weisfeiler-Lehman relabeling algorithm for an RDF graph is given in Algorithm 3, which is the same relabeling algorithm proposed in [85]. The algorithm takes as input the RDF graph $G = (V, E)$, a labeling function l, which returns a label of a vertex or edge in the graph based on an index, the subraph depth d and the number of iterations h. The algorithm returns the labeling functions for each iteration l_0 to l_h, and a label dictionary f. Furthermore, the neighborhood $N(v) = (v', v) \in E$ of a vertex is the set of edges going to the vertex v and the neighborhood $N((v, v')) = v$ of an edge is the vertex that the edge comes from.

The procedure of converting the RDF graph to a set of sequences of tokens goes as follows: (i) for a given graph $G = (V, E)$, we define the Weisfeiler-Lehman algorithm parameters, i.e., the number of iterations h and the vertex subgraph depth d, which defines the subgraph in which the subtrees will be counted for the given vertex; (ii) after each iteration, for each vertex $v \in V$ of the original graph G, we extract all the paths of depth d within the subgraph of the vertex v on the relabeled graph using Algorithm 1. We set the original label of the vertex v as the starting token of each path, which is then considered as a sequence of tokens. The sequences after each iteration will have the following pattern $v_r \rightarrow l_n(e_i, j) \rightarrow l_n(v_i, j)$, where l_n returns the label of the edges and the vertices in the n^{th} iteration. The sequecens could also be seen as $v_r \rightarrow T_1 \rightarrow T_1$ ... T_d, where T_d is a subtree that appears on depth d in the vertex's subgraph; (iii) we repeat step (ii) until the maximum iterations h are reached. (iv) The final set of sequences is the union of the sequences of all the vertices in each iteration $P_G = \bigcup_{i=1}^{h} \bigcup_{v \in V} P_v$.

Algorithm 3: Weisfeiler-Lehman Relabeling for RDF

Data: $G = (V, E)$: RDF Graph, l: labeling function for G = (V, E), d: subgraph depth, h: number of iterations
Result: l_0 to l_h: label functions, f label dictionary

for n = *0;* $n < h$; *i++* **do**
  # 1. Multiset-label determination
  foreach $v \in V$ *and* $e \in E$ *and* $j = 0$ *to* d **do**
    if $n = 0$ *and* $l(v, j)$ *is defined* **then**
      set $M_n(v, j) = l_0(v, j) = l(v, j)$
    end
    if $n = 0$ *and* $l(e, j)$ *is defined* **then**
      set $M_n(e, j) = l_0(e, j) = l(e, j)$
    end
    if $n > 0$ *and* $l(v, j)$ *is defined* **then**
      set $M_n(v, j) = \{l_{n-1}(u, j) | u \in N(v)\}$
    end
    if $n > 0$ *and* $l(e, j)$ *is defined* **then**
      set $M_n(e, j) = \{l_{n-1}(u, j + 1) | u \in N(e)\}$
    end
  end
  # 2. Sorting each multiset
  foreach $M_n(v, j)$ *and* $M_n(e, j)$ **do**
    sort the elements in $M_n(v, j)$, resp. $M_n(e, j)$, in ascending order and concatenate them into a string $s_n(v, j)$, resp. $s_n(e, j)$
  end
  foreach $s_n(v, j)$ *and* $s_n(e, j)$ **do**
    if $n > 0$ **then**
      add $l_{n1}(v, j)$, resp. $l_{n1}(e, j)$, as a prefix to $s_n(v, j)$, resp. $s_n(e, j)$
    end
  end
  # 3. Label compression
  foreach $s_n(v, j)$ *and* $s_n(e, j)$ **do**
    map $s_n(v, j)$, resp. $s_n(e, j)$, to a new compressed label, using a function $f : \sum^* \rightarrow \sum$, such that $f(s_n(v, j)) = f(s_n(v', j))$ iff $s_n(v, j) = s_n(v', j)$, resp. $f(s_n(e, j)) = f(s_n(e', j))$ iff $s_n(e, j) = s_n(e', j)$
  end
  # 4. Relabeling
  foreach $s_n(v, j)$ *and* $s_n(e, j)$ **do**
    set $l_n(v, j) = f(s_n(v, j))$ and $l_n(e, j) = f(s_n(e, j))$
  end
end

3.2. Neural Language Models – word2vec

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 their performance. 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, 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, like Feedforward Neural Net Language Model (NNLM) [3,10,83], with the recent advances in the field several efficient approaches have been proposed. One of the most popular and widely used approaches is the word2vec neural language model [41,42]. 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. The efficiency of the models comes as a result from the simplicity of the models by avoiding dense matrix multiplication, i.e., the non-linear hidden layer is removed from the neural network and the projection layer is shared for all words. Furthermore, the Skip-gram model has been extended to make the training even more efficient, i.e., (i) sub-sampling of frequent words, which significantly improves the model training efficiency, and improves the vector quality of the less frequent words; (ii) using simplified variant of Noise Contrastive Estimation [22], called negative sampling.

3.2.1. 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. 1a. The input layer is comprised of 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:

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

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

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

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:

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

3.2.2. 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. 1b). More formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, and a context window of size c, the objective of the skip-gram model is to maximize the following average log probability:

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

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

$$p(w_{t+c}|w_t) = \frac{exp(v'^T_{w_{t+c}} v_{w_t})}{\sum_{v=1}^{V} exp(v'^T_{w_v} v_{w_t})}, \quad (5)$$

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.

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 [42]. The empirical studies in the original paper [42] 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) are projected into a lower-dimensional feature space, and semantically similar words (or entities) are positioned close to each other.

4. Evaluation

We evaluate our approach on three different tasks: (i) standard machine-learning classification and regression; (ii) document similarity and entity relatedness; (iii) top-N recommendation task both with content-based and hybrid RSs. For all three tasks, we utilize two of the most prominent RDF knowledge graphs [59], i.e., DBpedia [37] and Wikidata [84]. DBpedia is a knowledge graph which is extracted from structured data in Wikipedia. The main source for this extraction are the key-value pairs in the Wikipedia infoboxes. Wikidata is a collaboratively edited knowledge graph, operated by the Wikimedia foundation[4] that also hosts various language editions of Wikipedia.

We use the English version of the 2015-10 DBpedia dataset, which contains $4,641,890$ instances and $1,369$ mapping-based object properties[5]. In our evaluation, we only consider object properties, and ignore datatype properties and literals.

For the Wikidata dataset, we use the simplified and derived RDF dumps from 2016-03-28[6]. The dataset contains $17,340,659$ entities in total. As for the DBpedia dataset, we only consider object properties, and ignore the data properties and literals.

The first step of our approach is to convert the RDF graphs into a set of sequences. As the number of generated walks increases exponentially [87] with the graph traversal depth, calculating Weisfeiler-Lehman subtrees RDF kernels, or all graph walks with a given depth d for all of the entities in the large RDF graph quickly becomes unmanageable. Therefore, to extract the entities embeddings for the large RDF datasets, we use only random graph walks entity sequences, generated using Algorithm 2. For both DBpedia and Wikidata, we first experiment with 200 random walks per

[4] http://wikimediafoundation.org/

[5] http://wiki.dbpedia.org/services-resources/datasets/dbpedia-datasets

[6] http://tools.wmflabs.org/wikidata-exports/rdf/index.php?content=dump_download.php\&dump=20160328

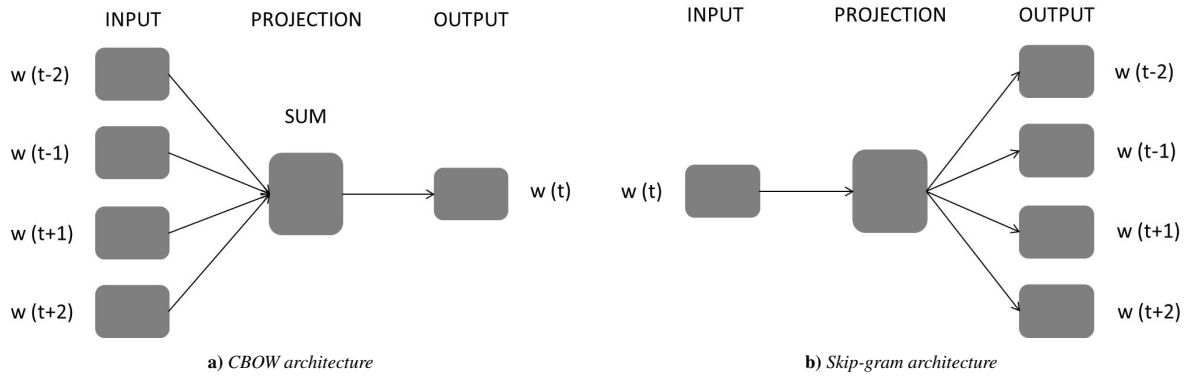


Fig. 1. Architecture of the CBOW and Skip-gram model.

entity with depth of 4, and 200 dimensions for the entities' vectors. Additionally, for DBpedia we experiment with 500 random walks per entity with depth of 4 and 8, with 200 and 500 dimensions for the entities' vectors. For Wikidata, we were unable to build models with more than 200 walks per entity, because of memory constrains, therefore we only experiment with the dimensions of the entities' vectors, i.e., 200 and 500.

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; with average input vector for CBOW. The parameter values are selected based on recommendations from the literature [41]. To prevent sharing the context between entities in different sequences, each sequence is considered as a separate input in the model, i.e., the sliding window restarts for each new sequence. We used the *gensim* implementation[7] for training the models. All the models, as well as the code, are publicly available.[8]

In the evaluation section we use the following notation for the models: *KB2Vec model #walks #dimensions depth*, e.g. *DB2vec SG 200w 200v 4d*, refers to a model built on DBpedia using the skip-gram model, with 200 walks per entity, 200 dimensional vectors and all the walks are of depth 4.

[7] https://radimrehurek.com/gensim/

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

5. Machine Learning with Background Knowledge from LOD

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. 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, Oscars won by the starring actors, etc.) can lead to a more accurate model.

5.1. Experimental Setup

For evaluating the performance of our RDF embeddings in machine learning tasks, we perform an evaluation on a set of benchmark datasets. The dataset contains three smaller-scale RDF datasets (i.e., AIFB, MUTAG, and BGS), where the classification target is the value of a selected property within the dataset, and five larger datasets linked to DBpedia and Wikidata, where the target is an external variable (e.g., the metacritic score of an album or a movie). The latter datasets are used both for classification and regression. Details on the datasets can be found in [71].

For each of the small RDF datasets, we first build two corpora of sequences, i.e., the set of sequences generated from graph walks with depth 8 (marked as W2V), and set of sequences generated from Weisfeiler-Lehman subtree kernels (marked as K2V). For the Weisfeiler-Lehman algorithm, we use 3 iterations and depth of 4, and after each iteration we extract all walks for each entity with the same depth. We use the corpora of sequences to build both CBOW and Skip-Gram models with the following parameters: window size = 5; number of iterations = 10; negative sampling for op-

Table 1
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. In case of classification, *c* indicates the number of classes.

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
AIFB	176	C (c=4)	AIFB
MUTAG	340	C (c=2)	MUTAG
BGS	146	C (c=2)	BGS

timization; negative samples = 25; with average input vector for CBOW. We experiment with 200 and 500 dimensions for the entities' vectors.

We use the RDF embeddings of DBpedia and Wikidata (see Section 4) on the five larger datasets, which provide classification/regression targets for DBpedia/Wikidata entities (see Table 1).

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 [60,68], and two strategies for features derived from graph sub-structures [87]:

- Features derived from specific relations. In the experiments we use the relations *rdf:type* (types), and *dcterms:subject* (categories) for datasets linked to DBpedia.
- 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. Furthermore, we combine both incoming and outgoing relations (rel in & out).
- Features derived from generic relations-values, i.e., we generate a feature for each incoming (rel-vals in) or outgoing relation (rel-vals out) of an entity including the value of the relation. Furthermore, we combine both incoming and outgoing relations with the values (rel-vals in & out).
- 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 [87] 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, $d = 2, h = 2$ (WL_2_2) and $d = 4, h = 3$ (WL_4_3).
  - The Intersection Tree Path kernel for RDF [87] 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 4 (WC_4) and 6 (WC_6).

Furthermore, we compare the results to the state-of-the art graph embeddings approaches: TransE, TransH and TransR. While there are many graph embedding approaches, the approaches based on translating embeddings have shown to outperform the rest of the approaches on the task of link predictions. Also, these approaches scale to large knowledge-graphs as DBpedia.[9] We use an existing implementation and build models on the small RDF datasets and the the whole DBpedia data with the default parameters.[10]

For all the experiments we don't perform any feature selection, i.e., we use all the features generated with the given feature generation strategy. For the baseline feature generation strategies, we use binary feature vectors, i.e., 1 if the feature exists for the instance, 0 otherwise.

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 (SVMs). For the SVM classifier we optimize the complexity constant *C*[11] 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). We measure accuracy for classification tasks, and root mean squared error (RMSE) for regression tasks. 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[12] [61,66]. The experiments, in-

[9] Because of high processing requirements we were not able to build the models for the Wikidata dataset.

[10] https://github.com/thunlp/KB2E/

[11] The complexity constant sets the tolerance for misclassification, where higher C values allow for "softer" boundaries and lower values create "harder" boundaries.

[12] http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension

cluding the feature generation and the evaluation, were performed using the RapidMiner data analytics platform.[13] The RapidMiner processes and the complete results can be found online.[14]. The experiments were run using a Linux machine with 20GB RAM and 4 Intel Xeon 2.60GHz CPUs.

5.2. Results

The results for the task of classification on the small RDF datasets are shown in Table 2.[15] Experiments marked with "\" did not finish within ten days, or have run out of memory. The reason for that is the high number of generated features for some of the strategies, as explained in Section 5.4. From the results we can observe that the K2V approach outperforms all the other approaches. More precisely, using the skip-gram feature vectors of size 500 in an SVM model provides the best results on all three datasets. The W2V approach on all three datasets performs closely to the standard graph substructure feature generation strategies, but it does not outperform them. K2V outperforms W2V because it is able to capture more complex substructures in the graph, like sub-trees, while W2V focuses only on graph paths. Furthermore, the related approaches, perform rather well on the AIFB dataset, and achieve comparable results to the K2V approach, however, on the other two dataset K2V significantly outperforms all three of them.

The results for the task of classification on the five datasets using the DBpedia and Wikidata entities' vectors are shown in Table 3, and the results for the task of regression on the five datasets using the DBpedia and Wikidata entities' vectors are given in Table 4. We can observe that the latent vectors extracted from DBpedia and Wikidata outperform all of the standard feature generation approaches. Furthermore, the RDF2vec approaches built on the DBpedia dataset continuously outperform the related approaches, i.e., TransE, TransH, and TransR, on both tasks for all the datasets, except on the Forbes dataset for the task of classification. In this case, all the related approaches outperform the baseline approaches as well as the RDF2vec approach. The difference is most significant when using the C4.5 classifier. In general, the DBpedia vectors work better than the Wikidata vectors, where the skip-gram vectors with size 200 or 500 built on graph walks of depth 8 on most of the datasets lead to the best performances. An exception is the AAUP dataset, where the Wikidata skip-gram 500 vectors outperform the other approaches. The reason for this could be that Wikidata contains more information about universities.

On both tasks, we can observe that the skip-gram vectors perform better than the CBOW vectors. Furthermore, the vectors with higher dimensionality and longer walks lead to a better representation of the entities and better performance on most of the datasets. However, for the variety of tasks at hand, there is no universal approach, i.e., a combination of an embedding model and a machine learning method, that consistently outperforms the others.

5.3. Semantics of Vector Representations

To analyze the semantics of the vector representations, we employ Principal Component Analysis (PCA) to project the entities' feature vectors into a two dimensional feature space. We selected seven countries and their capital cities, and visualized their vectors as points in a two-dimensional space. Figure 2a shows the corresponding DBpedia vectors, and Figure 2b shows the corresponding Wikidata vectors. The figure illustrates the ability of the model to automatically organize entities of different types, and preserve the relationships between different entities. For example, we can see that there is a clear separation between the countries and the cities, and the relation "capital" between each pair of country and the corresponding capital city is preserved. Furthermore, we can observe that more similar entities are positioned closer to each other, e.g., we can see that the countries that are part of the EU are closer to each other, and the same applies for the Asian countries.

5.4. Features Increase Rate

Finally, we conduct a scalability experiment, where we examine how the number of instances affects the number of generated features by each feature generation strategy. For this purpose we use the *Metacritic Movies* dataset. We start with a random sample of 100 instances, and in each next step we add 200 (or 300) unused instances, until the complete dataset is used, i.e., $2,000$ instances. The number of generated features

[13] https://rapidminer.com/

[14] http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/

[15] We do not consider the strategies for features derived from specific relations, i.e., types and categories, because the datasets do not contain categories, and all the instances are of the same type

Table 2
Classification results on the small RDF datasets. The best results are marked in bold. Experiments marked with "\" did not finish within ten days, or have run out of memory.

Strategy/Dataset	AIFB				MUTAG				BGS			
	NB	KNN	SVM	C4.5	NB	KNN	SVM	C4.5	NB	KNN	SVM	C4.5
rel in	16.99	47.19	50.70	50.62	\	\	\	\	61.76	54.67	63.76	63.76
rel out	45.07	45.56	50.70	51.76	41.18	54.41	62.94	62.06	54.76	69.05	72.70	69.33
rel in & out	25.59	51.24	50.80	51.80	\	\	\	\	54.76	67.00	72.00	70.00
rel-vals in	73.24	54.54	81.86	80.75	\	\	\	\	79.48	83.52	86.50	68.57
rel-vals out	86.86	55.69	82.39	71.73	62.35	62.06	73.53	62.94	84.95	65.29	83.10	73.38
rel-vals in&out	87.42	57.91	88.57	85.82	\	\	\	\	84.95	70.81	85.80	72.67
WL_2_2	85.69	53.30	92.68	71.08	91.12	62.06	92.59	93.29	85.48	63.62	82.14	75.29
WL_4_3	85.65	65.95	83.43	89.25	70.59	62.06	94.29	93.47	90.33	85.57	91.05	87.67
WC_4	86.24	60.27	75.03	71.05	90.94	62.06	91.76	93.82	84.81	69.00	83.57	76.90
WC_6	86.83	64.18	82.97	71.05	92.00	72.56	86.47	93.82	85.00	67.00	78.71	76.90
TransE	82.29	90.85	89.74	62.94	72.65	47.65	72.65	65.29	61.52	70.67	65.71	63.57
TransH	80.65	88.10	84.67	59.74	70.59	43.24	70.29	57.06	58.81	69.95	69.38	58.95
TransR	80.03	90.26	89.74	58.53	72.35	46.76	72.94	59.12	61.33	61.76	64.43	57.48
W2V CBOW 200	70.00	69.97	79.48	65.33	74.71	72.35	80.29	74.41	56.14	74.00	74.71	67.38
W2V CBOW 500	69.97	69.44	82.88	73.40	75.59	70.59	82.06	72.06	55.43	73.95	74.05	65.86
W2V SG 200	76.76	71.67	87.39	65.36	70.00	71.76	77.94	68.53	66.95	69.10	75.29	71.24
W2V SG 500	76.67	76.18	89.55	71.05	72.35	72.65	78.24	68.24	68.38	71.19	78.10	63.00
K2V CBOW 200	85.16	84.48	87.48	76.08	78.82	69.41	86.47	68.53	93.14	95.57	94.71	88.19
K2V CBOW 500	90.98	88.17	86.83	76.18	80.59	70.88	90.88	66.76	93.48	95.67	94.82	87.26
K2V SG 200	85.65	87.96	90.82	75.26	78.53	69.29	95.88	66.00	91.19	93.24	95.95	87.05
K2V SG 500	88.73	88.66	**93.41**	69.90	82.06	70.29	**96.18**	66.18	91.81	93.19	**96.33**	80.76

for each sub-sample of the dataset using each of the feature generation strategies is shown in Figure 3.

From the chart, we can observe that the number of generated features sharply increases when adding more samples in the datasets, especially for the strategies based on graph substructures.

In contrast, the number of features remains the same when using the RDF2Vec approach, as it is fixed to 200 or 500, respectively, independently of the number of samples in the data. Thus, by design, it scales to larger datasets without increasing the dimensionality of the dataset.

6. 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 extraction. 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 somehow lower relatedness value.

6.1. Approach

In this section, we introduce several approaches for entity and document modeling based on the previously built latent feature vectors for entities.

6.1.1. Entity Similarity

As previously mentioned, in the RDF2vec feature embedding space (see Section 3), 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:

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

Table 3: Classification results. The first number represents the dimensionality of the vectors, while the second number represent the value for the depth parameter. The best results are marked in bold. Experiments marked with "\" did not finish within ten days, or have run out of memory.

Strategy/Dataset	Cities				Metacritic Movies				Metacritic Albums				AAUP				Forbes			
	NB	KNN	SVM	C4.5	NB	KNN	SVM	C4.5	NB	KNN	SVM	C4.5	NB	KNN	SVM	C4.5	NB	KNN	SVM	C4.5
types	55.71	56.17	63.21	59.05	68.00	57.60	71.40	70.00	66.50	50.75	62.31	54.44	41.00	85.62	91.67	92.78	55.08	75.84	75.67	75.85
categories	55.74	49.98	62.39	56.17	75.25	62.70	76.35	69.50	67.40	54.13	64.50	56.62	48.00	85.83	90.78	91.87	60.38	76.11	75.70	75.70
rel in	60.41	58.46	71.70	60.35	52.75	49.90	60.35	60.10	51.13	62.19	65.25	60.75	45.63	85.94	90.62	92.81	50.24	76.49	75.16	76.10
rel out	47.62	60.00	66.04	56.71	52.90	58.45	66.40	62.70	58.75	63.75	62.25	64.50	41.15	85.83	89.58	91.35	64.73	75.84	75.73	75.92
rel in & out	59.44	58.57	66.04	56.47	52.95	59.30	67.75	62.55	58.69	64.50	67.38	61.56	42.71	85.94	89.67	92.50	22.27	75.96	76.34	75.98
rel-vals in	\	\	\	\	50.60	50.00	50.60	50.00	50.88	50.00	50.81	50.00	54.06	84.69	89.51	\	14.95	76.15	76.97	75.73
rel-vals out	53.79	35.91	55.66	64.13	78.50	54.78	78.71	\	74.06	52.56	76.99	\	57.81	85.73	91.46	91.78	67.09	75.61	75.74	76.74
rel-vals in&out	\	\	\	\	77.90	55.75	77.82	\	74.25	51.25	75.85	\	63.44	84.69	91.56	\	67.20	75.88	75.96	76.75
WL_2_2	70.98	49.31	65.34	75.29	75.45	66.90	79.30	70.80	73.63	64.69	76.25	62.00	58.33	91.04	91.46	92.40	64.17	75.71	75.10	76.59
WL_4_3	65.48	53.29	69.90	69.31	\	\	\	\	\	\	\	\	\	\	\	\	\	\	\	\
WC_4	72.71	47.39	66.48	75.13	75.39	65.89	74.93	69.08	72.00	60.63	76.88	63.69	57.29	90.63	93.44	92.60	64.23	75.77	76.22	76.47
WC_6	65.52	52.36	67.95	65.15	74.25	55.30	78.40	\	72.81	52.87	77.94	\	57.19	90.73	90.94	92.60	64.04	75.65	76.22	76.59
DB_TransE	65.79	75.71	74.63	61.50	65.75	64.17	68.96	61.16	62.81	60.48	64.17	56.86	80.28	84.86	28.95	89.65	92.88	79.98	74.37	**95.44**
DB_TransH	64.39	72.66	76.66	60.89	63.51	63.25	67.43	60.96	63.97	63.13	65.07	60.23	80.39	84.86	27.55	89.21	93.82	79.98	74.37	93.68
DB_TransR	63.08	67.32	74.50	59.84	64.38	60.16	64.43	52.04	63.56	59.68	66.41	60.39	79.19	84.86	28.95	89.00	93.28	79.98	74.37	93.70
DB2vec SG 200w 200v 4d	75.66	73.50	75.68	47.58	77.58	78.24	79.61	72.78	73.85	73.79	75.28	65.95	78.21	85.18	29.57	91.83	82.24	81.07	74.10	85.23
DB2vec CBOW 200w 200v 4d	68.45	68.39	71.53	60.34	70.64	72.83	75.43	69.11	70.03	65.31	73.53	60.84	73.09	85.29	29.57	91.40	87.34	80.38	74.10	84.40
DB2vec CBOW 500w 200v 4d	59.32	68.84	77.39	64.32	65.60	79.74	82.90	74.33	70.72	71.86	76.36	67.24	73.36	89.65	29.00	92.45	89.38	80.94	76.83	84.81
DB2vec CBOW 500w 500v 4d	59.32	71.34	76.37	66.34	65.65	79.49	82.75	73.87	69.71	71.93	75.41	65.65	72.71	89.65	29.11	92.01	89.02	80.82	76.95	85.17
DB2vec SG 500w 200v 4d	60.34	71.82	76.37	65.37	65.25	80.44	83.25	73.87	68.95	73.89	76.11	67.87	71.20	89.65	28.90	92.12	88.78	80.82	77.92	85.77
DB2vec SG 500w 500v 4d	58.34	72.84	76.87	67.84	65.45	80.14	**83.65**	72.82	70.41	74.34	78.44	67.49	71.19	89.65	28.90	92.23	88.30	80.94	77.25	84.81
DB2vec CBOW 500w 200v 8d	69.26	69.87	67.32	63.13	57.83	70.08	65.25	67.47	67.91	64.44	72.42	65.39	68.18	85.33	28.90	90.50	77.35	80.34	28.90	85.17
DB2vec CBOW 500w 500v 8d	62.26	69.87	76.84	63.21	58.78	69.82	69.46	67.67	67.53	65.83	74.26	63.42	62.90	85.22	29.11	90.61	89.86	80.34	78.65	84.81
DB2vec SG 500w 200v 8d	73.32	75.89	78.92	60.74	79.94	79.49	83.30	75.13	77.25	76.87	**79.72**	69.14	78.53	85.12	29.22	91.04	90.10	80.58	78.96	84.68
DB2vec SG 500w 500v 8d	**89.73**	69.16	84.19	72.25	80.24	78.68	82.80	72.42	73.57	76.30	78.20	68.70	75.07	**94.48**	29.11	94.15	88.53	80.58	77.79	86.38
WD2vec CBOW 200w 200v 4d	68.76	57.71	75.56	61.37	51.49	52.20	51.64	49.01	50.86	50.29	51.44	50.09	50.54	90.18	89.63	88.83	49.84	81.08	76.77	79.14
WD2vec CBOW 200w 500v 4d	68.24	57.75	85.56	64.54	49.22	48.56	51.04	50.98	53.08	50.03	52.33	53.28	48.45	90.39	89.74	88.31	51.95	80.74	78.18	80.32
WD2vec SG 200w 200v 4d	72.58	57.53	75.48	52.32	69.53	70.14	75.39	67.00	60.32	62.03	64.76	58.54	60.87	90.50	89.63	89.98	65.45	81.17	77.74	77.03
WD2vec SG 200w 500v 4d	83.20	60.72	79.87	61.67	71.10	70.19	76.30	67.31	55.31	58.92	63.42	56.63	55.85	90.60	89.63	87.69	58.95	81.17	79.00	79.56

Table 4: Regression results. The first number represents the dimensionality of the vectors, while the second number represent the value for the depth parameter. The best results are marked in bold. Experiments that did not finish within ten days, or that have run out of memory are marked with "\".

Strategy/Dataset	Cities			Metacritic Movies			Metacritic Albums			AAUP			Forbes		
	LR	KNN	M5	LR	KNN	M5	LR	KNN	M5	LR	KNN	M5	LR	KNN	M5
types	24.30	22.16	18.79	77.80	30.68	22.16	16.45	18.36	13.95	9.83	34.95	6.28	29.22	21.07	18.32
categories	18.88	22.68	22.32	84.57	23.87	22.50	16.73	16.64	13.95	8.08	34.94	6.16	19.16	21.48	18.39
rel in	49.87	18.53	19.21	22.60	41.40	22.56	13.50	22.06	13.43	9.69	34.98	6.56	27.56	20.93	18.60
rel out	49.87	18.53	19.21	21.45	24.42	20.74	13.32	14.59	13.06	8.82	34.95	6.32	21.73	21.11	18.97
rel in & out	40.80	18.21	18.80	21.45	24.42	20.74	13.33	14.52	12.91	12.97	34.95	6.36	26.44	20.98	19.54
rel-vals in	\	\	\	21.46	24.19	20.43	13.94	23.05	13.95	\	34.96	6.27	\	20.86	19.31
rel-vals out	20.93	23.87	20.97	25.99	32.18	22.93	\	15.28	13.34	\	34.95	6.18	\	20.48	18.37
rel-vals in&out	\	\	\	\	25.37	20.96	\	15.47	13.33	\	34.94	6.18	\	20.20	18.20
WL_2_2	20.21	24.60	20.85	\	21.62	19.84	\	13.99	12.81	\	34.96	6.27	\	19.81	19.49
WL_4_3	17.79	20.42	17.04	\	\	\	\	\	\	\	\	\	\	\	\
WC_4	20.33	25.95	19.55	\	22.80	22.99	\	14.54	12.87	9.12	34.95	6.24	\	20.45	19.26
WC_6	19.51	33.16	19.05	\	23.86	19.19	\	19.51	13.02	\	35.39	6.31	\	20.58	19.04
DB_TransE	14.22	14.45	14.46	20.66	23.61	20.71	13.20	14.71	13.23	6.34	57.27	6.43	20.00	21.55	17.73
DB_TransH	13.88	12.81	14.28	20.71	23.59	20.72	13.04	14.19	13.03	6.35	57.27	6.47	19.88	21.54	16.66
DB_TransR	14.50	13.24	14.57	20.10	23.37	20.04	13.87	15.74	13.93	6.34	57.31	6.37	20.45	21.55	17.18
DB2vec SG 200w 200v 4d	14.26	14.02	15.15	16.82	18.52	16.56	11.57	12.38	11.56	6.26	57.20	6.45	19.68	20.85	18.50
DB2vec CBOW 200w 200v 4d	14.13	13.33	15.46	18.66	20.61	18.39	12.32	13.87	12.32	6.35	56.85	6.37	19.67	20.84	18.72
DB2vec CBOW 500w 200v 4d	14.37	12.55	14.33	15.90	17.46	15.89	11.79	12.45	11.59	12.13	45.76	12.00	18.32	26.19	17.43
DB2vec CBOW 500w 500v 4d	14.99	12.46	14.66	15.90	17.45	15.73	11.49	12.60	11.48	12.44	45.67	12.30	18.23	26.27	17.62
DB2vec SG 500w 200v 4d	13.38	12.54	15.13	15.81	17.07	15.84	11.30	12.36	11.42	12.13	45.72	12.10	17.63	26.13	17.85
DB2vec SG 500w 500v 4d	14.73	13.25	16.80	15.66	17.14	15.67	11.20	12.11	11.28	12.09	45.76	11.93	18.23	26.09	17.74
DB2vec CBOW 500w 200v 8d	16.17	17.14	17.56	21.55	23.75	21.46	13.35	15.41	13.43	6.47	55.76	6.47	24.17	26.48	22.61
DB2vec CBOW 500w 500v 8d	18.13	17.19	18.50	20.77	23.67	20.69	13.20	15.14	13.25	6.54	55.33	6.55	21.16	25.90	20.33
DB2vec SG 500w 200v 8d	12.85	14.95	12.92	15.15	17.13	**15.12**	10.90	11.43	10.90	6.22	56.95	6.25	18.66	21.20	18.57
DB2vec SG 500w 500v 8d	11.92	12.67	**10.19**	15.45	17.80	15.50	**10.89**	11.72	10.97	6.26	56.95	6.29	18.35	21.04	**16.61**
WD2vec CBOW 200w 200v 4d	20.15	17.52	20.02	23.54	25.90	23.39	14.73	16.12	14.55	16.80	42.61	6.60	27.48	22.60	21.77
WD2vec CBOW 200w 500v 4d	23.76	18.33	20.39	24.14	22.18	24.56	14.09	16.09	14.00	13.08	42.89	6.08	50.23	21.92	26.66
WD2vec SG 200w 200v 4d	20.47	18.69	20.72	19.72	21.44	19.10	13.51	13.91	13.67	6.86	42.82	6.52	23.69	21.59	20.49
WD2vec SG 200w 500v 4d	22.25	19.41	19.23	25.99	21.26	19.19	13.23	14.96	13.25	8.27	42.84	**6.05**	21.98	21.73	21.58

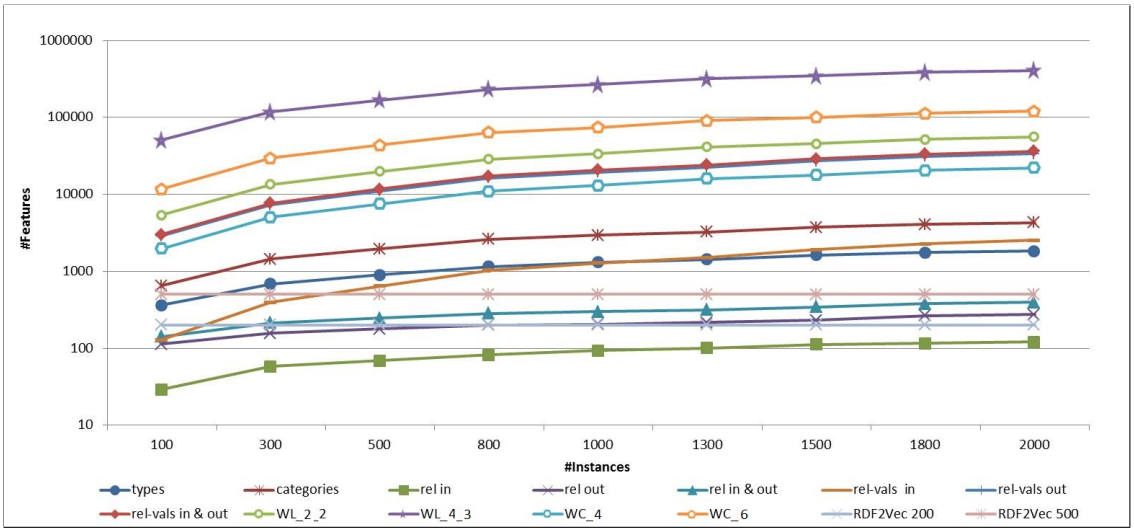


Fig. 3. Features increase rate per strategy (log scale).

6.1.2. Document Similarity

We use those entity similarity scores in the task of calculating semantic document similarity. We follow a similar approach as the one presented in [57], where two documents are considered to be similar if many entities of the one document are similar to at least 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 other pairs.

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:

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

6.1.3. Entity Relatedness

In this approach 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 3. Given a 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:

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

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

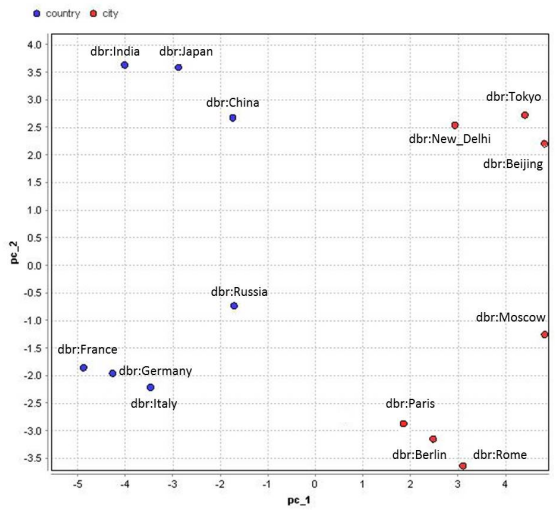
In the case of a skip-gram model, the probability is calculated as:

$$p(e_1|e_2) = \frac{exp(v'^T_{e_1} v_{e_2})}{\sum_{e=1}^{V} exp(v'^T_e v_{e_2})}, \tag{9}$$

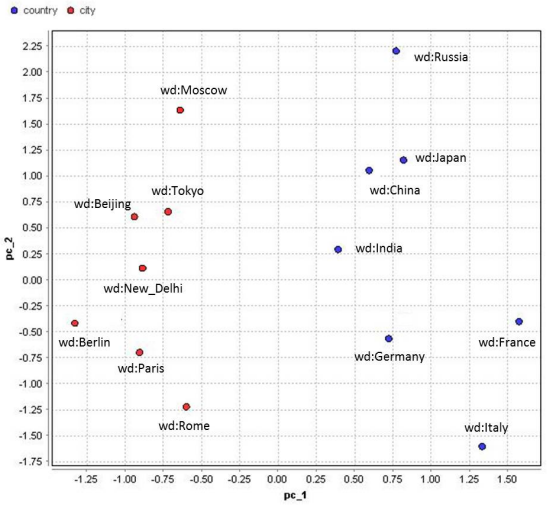
where v_e and v'_e are the input and the output vector of the entity e, and V is the complete vocabulary of entities.

6.2. Evaluation

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.



a) *DBpedia vectors*



b) *Wikidata vectors*

Fig. 2. Two-dimensional PCA projection of the 500-dimensional Skip-gram vectors of countries and their capital cities.

6.2.1. Entity Relatedness

For evaluating the entity relatedness approach, we use the KORE dataset [26]. 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.

We use two approaches for calculating the relatedness rank between the entities, i.e. (i) the entity similarity approach described in section 6.1.1; (ii) the entity relatedness approach described in section 6.1.3.

We evaluate each of the RDF2Vec models separately. Furthermore, we also compare to the Wiki2vec model[16], which is built on the complete Wikipedia corpus, and provides vectors for each DBpedia entity.

Table 5 shows the Spearman's rank correlation results when using the entity similarity approach. Table 6 shows the results for the relatedness approach. The results show that the DBpedia models outperform the Wikidata models. Increasing the number of walks per entity improves the results. Also, the skip-gram models outperform the CBOW models continuously. We can observe that the relatedness approach outperforms the similarity approach.

Furthermore, we compare our approaches to several state-of-the-art graph-based entity relatedness approaches:

- baseline: computes entity relatedness as a function of distance between the entities in the network, as described in [76].
- KORE: calculates keyphrase overlap relatedness, as described in the original KORE paper [26].
- CombIC: semantic similarity using a Graph Edit Distance based measure [76].
- ER: Exclusivity-based relatedness [29].

The comparison shows that our entity relatedness approach outperforms all the rest for each category of entities. Interestingly enough, the entity *similarity* approach, although addressing a different task, also outperforms the majority of state of the art approaches.

6.3. Document Similarity

To evaluate the document similarity approach, we use the LP50 dataset [35], 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 of all annotators are averaged. As a evaluation metrics we use Pearson's linear correlation coefficient and Spearman's rank correlation plus their harmonic mean.

Again, we first evaluate each of the RDF2Vec models separately. Table 8 shows document similarity results. As for the entity relatedness, the results show that the skip-gram models built on DBpedia with 8 hops lead to the best performances.

[16] https://github.com/idio/wiki2vec

Table 5
Similarity-based relatedness Spearman's rank correlation results

Model	IT companies	Hollywood Celebrities	Television Series	Video Games	Chuck Norris	All 21 entities
DB2vec SG 200w 200v 4d	0.525	0.505	0.532	0.571	0.439	0.529
DB2vec CBOW 200w 200v	0.330	0.294	0.462	0.399	0.179	0.362
DB2vec CBOW 500w 200v 4d	0.538	0.560	0.572	0.596	0.500	0.564
DB2vec CBOW 500w 500v 4d	0.546	0.544	0.564	0.606	0.496	0.562
DB2vec SG 500w 200v 4d	0.508	0.546	0.497	0.634	0.570	0.547
DB2vec SG 500w 500v 4d	0.507	0.538	0.505	0.611	0.588	0.542
DB2vec CBOW 500w 200v 8d	0.611	0.495	0.315	0.443	0.365	0.461
DB2vec CBOW 500w 500v 8w	0.486	0.507	0.285	0.440	0.470	0.432
DB2vec SG 500w 200v 8w	0.739	0.723	0.526	0.659	0.625	0.660
DB2vec SG 500w 500v 8w	**0.743**	**0.734**	**0.635**	**0.669**	**0.628**	**0.692**
WD2vec CBOW 200w 200v 4d	0.246	0.418	0.156	0.374	0.409	0.304
WD2vec CBOW 200w 500v 4d	0.190	0.403	0.103	0.106	0.150	0.198
WD2vec SG 200w 200v 4d	0.502	0.604	0.405	0.578	0.279	0.510
WD2vec SG 200w 500v 4d	0.464	0.562	0.313	0.465	0.168	0.437
Wiki2vec	0.613	0.544	0.334	0.618	0.436	0.523

Table 6
Context-based relatedness Spearman's rank correlation results

Model	IT companies	Hollywood Celebrities	Television Series	Video Games	Chuck Norris	All 21 entities
DB2vec SG 200w 200v 4d	0.643	0.547	0.583	0.428	0.591	0.552
DB2vec CBOW 200w 200v	0.361	0.326	0.467	0.426	0.208	0.386
DB2vec CBOW 500w 200v 4d	0.671	0.566	0.591	0.434	0.609	0.568
DB2vec CBOW 500w 500v 4d	0.672	0.622	0.578	0.440	0.581	0.578
DB2vec SG 500w 200v 4d	0.666	0.449	0.611	0.360	0.630	0.526
DB2vec SG 500w 500v 4d	0.667	0.444	0.609	0.389	0.668	0.534
DB2vec CBOW 500w 200v 8d	0.579	0.484	0.368	0.460	0.412	0.470
DB2vec CBOW 500w 500v 8d	0.552	0.522	0.302	0.487	0.665	0.475
DB2vec SG 500w 200v 8d	**0.811**	**0.778**	**0.711**	**0.658**	**0.670**	**0.736**
DB2vec SG 500w 500v 8d	0.748	0.729	0.689	0.537	0.625	0.673
WD2vec CBOW 200w 200v 4d	0.287	0.241	-0.025	0.311	0.226	0.205
WD2vec CBOW 200w 500v 4d	0.166	0.215	0.233	0.335	0.344	0.243
WD2vec SG 200w 200v 4d	0.574	0.671	0.504	0.410	0.079	0.518
WD2vec SG 200w 500v 4d	0.661	0.639	0.537	0.395	0.474	0.554
Wiki2vec	0.291	0.296	0.406	0.353	0.175	0.329

Furthermore, we compare our approach to several state-of-the-art graph-based document similarity approaches:

- TF-IDF: Distributional baseline algorithm.
- AnnOv: Similarity score based on annotation overlap that corresponds to traversal entity similarity with radius 0, as described in [57].
- Explicit Semantic Analysis (ESA) [17].
- GED: semantic similarity using a Graph Edit Distance based measure [76].
- Salient Semantic Analysis (SSA), Latent Semantic Analysis (LSA) [24].
- Graph-based Semantic Similarity (GBSS) [57].

The results for the related approaches were copied from the respective papers, except for ESA, which was copied from [57], where it is calculated via public ESA REST endpoint[17]. The results show that our document

[17] http://vmdeb20.deri.ie:8890/esaservice

Table 7
Spearman's rank correlation results comparison to related work

Approach	IT companies	Hollywood Celebrities	Television Series	Video Games	Chuck Norris	All 21 entities
baseline	0.559	0.639	0.529	0.451	0.458	0.541
KORE	0.759	0.715	0.599	**0.760**	0.498	0.698
CombIC	0.644	0.690	0.643	0.532	0.558	0.624
ER	0.727	0.643	0.633	0.519	0.477	0.630
DB_TransE	-0.023	0.120	-0.084	0.353	-0.347	0.070
DB_TransH	-0.134	0.185	-0.097	0.204	-0.044	0.035
DB_TransR	-0.217	0.062	0.002	-0.126	0.166	0.058
DB2Vec Similarity	0.743	0.734	0.635	0.669	0.628	0.692
DB2Vec Relatedness	**0.811**	**0.778**	**0.711**	0.658	**0.670**	**0.736**

Table 8
Document similarity results - Pearson's linear correlation coefficient (r) Spearman's rank correlation (ρ) and their harmonic mean μ

Model	r	ρ	μ
DB2vec SG 200w 200v 4d	0.608	0.448	0.516
DB2vec CBOW 200w 200v 4d	0.562	0.480	0.518
DB2vec CBOW 500w 200v 4d	0.681	0.535	0.599
DB2vec CBOW 500w 500v 4d	0.677	0.530	0.594
DB2vec SG 500w 200v 4d	0.639	0.520	0.573
DB2vec SG 500w 500v 4d	0.641	0.516	0.572
DB2vec CBOW 500w 200v 8d	0.658	0.491	0.562
DB2vec CBOW 500w 500v 8d	0.683	0.512	0.586
DB2vec SG 500w 200v 8d	**0.708**	**0.556**	**0.623**
DB2vec SG 500w 500v 8d	0.686	0.527	0.596
WD2vec CBOW 200w 200v 4d	0.568	0.383	0.458
WD2vec CBOW 200w 500v 4d	0.593	0.386	0.467
WD2vec SG 200w 200v 4d	0.606	0.385	0.471
WD2vec SG 200w 500v 4d	0.613	0.343	0.440
Wiki2vec	0.662	0.513	0.578
DB_TransE	0.565	0.432	0.490
DB_TransH	0.570	0.452	0.504
DB_TransR	0.578	0.461	0.513

Table 9
Comparison of the document similarity approach to the related work

Approach	r	ρ	μ
TF-IDF	0.398	0.224	0.287
AnnOv	0.590	0.460	0.517
LSA	0.696	0.463	0.556
SSA	0.684	0.488	0.570
GED	0.630	\	\
ESA	0.656	0.510	0.574
GBSS	0.704	0.519	0.598
DB2Vec	**0.708**	**0.556**	**0.623**

similarity approach outperforms all of the related approaches for both Pearson's linear correlation coefficient and Spearman's rank correlation, as well as their harmonic mean.

We do not compare our approach to the machine-learning approach proposed by Huang et al. [27], because that approach is a supervised one, which is tailored towards the dataset, whereas ours (as well as the others we compare to) are unsupervised.

7. Recommender Systems

As discussed in Section 2.3, the *Linked Open Data* (LOD) initiative [4] has opened new interesting possibilities to realize better recommendation approaches. Given that the items to be recommended are linked to a LOD dataset, information from LOD can be exploited to determine which items are considered to be similar to the ones that the user has consumed in the past, allowing to discover hidden information and implicit relations between objects. While LOD is rich in high quality data, it is still challenging to find an effective and efficient way of exploiting the knowledge for content-based recommendations. So far, most of the proposed approaches in the literature are supervised or semi-supervised, which means they cannot work without human interaction. New challenges, and at the same time new opportunities, arise from unsupervised approaches for feature learning.

RDF graph embeddings are a promising way to approach those challenges and build content-based recommender systems. As for entity similarity in the previous section, the cosine similarity between the latent

vectors representing the items can be interpreted as a measure of reciprocal proximity and then exploited to produce recommendations.

In this section, we explore the development of a hybrid RS, leveraging the latent features extracted with RDF2Vec. The hybrid system takes advantage of both RDF graph embeddings and Factorization Machines (FMs) [63], an effective method combining Support Vector Machines with factorization models.

7.1. Factorization Machines

Factorization Machines (FMs) [63] are a general predictor model which rely on the advantages of Support Vector Machines (SVMs) and factorization models. SVMs have been successfully applied in many classification and regression problems but they have been proved to be not effective in those settings like collaborative filtering which are characterized by huge sparsity. This happens since in sparse settings there is not enough data to estimate a "dense" factorization as SVMs would do. FM use a factorized parametrization, breaking the independence of the interaction parameters by factorizing them. The difference clearly emerges when the model equations of degree 2 for a factorization machine and for a SVM are directly compared. Being aware that in a prediction task the general goal is to estimate a function $y : \mathcal{R}^n \rightarrow T$, from a real valued feature vector $x \in \mathcal{R}^n$ to a target domain T, the model equation [63] for a FM of degree 2 is given by:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} (v_i^T v_j) x_i x_j \quad (10)$$

where $\hat{y}$ represents the estimate of function y. In (10) the parameters to be estimated are the global bias $w_0 \in \mathcal{R}$, the vector $w = [w_1, ..., w_n] \in \mathcal{R}^n$ whose component w_i models the strength of the i_{th} variable and a matrix $V \in \mathcal{R}^{n \times k}$, which describes the i_{th} variable with its i_{th} row v_i, i.e., with a vector of size k, where k is a hyperparameter defining the factorization dimensionality. On the other hand the model equation for a SVM of the same degree 2 is defined as:

$$\hat{y}(x) = w_0 + \sqrt{2} \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} w_{i,i}^{(2)} x_i^2 + \sqrt{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{i,j}^{(2)} x_i x_j \quad (11)$$

with model parameters $w_0 \in \mathcal{R}$, $w \in \mathcal{R}^n$ and the symmetric matrix $W^{(2)} \in \mathcal{R}^{n \times n}$. From the comparison of equations (10) and (11) we can realize that both model all nested interactions up to degree 2 but with a fundamental difference: the estimation of the interactions between variables in the SVM model is done directly and independently with $w_{i,j}$ whereas it is parametrized in the FM model. In other words, parameters $w_{i,j}$ and $w_{i,l}$ will be completely independent in the SVM case while parameters $v_i^T v_j$ and $v_i^T v_l$ overlap thanks to the shared parameter v_i.

The strength of FMs models is also due to linear complexity [63] and due to the fact that they can be optimized in the primal and they do not rely on support vectors like SVMs. Finally, [63] shows that many of the most successful approaches for the task of collaborative filtering are de facto subsumed by FMs. It is the case for example of Matrix Factorization [80,74] and SVD++ [32].

7.2. Experiments

We evaluate different variants of our approach on three datasets, and compare them to common approaches for creating content-based item representations from LOD, as well as to state of the art collaborative and hybrid approaches. Furthermore, we investigate the use of two different LOD datasets as background knowledge, i.e., DBpedia and Wikidata.

7.2.1. Datasets

In order to test the effectiveness of vector space embeddings for the recommendation task, we have performed an extensive evaluation in terms of ranking accuracy on three datasets belonging to different domains, i.e., `Movielens`[18] for movies, `LibraryThing`[19] for books, and `Last.fm`[20] for music. The first dataset, `Movielens 1M`, contains 1 million 1-5 stars ratings from 6,040 users on 3,952 movies. The dataset

[18] `http://grouplens.org/datasets/movielens/`
[19] `https://www.librarything.com/`
[20] `http://www.lastfm.com`

LibraryThing contains more than 2 millions ratings from 7,279 users on 37,232 books. As in the dataset there are many duplicated ratings, when a user has rated the same item more than once, her last rating is selected. The unique ratings are 749,401, in the range from 1 to 10. Both Movielens and LibraryThing datasets contain explicit ratings, and to test the approach also on implicit feedbacks, a third dataset built on the top of the Last.fm music system is considered. Last.fm contains 92,834 interactions between 1,892 users and 17,632 musical artists. Each interaction is annotated with the corresponding listening count.

The original datasets are enriched with background information using the item mapping and linking to DBpedia technique described in [54], whose dump is available at https://github.com/sisinflab/LODrecsys-datasets. Since not all the items have a corresponding resource in DBpedia, after the mapping, the versions of Movielens, LibraryThing and Last.fm datasets contain 3,883 movies, 11,695 books, and 11,180 musical artists, respectively.

The datasets are finally preprocessed to guarantee a fair comparison with the state of the art approaches described in [12]. Here, the authors propose to (i) remove popularity biases from the evaluation not considering the top 1% most popular items, (ii) reduce the sparsity of Movielens dataset in order to have at least a sparser test dataset and (iii) remove from LibraryThing and Last.fm users with less than five ratings and items rated less than five times. The final statistics on the three datasets are reported in Table 10.

Table 10
Statistics about the three datasets

	Movielens	**LibraryThing**	**Last.fm**
Number of users	4,186	7,149	1,875
Number of items	3,196	4,541	2,432
Number of ratings	822,597	352,123	44,981
Data sparsity	93.85%	98.90%	99.01%

7.2.2. Evaluation Protocol

The ranking setting for the recommendation task consists of producing a ranked list of items to suggest to the user and in practical situations turns into the so-called top-N recommendation task, where just a cut-off of the ranked list of size N is provided to the user. This setting has recently replaced the rating prediction, because of the increasing awareness that the user is not interested in an accurate prediction of the item rating, but is looking for a (limited) list of items extracted from the pool of available ones.

As evaluation ranking protocol for our comparison, we adopted the *all unrated items* methodology presented in [81] and already used in [12]. Such methodology asks to predict a score for each item not rated by a user, irrespective of the existence of an actual rating, and to compare the recommendation list with the test set.

The metrics involved in the experimental comparison are three well-known ranking measures for recommendation accuracy, i.e., precision, recall, F-score (F1) and nDCG.

- *precision@N* [65] represents the fraction of relevant items in the top-N recommendations.
- *recall@N* [65] indicates the fraction of relevant items, in the user test set, occurring in the top-N list. As relevance threshold, we set 4 for Movielens and 8 for LibraryThing, as previously done in [12].
- The *F1* score [65], i.e., the harmonic mean of precision and recall, is also pointed out to be thorough. Although precision and recall are good indicators to evaluate the accuracy of a recommendation engine, they are not rank-sensitive.
- The normalized Discounted Cumulative Gain *nDCG@N* [2] instead takes into account also the position in the recommendation list, being defined as

$$\text{nDCG@}\,N = \frac{1}{\text{iDCG}} \cdot \sum_{i=1}^{N} \frac{2^{rel(u,i)} - 1}{log_2(1+i)} \quad (12)$$

  where $rel(u,i)$ is a boolean function representing the relevance of item i for user u and iDCG is a normalization factor that sets $\text{nDCG@}\,N$ value to 1 when an ideal ranking is returned [2].

As suggested in [81] and set up in [12], in the computation of nDCG@N we fixed a default "neutral" value for those items with no ratings, i.e., 3 for Movielens and 5 for LibraryThing.

All the results have been computed @10, that is considering the top-10 list recommended to each user and then averaging across all users.

7.2.3. Experimental Setup

The target of this experimental section is two-fold. On the one hand, we show that the proposed graph embeddings technique outperforms other strategies for

feature creation. On the other hand, we combine it with a hybrid RS and compare the resulting system's relative performances with state of the art approaches. The first goal is pursued by implementing an item-based K-Nearest Neighbor method, hereafter denoted as ItemKNN, with cosine similarity among features vectors. As data mining features, the approaches based on direct relations and graph substructures, as detailed in Section 5.1, are considered.

For what concerns the comparison against some of the most promising collaborative and hybrid approaches currently available, we consider the results recently published in [12]. There, the authors show that the best effectivenesses in terms of ranking accuracy is reached by the following algorithms:

- `SLIM` [51] is a Sparse LInear Method for top-N recommendation that learns a sparse coefficient matrix for the items involved in the system by only relying on the users purchase/ratings profile and by solving a L1-norm and L2-norm regularized optimization problem.
- `BPR-SSLIM` is a Sparse LInear Method using item Side information (SSLIM) [52] and the Bayesian Personalized Ranking (BPR) optimization criterion [64].
- `SPRank` [12] is a novel hybrid recommender system that solves the top-N recommendation problem in a learning to rank fashion, exploiting the freely available knowledge in the Linked Open Data to build semantic path-based features.

In [12], it has been proved that `SPRank` is able to outperform `SLIM` and `BPR-SSLIM` (at least according to Precision and nDCG) in those context characterized by higher sparsity, where reasonably the contribution of the content is more essential. For the sake of completeness, it is necessary to remind that in [12] the aforementioned approaches are tested against well established collaborative algorithms for rating prediction, e.g., *Biased Matrix Factorization* [33], and for items ranking, such as *Bayesian Personalized Ranking* [64] and *Soft Margin Ranking Matrix Factorization* [89]. Furthermore, two algorithms that work on heterogeneous information networks are also involved in the comparison, i.e., *PathRank* [36], an extension of Personalized PageRank for heterogeneous networks, and *HeteRec* [92], which represents the connectivity between users and items through a matrix factorization algorithm that uses meta-path based latent features.

7.3. Results

We present the results by a purely content-based RS and a hybrid RS using RDF2Vec.

7.3.1. Content-based Approach

The first experiment aims to point out the validity of the proposed graph embeddings technique for feature generation in the context of content-based RS, and relies on a relatively simple recommendation algorithm, i.e., the item-based K-Nearest Neighbor approach [65] with cosine similarity. Formally, this method evaluates the closeness of items through cosine similarity between the corresponding features vectors and then selects a subset of those – the neighbors – for each item, that will be used to estimate the rating of user u for a new item i as follows:

$$r^*(u,i) = \frac{\sum_{j \in ratedItems(u)} cosineSim(j,i) \cdot r_{u,j}}{\sum_{j \in ratedItems(u)} |cosineSim(j,i)|} \tag{13}$$

where $ratedItems(u)$ is the set of items already evaluated by user u, $r_{u,j}$ indicates the rating for item j by user u and $cosineSim(j,i)$ is the cosine similarity score between items j and i. In our experiments, the size of the considered neighbourhood is limited to 5.

The vector representation of items made of latent features, output of the RDF graph embeddings with graph walks, is evaluated against all the other data mining features listed in Section 5.1. Tables 11, 12 and 13 contain the values of precision, recall, F-score (F1) and nDCG, respectively for `Movielens`, `LibraryThing` and `Last.fm`. The computation of recommendations has been done with the publicly available library RankSys.[21]

The first conclusion that can be drawn from Tables 11, 12 and 13 is that the best approach for all datasets is retrieved with a skip-gram model, 500 walks per entity and with a size of 200 for vectors built upon DBpedia. Altough on `Movielens`, the highest value of precision is achived using vector size of 500, the size 200 is prevalent according to the F1 measure. A substantial difference concerns the exploratory depth of the random walks, since for `Movielens` the results related to a depth of 4 outperform those computed with a depth of 8, while the tendency is reversed for both

[21] `http://ranksys.org/`

Table 11
Results of the ItemKNN approach on `Movielens` dataset with reference to different computation of features.

Strategy	Precision	Recall	F1	nDCG
types	0.00313	0.00145	0.00198	0.28864
categories	0.0305	0.02093	0.02482	0.30444
rel in	0.01122	0.00589	0.0077	0.29183
rel out	0.02844	0.01607	0.02053	0.30274
rel in & out	0.02852	0.01566	0.02021	0.3006
rel-vals in	0.03883	0.02293	0.02882	0.29411
rel-vals out	0.01279	0.00971	0.011	0.29378
rel-vals in & out	0.01174	0.00913	0.01027	0.29333
WC_4	0.00684	0.00343	0.0045	0.29032
WL_2_2	0.00601	0.00288	0.00389	0.28977
DB_TransE	0.03047	0.01411	0.01928	0.30385
DB_TransH	0.02649	0.01187	0.01639	0.30016
DB_TransR	0.00941	0.0043	0.0059	0.29216
DB2vec SG 200w 200v 4d	0.05423	0.02693	0.03598	0.31676
DB2vec CBOW 200w 200v 4d	0.03475	0.01637	0.02225	0.30426
DB2vec CBOW 500w 200v 4d	0.03893	0.02167	0.02784	0.30782
DB2vec CBOW 500w 500v 4d	0.03663	0.02088	0.02659	0.30557
DB2vec SG 500w 200v 4d	0.05681	**0.03119**	**0.04027**	**0.31828**
DB2vec SG 500w 500v 4d	**0.05786**	0.0304	0.03985	0.31726
DB2vec CBOW 500w 200v 8d	0.01064	0.00548	0.00723	0.29245
DB2vec CBOW 500w 500v 8d	0.01137	0.00567	0.00756	0.29289
DB2vec SG 500w 200v 8d	0.04424	0.02693	0.03347	0.30997
DB2vec SG 500w 500v 8d	0.02191	0.01478	0.01765	0.29863
WD2vec CBOW 200w 200v 4d	0.01217	0.00596	0.00800	0.29362
WD2vec CBOW 200w 500v 4d	0.01027	0.00427	0.0060	0.29211
WD2vec SG 200w 200v 4d	0.02902	0.01479	0.01959	0.30189
WD2vec SG 200w 500v 4d	0.02644	0.01246	0.01693	0.29967

`LibraryThing` and `Last.fm`. Secondly, the advantage of the Skip-Gram model over CBOW is a constant both on DBpedia and Wikidata and is particularly evident when the model involves longer random walks, i.e., with depth 8. Comparing the LOD datasets, it clearly emerges that DBpedia lets to gain higher values than Wikidata for each metric involved, but it turns out that Wikidata is quite effective on `LibraryThing`, where the skip-gram vectors with depth of 4 exceed the corresponding DBpedia vectors. Moving to the features extracted from direct relations, the contribution of the "categories" stands clearly out, together with relations-values "rel-vals", especially when just incoming relations are considered: these features allow to achieve better results than the approaches based on translating embeddings, i.e., DB_TransE, DB_TransH and DB_TransR. The use of methods based on kernels for features extraction, i.e., WC_4 and WL_2_2 approaches, seems not to provide significant advantages to the recommendation algorithm.

To point out that the latent features built upon RDF graph are able to capture its structure, placing closely semantically similar items, some examples of the neighbouring sets retrieved using the graph embeddings technique are provided. These sets are directly exploited by the ItemKNN algorithm to produce recommendations. Table 14 is related to movies and to the strategy "DB2vec SG 500w 200v 4d", and displays

Table 12
Results of the ItemKNN approach on `LibraryThing` dataset with reference to different computation of features.

Strategy	Precision	Recall	F1	nDCG
types	0.01854	0.04535	0.02631	0.16064
categories	0.06662	0.15258	0.09274	0.23733
rel in	0.04577	0.10219	0.06322	0.20196
rel out	0.04118	0.09055	0.05661	0.19449
rel in & out	0.04531	0.10165	0.06268	0.20115
rel-vals in	0.06176	0.14101	0.08589	0.22574
rel-vals out	0.06163	0.13763	0.08513	0.22826
rel-vals in & out	0.06087	0.13662	0.08421	0.22615
WC_4	0.00159	0.00306	0.00209	0.12858
WL_2_2	0.00155	0.00389	0.00221	0.12937
DB_TransE	0.01819	0.04705	0.02623	0.1585
DB_TransH	0.01466	0.03997	0.02145	0.15331
DB_TransR	0.00162	0.00341	0.00219	0.12947
DB2vec SG 200w 200v 4d	0.00442	0.00942	0.00601	0.13502
DB2vec CBOW 200w 200v 4d	0.00466	0.00933	0.00621	0.13595
DB2vec CBOW 500w 200v 4d	0.05127	0.11777	0.07143	0.21244
DB2vec CBOW 500w 500v 4d	0.05065	0.11557	0.07043	0.21039
DB2vec SG 500w 200v 4d	0.05719	0.12763	0.07898	0.2205
DB2vec SG 500w 500v 4d	0.05811	0.12864	0.08005	0.22116
DB2vec CBOW 500w 200v 8d	0.00836	0.02334	0.01231	0.14147
DB2vec CBOW 500w 500v 8d	0.00813	0.02335	0.01206	0.14257
DB2vec SG 500w 200v 8d	**0.07681**	**0.17769**	**0.10725**	**0.25234**
DB2vec SG 500w 500v 8d	0.07446	0.1743	0.10434	0.24809
WD2vec CBOW 200w 200v 4d	0.00537	0.01084	0.00718	0.13524
WD2vec CBOW 200w 500v 4d	0.00444	0.00984	0.00611	0.13428
WD2vec SG 200w 200v 4d	0.06416	0.14565	0.08907	0.23309
WD2vec SG 200w 500v 4d	0.06031	0.14194	0.08465	0.22752

that neighboring items are highly relevant and close to the query item, i.e., the item for which neighbors are searched for. Figure 4 depicts the 2D PCA projection of the movies in that table, showing that similar movies are actually projected closely to each other.

7.3.2. Hybrid Approach

To second goal of our experiments is to show that a hybrid recommender system, leveraging the item content built with the graph embeddings technique, can compete against some of the most promising state of the art approaches for recommendation. A hybrid RS based on Factorization Machines is utilized, namely the *ranking factorization machine* algorithm available in the *Graphlab* recommender toolkit[22]. This algorithm allows to define both users and items side data. Grounding on the capability to automatically cluster semantically similar items pointed out by the experiments of the previous section, the additional information added to each item, as side data, is the list of its 5 nearest neighbors. The tuning of the hyperparameters (specifically the numbers of latent factors and the maximum number of iterations) has been carried out through cross-validation on a validation set obtained selecting the 15% of the ratings of each user from the training set. The same choice for cross vali-

[22] `https://dato.com/products/create/docs/graphlab.toolkits.recommender.html`

Table 13
Results of the ItemKNN approach on `Last.fm` with reference to different computation of features.

Strategy	Precision	Recall	F1	nDCG
types	0.00525	0.03256	0.009	0.01826
categories	0.01762	0.09889	0.02991	0.06023
rel in	0.00625	0.03625	0.01066	0.02042
rel out	0.00519	0.02757	0.00873	0.01733
rel in & out	0.00718	0.04205	0.01226	0.02567
rel-vals in	0.0185	0.10502	0.03145	0.06733
rel-vals out	0.00805	0.04585	0.01369	0.0248
rel-vals in & out	0.00339	0.01774	0.00569	0.00982
WC_4	0.00086	0.00401	0.00141	0.00241
WL_2_2	0.00086	0.00344	0.00137	0.00215
DB_TransE	0.01117	0.06078	0.01887	0.03953
DB_TransH	0.01409	0.07928	0.02392	0.04881
DB_TransR	0.0011	0.00523	0.00181	0.00381
DB2vec SG 200w 200v 4d	0.01	0.05498	0.01692	0.02911
DB2vec CBOW 200w 200v 4d	0.00929	0.05227	0.01577	0.03288
DB2vec CBOW 500w 200v 4d	0.01749	0.09915	0.02973	0.06435
DB2vec CBOW 500w 500v 4d	0.01769	0.10016	0.03006	0.06404
DB2vec SG 500w 200v 4d	0.02015	0.11109	0.03411	0.07232
DB2vec SG 500w 500v 4d	0.02001	0.10978	0.03385	0.07448
DB2vec CBOW 500w 200v 8d	0.00944	0.05349	0.01604	0.03311
DB2vec CBOW 500w 500v 8d	0.00964	0.0563	0.01646	0.03166
DB2vec SG 500w 200v 8d	**0.0234**	**0.1359**	**0.03992**	**0.08719**
DB2vec SG 500w 500v 8d	0.02088	0.12248	0.03567	0.07789
WD2vec CBOW 200w 200v 4d	0.00133	0.00785	0.00227	0.00382
WD2vec CBOW 200w 500v 4d	0.001	0.00532	0.00168	0.00408
WD2vec SG 200w 200v 4d	0.00612	0.03388	0.01036	0.02157
WD2vec SG 200w 500v 4d	0.00658	0.03932	0.01127	0.02382

dation has been adopted for the competing recommendations algorithms, as already done in [12].

Tables 15, 16, and 17 report, respectively for `Movielens`, `LibraryThing` and `Last.fm`, the most promising approach according to the analysis conducted with the ItemKNN algorithm in Section 7.3.1 (Tables from 11 to 13) and the competing approaches `SPRank`, `SLIM` and `BPR-SSLIM`[23]. For `BPR-SSLIM`, the resources directly connected to the items are used as item side data, as in [12]. The prefix "$\mathcal{H}$-" is used as a notation for the hybrid RS grounding on the relative feature vectors. `SLIM` and `BPR-SSLIM` are implemented through the publicly available software library MyMediaLite[24] [18], while for `SPRank`, the implementation provided by its authors, available at `https://github.com/sisinflab/lodreclib`, is used. We repeated the experiments three times for each datasets and then averaged the results across the three runs (reporting also the standard deviation).

Looking at Tables 15, 16 and 17, we can observe performance improvements on all datasets and according to each of the metrics involved, with the only exception of the precision on the `Last.fm` dataset. In

[23]The complete version of these tables and the code for implementing the hybrid system are available at `http://sisinflab.poliba.it/recommender-systems/hybridSystemRDF2Vec.html`

[24]`http://www.mymedialite.net`

Table 14
Examples of K-Nearest Neighbor sets on Movielens.

Query Movie	K Nearest Neighbours
Batman	Batman Forever, Batman Returns, Batman & Robin, Superman IV: The Quest for Peace, Dick Tracy
Bambi	Cinderella, Dumbo, 101 Dalmatians , Pinocchio, Lady and the Tramp
Star Trek: Generations	Star Trek VI: The Undiscovered Country, Star Trek: Insurrection, Star Trek III: The Search for Spock, Star Trek V: The Final Frontier, Star Trek: First Contact

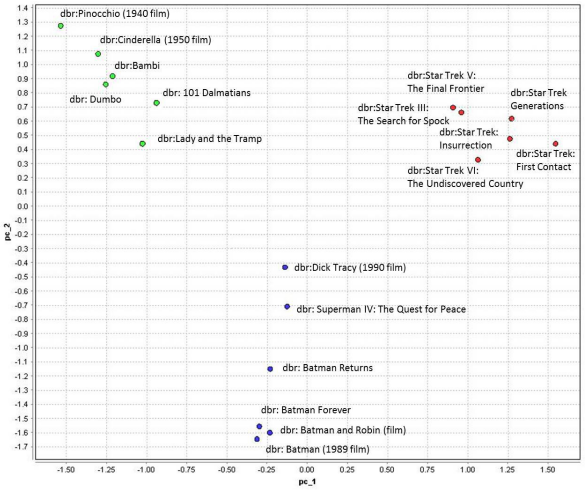


Fig. 4. Two-dimensional PCA projection of the 200-dimensional Skip-Gram vectors of movies in Table 14.

particular, on Movielens, the hybrid SPRank cannot compete with the collaborative method SLIM and to its extended version with side data, BPR−SSLIM, probably because of the key contribution of collaborative information on this dataset. In fact, Movielens is the denser dataset in the comparison, containing only experienced users with more than fifty ratings. Nevertheless, the approach "$\mathcal{H}$-DB2vec SG 500w 200v 4d" outperforms the competing ones, reasonably with the key contribution of factorization models in FM. On the other datasets, whose sparsity better reproduces real-world scenarios, the contribution of content appears more determinant. This emerges in particular on LibraryThing, where BPR−SSLIM and SPRank overtake SLIM and "$\mathcal{H}$-DB2vec SG 500w 200v 8d" improves the results even further. On the third dataset, i.e., Last.fm, the hybrid approaches generally perform better than SLIM (except on precision for BPR−SSLIM) and the approach based on graph embeddings reports a further enhancement in recall and nDCG. The results are statistically significant according to the Wilcoxon signed rank with $p < 0.001$, for less than a different specification. In conclusion, the choice of using Factorization Machines and of building item side data by means of the RDF graph embeddings technique, turns out to be favorable for fostering accuracy in general and getting improvements, on every dataset, on the rank-sensitive metric nDCG. We remind that the importance of providing to the user an opportunely *ranked* list of recommendations is nowadays recognized as fundamental in the RS research community.

8. Conclusion and Outlook

In this paper, we have presented RDF2Vec, an approach for learning latent numerical representations of entities in RDF graphs. In this approach, we first convert the RDF graphs in a set of sequences using two strategies, Weisfeiler-Lehman Subtree RDF Graph Kernels and graph walks, which are then used to build neural language models. The evaluation shows that such entity representations could be used in three different tasks. In each of those tasks, they were capable of outperforming standard feature generation approaches, i.e., approaches that turn (subsets of) RDF graphs into propositional models.

We have explored different variants of building embedding models. While there is no universally best performing approach, we can observe some trends. With respect to the first step of the transformation, i.e., the construction of sequences, kernel transformations lead to better results than (random) walks. However, they do not scale well to large-scale knowledge graphs, such as DBpedia or Wikidata. With respect to the second step, i.e., the actual computation of the embeddings, we have observed that Skip-Gram (SG) models in most cases outperform Continuous-Bag-of-Words (CBOW) models. The other characteristics of the models (e.g., the dimensionality of the embedding space) show less clear trends towards an optimal setting.

While constructing a vector space embedding for a large-scale knowledge graph, such as DBpedia or Wikidata, can be computationally expensive, we have shown that this step has to be taken only once, as the embeddings can be reused on various tasks. This is particularly interesting for such cross-domain knowledge

Table 15

Comparative results on `Movielens` dataset. The differences between "$\mathcal{H}$-DB2vec SG 500w 200v 4d" and the other methods are statistically significant according to the Wilcoxon signed rank with $p < 0.001$ (*) or $p < 0.05$ (**), with the only exception of `SLIM` in terms of recall. The scores for "$\mathcal{H}$-DB2vec SG 500w 200v 4d", averaged over three runs, are reported with the standard deviation.

Strategy	Precision	Recall	F1	nDCG
$\mathcal{H}$-DB2vec SG 500w 200v 4d	**0.2698** $\pm$ 0.0012	**0.1480** $\pm$ 0.0013	**0.1911** $\pm$ 0.0013	**0.4782** $\pm$ 0.0007
`SPRank`	0.1613*	0.0785*	0.1056	0.4056*
`SLIM`	0.2631*	0.1469	0.1885	0.4597*
`BPR-SSLIM`	0.2636*	0.1439**	0.1861	0.4674*

Table 16

Comparative results on `LibraryThing` dataset. The differences between "$\mathcal{H}$-DB2vec SG 500w 200v 8d" and the other methods are statistically significant according to the Wilcoxon signed rank with $p < 0.001$. The scores for "$\mathcal{H}$-DB2vec SG 500w 200v 8d", averaged over three runs, are reported with the standard deviation.

Strategy	Precision	Recall	F1	nDCG
$\mathcal{H}$-DB2vec SG 500w 200v 8d	**0.1061** $\pm$ 0.0016	**0.241** $\pm$ 0.0036	**0.1473** $\pm$0.0022	**0.3127** $\pm$ 0.005
`SPRank`	0.1019	0.2232	0.1399	0.3068
`SLIM`	0.0543	0.0988	0.07	0.2317
`BPR-SSLIM`	0.0962	0.2328	0.1361	0.3051

Table 17

Comparative results on `Last.fm` dataset. The differences between "$\mathcal{H}$-DB2vec SG 500w 200v 8d" and the other methods are statistically significant according to the Wilcoxon signed rank with $p < 0.001$. The scores for "$\mathcal{H}$-DB2vec SG 500w 200v 8d", averaged over three runs, are reported with the standard deviation.

Strategy	Precision	Recall	F1	nDCG
$\mathcal{H}$-DB2vec SG 500w 200v 8d	0.0607 $\pm$ 0.0021	**0.3408** $\pm$ 0.0143	0.103$\pm$0.003	**0.2266** $\pm$ 0.01
`SPRank`	**0.0841**	0.2043	**0.1191**	0.1918
`SLIM`	0.0723	0.1548	0.0985	0.1395
`BPR-SSLIM`	0.0465	0.2649	0.0791	0.1759

graphs, which can be used in a variety of scenarios and applications.

For the moment, we have defined some constraints for the construction of the embeddings. We do not use literal values, and we do not particularly distinguish between the schema and the data level of a graph. The former constraint has some limitations, e.g., when it comes to the tasks of determining entity similarity: for example, the similarity of two movies in terms of release date and budget or the similarity of two cities in terms of area and population is currently not captured by the models. Schema level and data level similarity are currently implicitly interwoven, but in particular for knowledge graphs with richer schemas (e.g., YAGO with its type hierarchy of several hundred thousand types), distinguishing embeddings of the schema and data level might become beneficial.

Apart from using vector space embeddings when exploiting LOD data sources, they may also become an interesting technique for improving those sources as such, for example knowledge base completion [38]. Among others, the proposed approach could also be used for link prediction, entity typing, or error detection in knowledge graphs [59], as shown in [43,48]. Similarly to the entity and document modeling, the approach can be extended for entity summarization, which is also an important task when consuming and visualizing large quantities of data [8].

Summarizing, we have shown that it is possible to adapt the technique of word embeddings to RDF graphs, and that those embeddings lead to compact vector representations of entities. We have shown that those vector representations help building approaches which outperform many state of the art tools on various tasks, e.g., data mining with background knowledge

or recommender systems. Furthermore, the proposed vector space representations are *universal* in the sense that they are not task specific, i.e., a vector space embedding for a general graph like DBpedia or Wikidata can be built once and reused for several tasks. As the embeddings used in this paper are publicly available, they are a versatile asset which can be exploited for various knowledge-intensive tasks in future research.

Acknowledgements The work presented in this paper has been partly funded by the German Research Foundation (DFG) under grant number PA 2373/1-1 (Mine@LOD), partially funded by the project PON03 PE_00136_1 Digital Services Ecosystem: DSE, and 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"). Jessica Rosati also acknowledges support of I.B.M. Ph.D. fellowship 2015-2016.

References

[1] Aggarwal, N., Buitelaar, P.: Wikipedia-based distributional semantics for entity relatedness. In: 2014 AAAI Fall Symposium Series (2014)

[2] Bellogín, A., Cantador, I., Castells, P.: A comparative study of heterogeneous item recommendations in social systems. Inf. Sci. 221, 142–169 (Feb 2013), http://dx.doi.org/10.1016/j.ins.2012.09.039

[3] Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. Journal of machine learning research 3(Feb), 1137–1155 (2003)

[4] Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – The Story So Far. International journal on semantic web and information systems 5(3), 1–22 (2009)

[5] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)

[6] Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems 30(1), 107 – 117 (1998)

[7] Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction 12(4), 331–370 (Nov 2002), http://dx.doi.org/10.1023/A:1021240730564

[8] Cheng, G., Tran, T., Qu, Y.: Relin: relatedness and informativeness-based centrality for entity summarization. In: The Semantic Web–ISWC, pp. 114–129 (2011)

[9] Cheng, W., Kasneci, G., Graepel, T., Stern, D., Herbrich, R.: Automated feature generation from structured knowledge. In: CIKM (2011)

[10] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research 12(Aug), 2493–2537 (2011)

[11] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE 41(6), 391–407 (1990)

[12] Di Noia, T., Ostuni, V.C., Tomeo, P., Di Sciascio, E.: Sprank: Semantic path-based ranking for top-n recommendations using linked open data. ACM Transactions on Intelligent Systems and Technology (TIST) (2016)

[13] Di Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D.: Exploiting the web of data in model-based recommender systems. In: Proceedings of the Sixth ACM Conference on Recommender Systems. pp. 253–256. RecSys '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2365952.2366007

[14] Di Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D., Zanker, M.: Linked open data to support content-based recommender systems. In: Proceedings of the 8th International Conference on Semantic Systems. pp. 1–8. I-SEMANTICS '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2362499.2362501

[15] Di Noia, T., Ostuni, V.C., Rosati, J., Tomeo, P., Di Sciascio, E., Mirizzi, R., Bartolini, C.: Building a relatedness graph from linked open data: A case study in the IT domain. Expert Syst. Appl. 44, 354–366 (2016)

[16] Fanizzi, N., d'Amato, C.: A declarative kernel for alc concept descriptions. In: Foundations of Intelligent Systems, pp. 322–331 (2006)

[17] Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: IJcAI. vol. 7, pp. 1606–1611 (2007)

[18] Gantner, Z., Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Mymedialite: A free recommender system library. In: Proceedings of the Fifth ACM Conference on Recommender Systems. pp. 305–308. RecSys '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/2043932.2043989

[19] de Gemmis, M., Lops, P., Musto, C., Fedelucio, N., Semeraro, G.: Semantics-aware content-based recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 119–159. Springer, 2nd edn. (2015)

[20] Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., Sharp, D.: E-commerce in your inbox: Product recommendations at scale. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1809–1818. KDD '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2783258.2788627

[21] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 855–864. ACM (2016)

[22] Gutmann, M.U., Hyvärinen, A.: Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. Journal of Machine Learning Research 13(Feb), 307–361 (2012)

[23] Harris, Z.S.: Mathematical Structures of Language. Wiley, New York, NY, USA (1968)

[24] Hassan, S., Mihalcea, R.: Semantic relatedness using salient semantic analysis. In: AAAI (2011)

[25] Hassanzadeh, O., Consens, M.: M.: Linked movie data base. In: In: Workshop on Linked Data on the Web (2009)

[26] Hoffart, J., Seufert, S., Nguyen, D.B., Theobald, M., Weikum, G.: Kore: keyphrase overlap relatedness for entity disambiguation. In: Proceedings of the 21st ACM international conference on Information and knowledge management. pp. 545–554. ACM (2012)

[27] Huang, L., Milne, D., Frank, E., Witten, I.H.: Learning a concept-based document similarity measure. Journal of the American Society for Information Science and Technology 63(8), 1593–1608 (2012)

[28] Huang, Y., Tresp, V., Nickel, M., Kriegel, H.P.: A scalable approach for statistical learning in semantic graphs. Semantic Web (2014)

[29] Hulpuş, I., Prangnawarat, N., Hayes, C.: Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In: International Semantic Web Conference. pp. 442–457. Springer (2015)

[30] Kappara, V.N.P., Ichise, R., Vyas, O.: Liddm: A data mining system for linked data. In: LDOW (2011)

[31] Khan, M.A., Grimnes, G.A., Dengel, A.: Two pre-processing operators for improved learning from semanticweb data. In: RCOMM (2010)

[32] Koren, Y.: Factorization meets the neighborhood: A multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 426–434. KDD '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1401890.1401944

[33] Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (Aug 2009), http://dx.doi.org/10.1109/MC.2009.263

[34] Kramer, S., Lavrač, N., Flach, P.: Propositionalization approaches to relational data mining. In: Relational Data Mining, pp. 262–291. Springer Berlin Heidelberg (2001)

[35] Lee, M., Pincombe, B., Welsh, M.: An empirical evaluation of models of text document similarity. Cognitive Science Society (2005)

[36] Lee, S., Park, S., Kahng, M., Lee, S.g.: Pathrank: A novel node ranking measure on a heterogeneous graph for recommender systems. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management. pp. 1637–1641. CIKM '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2396761.2398488

[37] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web Journal (2013)

[38] Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI. pp. 2181–2187 (2015)

[39] Lösch, U., Bloehdorn, S., Rettinger, A.: Graph kernels for rdf data. In: The Semantic Web: Research and Applications, pp. 134–148. Springer (2012)

[40] Middleton, S.E., Roure, D.D., Shadbolt, N.R.: Ontology-Based Recommender Systems, pp. 779–796. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-540-92673-3_35

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

[42] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)

[43] Minervini, P., Fanizzi, N., d'Amato, C., Esposito, F.: Scalable learning of entity and predicate embeddings for knowledge graph completion. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). pp. 162–167. IEEE (2015)

[44] Musto, C., Semeraro, G., De Gemmis, M., Lops, P.: Word embedding techniques for content-based recommender systems: an empirical evaluation. In: RecSys Posters, ser. CEUR Workshop Proceedings, P. Castells, Ed. vol. 1441

[45] Musto, C., Semeraro, G., Lops, P., de Gemmis, M.: Random Indexing and Negative User Preferences for Enhancing Content-Based Recommender Systems, pp. 270–281. Springer Berlin Heidelberg, Berlin, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-23014-1_23

[46] Musto, C., Semeraro, G., Lops, P., de Gemmis, M.: Contextual eVSM: A Content-Based Context-Aware Recommendation Framework Based on Distributional Semantics, pp. 125–136. Springer Berlin Heidelberg, Berlin, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-39878-0_12

[47] Mynarz, J., Svátek, V.: 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 (2013)

[48] Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. arXiv preprint arXiv:1503.00759 (2015)

[49] Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE 104(1), 11–33 (2016)

[50] Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Proceedings of the 28th international conference on machine learning (ICML-11). pp. 809–816 (2011)

[51] Ning, X., Karypis, G.: SLIM: sparse linear methods for top-n recommender systems. In: 11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011. pp. 497–506 (2011), http://dx.doi.org/10.1109/ICDM.2011.134

[52] Ning, X., Karypis, G.: Sparse linear methods with side information for top-n recommendations. In: Proceedings of the Sixth ACM Conference on Recommender Systems. pp. 155–162. RecSys '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2365952.2365983

[53] Nunes, B.P., Kawase, R., Fetahu, B., Dietze, S., Casanova, M.A., Maynard, D.: Interlinking documents based on semantic graphs. Procedia Computer Science 22, 231–240 (2013)

[54] Ostuni, V.C., Noia, T.D., Sciascio, E.D., Mirizzi, R.: Top-n recommendations from implicit feedback leveraging linked open data. In: ACM RecSys '13. pp. 85–92 (2013)

[55] Ostuni, V.C., Oramas, S., Di Noia, T., Serra, X., Di Sciascio, E.: Sound and music recommendation with knowledge graphs. ACM Transactions on Intelligent Systems and Technology (TIST) (2016), to appear

[56] Ozsoy, M.G.: From word embeddings to item recommendation. arXiv preprint arXiv:1601.01356 (2016)

[57] Paul, C., Rettinger, A., Mogadala, A., Knoblock, C.A., Szekely, P.: Efficient graph-based document similarity. In: International Semantic Web Conference. pp. 334–349. Springer (2016)
[58] Paulheim, H.: Exploiting linked open data as background knowledge in data mining. In: Workshop on Data Mining on Linked Open Data (2013)
[59] Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web (Preprint), 1–20 (2016)
[60] Paulheim, H., Fümkranz, J.: Unsupervised generation of data mining features from linked open data. In: Proceedings of the 2nd international conference on web intelligence, mining and semantics. p. 31. ACM (2012)
[61] Paulheim, H., Ristoski, P., Mitichkin, E., Bizer, C.: Data mining with background knowledge from the web. RapidMiner World (2014)
[62] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710. ACM (2014)
[63] Rendle, S.: Factorization machines with libfm. ACM Trans. Intell. Syst. Technol. 3(3), 57:1–57:22 (May 2012), http://doi.acm.org/10.1145/2168752.2168771
[64] Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. pp. 452–461. UAI '09, Arlington, Virginia, United States (2009), http://dl.acm.org/citation.cfm?id=1795114.1795167
[65] Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.: Recommender Systems Handbook. Springer-Verlag New York, Inc., New York, NY, USA, 2nd edn. (2015)
[66] Ristoski, P., Bizer, C., Paulheim, H.: Mining the web of linked data with rapidminer. Web Semantics: Science, Services and Agents on the World Wide Web 35, 142–151 (2015)
[67] Ristoski, P., Mencía, E.L., Paulheim, H.: A hybrid multi-strategy recommender system using linked open data. In: Semantic Web Evaluation Challenge, pp. 150–156. Springer (2014)
[68] Ristoski, P., Paulheim, H.: A comparison of propositionalization strategies for creating features from linked open data. In: Linked Data for Knowledge Discovery (2014)
[69] Ristoski, P., Paulheim, H.: Rdf2vec: Rdf graph embeddings for data mining. In: International Semantic Web Conference (To Appear). Springer (2016)
[70] Ristoski, P., Paulheim, H.: Semantic web in data mining and knowledge discovery: A comprehensive survey. Web Semantics: Science, Services and Agents on the World Wide Web (2016)
[71] Ristoski, P., de Vries, G.K.D., Paulheim, H.: A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In: International Semantic Web Conference (To Appear). Springer (2016)
[72] Rosati, J., Ristoski, P., Noia, T.D., Leone, R.D., Paulheim, H.: Rdf graph embeddings for content-based recommender systems. In: Proceedings of the 3rd Workshop on New Trends in Content-based Recommender Systems (CBRecSys 2016) (September 2016)
[73] Sahlgren, M.: An introduction to random indexing. In: In Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005 (2005)
[74] Salakhutdinov, R., Mnih, A.: Bayesian probabilistic matrix factorization using markov chain monte carlo. In: Proceedings of the 25th International Conference on Machine Learning. pp. 880–887. ICML '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1390156.1390267
[75] Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: The Semantic Web–ISWC (2014)
[76] Schuhmacher, M., Ponzetto, S.P.: Knowledge-based graph document modeling. In: Proceedings of the 7th ACM international conference on Web search and data mining. pp. 543–552. ACM (2014)
[77] Semeraro, G., Lops, P., Basile, P., de Gemmis, M.: Knowledge infusion into content-based recommender systems. In: Proceedings of the Third ACM Conference on Recommender Systems. pp. 301–304. RecSys '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1639714.1639773
[78] Shervashidze, N., Schweitzer, P., Van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. The Journal of Machine Learning Research 12, 2539–2561 (2011)
[79] Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in neural information processing systems. pp. 926–934 (2013)
[80] Srebro, N., Rennie, J.D.M., Jaakola, T.S.: Maximum-margin matrix factorization. In: Advances in Neural Information Processing Systems 17. pp. 1329–1336. MIT Press (2005)
[81] Steck, H.: Evaluation of recommendations: Rating-prediction and ranking. In: Proceedings of the 7th ACM Conference on Recommender Systems. pp. 213–220. RecSys '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2507157.2507160
[82] Thiagarajan, R., Manjunath, G., Stumptner, M.: Computing semantic similarity using ontologies. HP Laboratories). Technical report HPL-2008-87 (2008)
[83] Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th annual meeting of the association for computational linguistics. pp. 384–394. Association for Computational Linguistics (2010)
[84] Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM 57(10), 78–85 (2014)
[85] de Vries, G.K.D.: A fast approximation of the Weisfeiler-Lehman graph kernel for RDF data. In: ECML/PKDD (1) (2013)
[86] de Vries, G.K.D., de Rooij, S.: A fast and simple graph kernel for rdf. In: DMLOD (2013)
[87] de Vries, G.K.D., de Rooij, S.: Substructure counting graph kernels for machine learning from rdf data. Web Semantics: Science, Services and Agents on the World Wide Web 35, 71–84 (2015)
[88] Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI. pp. 1112–1119. Citeseer (2014)
[89] Weimer, M., Karatzoglou, A., Smola, A.: Improving max-

imum margin matrix factorization. Mach. Learn. 72(3), 263–276 (Sep 2008), http://dx.doi.org/10.1007/s10994-008-5073-7

[90] Widdows, D.: Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1. pp. 136–143. ACL '03, Association for Computational Linguistics, Stroudsburg, PA, USA (2003), http://dx.doi.org/10.3115/1075096.1075114

[91] Yanardag, P., Vishwanathan, S.: Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1365–1374. ACM (2015)

[92] Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., Khandelwal, U., Norick, B., Han, J.: Personalized entity recommendation: A heterogeneous information network approach. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining. pp. 283–292. WSDM '14, ACM, New York, NY, USA (2014)