RDF Data Clustering based on Resource and Predicate Embeddings

Siham Eddamiri[1], El Moukhtar Zemmouri[1] and Asmaa Benghabrit[2]
[1]*LM2I Laboratory, Moulay Ismail University, ENSAM, Meknes, Morocco*
[2]*LMAID Laboratory, Mohammed V University, ENSMR, Rabat, Morocco*

Keywords: Machine Learning, Linked Data, RDF, Clustering, Word2vec, Doc2vec, K-means.

Abstract: With the increasing amount of Linked Data on the Web in the past decade, there is a growing desire for machine learning community to bring this type of data into the fold. However, while Linked Data and Machine Learning have seen an explosive growth in popularity, relatively little attention has been paid in the literature to the possible union of both Linked Data and Machine Learning. The best way to collaborate these two fields is to focus on RDF data. After a thorough overview of Machine learning pipeline on RDF data, the paper presents an unsupervised feature extraction technique named Walks and two language modeling approaches, namely Word2vec and Doc2vec. In order to adapt the RDF graph to the clustering mechanism, we first applied the Walks technique on several sequences of entities by combining it with the Word2Vec approach. However, the application of the Doc2vec approach to a set of walks gives better results on two different datasets.

1 INTRODUCTION

In recent years, the Web evolved from a web of documents to a web of data (Bizer et al., 2009). In fact, the Web emerges from a global information space of interlinked documents to one where both documents and data are linked. Allowing to provide a solid foundation for this evolution to be best practices for publishing and interlinking structured data on the Web known as Linked Data principles (Bizer, 2011).

Meanwhile, the combination of Linked Data and the field of Machine Learning hasnt paid much, while both fields have seen an exponential growth in popularity in the past decade. Therefore, the best solution is to focus on the Resource Description Framework (RDF) which is in the lowest layers in the Semantic Web stack. Thus, to understand RDF, the ML researcher do not need to know about ontologies, reasoning and SPARQL to have the benefit.

As a matter of fact, the RDF data-model (Bloem et al., 2014), is not suited for traditional machine learning algorithms. In fact, traditional machine learning techniques requires a tabular as input and not on large interconnected graphs such as RDF graphs. Therefore, expressing such data in RDF will add many relations and concepts on top of its native structure, which will add much inferencing, harmonization, and accessibility, and also a new impediments and challenges.

Moreover, in the semantic web, a dataset doesnt separate any longer by instances, or emerge from a single learning task, even the standard methods of evaluation dont adapt well. Therefore, a new pipeline for RDF must be adopted in order to provide a comprehensive framework and to fit the common steps for traditional machine learning techniques. While the transformation to RDF is reversible, reconstructing the original data requires manual effort or domain-specific methods in order to suit the pipeline. Hence, the preprocessing of RDF data to a similar form is required to process large amounts of RDF data by generic methods.

In this paper, we introduce an overview for machine learning on RDF data, and a set of common techniques for data pre-processing, in order to answer most of common tasks in machine learning. In this regard, we first convert the RDF graph into a set of sequences using two approaches for generating graph walks. Then, we adopt two language modelling approaches for latent representation of entities in RDF graphs to train the sequence of entities, finally we use K-means to group those entities vectors.

The rest of this paper is structured as: in section 2, we give some necessary preliminaries followed by related works in Section 3. Section 4 describes our approaches and then we present in section 5 an evaluation of our propositions. Finally, we conclude with a summary.

Eddamiri, S., Zemmouri, E. and Benghabrit, A.
RDF Data Clustering based on Resource and Predicate Embeddings.
DOI: 10.5220/0007228903670373
In *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018) - Volume 1: KDIR*, pages 367-373
ISBN: 978-989-758-330-8

2 PRELIMINARIES

The Resource Description Framework (RDF) is a language for representing metadata about Web resources (Manola and Miller, 2004). It has been introduced and recommended by the World Wide Web Consortium (W3C) as a fundamental building block of Linked Data and the Semantic Web. The central idea of RDF is to enable the encoding, exchange and reuse of structured metadata.

The atomic construct of RDF data model are statements about resources in the form (subject, predicate, object) called triples. A triple (s,p,o) in a set of triples $\mathcal{T}$ is composed by a subject s that is a resource identified by URI, a property/predicate p also identified by URI and an object o that is the value of the property. An object can be either another resource or a literal. An RDF resource can also be a blank node (Manola and Miller, 2004).

This simple model of assertions enables RDF to represent a set of triples as a directed edge-labelled graph $G = (V,E)$ where s and o are nodes in V, and $(s,o) \in E$ is an edge oriented from node s to node o and labeled with predicate p.

In this paper, we adopt the RDF formalization introduced by (Tran et al., 2009) given as follow:

Definition 1 (RDF Data-Graph). An RDF data-graph G representing a set of triples $\mathcal{T}$ is defined as a tuple $G = (V,E,L)$ where:

- $V = V_E \cup V_l \cup V_b$ define a finite set of vertices as the union of V_E the set of resource vertices, V_l the set of literal vertices and V_b the set of blank node vertices.
  $V = \{v \mid \exists\, x,y\ (v,x,y) \in \mathcal{T} \vee (x,y,v) \in \mathcal{T}\}$
- E is a finite set of directed edge $e(v_1,v_2)$ that connect the subject v_1 and the object v_2:
  $E = \{(v_1,v_2) | \exists x (v_1,x,v_2) \in \mathcal{T} \vee (v_2,x,v_1) \in \mathcal{T}\}$
- $L = L_V \cup L_E$ is a finite set of labels as the union of L_V a set of vertex labels and L_E a set of edge labels:
  - If $v \in V_l$ then $L(v)$ is a literal value.
  - If $v \in V_E$ then $L(v)$ is the resource corresponding URI.
  - If $v \in V_b$ then L(v) is set to NULL.
  - If $e \in E$ then $L(e)$ is the property corresponding URI.

  $L = l(v) = v | v \in V \cup l(e) = e$

This formalization fails when predicate terms are also used in the subject or object position. Therefore, we must distinct between the position of the predicate and the position of the subject and object, even if they have the same label.

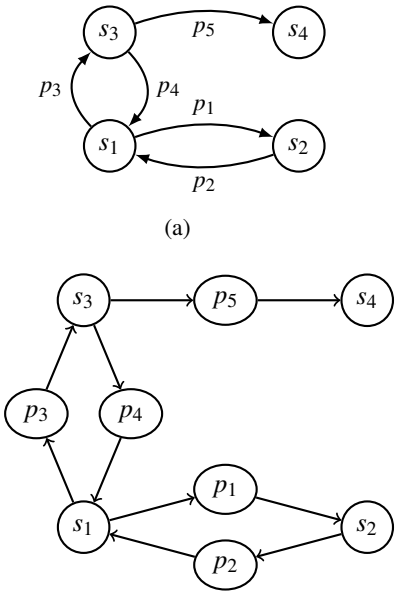

Figure 1: (a) Example of RDF graph that is edge-labeled multigraph; (b) The obtained RDF bipartite graph.

As a matter of fact, the main challenge of RDF data which defined as an example of labeled multigraph (edge-labeled multigraph) is that there are many edges between a pair of vertices. Therefore, we transform an edge-labeled multigraph RDF to an RDF bipartite graph (Figure 1) that incorporate statements and properties as nodes into the graph, in order to deliver a richer sense of connectivity than the standard directed labeled graph representations. To do so, we opt to reify the edges by adopting the formalization introduced by (de Vries et al.,2013), describing an RDF bipartite graph as:

So, there is a vertex for each unique subject and object. And for each triple we create two edges, one for connecting subject to triple and other for connecting triple to object. This definition has the additional advantage that occur in applying any algorithms that use labeled simple graphs, regardless of whether or not they are constructed from a triple store.

Definition 2 (RDF Bipartite Graph). An RDF bipartite data graph G is defined as a tuple (V,E,l) where:

- $V = V_E \cup V_l \cup V_b \cup V_T$ a finite set of vertices as the union of V_E the set of resource vertices, V_l the set of literal vertices, V_b the set of blank nodes vertices, and V_T the set of triple vertices.
  $V = \{v \mid \exists\, x,y\ (v,x,y) \in \mathcal{T} \vee (x,y,v) \in \mathcal{T}\} \cup \mathcal{T}$
- E is a finite set of directed edges that connect a subject v_1 to a triple and a triple to object v_2:

$E = \{(v_1, v_2) | \exists x (v_1, x, v_2) \in \mathcal{T} \vee (v_2, x, v_1) \in \mathcal{T}\}$

- $L = L_V \cup L_E$ is a finite set of labels as the union of L_V a set of vertex labels and L_E a set of edge labels:
  - If $v \in V_l$ then $L(v)$ is a literal value.
  - If $v \in V_E$ then $L(v)$ is the resource corresponding URI.
  - If $v \in V_b$ then L(v) is set to NULL.
  - If $e \in E$ then $L(e)$ is the property corresponding URI.

  $L = l(v) = v | v \in V \cup l(e) = e$

So, there is a vertex for each unique subject and object. And for each triple we create two edges, one for connecting subject to triple and other for connecting triple to object. This definition has the additional advantage that occur in applying any algorithms that use labeled simple graphs, regardless of whether or not they are constructed from a triple store.

3 RELATED WORK

The combination of Machine Learning and Sematic Web focuses on making an RDF graph as input to an existing ML algorithm, by creating a generic pipeline which will help to have some standardized methods. We have drawn an example pipeline (Figure 2), to summon up from RDF to an ML model by the following steps:

Pre-processing consists on extracting the most efficient and more relevant information from RDF data, by dealing with three common problems: (1) Cleaning data based on detecting and correcting or even removing a vertex/property/triples from an RDF data; (2) Hub removal based on the principle that node which have many links to other nodes add a little information about an instance, hence we could relabel the hub vertex by the combination of edges and vertex labels (Bloem et al., 2014),on one hand, or removing all the frequent pairs from the graph using a parameter k and replacing the pair that have the lowest frequency by the concatenation of the labels (de Vries, 2013) on the other hand; (3) Relabeling based on giving a new label to a vertex (blank nodes or a hub vertex).

Instance Extraction consists on extracting sets of subgraphs that are relevant to a given resources (vertices) from an RDF graph. For this, there are several approaches which we can categorize into three approaches: (1) Immediate Properties/Nave Extraction is a direct approach that consider the immediate properties of resources (Colucci et al., 2014; Ristoski and Paulheim, 2016);(2) Concise Bounded Description (CBD) is the improvement over the immediate properties which consider the types of nodes into account in the graph;(3) Depth Limited Crawling (DLC) is limited the sub-graph by depth and simply traverse the graph of certain number of steps from the starting node, which could be traversing forward or/and backwards edges from the root node in the graph such as (Lösch et al., 2012; de Vries, 2013; Bach, 2008; Zhao and Karypis, 2002).

Feature Extraction consists on transforming RDF graph to features which reduce irrelevant and redundant variables, while describing the data with sufficient accuracy. The current state of the art for this step given by Weisfeiler-Lehman (WL) graph kernel (de Vries and de Rooij, 2015; Bach, 2008) aimed at improving the computation time of the kernel while applied RDF, which can be seen as a graph or as a learner. The WL algorithm was adapted to compute the kernel directly on the underlying graph, while maintaining a subgraph perspective for each instance as (Lösch et al., 2012; de Vries, 2013).

Learning consists on feeding the feature vectors or the subgraphs to a learner which we can divide to two types : (1) Supervised Learning based on training a data sample from labeled data source (Lösch et al., 2012); (2) Unsupervised learning is a Self-Organizing neural network learn which based on identifying a hidden pattern in unlabelled input data (Grimnes et al., 2008; Colucci et al., 2014; Ristoski and Paulheim, 2016).

4 THE METHODOLOGY

4.1 Overview

As a matter of fact, in case of RDF graphs, we consider entities and relations between them instead of word sequences. Therefore, we convert the RDF graph data into sequences of entities (vertices + edges) in order to apply such approaches. After that, we train these sequences to the neural language models to generate vectors of numerical values in a latent feature space, then we cluster them by using K-means algorithm in order to obtain groups. To do so, in this paper we use two approaches for extracting instances and two approaches for generating embedding vectors from an RDF graphs.

4.2 Feature Vector using Word2vec

In this approach depicted in figure 3, for a given RDF graph $G = (V, E, l)$ we extract first the vertices (instances) that we would cluster, then for each vertex v we generate all walks of depth d starting with v by

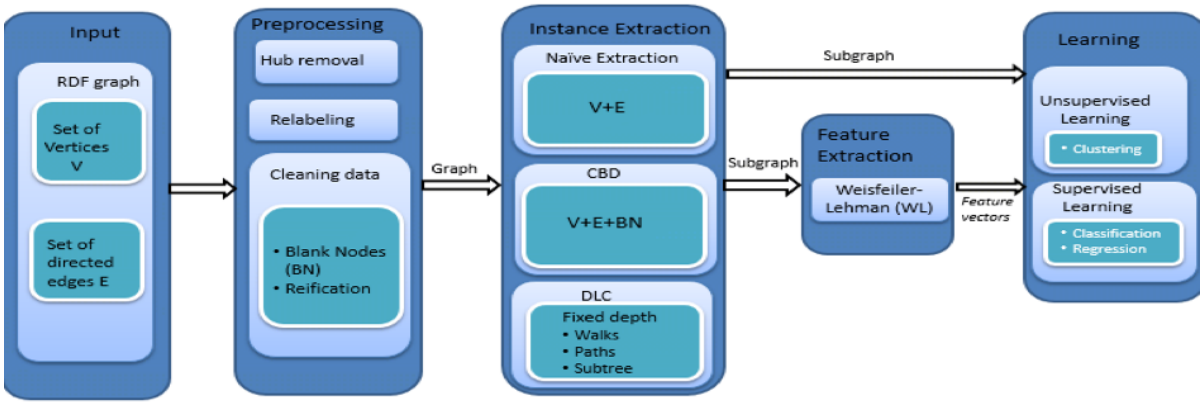

Figure 2: An overview of a Machine Learning pipeline for RDF.

the *BFS* (Breath-First Search) algorithm. The set of sequences for graph G is the union of all the walks generated by exploring the direct outgoing edges of the root vertex v_r, while for each explored edge r_{1i} we extract the neighbourhood vertex, so for the depth $d = 1$ the sequences has the following structure v_r, r_{1i}, v_{1i} and vice versa for the neighborhood vertex until the depth d.

Definition 3: A Walks is defined as a sequence of vertices and edges with a specific depth d as the following formalization :

$$Walks = \{(v_{i-1}, e_i, v_i) \in \mathcal{T} | 0 \leq i \leq d\}$$

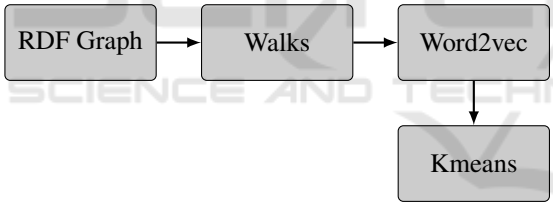

Figure 3: The overall process to map the walks toward word2vec.

For the training, we use word2vec (Mikolov et al., 2013), to construct a vocabulary from the training file and then learns high dimensional vector representations of words. according to the parameter in Table 1.

4.3 Feature Vector using Doc2vec

In this approach (Fig. 4), we use the same technique as the first approach to generate the walks, rooted with a specific vertex v_r by a depth d. however, the difference here is to gather all the walks that start with the same vertex into a file, hence we will have for each vertex a file that contain all the walks, a set of walks, as defined at the beginning of 4.2, with a depth d rooted with this vertex. Then we use doc2vec (Le and Mikolov, 2014), which is inspired by word2vec

Table 1: Training parameters using Word2vec.

Parameters	Explanations	Default value
Train	Name of input file	Train.txt
Output	Name of output file	Vectors.bin
SG	Choice of training model 0: Skip-gram model 1: CBOW model	0
Size	Dimension of vectors	200/300
Negative	Number of negative samples	10
Windows	Distance between the current and predicted word	4

for larger blocks of texts (documents) to learn embeddings and then learns high dimensional vector representations of documents according to the parameter in Table 2.

Definition 4 : A set of Walks is defined as the union of walks with a specific vertex v_r by a depth d as the following formalization :

$$\sum Walks(v_r) = \cup\{(v_r, e_{r+1}, v_{r+1}) \in \mathcal{T} | 0 \leq i \leq d\}$$

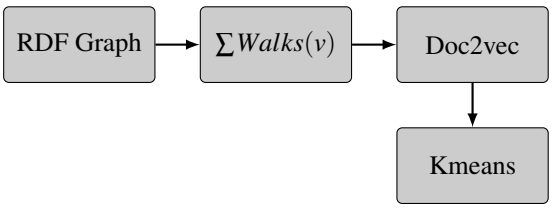

Figure 4: The overall process to map the walks toward word2vec.

5 RESULTS AND DISCUSSION

Evaluate both approaches on two datasets and two different feature extraction strategies (Walk and the set of walks), combined with two different learning algorithms (Word2vec and Doc2vec).

Table 2: Training parameters using Doc2vec.

Parameters	Explanations	Default value
Train	Name of input file	Train.txt
Output	Name of output file	Vectors.bin
DM	Choice of training model 0: distributed bag of words model 1: distributed memory model	0
Size	Dimension of vectors	200/300
Negative	Number of negative samples	10
Windows	Distance between the current and predicted word	4

5.1 Dataset

In order to evaluate our approaches, we use two types of RDF Graphs, derived from existing RDF datasets, as showed in Table 3. The value of a certain property is used as a classification target. However, first we specifiedd the number of clusters generated for both datasets based on the preexisting classification and calculated purity, Fmeasure.

The AIFB dataset represented in (Bloehdorn and Sure, 2007) describes the AIFB research institute in terms of its staff, research groups, and publications. In total this dataset contains 178 persons that belong to 1 of 5 research groups. However, one of these groups has only 4 embers, which we ignore from the dataset, leaving 4 groups. The goal of the experiment is to training and grouping the 174 persons into 4 groups

The BGS dataset represented in (de Vries, 2013) was created by the British Geological (BGS) Survey and describes geological measurements in Great Britain in the form of Named Rock Units. For these named rock units, we have two largest classes for lithogenesis property, each class have 93 and 53 instances. The aim of our experiment is to training and grouping these 163 instances into 2 groups.

Table 3: The parameters of training command using Word2vec.

Dataset	Instances	Clusters	Source
AIFB	176	4	AIFB
BGS	146	2	BGS

5.2 Performance Measures

The effectiveness of the clustering step can be evaluated by two measures. They allow evaluating the reliability of the clustering mechanism by comparing the obtained clusters to known classes. The first measure is the F-measure, it is based on the fundamental informational retrieval parameters, precision an recall which are defined as :

$$P(i,j) = \frac{n_{ij}}{n_j} \tag{1}$$

$$R(i,j) = \frac{n_{ij}}{n_i} \tag{2}$$

Where n_i is the number of the vertices of the class i, n_j is the number of the vertices of the cluster j and n_{ij} is the number of the class i in the cluster j. The F-measure of a cluster and a class i is given by :

$$F(i,j) = \frac{2P(i,j) * R(i,j)}{P(i,j) + R(i,j)} \tag{3}$$

For the entire clustering result, the F-measure is computed as following:

$$F = \sum_i \frac{n_i}{n} \max_j (F(i,j)) \tag{4}$$

Where n is the total number of vertices. A better clustering result is measured by the largest F-measure (Steinbach et al., 2000). The second measure is the purity of a cluster represents the ratio of the dominant class in the cluster to the size of the cluster. Thus, the purity of the cluster is defined as:

$$purity(j) = \frac{1}{n_j} \max_i (n_{ij}) \tag{5}$$

The global value for the purity is the weighted average of all purity values. It is given by the following formula:

$$purity = \sum_j \frac{n_j}{n} purity(j) \tag{6}$$

A better clustering result is measured by the largest purity value (Zhao et al.,2004).

5.3 Results

The results for the task of clustering on the two RDF datasets are given in Tables 4 and 5. From the results we can observe that the combination of set of walks with doc2vec approach outperforms all the other approach. More precisely, using the skip-gram feature vectors of size 300 provides the best results on the two datasets. The combination of the walks with word2vec approach on all two datasets performs closely to the standard graph substructure feature generation strategies, but it does not outperform them. Doc2vec approach generate vectors description of an instance based on all the walks related to this instance while, word2vec approach consider each walk separately. Hence, we observe that the skip-gram perform

Table 4: The results of similar features clustering based on word2vec and doc2vec in terms of Purity.

	AIFB Dataset				BGS Dataset			
	Word2Vec		Doc2Vec		Word2Vec		Doc2Vec	
	SG	CBOW	PV-DBOW	PV-DM	SG	CBOW	PV-DBOW	PV-DM
$Size = 200$								
$d = 2$	46.33	46.32	55.71	63.03	67.86	66.67	85.62	78.77
$d = 3$	49.15	47.46	62.50	73.58	60.36	57.14	76.71	72.60
$d = 4$	64.15	48.59	55.71	68.09	76.19	82.53	69.86	71.23
$Size = 300$								
$d = 2$	47.46	48.02	**79.67**	72.77	60.00	66.67	**86.99**	78.08
$d = 3$	79.10	49.15	63.38	68.06	61.60	63.91	67.12	71.92
$d = 4$	47.46	48.02	64.79	73.50	78.12	61.90	77.40	71.92

Table 5: The results of similar features clustering based on word2vec and doc2vec in terms of F-measure.

	AIFB Dataset				BGS Dataset			
	Word2Vec		Doc2Vec		Word2Vec		Doc2Vec	
	SG	CBOW	PV-DBOW	PV-DM	SG	CBOW	PV-DBOW	PV-DM
$Size = 200$								
$d = 2$	44.51	45.01	50.04	57.01	62.72	62.21	62.92	59.23
$d = 3$	44.21	45.72	58.40	73.70	60.08	63.55	71.11	59.28
$d = 4$	58.81	46.51	49.61	64.11	76.19	82.36	59.45	68.77
$Size = 300$								
$d = 2$	44.52	45.84	56.51	64.73	60.00	59.48	63.49	58.86
$d = 3$	56.91	47.01	59.50	67.84	61.19	59.48	56.85	66.90
$d = 4$	44.13	46.51	59.80	63.70	69.10	57.45	71.90	69.54

better than the CBOW in the approach word2vec with size 300 and the PV-DBOW perform better than PV-DM in Doc2vec in term of purity.

6 CONCLUSIONS

In this paper, we have presented an overview of a Machine Learning pipeline on RDF data, and a set of common techniques for data preprocessing, in order to get solve the most common tasks in machine learning from linked data by requiring steps such as : Preprocessing, instance extraction, feature extraction and learning. Then we have identified two mean challenges for performing clustering of Semantic Web data, which are instances extraction instance from a large RDF graph. Moreover, we have evaluated two different approaches on AFIB and BGS datasets. After comparing these approaches, we opted for the set of walks approach to generate an instance extraction from RDF graph. Finally, we believe that we have shown that clustering of RDF resources is an interesting area, where many questions remain unanswered. Our middle-term goal is to choose the optimal instance extraction methods, distance metrics and clustering algorithms to perform our RDF clustering.

REFERENCES

Bach, F. R. (2008). Graph kernels between point clouds. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 25–32, New York, NY, USA. ACM.

Bizer, C. (2011). Evolving the web into a global data space. In *Proceedings of the 28th British National Conference on Advances in Databases*, BNCOD'11, pages 1–1, Berlin, Heidelberg. Springer-Verlag.

Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data - the story so far. *International journal on semantic web and information systems*, 5(3):1–22.

Bloehdorn, S. and Sure, Y. (2007). Kernel methods for mining instance data in ontologies. In Aberer, K., Choi,

K.-S., Noy, N. F., Allemang, D., Lee, K.-I., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudr-Mauroux, P., editors, *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 58–71. Springer.

Bloem, P., Wibisono, A., and de Vries, G. (2014). Simplifying RDF data for graph-based machine learning. In *KNOW@LOD*, volume 1243 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Colucci, S., Giannini, S., Donini, F. M., and Sciascio, E. D. (2014). A deductive approach to the identification and description of clusters in linked open data. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, ECAI'14, pages 987–988, Amsterdam, The Netherlands, The Netherlands. IOS Press.

de Vries, G. K. (2013). A fast approximation of the weisfeiler-lehman graph kernel for rdf data. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Volume 8188*, ECML PKDD 2013, pages 606–621, New York, NY, USA. Springer-Verlag New York, Inc.

de Vries, G. K. D. and de Rooij, S. (2015). Substructure counting graph kernels for machine learning from rdf data. *Web Semant.*, 35(P2):71–84.

Grimnes, G. A., Edwards, P., and Preece, A. (2008). Instance based clustering of semantic web resources. In Bechhofer, S., Hauswirth, M., Hoffmann, J., and Koubarakis, M., editors, *The Semantic Web: Research and Applications*, pages 303–317, Berlin, Heidelberg. Springer Berlin Heidelberg.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

Lösch, U., Bloehdorn, S., and Rettinger, A. (2012). Graph kernels for rdf data. In Simperl, E., Cimiano, P., Polleres, A., Corcho, O., and Presutti, V., editors, *The Semantic Web: Research and Applications*, pages 134–148, Berlin, Heidelberg. Springer Berlin Heidelberg.

Manola, F. and Miller, E. (2004). RDF primer. World Wide Web Consortium, Recommendation REC-rdf-primer-20040210.

Mikolov, T., Yih, S. W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics.

Ristoski, P. and Paulheim, H. (2016). Rdf2vec: Rdf graph embeddings for data mining. In Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., and Gil, Y., editors, *The Semantic Web – ISWC 2016*, pages 498–514, Cham. Springer International Publishing.

Steinbach, M., Karypis, G., and Kumar, V. (2000). A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*.

Tran, T., Wang, H., and Haase, P. (2009). Hermes: Data web search on a pay-as-you-go integration infrastructure. *Web Semant.*, 7(3):189–203.

Zhao, Y. and Karypis, G. (2002). Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 515–524, New York, NY, USA. ACM.