Simplifying RDF Data for Graph-Based Machine Learning

Peter Bloem[1], Adianto Wibisono[12], and Gerben K.D. de Vries[1]

[1] System and Network Engineering Group
Informatics Institute, University of Amsterdam
the Netherlands
`uva@peterbloem.nl, g.k.d.devries@uva.nl`
[2] Knowledge Representation and Reasoning Group
Vrije Universiteit Amsterdam
the Netherlands
`a.wibisono@vu.nl`

Abstract. From the perspective of machine learning and data mining applications, expressing data in RDF rather than a domain-specific format can add complexity and obfuscate the internal structure. We investigate and illustrate this issue with an example where bio-molecular graph datasets are expressed in RDF. We use this example to inspire preprocessing techniques which reverse some of the complications of adding semantic annotations, exposing those patterns in the data that are most relevant to machine learning. We test these methods in a number of classification experiments and show that they can improve performance both for our example datasets and real-world RDF datasets.

1 Introduction

Linked Data is fast establishing itself as the principal method for reliable, accessible, long-term storage of data [1]. It contrasts starkly with the datasets traditionally used and exchanged in machine learning research, which are commonly distributed as application-specific files (such as MATLAB formatted data), CSV files, or relational databases. The quality of this data varies greatly and in many cases, the original semantic context required to interpret it is missing.

We can combat these problems with Linked Data and RDF. However, for machine learning applications, this also poses new challenges. Commonly, machine learning data is separated into instances and expressed in a canonical form like feature vectors, trees, sequences or small graphs. Expressing such data in RDF will add many relations and concepts on top of its native structure. If this raw form is well suited to machine learning, we may expect that the added information, while helpful for inference, harmonization and accessibility, can be detrimental to machine learning.

While the transformation to RDF is reversible, reconstructing the original data will usually require manual effort or domain-specific methods. To process large amounts of RDF data by generic methods, we require automatic pre-processing that simplifies the raw RDF data to something similar to its native form.

In this paper we introduce a number of such methods and show that these methods can (at least partly) deal with the problems introduced in our illustrative example, which uses typical graph datasets from machine learning, converted to RDF. Furthermore, we test our techniques on a number of real-world RDF machine learning classification datasets and show that simplification can, for the right parameters, improve performance on such datasets as well.

For an in-depth overview of machine learning in the context of Linked Data, we refer the reader to [2]. For examples of machine learning in a Semantic Web context, see [3,4]. For this paper we take the approach of using graph kernels for learning from RDF data [5,6,7] as a graph-learning workflow to test our methods.

2 An illustrative example

To illustrate the tension that exists between principles of well-authored RDF data and data that facilitates machine learning, we adapt the MUTAG and ENZYMES datasets, as used in [8,9] and translate these into an RDF representation.

These datasets contain the molecular form of enzymes and other molecules, each encoded straightforwardly as a graph with a node per atom or tertiary structure (colored by type), and an unlabeled link for each bond. To represent this in RDF we create a blank node for each atom or tertiary structure. We create a node for each label and connect the blank nodes to these by an `a` edge. The bonds are represented by two symmetric `bond` relations and we group the blank nodes belonging to a single molecule by connecting them to an instance node by `has` and `partOf` relations. See Fig. 1 for an example.

This process turns the set of single, small graphs into a densely connected web. On the one hand, we can imagine that this obscures the graph structure of the original graph. On the other, it may provide links which help the learning algorithms to connect related instances. We will see in the experiments in Sect. 4 that this translation to RDF diminishes classification performance for the MUTAG and ENZYME datasets. Ultimately it will depend on the learning algorithm used and the learning context whether the transformation to RDF will harm or help. All we can say from this example, is that the form of the data is radically changed.

Of course, the non-RDF form of the data will always be recoverable from the RDF, but, in general, doing so will require domain specific processing, possibly by hand. If linked data becomes the standard means of representing data for safe, long term storage, the easiest way to use it in machine learning applications

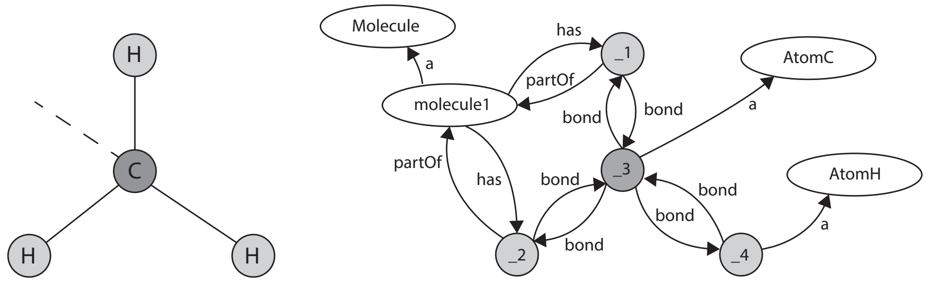

Fig. 1: A common graph-learning problem: molecular structure. The figure on the left shows the basic form of the data, the figure on the right shows an RDF representation. For the sake of clarity, not all RDF relations of the type `a`, `partOf` and `has` are shown.

would be in that form, without manual pre-processing. In the following section we investigate two automatic pre-processing methods that will simplify the data for machine learning purposes.

It may seem that removing the links to ontology nodes reverses the process, but this would leave the remaining nodes with unique labels, removing the atom types. Additionally, we expect that some of the RDF enrichment might help the learning process. Ultimately, we would like to have an algorithm which can simplify a graph based on the local graph structure, rather than hard-coded assumptions about the role of RDF, so that it removes only the annotations that will hurt learning.

3 Pre-processing Methods

As we saw in the previous example, exposing data as RDF can have a profound effect on its graph structure. We will present two methods to deal with this issue. We view an RDF dataset as a directed multigraph with the resources, literals and blank nodes as nodes in the graph, and their relations as directed, labeled edges. Literals that occur more than once are seen as the same node, and the fact that edges can be linked to nodes (i.e. predicates can be subjects or objects) is ignored.

Hub removal Our first approach to simplifying graphs is based on the principle that the hubs in the RDF graph—those nodes that are connected to many other nodes—are likely to add little information about an instance.

This approach is partly inspired by the Slash-and-burn algorithm [10]. This algorithm uses the property of scale-free graphs that removing the hubs causes the graph to fall apart in to disconnected components [11]: it splits a graph into a small collection of hubs and a long tail of disconnected islands. Our intuition is that these islands represent the innate instances of the data, and that the hubs represent the added semantic relations.[3]We use three methods to find the list of hubs:

rdf-type We consider as hubs only those nodes which are the object in an `rdf:type` relation. We sort by the number of such relations the node is a part of.

degree-plain We sort the nodes by degree (the sum of in- and out-degree) and choose the top h nodes as hub.

degree-signature In this method we group the edges around the node by the combination of the direction of the edge and its label (we call this combination the signature). We take the frequency of the most frequent signature among the node's neighboring edges and sort all nodes by this, removing the top h nodes with the highest such frequency. The idea is that a node which means a lot of different things to other nodes does not make a hub. Only when it means the same thing to a lot of nodes does it become a hub.

Relabeling Removing hubs will go some way to reducing the complexity of the graph, but some issues remain. To illustrate, we can see that removing hubs from the RDF version of our molecule dataset still leaves the most important nodes blank: those representing atoms. For these nodes, their node label is irrelevant, but the `a` relation and the label of the node it points to (the atomic symbol) are essential information.

We therefore introduce a relabeling operation. We investigate all nodes that are neighbors of removed hubs. For these nodes, we take the hub with the lowest degree to which it is connected, and relabel it with the concatenation of the relation to the hub and the hub's label. We test this method both together with node removal and on its own (where we detect the nodes, but do not remove them).

4 Experiments

In the following experiments, we test our simplification methods on 5 classification tasks. The first two tasks deal with the regular and RDF versions of the molecule datasets MUTAG and ENZYMES. The goal of these two tasks is to test whether the problems introduced in our illustrative example in Sect. 2 can be

[3] Using the Slash-and-burn algorithm explicitly to detect hubs turned out to be equivalent to simply removing the top h hubs by degree, since RDF graphs are not usually scale free in a strict sense, so they can remain connected even if many hubs are removed.

dealt with using our simplification techniques. The other three tasks use regular, real-world RDF datasets. For machine learning, we use two representative graph kernels for RDF: the Weisfeiler-Lehman and the Intersection SubTree graph kernels. These kernels are combined with a Support Vector Machine (SVM) [12] as our classification algorithm.

Weisfeiler-Lehman The Weisfeiler-Lehman (WL) graph kernel [8] is a fast, state-of-the-art kernel that iteratively constructs features that represent the subtrees that occur in the graph. These features are constructed by an efficient rewrite procedure, which creates a new multiset label for a node based on the neighbors of that node. This multiset is sorted and together with the original label concatenated into a string, which is the new label. For each unique string, a new (shorter) label is introduced to replace the original node label. The rewriting process can be efficiently implemented using counting sort, see [8] for details.

The Weisfeiler-Lehman algorithm was adapted for RDF in [6]. This adaptation computes the features for all instances on the full RDF graph at once. To make sure that as much of the results in the experiments can be attributed to the graph pre-processing, we use a simpler adaptation, which is as close to the regular WL kernel as possible. The version in this paper is adapted to RDF to handle directed edges and edge labels, but is computed on separate instance subgraphs. It also forgoes the weighting of the rewrite iterations and has the labels 'travel' along the reverse direction of the edge.[4]

Intersection SubTree The Intersection SubTree (IST) kernel was defined specifically for RDF in [5] and takes a different approach to comparing RDF instances than the WL kernel. This kernel works by counting the number of full subtrees in the intersection tree that is created by taking the common children of the two instance root nodes and iterating this to a certain depth, 3 in our experiments. A version that counts partial subtrees is also defined in [5], but both in [5] and [7] very little performance difference with the full subtree version is shown, so we do not include it here. We only test the IST kernel for the real-world RDF datasets, since it was not developed to handle the regular graphs like the molecule datasets and performs poorly on them.

Experimental set-up We test the cross-product of our hub removal and relabeling methods. This leads to testing: rdf-type, degree-plain and degree-signature, with three settings: remove links to hubs (Li), relabel nodes (La) and both (LiLa). Each of these 9 simplification variants are tested for a number of different total hub settings h, differing per task. Instance subgraphs are extracted from the simplified graph up to depth 3 (respecting edge directions). No additional inferencing by the triple-store is done.

[4] The algorithm we use is one of the comparison algorithms of [6], without the iteration weighting and the labels moving in the opposite direction along the edges.

For each setting an SVM classifier is evaluated using 10-fold cross-validation, repeated 10 times to remove random fold assignment effects. Within folds, the C parameter of the SVM is optimized using an inner 10-fold cross-validation loop. As performance measure we use classification error, i.e. the fraction of misclassified instances. For the WL kernel we optimize over the number of iterations parameter and for the IST kernel we use the settings from the original paper [5].

All of our experiments are implemented in Java and are available online, together with the datasets used[5]. For our SVM we use the Java version of the LibSVM [13] library and for dealing with the RDF data we use the SESAME library.[6]

4.1 Molecular datasets

The first two tasks involve classifying molecular data from the MUTAG and ENZYMES datasets. In the MUTAG dataset there are 188 instances in 2 classes, and in the ENZYMES set we have 600 instances and 6 classes. We test the performance of the regular Weisfeiler-Lehman (WL) algorithm on the original data and the performance of WL on the RDF versions of these datasets with our simplification methods. These experiments aim to show that indeed our illustrative example from Sect. 2 leads to problems and that our pre-processing methods can (partially) alleviate these problems.

We include the 0 hubs setting as a baseline; under this setting the graph is not simplified in any way. We test up to 6 hubs removed, since we know by our construction that there are no more hubs in the datasets. For the regular WL kernel we optimize over the iterations parameter from $0, 1, 2, 3, 4, 5, 6$ and for the WL RDF version from $0, 2, 4, 6, 8, 10, 12$.[7]

The results for the MUTAG and ENZYME datasets are presented in Table 1. Bold indicates the best score for the dataset, or those scores that do not have a significant difference with the best score under a Student t-test with $p < 0.05$.

Discussion In both experiments we see that using unsimplified graphs ($h = 0$) shows clearly worse performance than when using the original data. For the MUTAG dataset we see that using simplification allows us to reach a better performance than on the original data. This is quite remarkable, and likely has something to do with the fact that WL for RDF takes edges into account and therefore the feature vectors contain a feature that represents the number of edges. In the case of the ENZYMES dataset, simplification allows us to get substantially closer to the performance on the original data. In both experiments

[5] `http://www.data2semantics.org/publications/pre-processing-eswc-2014`

[6] `http://www.openrdf.org`

[7] The RDF version of Weisfeiler-Lehman includes edges, therefore 2 iterations are ‘equal’ to 1 iteration of the regular WL algorithm.

Table 1: Mean error for the simplification experiments on the molecular datasets. 'Li' indicates removal of the links to the hubs, 'La' indicates relabeling of the node with the labels of the link and hub. h indicates the number of hubs used for simplification.

MUTAG									
	regular data: 0.127								
		rdf-type			degree-plain			degree-signature	
h	Li	La	LiLa	Li	La	LiLa	Li	La	LiLa
0	0.178	0.178	0.178	0.178	0.178	0.178	0.178	0.178	0.178
1	0.239	0.141	0.146	0.239	0.141	0.146	0.239	0.141	0.146
2	0.190	**0.116**	**0.115**	0.190	**0.116**	**0.115**	0.190	**0.116**	**0.115**
3	0.222	**0.114**	0.118	0.222	**0.114**	0.118	0.222	**0.114**	0.118
4	0.257	**0.102**	**0.110**	0.257	**0.102**	**0.110**	0.257	**0.102**	**0.110**
5	0.267	**0.118**	**0.120**	0.267	**0.118**	**0.119**	0.267	**0.118**	**0.120**
6	0.265	**0.118**	0.122	0.265	**0.117**	0.122	0.265	**0.118**	0.122
ENZYME									
	regular data: **0.474**								
		rdf-type			degree-plain			degree-signature	
h	Li	La	LiLa	Li	La	LiLa	Li	La	LiLa
0	0.805	0.805	0.805	0.805	0.805	0.805	0.805	0.805	0.805
1	0.777	0.748	0.745	0.777	0.748	0.745	0.777	0.748	0.745
2	0.820	0.744	0.749	0.820	0.744	0.749	0.820	0.744	0.749
3	0.824	0.591	0.596	0.824	0.591	0.596	0.824	0.591	0.596
4	0.828	0.581	0.584	0.828	0.581	0.584	0.828	0.581	0.584
5	0.828	0.581	0.584	0.828	0.581	0.584	0.828	0.581	0.584
6	0.828	0.581	0.584	0.827	0.580	0.583	0.827	0.580	0.583

relabeling (La) of the RDF graph is needed to improve performance. The experiments make virtually no distinction between the three simplification methods.

4.2 Affiliation Prediction

For our first experiment with a real-world RDF dataset we repeat the affiliation prediction experiment from [6], which was introduced in [14] and repeated in [5]. In this experiment, data is used from the semantic portal of the AIFB research institute. The goal is to predict one of four affiliations for the people in the institute. The fifth affiliation is ignored, since it only has 4 members, leaving a total of 174 instances. Since the affiliations are known, the `affiliation` relation (and its inverse the `employs` relation) are removed from the RDF for training.

Table 2 shows the results for the graph simplification experiments. The h setting is varied from 0 to 40. For the WL algorithm we optimize over the iterations parameter from $0, 2, 4, 6$. Again, bold type means the best score or not significantly different from the best score.

Table 2: Mean error for the AIFB simplification experiment.

	rdf-type			degree-plain			degree-signature		
h	Li	La	LiLa	Li	La	LiLa	Li	La	LiLa
	Weisfeiler-Lehman								
0	0.097	0.097	0.097	0.097	0.097	0.097	0.097	0.097	0.097
1	0.098	0.118	0.119	0.099	0.119	0.119	0.098	0.118	0.119
2	0.097	0.132	0.137	0.095	0.134	0.137	0.097	0.132	0.137
3	0.093	0.133	0.141	**0.085**	0.112	0.117	**0.082**	0.111	0.111
4	0.095	0.116	0.129	0.091	0.110	0.104	0.094	0.109	0.113
5	0.092	0.108	0.115	**0.085**	0.107	0.106	0.096	0.105	0.107
10	0.095	0.108	0.124	0.091	0.100	**0.088**	0.089	0.097	0.096
20	0.092	0.107	0.125	0.103	0.102	0.101	0.095	0.108	0.099
30	0.093	0.108	0.131	0.109	0.107	0.099	0.102	0.092	**0.085**
40	0.093	0.108	0.131	0.095	0.102	0.107	0.098	0.094	0.111
	Intersection SubTree								
0	0.172	0.172	0.172	0.172	0.172	0.172	0.172	0.172	0.172
1	0.171	0.546	0.546	0.172	0.546	0.546	0.171	0.546	0.546
2	0.167	0.546	0.571	0.167	0.546	0.566	0.167	0.546	0.571
3	0.167	0.551	0.600	0.172	0.345	0.367	0.171	0.327	0.328
4	0.168	0.517	0.562	0.172	0.408	0.444	0.167	0.421	0.459
5	0.168	0.540	0.576	0.172	0.379	0.412	0.167	0.362	0.402
10	0.169	0.719	0.734	0.166	0.151	0.178	0.165	0.205	0.226
20	0.160	0.712	0.725	0.171	0.268	0.348	0.165	0.227	0.259
30	0.162	0.712	0.727	0.171	0.253	0.318	0.165	0.217	0.254
40	0.162	0.712	0.727	0.164	0.544	0.580	0.178	0.533	0.551

Discussion Using graph simplification, it is possible to improve the performance for this task by nearly 16% (0.097 to 0.082) for the WL kernel, but this depends very much on the number of hubs considered. Contrary to the molecule tasks, there is a difference between the methods, with the degree-plain and degree-signature techniques performing better. Also, relabeling (La) is not a necessary step in this experiment. The intersection subtree kernel performs substantially worse on this task. However, there are cases were performance is improved over the $h = 0$ baseline. It is interesting to see that some pre-processing settings can have a real detrimental effect on performance for the IST kernel, whereas, although the performance for the WL kernel can also drop, it is never by such a large amount.

4.3 ISWC Program chair prediction

For our second experiment, we use RDF data about the ISWC conferences of 2010 and 2011 to predict members of the program committee for the 2012 conference[8]. A total of 302 people went to all three conferences, of which 115 where part of the 2012 program committee for the research track.

The results are presented in Table 3. The experimental settings are similar to the AIFB experiment.

Table 3: Mean error for the ISWC simplification experiment.

	rdf-type			degree-plain			degree-signature		
h	Li	La	LiLa	Li	La	LiLa	Li	La	LiLa
	Weisfeiler-Lehman								
0	0.233	0.233	0.233	0.233	0.233	0.233	0.233	0.233	0.233
1	0.233	0.236	0.232	0.233	0.236	0.232	0.233	0.236	0.232
2	0.233	0.240	0.240	0.233	0.234	0.232	0.233	0.234	0.232
3	0.233	0.236	0.238	0.233	0.240	0.239	0.233	0.24	0.239
4	0.233	0.236	0.237	0.233	0.236	0.237	0.233	0.236	0.237
5	0.233	0.236	0.237	0.341	0.233	0.328	0.233	0.236	0.237
10	0.233	0.238	0.238	0.344	0.235	0.343	0.277	0.233	0.285
20	0.233	0.240	0.242	0.352	0.236	0.362	0.245	0.239	0.245
30	0.233	0.240	0.242	0.261	0.239	0.255	0.227	0.239	**0.225**
40	0.233	0.240	0.242	0.284	0.240	0.260	**0.226**	0.237	**0.223**
	Intersection SubTree								
0	0.228	0.228	0.228	0.228	0.228	0.228	0.228	0.228	0.228
1	0.228	0.228	0.228	0.228	0.228	0.228	0.228	0.228	0.228
2	0.228	0.234	0.235	0.228	0.228	0.228	0.228	0.228	0.228
3	0.228	0.251	0.252	0.228	0.234	0.235	0.228	0.234	0.235
4	0.228	0.251	0.251	0.228	0.251	0.252	0.228	0.251	0.252
5	0.228	0.251	0.251	0.336	0.251	0.339	0.228	0.251	0.251
10	0.228	0.252	0.253	0.337	0.442	0.448	0.263	0.248	0.291
20	0.228	0.456	0.459	0.306	0.429	0.441	0.227	0.242	0.257
30	0.228	0.456	0.459	0.267	0.432	0.401	0.228	0.237	0.280
40	0.228	0.456	0.459	0.259	0.438	0.357	0.227	0.234	0.250

Discussion As in the affiliation prediction task, performance can be improved for the WL kernel, but it does depend greatly on the h setting. The degree-signature methods show the best performance. Removing links (Li) for the degree-plain method has a clear negative impact. The performance difference between the IST and WL kernels as not as large as in the AIFB experiment (for the $h = 0$ baseline, IST even performs slightly better). However, pre-processing does not improve performance for the IST kernel and some settings can impact the performance quite severely.

[8] Available from `http://data.semanticweb.org/`

4.4 Lithogenesis prediction

For our last experiment we repeat the Lithogenesis prediction task from [6]. In this task, data from the British Geological Survey[9] is used, which contains information about geological measurements in Britain. The things measured by this survey are 'Named Rocked Units'. For these named rock units we try to predict the lithogenesis property, for which the two largest classes have 93 and 53 instances. Triples related to this property are removed from the dataset.

Experimental settings are as in the previous two experiments. Results are in Table 4.

Table 4: Mean error for the BGS simplification experiment.

	rdf-type			degree-plain			degree-signature		
h	Li	La	LiLa	Li	La	LiLa	Li	La	LiLa
	Weisfeiler-Lehman								
0	0.095	0.095	0.095	0.095	0.095	0.095	0.095	0.095	0.095
1	0.093	0.092	0.093	0.093	0.092	0.093	0.093	0.092	0.093
2	0.099	0.091	0.093	0.093	0.094	0.095	0.092	0.093	0.089
3	0.099	0.091	0.093	0.097	0.107	0.106	0.091	0.097	0.090
4	0.099	0.091	0.093	0.097	0.103	0.110	0.092	0.097	0.096
5	0.099	0.091	0.093	0.098	0.098	0.111	0.096	0.096	0.103
10	0.099	0.091	0.093	0.101	0.099	0.112	0.097	0.101	0.105
20	0.099	0.091	0.093	0.101	0.090	0.097	0.105	0.093	0.105
30	0.099	0.091	0.093	0.104	0.084	0.092	0.095	**0.075**	**0.068**
40	0.099	0.091	0.093	0.127	0.085	0.097	0.116	0.084	0.079
	Intersection SubTree								
0	0.145	0.145	0.145	0.145	0.145	0.145	0.145	0.145	0.145
1	0.144	0.497	0.497	0.144	0.497	0.497	0.144	0.497	0.497
2	0.147	0.497	0.498	0.144	0.453	0.451	0.144	0.503	0.503
3	0.147	0.497	0.498	0.150	0.551	0.549	0.144	0.501	0.503
4	0.147	0.497	0.498	0.150	0.547	0.548	0.144	0.502	0.499
5	0.147	0.497	0.498	0.150	0.547	0.546	0.144	0.510	0.510
10	0.147	0.497	0.498	0.151	0.503	0.488	0.148	0.495	0.495
20	0.147	0.497	0.498	0.140	0.246	0.237	0.138	0.388	0.402
30	0.147	0.497	0.498	0.147	0.246	0.244	0.151	0.218	0.220
40	0.147	0.497	0.498	0.150	0.171	0.162	0.153	0.369	0.370

Discussion In this task only very few simplification settings show a performance improvement. However, the degree-signature method does show a good improvement (20%) for the $h = 30$ setting and the WL kernel. A possible reason for the difficulty in improving performance is that this dataset is relatively hierarchical in nature, compared to the previous two datasets, making hubs less important.

[9] `http://data.bgs.ac.uk/`

Again, pre-processing does not improve the IST kernel and often has a severe negative impact.

5 Conclusions and future work

We have dealt with the problem of applying graph-based machine learning directly to Linked Data. We have shown that the annotations that well-written RDF contains, can obfuscate the patterns exploited by machine learning algorithms.

We have also shown that this effect can be mitigated by automatic pre-processing, although validation is still required to choose a pre-processing method for a given learning task.

For tasks where no raw form of the data is available (our real-world RDF datasets), we show that hub removal and relabeling can still improve performance, although it is difficult to select a priori the correct number of hubs to remove, and to decide whether or not to apply relabeling. Furthermore, the Weisfeiler-Lehman benefits better from this preprocessing then the Intersection SubTree Kernel.

While our pre-processing methods show improvements in certain cases, and at times with consistent parameter values across datasets, they also show that there is room for improvement. Specifically, there is room to investigate the reason for the variation in performance with respect to the number of hubs removed, and to investigate the cause of the good performance at $h = 30$, for the WL kernel, across datasets. Since the top 30 hubs differ entirely across these datasets, it is surprising that good performance should occur at an absolute value for these three tasks.

Pre-processing is just one issue in the task of bringing machine learning to a world where data is represented in RDF. The following questions remain:

- Given a node representing an instance, how do we determine which nodes in its neighborhood we should extract to create an RDF subgraph representing the instance? Currently we simply extract to a fixed depth.
- Given a subgraph, can we translate it to feature vectors? Translating to feature vectors, rather than applying graph-based methods would allow us to use almost any traditional classifier or clustering algorithm. The WL algorithm can be used as a feature extractor. [10]
- What is the best way to evaluate a classifier on RDF data? There is no straightforward way to split the whole dataset in disconnected testing and training parts. Here, we have removed all target relations, and provided these to the training algorithm separately, but other schemes can be imagined.

[10] This is analogous to the technique of 'propositionalization' in knowledge discovery in relational databases.

Finally, since most of our methods apply to graphs in general, not just to RDF, they might be useful in any learning task where the raw data consists of a single large graph, and the instances are represented by its nodes.

Acknowledgments We thank the reviewers for their insightful comments. This publication was supported by the Dutch national program COMMIT. We thank the authors of [5] for the AIFB dataset.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data—the story so far. Int. J. Semantic Web Inf. Syst. **5**(3) (2009) 1–22
2. Rettinger, A., Lösch, U., Tresp, V., d'Amato, C., Fanizzi, N.: Mining the semantic web—statistical learning for next generation knowledge bases. Data Min. Knowl. Discov. **24**(3) (2012) 613–662
3. Fanizzi, N., d'Amato, C., Esposito, F.: Induction of robust classifiers for web ontologies through kernel machines. J. Web Sem. **11** (2012) 1–13
4. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In Getoor, L., Scheffer, T., eds.: ICML, Omnipress (2011) 809–816
5. Lösch, U., Bloehdorn, S., Rettinger, A.: Graph kernels for RDF data. In Simperl, E., Cimiano, P., Polleres, A., Corcho, Ó., Presutti, V., eds.: ESWC. Volume 7295 of Lecture Notes in Computer Science., Springer (2012) 134–148
6. de Vries, G.K.D.: A fast approximation of the Weisfeiler-Lehman graph kernel for RDF data. In Blockeel, H., Kersting, K., Nijssen, S., Zelezný, F., eds.: ECML/PKDD (1). Volume 8188 of Lecture Notes in Computer Science., Springer (2013) 606–621
7. de Vries, G.K.D., de Rooij, S.: A fast and simple graph kernel for RDF. In d'Amato, C., Berka, P., Svátek, V., Wecel, K., eds.: DMoLD. Volume 1082 of CEUR Workshop Proceedings., CEUR-WS.org (2013)
8. Shervashidze, N., Schweitzer, P., van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-Lehman graph kernels. J. Mach. Learn. Res. **12** (November 2011) 2539–2561
9. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R.I., Borgwardt, K.M.: Graph kernels. Journal of Machine Learning Research **11** (2010) 1201–1242
10. Kang, U., Faloutsos, C.: Beyond'caveman communities': Hubs and spokes for graph compression and mining. In: Data Mining (ICDM), 2011 IEEE 11th International Conference on, IEEE (2011) 300–309
11. Albert, R., Jeong, H., Barabási, A.L.: Error and attack tolerance of complex networks. Nature **406**(6794) (2000) 378–382
12. Vapnik, V.: The nature of statistical learning theory. Springer (2000)
13. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2** (2011) 27:1–27:27 Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
14. Bloehdorn, S., Sure, Y.: Kernel Methods for Mining Instance Data in Ontologies. The Semantic Web (2007) 58–71