Web Semantics: Science, Services and Agents on the World Wide Web 35 (2015) 71–84

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Substructure counting graph kernels for machine learning from RDF data

Gerben Klaas Dirk de Vries [a,*], Steven de Rooij [b,a]

[a] System and Network Engineering Group, Informatics Institute, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands

[b] Knowledge Representation and Reasoning Group, Department of Computer Science, VU University Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands

HIGHLIGHTS

- Systematic graph kernel framework for RDF.
- Fast computation algorithms.
- Low frequency labels and hub removal on RDF to enhance machine learning.

ARTICLE INFO

Article history:
Received 15 December 2014
Received in revised form
16 June 2015
Accepted 3 August 2015
Available online 10 August 2015

Keywords:
Graph kernels
Machine learning for RDF
Weisfeiler–Lehman
Hub removal

ABSTRACT

In this paper we introduce a framework for learning from RDF data using graph kernels that count substructures in RDF graphs, which systematically covers most of the existing kernels previously defined and provides a number of new variants. Our definitions include fast kernel variants that are computed directly on the RDF graph. To improve the performance of these kernels we detail two strategies. The first strategy involves ignoring the vertex labels that have a low frequency among the instances. Our second strategy is to remove hubs to simplify the RDF graphs. We test our kernels in a number of classification experiments with real-world RDF datasets. Overall the kernels that count subtrees show the best performance. However, they are closely followed by simple bag of labels baseline kernels. The direct kernels substantially decrease computation time, while keeping performance the same. For the walks counting kernel this decrease in computation time is so large that it thereby becomes a computationally viable kernel to use. Ignoring low frequency labels improves the performance for all datasets. The hub removal algorithm increases performance on two out of three of our smaller datasets, but has little impact when used on our larger datasets.

1. Introduction

In recent years graph kernels have been introduced as a promising method to perform data mining and machine learning on Linked Data and the Semantic Web. These methods take Resource Description Framework (RDF) data as input.

One main advantage of this approach is that the techniques are therefore very widely applicable to all kinds of Linked Data. Almost no assumptions are made on the semantics of the data and its content, other than that it is in RDF. So, additionally, these methods require very little knowledge of Semantic Web technologies to be employed.

Another advantage is the host of existing machine learning algorithms, called kernel methods [1,2], that can be used with these graph kernels. The most well known of these algorithms is the Support Vector Machine (SVM) for classification and regression. However, algorithms exist for ranking [3,4], clustering [5], outlier detection [6], etc., which can all be used directly with these kernels. More recently, interest has increased for large scale linear classification [7] for larger datasets, for which a number of graph kernels can also be used.

In this paper we give a comprehensive overview of graph kernels for learning from RDF data. We introduce a framework for these kernels, which are based on counting different graph substructures, that encompasses most of the graph kernels previously introduced for RDF, but also introduces new variants. The

[*] Corresponding author. Tel.: +31 20 525 7522.
E-mail addresses: g.k.d.devries@outlook.com (G.K.D. de Vries), steven.de.rooij@gmail.com (S. de Rooij).

http://dx.doi.org/10.1016/j.websem.2015.08.002
1570-8268/© 2015 Elsevier B.V. All rights reserved.

framework includes fast kernel variants that are computed *directly* on the RDF graph. We also detail the necessary adaptation of the Weisfeiler–Lehman graph kernel [8] needed to compute a number of kernels in our framework. Furthermore, we give two strategies to further improve the machine learning performance with these kernels. The first strategy ignores vertex labels which have a low frequency of occurrence among the instances and the second strategy removes hubs to simplify the RDF graphs. All of our kernels defined in the framework can be used with large scale linear classification methods.

The kernels are studied in a number of classification experiments on different RDF datasets. The goal of these experiments is to study the influence of the different choices for graph kernels defined in our framework. It turns out that kernel performance differs per dataset. Overall, kernels that count subtrees in the graphs are the best choice. However, simple bag of labels baseline kernels also perform well and are significantly cheaper to compute. The strategy to ignore low frequency labels has a positive effect on performance in all tasks, whereas the hub removal strategy only has a positive effect in a number of tasks and has no influence for larger datasets.

The work presented in this paper consolidates and expands our earlier papers on graph kernels for RDF [9,10] and hub removal [11].

The rest of this paper is structured as follows. We begin with an overview of related work. In Section 3 we introduce our kernel framework and algorithms. Section 4 covers our experiments with these kernels. We end with conclusions and suggestions for future work.

2. Related work

Graph kernels, such as those introduced in [12,8,13], are methods to perform machine learning on graph structured data, using kernel methods [1,2].

For learning from RDF data, the intersection subtree and intersection graph kernels were introduced in [14]. A fast approximation of the Weisfeiler–Lehman graph kernel [8], specifically designed for RDF was introduced in [9]. In [10] a fast and simple graph kernel, similar to the intersection subtree kernel was defined. In the context of recommender systems using linked data [15] and [16] introduce kernels based on the labels in the neighborhood of a vertex. The current paper defines a framework which systematically covers most of these previously introduced kernels.

There are a number of papers that cover machine learning from Linked Data and the Semantic Web using kernel methods, which focus on the formal/ontological level of the data, and which are therefore less generally applicable. One of the first papers to introduce kernel methods to learn from the Semantic Web was [17]. In that paper, kernels are defined manually using task relevant properties of the instances. Other approaches are based on description logics, such as [18], and inductive logic programming [19].

The specific task of link prediction on RDF graphs is tackled in several papers. One approach is to use matrix factorization [20,21]. Another, related approach, is to use tensor factorization [22,23]. Finally, there are also approaches using (tensor) neural networks [24,25] and very recently Gaussian processes [26]. The graph kernel methods of this paper are applicable to a more wide range of machine learning and data mining tasks than these link prediction methods. For instance, graph kernels can be used in clustering, to predict labels/classes that are not links in the graph.

The framework for graph kernels for RDF that we define in this paper has some similarities to the comparison of propositionalization [27] strategies for Linked Data in [28], in the sense that some of the propositionalizations are similar to the graph substructures that our graph kernels work with. However, they do not use kernel methods and use Linked Data as background knowledge to enhance data mining on 'normal' data [29].

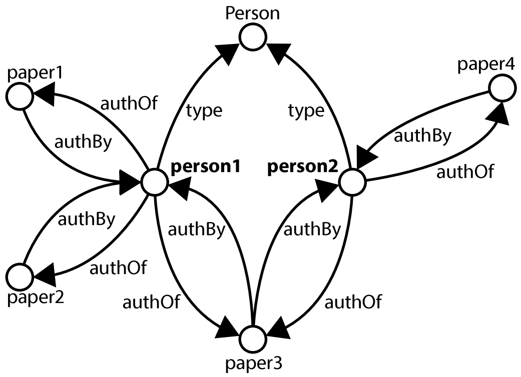

Fig. 1. Small example RDF graph of an author network.

For a relatively recent overview of data mining and machine learning from Linked Data and the Semantic Web, see [30].

3. Graph kernels for RDF

The Resource Description Framework (RDF)[1] is the foundation of Linked Data and the Semantic Web. The central idea is to store statements about resources in subject–predicate–object form, called *triples*, which define relations between a set of terms. A triple (s, p, o) in a set of triples $\mathcal{T}$ specifies that the subject term s is related to the object term o by the predicate p.

Often a set of triples $\mathcal{T}$ is referred to and visualized as a graph, where the subject and object terms in the data constitute the vertices, and a triple (s, p, o) specifies that s and o are connected by an edge with label p. This graph interpretation breaks down when predicate terms are also used in the subject or object position, for example to define properties of the relations themselves [31]. However, most datasets that occur in practice define only a limited number of facts *about* the relations, so for simplicity we choose to interpret predicates in the predicate position and predicates in the subject or object positions as distinct, even if they have the same label.

More expressive Semantic Web knowledge representation formalisms, such as the Web Ontology Language (OWL) and RDF Schema (RDFS) can be represented in RDF. Therefore, RDF triplestores, often include reasoning engines to automatically derive new triples, if the RDF represents data modeled using these more expressive formalisms.

Fig. 1 is an illustration of a simple RDF graph of two Persons (person1 and person2) that authored a couple of papers, one of them (paper3) together.

Learning from RDF using graph kernels takes the approach that instances, which are vertices in a large RDF graph, are represented by their neighborhood, i.e. the triples around them, up to a certain depth. For example, in Fig. 1, the instances could be person1 and person2, and their neighborhood of depth 1 would be all the triples in which they are the subject.

The graph in Fig. 1 is an intuitive visualization of RDF data as a directed labeled multigraph. But when described formally, labeled multigraphs are more complicated than necessary for our purpose. Moreover, such a representation requires us to always distinguish between edges and vertices, which turns out not to be necessary in the description of any of our algorithms. So instead we opt to reify

[1] http://www.w3.org/standards/semanticweb/.