Efficient Graph Kernels for RDF data using Spark

Bernhard Japes[1] and Shinho Kang[2]

[1] Informatik III, Universität Bonn, Germany
bernhard.japes@uni-bonn.de
[2] Informatik III, Universität Bonn, Germany
TODO

Abstract

In this paper we study the application of graph kernels for RDF data using the popular Apache Spark[1] engine in combination with the SANSA-Stack[2] data flow utilities. We focus on an implementation of the Intersection Tree Path (ITP) Kernel, published by Gerben Klaas Dirk de Vries and Steven de Rooij in [2], that is based on the concept of constructing a tree for each instance and counting the number of paths in that tree.

TODO: Add further information about implementation and/or results

1 Introduction

The increasing availability of structured data and the rise of the semantic web pose new challenges for machine learning and data mining. As an official standard, the *Resource Description Framework* (RDF) is commonly used to represent those graphs, which led to research on how to use the RDF structure to predict links and labels of instances efficiently. Most of the current approaches to mining structured graph-data focus on specific semantic properties and are individually designed for different problems [4, 3].

Kernels, however, have already been proven to be useful as a much more flexible approach for Pattern Analysis in different areas [5], which resulted in further research on specific graph kernels for RDF. The main drawback of most of these graph kernels, including the state-of-the-art *Weisfeiler-Lehman* (WL) RDF kernel, is their computation time as shown in [1]. In [2] Gerben Klaas Dirk de Vries and Steven de Rooij present a *Fast and Simple Graph Kernel for RDF* with just a slightly worse prediction performance than the WL graph kernel, but the huge upside of being 10 times faster in practice. Their idea of a fast and simple, but scalable kernel also seems to be promising for big data applications. However several adaptions of their algorithm are required to ensure consistent computations on distributed data sets using the Apache Spark engine.

2 Approach

The graph kernel...

[1] http://spark.apache.org
[2] http://www.sansa-stack.net

3 Implementation

4 Evaluation

5 Conclusion

5.1 Project timeline

5.2 Further ideas

References

[1] Gerben K. D. de Vries. *A Fast Approximation of the Weisfeiler-Lehman Graph Kernel for RDF Data*, pages 606–621. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[2] Gerben Klaas Dirk De Vries and Steven De Rooij. A fast and simple graph kernel for rdf. In *Proceedings of the 2013 International Conference on Data Mining on Linked Data - Volume 1082*, DMoLD'13, pages 23–34, Aachen, Germany, Germany, 2013. CEUR-WS.org.

[3] Yi Huang, Volker Tresp, Markus Bundschus, Achim Rettinger, and Hans-Peter Kriegel. *Multivariate Prediction for Learning on the Semantic Web*, pages 92–104. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[4] Achim Rettinger, Matthias Nickles, and Volker Tresp. *Statistical Relational Learning with Formal Ontologies*, pages 286–301. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[5] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

Algorithm 1: The Intersection Tree Path Kernel as introduced in [2]

Data: a set of RDF triples R, a set of instances I and a max depth d_{max}
Result: a set of feature vectors F corresponding to the instances I
Comment: $pathMap$ is a global map between paths of vertices and edges and integers

- set $pathIdx = 1$
- for each $i \in I$
  - create a new feature vector fv
  - do $processVertex(i, i, [], fv, d_{max})$
  - add fv to F

function $processVertex(v, root, path, fv, d)$
- if $d = 0$, return
- for each $(v, p, o) \in R$
  - if o is $root$, set $path = [path, p, rootID]$
  - else, $path = [path, p, o]$
  - if $pathMap(path)$ is undifned
    - set $pathMap(path) = pathIdx$ - set$pathIdx = pathIdx + 1$
  - set $fv(pathMap(path)) = fv(pathMap(path)) + 1$
  - do $processVertex(o, root, path, fv, d - 1)$