RMonto: Ontological Extension to RapidMiner

Jedrzej Potoniec, Agnieszka Ławrynowicz
Institute of Computing Science,
Poznań University of Technology, Poznań, Poland,
email: {jpotoniec,alawrynowicz}@cs.put.poznan.pl

Abstract. We present RMonto, an ontological extension to RapidMiner, that provides possibility of machine learning with formal ontologies. RMonto is an easily extendable framework, currently providing support for unsupervised clustering with kernel methods and (frequent) pattern mining in knowledge bases. One important feature of RMonto is that it enables working directly on structured, relational data. Additionally, its custom algorithm implementations may be combined with the power of RapidMiner through transformation/extraction from the ontological data to attribute-value data.

1 Introduction

The Web of Data of today is composed of billions of RDF triples with billions of links between data items. This ongoing publishing effort requires new methods, and new tools for data analysis, mining, and knowledge extraction that would be able to cope with semantic expressivity of the data, as well as with its heterogeneity. To address the challenge of mining semantic data, especially that represented with terms coming from formal ontologies, we decided to develop a tool, named RMonto. RMonto is an extension to a world-leading open source data mining, predictive analytics, and business analytics environment, RapidMiner [5]. RapidMiner is used for research, education, rapid prototyping, application development, and industrial applications. It provides data mining and machine learning procedures such as data loading and transformation, data preprocessing and visualization, modelling, evaluation, and deployment. Despite RapidMiner is a mature and flexible tool, it is concentrated on tabular data and thus not prepared to be used with relational data, even that simple as RDF.

Importantly, the analysis of the state-of-art in available tools, or more essentially frameworks for mining semantic data reveals almost luck of such tools in the market. An exception is DL-Learner [4], which provides a framework for learning in description logics and OWL, currently mostly addressing the concept learning task. Our approach may be distinguished from DL-Learner in exploiting the framework of RapidMiner to go further with a modularized, and compositional approach, and making possible design of data mining workflows ultimately more arbitrary than what is currently available to be done with DL-Learner.

There also exists an extension to RapidMiner called rapidminer–semweb [2] that provides operators for extracting an RDF graph from a repository and transforming it into a feature vector. However, it comes without actual learning algorithms that would tackle the problem of learning directly from expresssive and semantically rich data such as the one expressed in OWL.

In this context, our tool is conceived to provide both: relational learning algorithms, as well as a suite of data extraction and transformation methods to allow to use classical, propositional learning algorithm implementations on semantic data. Through the use of the framework of RapidMiner it also allows for flexible modelling of complex data mining processes as workflows.

2 System Description

The architecture of RapidMiner enables building data mining processes (workflows) from small blocks (operators) by connecting their inputs and outputs. It enables our custom operators (e.g. Semantic Web data extraction blocks or data mining algorithm implementations) to be interconnected with existing RapidMiner operators and provides added value to both. RMonto is implemented in Java, similarly to RapidMiner. It has been split into several libraries to easily maintain code reusability and separability. They create three layers:

1. Libraries implementing common interface between reasoning software (e.g. Pellet[1] or OWLim[2]) and algorithm's implementations (called PutOntoAPI).
2. Actual algorithm implementations, put to separated libraries (SeSiL – Semantic Similarity measures Library, Fr-Ont – Frequent patterns in Ontologies) for easier usage in other projects.
3. Extension, being a bridge between above layers and RapidMiner.

Providing common architecture to different reasoners has additional advantage that actual implementation of those interfaces does not have to be known during software building. RMonto makes use of this development by a run-time look up for libraries implementing the interfaces. The whole RapidMiner home directory is being searched and manifest of every JAR file is checked in order to choose libraries actually implementing facade to a reasoner. Dependencies, as file names, can also be specified in manifest and they will be loaded during loading JAR. Such flexible, plugin architecture allows to avoid one more issue about different software licences, i.e. makes possible for us to distribute only bridge library and to leave the responsibility to provide reasoner libraries to the user (this is the case i.e. with OWLim).

Below we describe currently implemented operators grouping them w.r.t RMonto RapidMiner's tree hierarchy.

Loading We developed three operators for loading data: *Load from file*, *Load from SPARQL endpoint* and *Build knowledge base*. *Load from file* defines access to files accessible with normal file-system operations. *Load from SPARQL endpoint* makes possible to download a part of remote data from a SPARQL endpoint in the form of a subgraph. Those graphs are constructed with SPARQL CONSTRUCT query. *Load...* operators supply the third with parameters of data sources. It is to avoid situation, in which the whole data loading logic is hidden in one operator, what would happen if those parameters were given as *Build knowledge base's*. Data loading is invoked inside *Build knowledge base*, as callback methods of objects received from *Load...* operators.

[1] `http://clarkparsia.com/pellet/`
[2] `http://www.ontotext.com/owlim`

ABox *SPARQL selector* and *ABox extractor* operators both provide list of URIs identifying objects to be used in the learning process. Those URIs must be valid identifiers of individuals in a knowledge base built by *Build knowledge base*. The first operator uses SPARQL query to extract data and the second one simply extracts list of URIs of all the individuals in a KB.

TBox Operators *All known classes* and *Features selector* work in the similar way as those in *ABox* section, but build list of classes instead. Both generate list of classes in the internal RMonto format. We resigned from URI-list representation in favor of our internal format to provide more complex language, which makes usage of complex classes not defined in a KB (i.e. intersection of two defined classes). This is the way the *Features selector* operator works, providing an editor to edit list of classes by hand. *All known classes* extracts information about all named classes in the knowledge base.

Kernels This section contains operators implementing kernel functions for description logics. They compute dissimilarity matrix, that is pairwise dissimilarities between objects.

Clustering Currently, there are implementations of two algorithms. The first one, *Semantic k–Medoids* is a simple *k–Medoids* algorithm implemented to be able to work with dissimilarity matrix computed by kernel. *Agglomerative hierarchical clustering* is extended to produce a semantic description of computed clusters.

Pattern Mining Operator *Fr-ONT-Qu* is an implementation of a transformed version of *Fr-ONT* algorithm described in [3]. The implemented version discovers patterns in the form of SPARQL graph patterns.

Data Transformation *Add label from KB* extends the list of URIs generated for example by *SPARQL selector* with additional column, to be interpreted as label in supervised learning. The *Propositionalisation* uses output of *Fr-ONT-Qu* and list of (labelled) examples to build binary matrix of features.

The set of operators implemented within RMonto enables Inductive Logic Programming style applications with Semantic Web data. In particular, it supports an upcoming field of research coined *semantic data mining* [1]: a data mining approach where domain ontologies are used as background knowledge.

3 Proposed Demo

We will present RMonto on two types of workflows. The first one will cover the topic of unsupervised clustering, exploiting kernels for structural, semantic data, and using data downloaded on–line by querying DBpedia[3] SPARQL endpoint. A similar demo is available on RMonto website at `http://semantic.cs.put.poznan.pl/RMonto/`.

The workflow on discovering frequent patterns will be presented on a dataset selected from those distributed with DL-Learner. On this workflow we will demonstrate how we construct complex, first-order features by relational pattern mining from ontologies, and further couple it with a classical, RapidMiner's propositional regression or classification. Such a workflow is presented in Fig. 1 (for a description of the workflow see [1]).

[3] `http://dbpedia.org/`

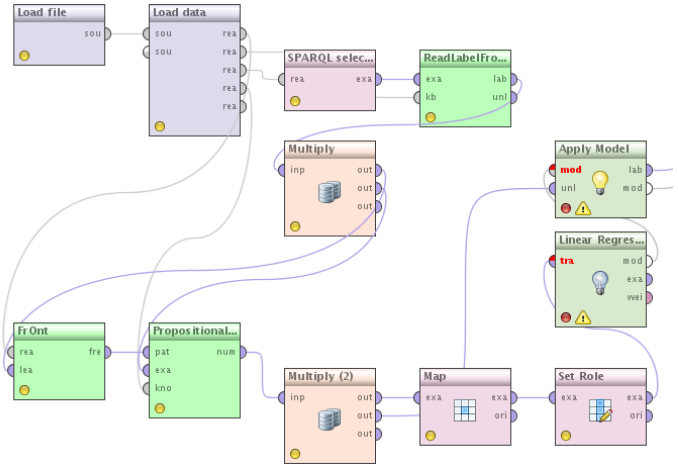

Fig. 1: Example workflow with frequent pattern mining for feature construction.

4 Conclusion and Future Work

The proposed tool, RMonto, provides an extensible framework for semantic data mining. In the nearest future, we will extend RMonto by recently proposed by our group kernels for $\mathcal{EL}^{++}$ description logic. We will also use RMonto to develop and test approaches to ontology-based meta-mining of data mining workflows in the field of recommender systems to automatically find characteristcs of a good workflow in this domain. In the future, we plan to provide more algorithm implementations, as well as methods for data transformation/extraction from the semantic data to the tabular data, complementing the ones currently available.

Acknowledgements. We acknowledge the support from European Community 7th framework program ICT-2007.4.4 (grant 231519 "e-LICO: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science") and from the Polish Ministry of Science and Higher Education (grant N N516 186437).

References

1. Hilario, M., Kalousis, A., Lavrac, N., Lawrynowicz, A., Potoniec, J., Vavpetic, A.: Semantic data mining tutorial at ECML/PKDD'2011 `http://semantic.cs.put.poznan.pl/SDM-tutorial2011/`
2. Khan, M.A., Grimnes, G.A., Dengel, A.: Two pre-processing operators for improved learning from SemanticWeb data. In: RapidMiner Community Meeting and Conference: RCOMM 2010 proceedings (2010), `http://www.cs.jyu.fi/ai/papers/IJIIT-2005.pdf`
3. Lawrynowicz, A., Potoniec, J.: Fr-ont: An algorithm for frequent concept mining with formal ontologies. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Ras, Z.W. (eds.) ISMIS. Lecture Notes in Computer Science, vol. 6804, pp. 428–437. Springer (2011)
4. Lehmann, J.: Dl-learner: Learning concepts in description logics. Journal of Machine Learning Research 10, 2639–2642 (2009)
5. Mierswa, I., Scholz, M., Klinkenberg, R., Wurst, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 935–940. ACM Press (2006)