Exploring Reasoning with the DMOP Ontology

C. Maria Keet[1], Claudia d'Amato[2], Zubeida Khan[1], and Agnieszka Ławrynowicz[3]

[1] Department of Computer Science, University of Cape Town, South Africa
keet@ukzn.ac.za,zkhan@csir.co.za
[2] Dipartimento di Informatica, Universita degli Studi di Bari, Italy
claudia.damato@uniba.it
[3] Institute of Computing Science, Poznan University of Technology, Poland
agnieszka.lawrynowicz@cs.put.poznan.pl

Abstract. We describe the Data Mining OPtimization Ontology (DMOP), which was developed to support informed decision-making at various choice points of the knowledge discovery (KD) process. DMOP contains in-depth descriptions of DM tasks, data, algorithms, hypotheses, and workflows. Its development raised a number of non-trivial modeling problems, the solution to which demanded maximal exploitation of OWL 2 representational potential. The choices made led to v5.4 of the DMOP ontology. We report some evaluations on processing DMOP with a standard reasoner by considering different DMOP features.

1 Introduction

Ontologies can enhance data mining in several ways, ranging from early experiments to improve clustering results [1] to let the miner learn from past mining experiments to optimize the data mining [2]. The latter constitutes a 'learning to learn' (meta-learning), which is defined as the application of machine learning techniques to meta-data about past machine learning experiments with the aim of improving the performance of the resulting model. The traditional meta-learning regarded only single machine learning algorithms and treated them as black boxes. However, the quality of the mined model not only depends on the output (learned model) of a single algorithm with characteristics of its input (data), but also on other phases of the data mining process, such as data cleaning and feature selection. This brings to the fore the requirement to describe the knowledge on how the different components of the data mining process interact so as to optimise the data mining/knowledge discovery (DM/KD) processes. CRISP-DM [3] provides a high-level standard model for the process of data mining. Several data mining ontologies exist, such as KDDONTO [4], KD ontology [5], DMWF [6] (for DM workflows), and OntoDM [7] (for definitions of the basic DM concepts). However, they lack the ability to manage the characteristics of algorithms.

To fill this gap, the Data Mining OPtimization Ontology (DMOP, pronounced deemope; http://www.dmo-foundry.org/) was developed. Its primary goal is to support all decision-making steps that determine the outcome of the DM process. Its primary use case has been in the Intelligent Discovery Assistant (IDA, comprised of an AI planner and semantic meta-miner) built within the e-LICO project (http://www.

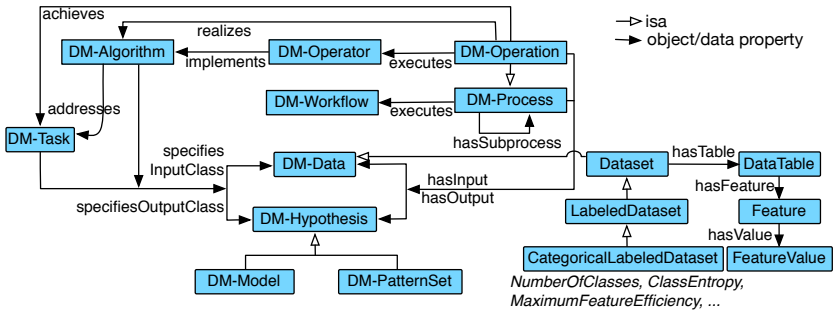

Fig. 1. Graphical rendering of the core concepts of DMOP, with a selection of three DataCharacteristics that are related to CategoricallabeledDataset.

`e-lico.eu`) that is deployed in the popular data mining tool RapidMiner. DMOP provides a richly axiomatised—using almost all features of OWL 2 DL—unified framework for analyzing DM tasks, algorithms, models, datasets, workflows and performance metrics, and their relationships and constraints. We provide a summary of DMOP v5.4 in Section 2; further details are described in [2] from a DM perspective, and in [8] from an ontology perspective. Anecdotal reports about reasoning over DMOP indicated that it was time-consuming (10-20 mins to classify the ontology), and we made several modelling decisions that may affect reasoner performance. To gain some insight into this, we performed a few evaluations from a modeller and reasoner end-user viewpoint. They are: establishing a baseline, the effect of the presence of the DOLCE foundational ontology to which DMOP is linked, the effect of using inverse object properties versus OWL 2 DL's feature `ObjectInverseOf()`, and charting the landscape of automated modularization of DMOP (see Section 3). DOLCE had a major effect on reasoner performance (10 times longer with DOLCE), using the `ObjectInverseOf()`, reduced the time by about a third, and there is room for improvement of the modularization tools.

2 Overview of DMOP's contents

The core concepts of DMOP (Fig. 1) are those essential for the data mining process (DM-Process). The input consists of a task specification (DM-Task) and training/test data (DM-Data); the output is a hypothesis (DM-Hypothesis), in the form of a global model (DM-Model) or a set of local patterns (DM-PatternSet). A DM-Task specifies a DM-process (or any part thereof) in terms of its input/output; a DM-Algorithm is the specification that addresses a DM-Task. A DM-Operator is a program that implements a given DM-Algorithm and instances of DM-Task and DM-Algorithm specify their input/output types; note that only processes, called DM-Operations, have actual inputs and outputs. A process that executes a DM-Operator also realizes the DM-Algorithm implemented by the operator and achieves the DM-Task addressed by the algorithm.

Each main class has many subclasses that are richly axiomatised. Data is the primary resource that feeds the knowledge discovery process, and their characteristics are

represented in DMOP. Most characteristics concern statistical measures (e.g., the number of instances of a data set) or the absolute or relative frequency of a categorical feature value, and others are information-theoretic measures. The right-hand side of Fig. 1 shows a small sample of characteristics associated with one of the Data subclasses.

The top levels of the algorithm hierarchy match those of the task hierarchy, since each algorithm class is defined by the task it addresses, but the overall hierarchy is much deeper, since there is an often dense subhierarchy of algorithms that specify diverse ways of addressing each task. DMOP contains in-depth knowledge of algorithms as expressed in their elaborate network of object properties (see [2] for additional details). Among others: has-quality relates a DM-Algorithm to an AlgorithmCharacteristic, such as LearningPolicy (Eager/Lazy) and ToleranceToClassImbalance.

DMOP v5.4 has 758 classes, 169 object properties, 15 data properties and 4584 axioms. It uses almost all of the OWL 2 DL features. DMOP has 100 object sub-property axioms, 110 equivalent classes axioms, qualified cardinality constraints, many properties have domain and range axioms declared (155 and 154 axioms, respectively) and appear often in class descriptions, there are 4 transitive object properties and 45 with inverse, and an `owl:import` for obsolete aspects. In addition, it uses punning extensively and has 9 property chains that are comprehensive with domain and range axioms and they link different ontology branches; as with punning, they cannot really be readily removed, for they contribute to deriving a substantial amount of knowledge. DMOP uses the weak form of punning available in OWL 2 to solve the modelling problem with algorithms and input/output objects [8]. It is applied only to leaf-level classes of IO-Object. Non-leaf classes are not punned, but represented by associated meta-classes; e.g., the IO-Object subclass of DataSet maps to the IO-Class subclass of DataSetClass. Another example is that the instances of the DM-Hypothesis class, which represents hypotheses generated by running an algorithm on the particular dataset, and the instances of its associated meta-class DM-HypothesisClass are the leaf-level descendant classes of DM-Hypothesis. Finally, there are many relations between all the branches in the class hierarchy.

3 Brief reasoner evaluation

We outline the four tests conducted and the main motivation behind them. First, we have anecdotal evidence of the time it takes to classify the ontology, which varied by machine and version, ranging from about 10 minutes for v5.2 to over 20 minutes for v5.2-v5.3. The first step is to obtain actual statistics for the current version in an ODE (Protégé).

A substantial part of the OWL DL-formalised highly axiomatised DOLCE [9] with ExtendedDnS (from the DLP397 archive) was extracted and merged with DMOP, adding to DMOP 43 classes, 78 object properties, and 593 axioms. It may be that the relatively slow performance is due to the features used in DMOP or a by-product from linking to the DOLCE fragment. We will compare the 'full DMOP' v5.4 with performance obtained with 'DMOP without DOLCE'. If there is little difference, then the long time is due to the rich axiomatization in DMOP itself.

Third, we hypothesise that new OWL 2 features may contribute to the performance. $\mathcal{SROIQ}$—and following from that, OWL 2 and Protégé 4.x too—has a function Inv on

roles, that can be used directly in the axiom. That is, instead of only `InverseObjectProperties(OPE1 OPE2)` for two object properties in the ontology, such as addresses with as inverse addressed by, one now can use `ObjectInverseOf(OP)`, and have only addresses whilst using (in Protegé notation) inverse(addresses) in an axiom (instead of extending the vocabulary with addressed by). Both options have been used, but for consistency of modelling, the former was chosen eventually. This test compares the baseline with a modified DMOP-with-inverseOf. If performance is worse, then the algorithms for the new feature may have room for improvement; if better, then that new feature of OWL 2 is an improvement over OWL 1.

Fourth, modularization is used for, among others, divide-and-conquer reasoning and collaborative ontology development, and it thus could be useful in DMOP reasoning and maintenance. There are several tools (discussed below), and those algorithms only work i) to find isolated subsections or ii) breaking it up into main branches or iii) make a profile out of it. It is known that DOLCE does not modularise well with the extant tools, because it is so dense across branches, which suggests it might not work well with DMOP v5.4 either.
Data of the experiments are available at `www.meteck.org/files/ORE14DMOP.zip`.

Performance To find a baseline on a general good machine, DMOP v5.4 was loaded in Protégé v4.2 and we measured the time for classifying all components (classes, properties, individuals) of the ontology. A MacBook Pro, MAC OS X v10.6.8, CPU 3.06 GHz Intel core 2 Duo, RAM 4 GB 1067 MHz DDR3 was used for the purpose. Once loaded, HermiT v1.3.8 was selected and started (for FACT++, an error message "Reasoner Died"' was returned). The classification times by component are included in Table 1, first column.

Table 1. Classification times (in minutes) of DMOP and DMOP with `ObjectInverseOf()`.

Component of classification	**DMOP v5.4**	**DMOP v5.4 inverses**
Class Hierarchy	6 mins	3 mins
Object Property Hierarchy	2 mins	1 min
Data Type Property Hierarchy	<1 min	few secs
Class instances	about 1 min	<1 min

As regards the third test, given the current version of DMOP, for each object property having a corresponding inverse object property within the vocabulary (n=45), we removed the one having minor usage and we used the inverse(propertyName) in Protégé. When the removed property was used, we adjusted the ontology in agreement of the usages of the removed object property. Information regarding the time for classifying all components on the current DMOP version, computed as for the baseline, are shown in Table 1, second column. These results show that the `ObjectInverseOf()` feature of OWL 2 improves the reasoner performance in the ODE by at least a third.

DMOP with and without DOLCE All of the DOLCE terms (ExtendedDnS that imports DOLCE-lite) that had been merged into DMOP were removed except three object

Table 2. Classification times (in minutes) of DMOP with and without DOLCE.

Ontology version	HermiT	Pellet	TrOWL
DMOP	1.3180	9.22375	0.14555
DMOP with DOLCELite&ExtendedDnS fragment	14.5005	timeout (over 8 hrs)	0.2692

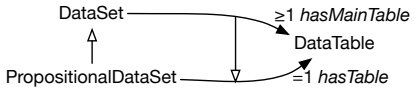

Fig. 2. Illustration of the problem of wrongly unsatisfiable class PropositionalDataSet.

properties that are widely used in DMOP's axioms: DOLCE-Lite:has-quale, DOLCE-Lite:has-quality, and DOLCE-Lite:inherent-in. Note, that since these properties were not linked to any other DOLCE entities anymore (all of which had been removed) they could be treated equivalently to as being local in DMOP. The tests were performed within Protege v4.3 with MacBook Air with CPU 1.86 GHz Intel Core 2 Duo, RAM 2GB 1067 MHz DDR3, running Mac OS X v10.6.8. The tested reasoners included: FaCT++ (v1.6.2), HermiT (v1.3.8), Pellet (v2.2.0), MORe (v0.1.5 HermiT/JFact/Experimental), TrOWL (v1.4).

The classification times are reported in Table 2; FaCT++ and MORe returned a "Reasoner Died" message and therefore they are not included in the table. The results show that HermiT was more than 10 times faster when classifying DMOP without DOLCE than when it classified DMOP with DOLCE. Pellet did not finish classifying DMOP with DOLCE and DnS in 8 hours, while it classified DMOP alone in less than 10 minutes.

Both Pellet and TrOWL inferred inconsistent classes. In the case of Pellet, the inconsistencies concern the use of datatypes. For instance, the range of DMOP:hasDataValue property is `xsd:anyType`, but in several axioms, more specific types are used, such as ModelParameterCount $\sqsubseteq$ =1 hasData$\mathrm{Value}_{\texttt{xsd:nonNegativeInteger}}$. Such axioms are correct, for `xsd:nonNegativeInteger` is a specialization of `xsd:anyType`, but nevertheless Pellet returns an inconsistency. In case of TrOWL, the wrongly computed unsatisfiable class concerns the use of role hierarchies, which is illustrated in Fig. 2, where domain and range of both properties are declared as DataSet and DataTable, respectively, which are non-disjoint sibling classes subsumed by DM-Data, and PropositionalDataSet $\equiv$ DataSet $\sqcap$ =1 hasTable.DataTable. According to the explanation, PropositionalDataSet is inconsistent due to DataSet $\sqsubseteq \exists$hasTable.DataTable and hasMainTable $\sqsubseteq$ hasTable. Although TrOWL is an approximate reasoner, unsatisfiability should not have been inferred: PropositionalDataSet can be kept satisfiable by deriving hasMainTable $\sqsubseteq$ hasTable and DataSet $\sqsubseteq$ =1 hasMainTable.DataTable; HermiT did not derive anything. From a modelling viewpoint, one would expect a derivation due to the subsumptions in opposite directions (see also *SubProS* in [10]) and the cardinality constraints (the representation will be revised anyway).

Modularization We modularize the DMOP ontology using the extraction functionality in Protégé v4.3. SWOOP [11], OWL API locality module extractor [12] and OWL Module extractor (`http://mowl-power.cs.man.ac.uk:8080/modularity/`) were ex-

Table 3. Ontology metrics for DMOP and its related modules. #X represents the number of each type of entity.

Ontology	**#Classes**	**#OPs**	**#DPs**	**#Individuals**	**#Axioms**
DMOP.owl	758	169	15	459	4584
DMOP-profile-EL.owl	758	169	15	459	4214
DMOP-branch-Endurant.owl	512	169	15	459	3409
DMOP-branch-Perdurant.owl	19	169	15	459	1376
DMOP-branch-Abstract-Quality.owl	231	169	15	459	2131
DMOP-branch-Toplevel.owl	44	169	15	459	1428
DMOP-branch-Merge.owl	758	169	15	459	4359

perimented with, but they created modules that were too large because entities within the DMOP ontology have many dependencies between them; there were no isolated branches of the ontology that could be modularized. We use the 'axioms by reference' method in Protégé to select entities from an ontology to copy, move or delete. The copy and move function returned an error, therefore we used the delete option to remove unwanted entities from the module.This resulted in four branch modules, a module of wholly-present entities (DMOP-Branch-Endurant), a module of entities that unfold in time (DMOP-Branch-Perdurant), a module of entities that exist in neither space nor time and property related entities (DMOP-Branch-Abstract-Quality), and a module that has only DMOP's top-level entities (DMOP-Branch-Toplevel). Thereafter, we merged the axioms of the branch modules (DMOP-Branch-Merge) in order to compare it to the original DMOP ontology. We also created an EL profile (DMOP-Profile-EL).

Table 3 displays a comparison of metrics for the DMOP modules. Note that merging the branches does not result in the original ontology, as can be observed from the difference in number of axioms in the merged ontology cf. the original DMOP. It is apparent that the module extraction feature in Protégé extracts classes in isolation and includes the object properties, data properties, and individuals of the original ontology, including those that do not relate to the classes in the module. For instance, in the DMOP-Branch-Perdurant.owl module, the object property, solves exists. This has no dependencies to the classes in the module and should not be present. The modularity tools could be improved by taking into consideration other entity types that reference the selected entity and also by allowing the user the choice of relaxing on logical principles such as completeness, or by preserving the ontology's knowledge by linking between modules.

4 Conclusions

The paper described the subject domain contents and some OWL 2 DL features used in the richly axiomatised Data Mining OPtimization ontology. We explored the effect of such modelling features on the performance of automated reasoners. Removing most of DOLCE reduced the classification times tenfold, using the `ObjectInverseOf` reduced the time by about a third, and there is room for improvement of the modularization tools.

Acknowledgements. We thank the contributors to DMOP: H. Do, S. Fischer, D. Gamberger, M. Hilario, L. Al-Jadir, S. Jupp, A. Kalousis, P. Kralj Novak, B. Mougouie, P. Nguyen, R. Palma, R. Stevens, A. Vavpetic, J. Wang, D. Wijaya, A. Woznica.

References

1. Zhou, Y., Young, J.A., Santrosyan, A., Chen, K., Yan, S.F., Winzeler, E.A.: In silico gene function prediction using ontology-based pattern identification. Bioinformatics **21**(7) (2005) 1237–1245
2. Hilario, M., Nguyen, P., Do, H., Woznica, A., Kalousis, A.: Ontology-based meta-mining of knowledge discovery workflows. In: Meta-Learning in Computational Intelligence. Volume 358 of Studies in Computational Intelligence. Springer (2011) 273–315
3. Shearer, C.: The CRISP-DM model: The new blueprint for data mining. Journal of Data Warehousing **5**(4) (2000) 13–22
4. Diamantini, C., Potena, D., Storti, E.: Supporting users in KDD processes design: a semantic similarity matching approach. In: Proc. of the Planning to Learn Works. (2010) 27–34–134
5. Záková, M., Kremen, P., Zelezný, F., Lavrac, N.: Automating knowledge discovery workflow composition through ontology-based planning. IEEE Trans. Automation Science & Engineering **8**(2) (2011) 253–264
6. Kietz, J., Serban, F., Bernstein, A., Fischer, S.: Data mining workflow templates for intelligent discovery assistance and auto-experimentation. In: Proc of the ECML/PKDD'10 Workshop on Third Generation Data Mining (SoKD'10). (2010) 1–12
7. Panov, P., Dzeroski, S., Soldatova, L.N.: OntoDM: An ontology of data mining. In: ICDM Workshops, IEEE Computer Society (2008) 752–760
8. Keet, C.M., Lawrynowicz, A., d'Amato, C., Hilario, M.: Modeling issues, choices in the data mining optimization ontology. In Rodriguez-Muro, M., et al., eds.: OWLED. Volume 1080 of CEUR Workshop Proceedings., CEUR-WS.org (2013)
9. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: Ontology library. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003). (2003) http://wonderweb.semanticweb.org.
10. Keet, C.M.: Detecting and revising flaws in OWL object property expressions. In ten Teije, A., et al., eds.: Proc. of EKAW'12. Volume 7603 of LNAI., Springer (2012) 252–266
11. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca Grau, B., Hendler, J.A.: Swoop: A web ontology editing browser. Journal of Web Semantics **4**(2) (2006) 144–153
12. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research (JAIR) **31** (2008) 273–318