ECML PKDD 2009
European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases

THIRD GENERATION DATA MINING:

TOWARDS SERVICE-ORIENTED KNOWLEDGE DISCOVERY

SoKD'09

September 7, 2009

Bled, Slovenia

Editors:

Vid Podpečan
Nada Lavrač
Jožef Stefan Institute, Ljubljana, Slovenia

Joost N. Kok
Jeroen de Bruin
LIACS, Leiden University, Leiden, The Netherlands

Preface

A major challenge for third generation data mining and knowledge discovery systems is the integration of different data/knowledge resources (which are highly diverse in nature in terms of representation and data formats) and computer systems (tools for data integration, data mining and knowledge discovery) which are distributed across the network. While the first generation data mining systems supported a single algorithm or a small collection of algorithms that are designed to mine attribute-valued data, today's second generation systems are characterized by supporting high performance interfaces to databases and data warehouses and by providing increased scalability and increased functionality; for example, second generation systems can mine larger and more complex data sets and provide increased flexibility by supporting a data mining schema and a data mining query language.

The emerging third generation data mining and knowledge discovery systems should be able to mine distributed and highly heterogeneous data found on intranets/extranets/grid and integrate efficiently with operational data/knowledge management and data mining systems. The key technologies which will make third generation data mining and knowledge discovery possible is to provide meta-data (semantic annotations) of different information resources (data, human-coded knowledge, and machine-induced patterns and predictive models) and data mining and knowledge discovery systems (pattern mining and model discovery tools) and implementation of data mining and knowledge discovery tools as services available on the web. Such service-oriented data mining and knowledge discovery systems will enable meta-level search of data/knowledge resources and systems, enabling the construction of knowledge discovery workflows (representing potentially repeatable sequences of data mining and data integration steps), resulting in improved pattern and model discovery.

Papers presented at the workshop addressed the following main topics: *service-oriented knowledge discovery*, *data mining services*, *data mining ontologies*, *ontologies as background knowledge used for learning*, *information fusion*, *dealing with heterogenous data sources*, and *data mining workflows*.

Ljubljana, Leiden, August 2009

Vid Podpečan
Nada Lavrač
Joost N. Kok
Jeroen de Bruin

Workshop Organization

Workshop Chairs

Vid Podpečan (Jožef Stefan Institute)
Nada Lavrač (Jožef Stefan Institute)
Joost N. Kok (Leiden University)
Jeroen de Bruin (Leiden University)

Program Committee

Vid Podpečan
Nada Lavrač
Joost N. Kok
Jeroen de Bruin
Michael Berthold
Igor Mozetič
Filip Železny
Hendrik Blockeel
Melanie Hilario

Additional Reviewers

Darko Čerepnalkoski
Joaquin Vanschoren
Celine Vens
Fabian Guiza

Table of Contents

Towards Cooperative Planning of Data Mining Workflows 1
Jörg-Uwe Kietz, Floarea Serban, Abraham Bernstein, and Simon Fischer

KDDONTO: An Ontology for Discovery and Composition of KDD Algorithms . . 13
Claudia Diamantini, Domenico Potena, and Emanuele Storti

Towards a Service-Oriented Knowledge Discovery Platform 25
Vid Podpečan, Matjaž Juršič, Monika Žáková, and Nada Lavrač

Advancing Data Mining Workflow Construction: A Framework and Cases using the Orange Toolkit 39
Monika Žáková, Vid Podpečan, Filip Železný, and Nada Lavrač

The Fantom Service for Subgroup Discovery in Score Lists 52
Jeroen de Bruin, Nada Lavrač, and Joost N. Kok

Definition of a Metadata Schema for Describing Data Preparation Tasks 64
Miguel Hidalgo, Ernestina Menasalvas, and Santiago Eibe

A Data Mining Ontology for Algorithm Selection and Meta-Mining 76
Melanie Hilario, Alexandros Kalousis, Phong Nguyen, and Adam Woznica

Stand on the Shoulders of Giants: Towards a Portal for Collaborative Experimentation in Data Mining 88
Joaquin Vanschoren and Hendrik Blockeel

Towards Service-Oriented Knowledge Discovery in Biomedical Research 100
Tu Bao Ho, Katsuhiko Takabayashi, Pham Tho Hoan, Nguyen Thanh Phuong, Saori Kawasaki, and Tran Dang Hung

OntoDM: Towards an Ontology of Data Mining Investigations (Ext. Abstract) . . . 114
Pance Panov, Larisa N. Soldatova, and Sašo Džeroski

Towards Cooperative Planning of Data Mining Workflows

Jörg-Uwe Kietz[1], Floarea Serban[1], Abraham Bernstein[1], and Simon Fischer[2]

[1] University of Zurich, Department of Informatics,
Dynamic and Distributed Information Systems Group,
Binzmühlestrasse 14, CH-8050 Zurich, Switzerland
{kietz|serban|bernstein}@ifi.uzh.ch
[2] Rapid-I GmbH, Stockumer Str. 475, 44227 Dortmund, Germany
fischer@rapid-i.com

Abstract. A major challenge for third generation data mining and knowledge discovery systems is the integration of different data mining tools and services for data understanding, data integration, data preprocessing, data mining, evaluation and deployment, which are distributed across the network of computer systems. In this paper we outline how an intelligent assistant that is intended to support end-users in the difficult and time consuming task of designing KDD-Workflows out of these distributed services can be built. The assistant should support the user in checking the correctness of workflows, understanding the goals behind given workflows, enumeration of AI planner generated workflow completions, storage, retrieval, adaptation and repair of previous workflows. It should also be an open easy extendable system. This is reached by basing the system on a data mining ontology (DMO) in which all the services (operators) together with their in-/output, pre-/postconditions are described. This description is compatible with OWL-S and new operators can be added importing their OWL-S specification and classifying it into the operator ontology.

1 Introduction

In the early days of data mining the biggest challenge for data miners was finding the right algorithm for their task. Todays typical 2nd generation KDD Support Systems (KDDSS) overcome these issues by providing a plethora of operators. The commercial systems such as SAS Enterprise Miner and SPSS Clementine or the Open-Source Systems RapidMiner and MiningMart have 100+ (RapidMiner which includes WEKA even 500+) different operators to support modeling the KDD process. These KDDSS have eased the problem of providing access to applicable data mining operators. They do, however, give rise to a new problem: *How can data miners navigate the multitude of data mining operators to construct a valid and applicable data mining process?*

Indeed this problem gets even aggravated. When considering the whole KDD process [7] the universe of possible combinations is enormous. Consider CRISP-DM [4]: It consists of 6 phases and 24 tasks. Granted 6 of the tasks are of an organizational nature, but the remaining 18 technical tasks open a huge design space [16] of possible KDD processes. As a consequence, most users are overwhelmed with the decisions they have to face and only explore a small part of the possible design space.

In this paper we propose that one should use a *cooperative planning* approach for designing the KDD process. A third generation KDDSS should employ a mixed initiative planning [8] approach to user interaction simplifying the following tasks:

– Effective representation of hundreds of operators used in KDD-workflows.
– Checking the correctness of KDD workflows
– Enumeration of (partial) KDD workflows
– Retrieval of previous (partial) KDD workflows
– Understanding and explanation of given KDD workflows
– Adaptation and repair of KDD workflows

In this paper we focus on the base of such a cooperative KDDSS. Specifically, we present a Data Mining Ontology (DMO) able to effectively organize hundreds of operators, which is the base for checking the correctness of KDD workflows and an HTN based planning component able to effectively enumerate useful KDD-workflows[3]. This is the base for our future work concerning a KDD-workflow explanation component and a KDD process adaptation component, which can be seen as a case-based planning component assuming that a case base is available.

In the remainder of the paper we first outline related work, and then discuss DMO and HTN in detail.

2 Previous Work

Several attempts have been made to create KDD support systems but none of them provides full support for generation of KDD workflows.

Žáková et. al [18] tried to automatically generate workflows using a knowledge ontology (DM ontology) and a planning algorithm based on the FastForward (FF) system. They limit themselves only to return the optimal workflow with the smallest number of processing steps, any other alternative workflows are not generated. The system does not involve user interaction during workflow generation.

The IDEA system [2] consists of an Intelligent Discovery Assistant (IDA) which provides users with systematic enumerations of valid DM processes and effective process rankings by different criteria (speed, accuracy, etc.). It is based on an ontology of DM operators that guides the workflow composition and contains heuristics for the ranking of different alternatives. The user is guided in her choices by choosing weights to trade-off the rankings of the alternatives along

[3] This is a report about work in progress. Check http://www.e-lico.eu/eProPlan to see the current state of the ontology as well as to download the Protege plug-ins we released so far.

the different dimensions (e.g., speed, accuracy, comprehensiblity) of her desiderata. The rankings are based on heuristics contained in the ontology as well as auto-experimentation.
The NExT system [1] is an extension of IDEA using Semantic Web technology Specifically it employs an OWL-S [11] ontology to specify the DM operators and relies on XPlan [10] for planning. While IDEA and NExT provide the user with a number of alternatives and guide the choice among them they are limited to proposing simple, non-splitting process flows, unless specific splitting templates are provided.

MiningMart [12] is a KDD preprocessing tool specialized in data mining on large data stored in relational databases. The meta-model M4 is used to store data on two levels: the logic level – describes the database schema and allows consistent access to information and the conceptual (ontology) level – uses concepts with features and relationships to model data [6]. The ontology captures all the data preprocessing therefore gives a better understanding and reuse of the data. However this meta-model is expressed in UML/XML and not in an ontology language. The system lacks automatic workflow creation of DM processes but enables the reuse of preprocessing phases across applications through case-based reasoning.

The CITRUS project [17] consists of an IDA which offers user-guidance through mostly a manual process of building the workflows. It uses planning for plan decomposition and refinement. The system is based on an extension of SPSS Clementine, which provides a visual interface to construct DM processes manually.

3 A Data Mining Ontology (DMO) for Planning

We designed a Data Mining Ontology (DMO) to contain all the information necessary to support a 3rd generation KDDSS. As Figure 1 shows (left to right), the DMO contains information about the *objects manipulated* by the KDDSS (*I/O-Object*), the *Meta Data* needed, the *operators* (i.e., algorithms) used by the tool, and a *goal description* that formalizes the user's desiderata. Here we succinctly describe the *I/O-Object* and the Meta Data before providing a slightly more extensive discussion of the Goals and Operators in the following subsections.

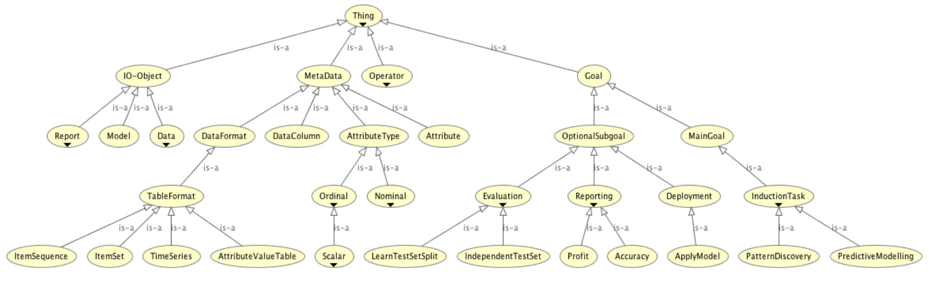


Fig. 1. The Upper-Structure of the Data Mining Ontology (simplified)

*I/O-Object*s are everything that is used or produced by operators. Every *I/O-Object* produced by an operator can be used as an input to any operator occurring at some later stage of the KDD workflow. Examples of *I/O-Object*s are *Data* (used and produced), *Model*s (used and produced) and *Report*s (produced only).

All of the above have several sub-concepts to specify the description of operator inputs and outputs I/O in a more specific fashion and to be used as conditions and effects of operators.

Sub-concepts of *Data* could be *AttributeValueDataTable*, *MissingValueFreeData*, *ScalarDataTable*, *NominalDataTable*, *TimeSeries*, *UnivariateTimeSeries*, *MultivariateTimeSeries*, etc. Prediction models, delivered by data mining operators are, e. g., *DecisionTree*, *Decisionlists*, *RuleSet*, *LinearModel*, ...

MetaData is used to describe *I/O-Object*s in more detail, i.e., the *DataFormat* such as tables, relations, images, etc. *DataColumns* can take specific value types like numerical or categorical values, and have particular roles such as “weight”, “id”, or “label”.

Furthermore, the meta data can contain aggregate information about the values found in this column. Examples are mean, variance, range, modal value, or a flag indicating whether there are missing values.

3.1 Specification of KDD goals and input

A planner requires a goal description consisting of a start state and a final (or goal) state. A KDD-Workflow final state is reached when the workflow solves the "Data Mining Goal" ([4], Sec. II.1.3) and also contains all the evaluation and reporting needed to let the user assess if it fulfills the "Success Criterion". Additionally, all conditions of the operators included in the workflow have to be fulfilled.

In the DMO we model goal descriptions using subclasses of *Goal* (see Figure 1 for a simplified extract.). When specifying the "Data Mining Goal" the user has to choose (or compose) a subclass of *MainGoal*. This main goal can be extended with compatible *OptionalSubGoal*s (which ones are compatible is modeled with object-properties in the DMO). For example, the user can specify "I would like a KDD-Workflow for *PredictiveModeling*, evaluated on an *IndependentTestSet*, where the performance is reported by a *Profit*-Chart."

Several of these Goals require (again modeled as an object-property) that input (data) is available. Namely, *PredictiveModeling* requires *trainingData* and *IndependentTestSet* requires *testData*. So the user is prompted to specify a data file to be analyzed and to extend it according the meta-data in the DMO.

3.2 Operator Ontology

Operators in KDD-workflows differ from operators usually used in planning in two very important aspects. First, they do produce new objects. Consequently, we face a (potentially) infinite planning domain and not a finite one as most previous work about planning assumes. Second, they only produce new things but never destroy old things. Thus, our world is monotonically growing with additional operator applications, i.e., everything that was valid (or known) in our world before executing an operator is still valid (known) afterwards. An infinite

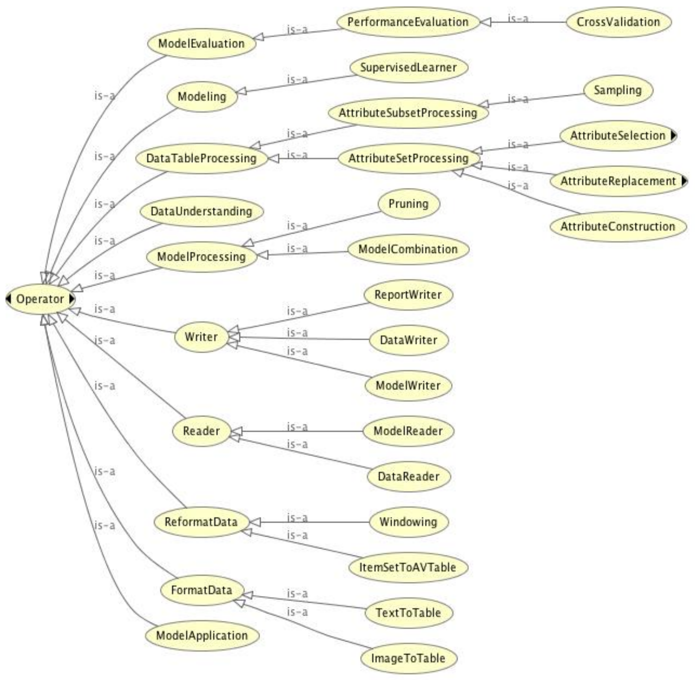


Fig. 2. The Top-Level Abstract Operators in the DMO

planning domain usually means that it is undecidable if a planning problem has a solution. Fortunately, the existence of plans is not really a concern for KDD-workflows. They always have a number of trivial solutions. E. g. for prediction tasks we could just compute the mean value or modal value of the target attribute, and build a model that always predicts that value, ignoring everything else.

Thus, we are not only looking for a plan, but rather for a good plan. The quality of a plan can be guided by the meta data information by which we enrich the objects exchanged by our operators. Note that the quality of a plan can be measured in various ways, but that some of the information is not available at plan-time. Especially not, the quality of a plan measured as expected predictive quality of a prediction model, e.g. its accuracy. This main success criterion of Data Mining is not available before executing the plan. Also, in real-world applications, we have to respect estimated non-functional constraints such as memory consumption and computation time of the workflow during planning.

As an example, consider the problem of cleaning a data set from missing values. Whereas a missing value imputation operation which replaces missing values by training a model for each column containing missing values, is usually the preferable method, this may be prohibitively expensive in terms of computation time when the number of columns is large. In that case, it may be better to go with the simpler solution of filling in missing values by using the column mean value. In order for a planner to be capable of taking these considerations into account, we annotate operators and I/O-Objects with the respective information.

"RapidMiner.ID3":

Superclass:	ClassificationLearning **and**
	(*uses* **exactly 1** *AttributeValueDataTable*) **and**
	(*produces* **exactly 1** *Model*) **and**
	(*simpleParameter1*(name="minimal_size_for_split") **exactly 1** integer) **and**
	(*simpleParameter2*(name="minimal_leaf_size") **exactly 1** integer) **and**
	(*simpleParameter3*(name="minimal_gain") **exactly 1** real)
Condition:	(*AttributeValueDataTable* **and** *MissingValueFreeData* **and**
	(*inputAttribute* **only** (*hasAttributeType* **only** *Categorial*)) **and**
	(*targetAttribute* **exactly 1** (*hasAttributeType* **only** *Categorial*))
	)(?D), noOfRecords(?D,?Size), ?P1 is ?Size / 100
	→ *uses*(**this**,?D), *simpleParameter2*(**this**,?P1)
Effect:	*uses*(**this**,?D), *hasFormat*(?D,?F), *inputAttribute*(?D,?IA),*targetAttribute*(?D,?TA),
	→ new(?M,?D), *DecisionTree*(?M), produces(**this**,?M), hasFormat(?M,?F)
	inputAttribute(?M,?IA),*predictedAttribute*(?M,?TA),

Fig. 3. A basic operator: RapidMiner.ID3

Operator in- and output and parameters. The behaviour of each Data Mining operator is controlled by a set of parameters usually specified as key-value-pairs for an operator. Such parameters can be quite simple (e.g. k in a k-fold cross validation), quite complex (such as a reference to a column in a *DataTable* on which the operator is supposed to operate or an SQL statement as a target expression for attribute construction), or sometimes even structured (like a list of parameters to be optimized by an optimization operator).

Such restrictions on in- and output and parameters of operators are specified in OWL-DL. Every input of any operator must be a sub-object-property of *uses*, every output of any operator must be a sub-object-property of *produces*, and every parameter of any operator must be a sub-data-property of *simpleParameter*. *uses* and *produces* have a domain restriction to *Operator* and a range restriction to *I/O-Object*. Figure 3 shows an example.

Operator conditions and effects. Operators may specify restrictions on their input. E. g., most data mining operators will only operate on data sets satisfying particular properties (e.g., having numerical or categorical attributes, or not containing any missing values).

The effects of operators depend on parameter values and on the operator's input. E. g., a discretization operator's output is equal to its input with the exception of a set of columns, which are specified by a parameter. In the output data set, these columns' value types will be changed to categorical.

Operator conditions and effects are described as rules expressed in the Semantic Web Rule Language (SWRL [9]). As an extension to SWRL we have the special object reference **this**, which is the operator instance to which the rule refers. Also, we added three new built-in predicates that allow the creation of new unique object IDs for the generated outputs in the operator effects. They are permitted in the consequent of effect-rules only. These are:

new(NewObject). Exactly one new unique object ID is generated and bound to the argument variable for each such literal (independent of how many solutions satisfy the premisses).

new(NewObject,OldObject). Exactly one new unique object ID is generated and bound to the argument variable for EVERY different OldObject computed by the preconditions (independently of how often this OldObject is part of a solution satisfying the premisses).

copy(NewObject,OldObject,Except). One new unique object ID is generated and bound to the argument variable for EVERY different OldObject computed by the preconditions (independently how often this OldObject is part of a solution satisfying the premisses), additionally all stored property facts involving the OldObject except those in Except are copied to the NewObject. Except is a conjunction of property literals where the OldObject must be one of the arguments.

The condition of an operator is a set of SWRL rules, where the antecedent specifies the condition and the consequent contains all the input- and parameter-properties of the operator. Effects are also sets of SWRL rules, they can contain all three new built-in predicates, must contains all the output-properties of the operator and may also create new meta-data to describe the new *I/O-Objects*. All variables that occur in the conclusion of a condition or effect rule must be either bound by the premisses of the effect rule or be returned as a NewObject by one of the above special literals. If there is more than one rule for conditions and or effects, the resulting condition/effect-rule is the union of all antecedent of the condition/effect-rules imply the union of all consequents of the condition/effect-rule.

Operator nesting. Operators can be nested, i.e., they can contain sub-workflows. E. g., a cross-validation can contain a data mining operator for generating a prediction model from a training set and an operator for applying the model on a test set. But usually, such a nesting is not restricted to single operators, but may involve complex workflows. Therefore we allow such operators to reference nested HTN tasks (see Section 4). Similar to the post-conditions specified for each operator, we specify conditions that the enclosing operator guarantees to and requires from its nested workflow.

Operator Inheritance & Subsumption Any operator not only inherits the IO restrictions from it's super classes (via normal ontological reasoning), it also inherits all conditions and effects from them (to be handled by our operator extension). If an operator has several rules for conditions or effects, the resulting condition- or effect-rule is the union of all antecedent → the union of all consequent. This leads to the following intuitive definition of an operator subsuming another:

1. the more special operator satisfies all input, output, and parameter restrictions of the more general operator,
2. the more general operator is applicable in all the situations, the more specific operator can be applied to, and
3. the more specific operator has at least all the effects of the more general one.

Formally, this can be captured by combining class subsumption ($\sqsubseteq_{DL}$) for I/O restrictions and instance reasoning with respect to a set of ontological axioms (a tbox) ($\models_{(tbox)}$) together with θ-subsumption for conditions and effects.

Definition 1 (Operator Subsumption). *An operator OP_1 subsumes another operator OP_2 (written $OP_1 \sqsubseteq_{OP} OP_2$), iff $OP_1 \sqsubseteq_{DL} OP_2$, and there exist a substitution θ_C for the condition and θ_E for the effects such that:*

– $antecedent(condition(OP_2)) \models_{(tbox)} \theta_C\, antecedent(condition(OP_1))$,
– $consequent(condition(OP_2)) \models_{(tbox)} \theta_C\, consequent(condition(OP_1))$
– $antecedent(effects(OP_2)) \models_{(tbox)} \theta_E\, antecedent(effects(OP_1))$, *and*
– $consequent(effects(OP_2)) \models_{(tbox)} \theta_E\, consequent(effects(OP_1))$.

3.3 Using OWL-S to import operators

The operators present in the DM ontology are of course limited to the existing operators sources (RapidMiner, Weka, etc.). As new operators will be developed our ontology must be extendible, to allow the users to use them for planning. Thus, we envisaged a way of extending the ontology with new operators based on a description of the operators in a language compatible with the specification from our ontology.

Each new operator has to be available as a Web service, described using a description language (WSDL) and needs to be semantically described in OWL-S [11]. I.e. it describes what it does in the *ServiceProfile*, how it works in the *ServiceModel* and how to access it in the *ServiceGrounding*. An operator can be added in the DM ontology if the given OWL-S description matches our operator description. From the OWL-S description we are mainly interested in the *ServiceProfile* and *ServiceGrounding*, while the *ServiceModel* does not concern our approach for now.

The *ServiceProfile* describes what the service does, it is similar to our *Operator* concept from the DM ontology. Both the ontology and OWL-S have to specify the Inputs, Outputs, Preconditions and Effects. The Preconditions and Effects should be specified in SWRL to be compatible to our representation from the ontology. The Inputs and Outputs need to be described by using the concepts from our ontology.

ServiceGrounding specifies how the service can be accessed and executed. It contains the WSDL location of a given operator that can be stored in our ontology as an annotation of the operator to be passed to an execution engine.

Therefore enhacing the operators with OWL-S facilitates the process of importing new operators thus allowing a flexible and standardized way to enrich the DM ontology.

4 An HTN for Data Mining

Hierarchical Task Network planning (HTN) [14, 13] originates from more than 30 years ago. It provides a powerful planning strategy using hierarchical abstraction and by that is able to handle large and complicated real world domains [13]. Also it is more expressive than classical planning being able to express undecidable problems, and therefore it is also[4] undecidable, if a plan exists in general [13].

Recently, AI planning techniques have been proposed as a way to automate (totally or partially) workflow composition especially Web services composition [15, 10]. Several planning techniques were compared in the context of Web services composition and as a conclusion HTN planning performs best in automation of Web services composition [3].

For us an HTN (similar to [5, 13]) consists of the following:

– A set of available *task*s to achieve the possible *Goal*s.
– Each *task* has an I/O specification (a list of property - ?variable : class) and a set of associated *method*s (that share the I/O specification of the task) that can execute it.
– Each *method* has a
  condition restricting it's possible applications, and a
  contribution specifying which *Goal*s it reaches, and a
  decomposition into a sequence of (sub-)*task*s and/or *operator*s, that - executed in that order - achieve the task.

Therefore an HTN planning problem consists of decomposing the initial task using applicable method that contribute to the current goals into applicable operators and sub-tasks and then recursively decompose each sub-task until we obtain a sequence of applicable operators only.

Fig.4 shows an example HTN to illustrate how one could specify the generation of KDD-workflows.

Even this simple HTN already contains some special built-in *task*s, namely the getPlan/reapplyPlan and the chooseOp/applyOp pairs of *task*s. It also illustrates the use of negation as failure (unknown) available in conditions. getPlan normaly plans the *task* it gets as it's first argument, it only additionally returns the plan it generated for that *task*, such that it can be reused later again. If the training data need a transformation before modeling this transformation needs

[4] The introduction of new objects, leads to potentially infinite - and therefore undecidable - planning problems (Sec.3.2).

Task: DoDatamining
I/O: [*goal* - ?G:*UserGoal*].
Method: "Propositional predictive Data Mining of attribute value data".
condition: *trainingData*(?G,?RTD), *AttributeValueDataTable*(?RTD),
unknown(*subgoal*(?G, ?E), *LearnTestSetSplit*(?E))
contribution: *PredictiveModeling*(?G).
decomposition:
ReadData([produces - ?RTD]),
chooseOp(*SupervisedLearner*([in - ?RTD]),?G,?ModelOp,?Requirements),
getPlan(TransformData([in - ?RTD, out - ?TD, goal -?Requirements]),?DataTransformationPlan),
applyOp(?ModelOp,[uses -?TD, produces - ?M]),
ReadData([produces - ?RED]),
OptionalEvaluation([goal - ?G, *model* - ?M, preprocessing - ?DataTransformationPlan]),
OptionalApplication([goal - ?G, *model* - ?M, preprocessing - ?DataTransformationPlan]).
Method:
...
Task: OptionalEvaluation
I/O: [*goal* - ?G:*UserGoal*, *model* - ?M:*Model*, preprocessing - ?DataTransformationPlan:Plan]
Method: "No evaluation wanted"
condition: unknown(*subgoal*(?G, ?E), *Evaluation*(?E))
contribution: .
decomposition: .
Method: "Evaluation of model performance on independent testdata"
condition: *subgoal*(?G, ?E), *testData*(?E,?RED), *AttributeValueDataTable*(?RED).
contribution: *subgoal*(?G, ?E), *IndependentTestSet*(?E).
decomposition:
reapplyPlan([in- ?RED, out - ?ED],?DataTransformationPlan]),
EvaluateModel([data - ?ED, model - ?M, produces - ?E),
GenerateReport([uses - ?E]).

Fig. 4. A simple extract from our HTN under development

to be reapplied (using reapplyPlan) on evaluation and application data, before the learned model can be applied to them as well. This is not just a simple "do the same again". The data transformation generated by the planner may involve operations like "feature selection", "Discretisation", "random sampling" that would behave differently on different data. Therefore the reapplication of plans must adapt the plans to corresponding dual operations "Select the same features", "Discretize into same Bins" "don't sample", i.e. all data dependent operations have to store their choices into "preprocessing models" and must have a dual re-application operator (specified in the ontology). HTN planning does straight forward planning, i.e. when an operator is applied all its input is already produced and available. However, in KDD-workflows selection of the modeling operator is usually done before preprocessing to set up the goals of preprocessing. This is achieved with the chooseOp/applyOp pair. chooseOp selects[5] an operator from the operator class specified in its first argument just as normal planning does, but opposed to planning it doesn't test applicability nor does it execute the effects, it only returns the conditions and the chosen operator. applyOp is just a meta-call that allows the operator to be a variable.

[5] Currently just by enumeration, maybe later with some heuristic guidance

5 Conclusions

In this paper we presented the basis of an open system for cooperative planning of KDD-Workflows. The system is currently developed within the e-LICO project as a set of Protege 4 plugins. The plugins (goal editor, condition/effect editor, OWL-S im-/export, restricted abox/rule reasoning, operator subsumption, workflow-checking and an HTN-planner) as well as the DMO are to be released to the public within this year.

We expect HTN-planning to be more successful in generating useful KDD-workflows, in the presence of a realistic (high) amount of operators, than forward planning tried in previous work so far, as the HTN grammar rules allow additional expert knowledge to be coded into the plan generation process, which is not possible in simple STRIPS-like forward planning. Nevertheless, we expect the most gain in user-productivity of the 1st version described here lies in the correctness-checking of (user-designed) KDD-workflows at design-time – checking for slight errors in operator applicability that are already detected before execution and will not crash the workflow execution (after some hours).

Planning of KDD-workflows is always limited by the information available at design/plan time. However designing optimal performing KDD-workflows is still a highly creative and interactive process, where most insights and design decisions are based on and revised according to the performance measured in the evaluation-phase of the KDD-workflows, i.e. with information available after execution and not at plan-time.

Data-Mining-Performance optimized KDD-workflows resulting from this expensive iterative design cycle are, therefore, very valuable and should be recorded in a case-base. Especially as the current knowledge on data mining does not allow to distinguish good from bad performing workflows without executing them. For future work we plan to extend the system to case-based planning, i.e. retrieval and adaptation of partial fitting KDD-workflows. This will allow to make this valuable and costly acquired knowledge available to later workflow design.

Acknowledgements: This work is partially supported by the European Community 7^{th} framework program ICT-2007.4.4 under grant number 231519 "e-Lico: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science". The DMO described in this paper is the result of ongoing collaborative work within the e-LICO project of Ingo Mierswa, Melanie Hilario, Alexandros Kalousis, Nguyen Phong and the authors. For ease of presentation we made some simplifications with respect to the full e-Lico DMO.

References

1. A. Bernstein and M. Daenzer. The NExT System: Towards True Dynamic Adaptions of Semantic Web Service Compositions (System Description). In *Proceedings of the 4th European Semantic Web Conference (ESWC '07)*. Springer, March 2007.
2. A. Bernstein, F. Provost, and S. Hill. Towards Intelligent Assistance for a Data Mining Process: An Ontology-based Approach for Cost-sensitive Classification.

IEEE Transactions on Knowledge and Data Engineering, 17(4):503–518, April 2005.
3. K. S. M. Chan, J. Bishop, and L. Baresi. Survey and comparison of planning techniques for web services composition. Technical report, Univ. of Pretoria, 2007.
4. P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. Crisp–dm 1.0: Step-by-step data mining guide. Technical report, The CRISP–DM Consortium, 2000.
5. K. Erol, J. Hendler, and D. Nau. HTN planning: Complexity and expressivity. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1123–1123. JOHN WILEY & SONS LTD, 1995.
6. T. Euler and M. Scholz. Using Ontologies in a KDD Workbench. In P. Buitelaar, J. Franke, M. Grobelnik, G. Paa?, and V. Svatek, editors, *Workshop on Knowledge Discovery and Ontologies at ECML/PKDD '04*, pages 103–108, 2004.
7. U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34, 1996.
8. G. Ferguson, J. Allen, and B. Miller. Trains-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 70–77. AAAI Press, 1996.
9. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. http://www.w3.org/Submission/SWRL/, 2004.
10. M. Klusch, A. Gerber, and M. Schmidt. Semantic Web Service Composition Planning with OWLS-XPlan. In *Proceedings of the 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web*. AAAI Press, 2005.
11. D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. *OWL-S: Semantic Markup for Web Services*. http://www.w3.org/Submission/OWL-S/, 2004.
12. K. Morik and M. Scholz. The MiningMart Approach to Knowledge Discovery in Databases. In N. Zhong and J. Liu, editors, *Intelligent Technologies for Information Analysis*, pages 47–65. Springer, 2004.
13. D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2004.
14. D. Nau, S. Smith, and K. Erol. Control strategies in HTN planning: Theory versus practice. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1127–1133. JOHN WILEY & SONS LTD, 1998.
15. E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. HTN planning for web service composition using SHOP2. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):377–396, 2004.
16. K. T. Ulrich and S. D. Eppinger. *Product Design and Development*. McGraw-Hill, New York, 3rd rev. ed. edition, 2003.
17. R. Wirth, C. Shearer, U. Grimmer, T. P. Reinartz, J. Schlösser, C. Breitner, R. Engels, and G. Lindner. Towards process-oriented tool support for knowledge discovery in databases. In *PKDD '97: Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 243–253, London, UK, 1997. Springer-Verlag.
18. M. Žáková, P. Křemen, F. Železný, and N. Lavrač. Planning to learn with a knowledge discovery ontology. In *Planning to Learn Workshop (PlanLearn 2008) at ICML 2008*, 2008.

KDDONTO: An Ontology for Discovery and Composition of KDD Algorithms

Claudia Diamantini, Domenico Potena and Emanuele Storti

Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione "M. Panti",
Università Politecnica delle Marche - via Brecce Bianche, 60131 Ancona, Italy
{diamantini,potena,storti}@diiga.univpm.it

Abstract. Knowledge Discovery in Databases (KDD) is a highly complex process, where a lot of tools are needed to achieve the discovery goal. It implies that an user has both to choose the algorithms more suitable to her goal, and to compose them for designing a process. In order to support the user, in this paper we introduce KDDONTO, an ontology formalizing the domain of KDD algorithms. For the design of KDDONTO we follow a formal ontology building methodology aimed to define goal-oriented ontologies satisfying quality requirements. We first identify the basic terms characterizing algorithms, and analyzing them we formally derive classes and relations of the ontology. Finally, an OWL-DL implementation is proposed and its evaluation is discussed.

1 Introduction

Knowledge Discovery in Databases (KDD) is the research field studying advanced techniques and methodologies aimed at the extraction of previously unknown, potentially useful knowledge from data [1]. A KDD process is a highly complex, iterative and interactive process, with a goal-driven and domain dependent nature. The huge amount of algorithms for data manipulation and the continuous development of new ones, their many characteristics, the performances shown by different algorithms on different kinds of data, introduce some issues to deal with for effectively designing a KDD process.

In last years researchers in Data Mining and Knowledge Discovery in Databases fields have shown more and more interest in knowledge representation techniques for giving support in the design of knowledge discovery process. Early works represent such a knowledge by exploiting relational meta-model [2], object-oriented paradigm [3] and conceptual hierarchy [4]. In particular, models in [2] and [3] are introduced for supporting the building of a workflow of KDD tools, whereas [4] introduces a taxonomy of KDD algorithms, that is exploited for designing a KDD process facing with cost-sensitive classification problems.

Following the mainstream of Semantic Web, recent researches experienced the introduction of ontologies for formally represent knowledge. Most of the available ontologies focus on software and algorithms for Data Mining, which is one of the phases of the wider and more complex KDD field [1, 5]. The first ontology of this kind is *DAMON* (DAta Mining ONtology) [6], that is built for simplifying the development of distributed KDD applications on the Grid, offering to domain experts a taxonomy for discovering tasks, methods and software more suitable for a given goal. In [7], the ontology is exploited for selecting algorithms on the basis of the specific application domain they are used for. Finally, an interesting work is [8], where *OntoDM* is proposed as a general purpose top-level ontology aimed to describe the whole Data Mining domain. Since OntoDM is not conceived for achieving specific support requirements, like discovery of algorithms and process composition, systems based on such an ontology can be used for different activities, but providing inefficient supports in each of them. Recently, [9] introduces a KDD ontology representing algorithms and any piece of knowledge involved in a KDD process (dataset and model), that is exploited for guiding a planning algorithm in the design of a KDD workflow.

Unlike [8], previous approaches do not exploit a formal and explicit methodology for building the ontology. We think that the lack of a formal design methodology could lead to an ambiguous, inconsistent and incomplete ontology, that does not satisfy quality requirements as those discussed in [10].

The main contribution of this work is the design of KDDONTO, an ontology for supporting both the discovery of suitable KDD algorithms and the composition of KDD processes. To this end, we use an ontology building methodology aimed to define goal-oriented ontologies satisfying quality requirements. Each step of this methodology returns as output a valid ontology represented in a different language. At first, the basic terms characterizing KDD algorithms are identified and organized in a glossary, then classes and relations of the ontology are formally derived from it in axiomatic form. Finally, an OWL-DL implementation of the KDDONTO is obtained.

This work is part of the Knowledge Discovery in Databases Virtual Mart (KDDVM) project[1], a more general project for the development of an open and extensible environment where users can look for implementations, suggestions, evaluations, examples of use of tools implemented as services. In this framework, the KDDONTO is exploited for supporting both the discovery of web services implementing KDD algorithms, and their composition for building KDD processes. In particular, the use of such an ontology guarantees to obtain valid, useful and unknown results [11, 12].

The next Section of the paper discusses the ontology building methodology that will be used throughout the paper. In Section 3 such a methodology is exploited for designing KDDONTO, whereas in Section 4 a discussion on the evaluation of KDDONTO is provided. Finally, Section 5 ends the paper.

2 Design Methodology

Ontology building is a complex task, which requires formal care and a proper domain knowledge to be accomplished. In order to avoid errors and to obtain a valid and useful ontology, a knowledge engineer has firstly to define the ontology

[1] http://boole.diiga.univpm.it

goals and then to choose the design methodology. An ontology can be developed either for realizing an unifying framework for a specific domain, or for describing useful domain information that can be exploited for giving support to specific applications, e.g. KDD algorithm discovering and process composition.

At present, a standard methodology for ontology design has not been identified, and methodologies proposed in literature are mainly borrowed from software engineering field, among others [13–16]. These methodologies define interactive and iterative steps which, starting from the identification of main representative terms of the domain, lead to the final ontology through step-wise refinements. In particular, [15] describes an ontology development process made of a set of activity to be performed in order to build a sound ontology, namely: conceptualization/formalization/implementation/evaluation. The methodology *Ontology Development 101* proposed in [16] identifies a set of standard questions about the ontology goal, helping the engineer to understand the fundamental terms of the domain and their properties. Then terms become classes, while properties are used to group classes into taxonomies and to define other relations among classes. This methodology defines also some techniques for verifying the correctness of the ontology at each refinement step, for example through checking the balance of the classes in the taxonomies.

Although there are many similarities between software engineering and ontology fields, like class and property concepts, the goal of a software engineer is the development of implementable classes, while a knowledge engineer focuses on the representation of the domain.

In [10], authors propose some criteria specifically conceived for ontology design, which are based on concepts and theories derived from logics, linguistics and analytical philosophy areas. The main idea is to realize an ontology as an axiomatic theory, that is a set of logical axioms formalizing relationships among domain concepts. The first step is individuate "primitive" concepts, from which any other domain concept is then derived. Formalization guarantees some benefits: it helps to clarify and disambiguate the meaning of terms, it represents a common logical language, understandable by scientists with different backgrounds, and supporting the integration and comparison of distinct theories about a given domain. Moreover it is general, therefore is it can be used in various domains for efficiently making meaning explicit.

An original and important characteristic of such an approach is the introduction of a set of quality requirements that a formal ontology has to satisfy. These properties are:

– *coherence*: the ontology must be intrinsically non-contradictory, that is if a sentence that can be inferred from the axioms contradicts a definition, then the ontology is incoherent;
– *clarity*: an ontology should effectively communicate the intended meaning of defined terms in a non-ambiguous fashion. All definitions should be documented with natural language;
– *extensibility*: the ontology should be extended and specialized for satisfying new goals without a revision of the existing definitions;

- *minimization of encoding bias*: an encoding bias results when representation choices are made purely for the convenience of notation or implementation. The conceptualization should be specified at the knowledge level without depending on a particular symbol-level encoding;
- *minimization of ontological commitment*: an ontology should require the minimal ontological commitment sufficient to support the intended knowledge sharing activities. It can be minimized by specifying the weakest theory (allowing the most models) and defining only those terms that are essential to the communication of knowledge consistent with that theory.

It is to be noted that [10] gives the design criteria for building an ontology, without clearly defining the designing steps. For this reason, the methodology we use for building KDDONTO is based the steps described in [15] and on the goal-oriented step-wise strategy defined in [16] where the previous introduced quality requirements are taken into account. The steps of the employed methodology are the following:

- *Concepts identification and definition:* this step allows to identify the primitive concepts of domain and to provide as accurate as possible definitions of concepts in natural language. The output of such a phase is the glossary of terms describing domain concepts;
- *Building of a conceptual schema:* concepts are represented through classes and relations. Each concept definition should be represented, whether possible, throught axioms and formal statements. At the end of this phase the axiomatic theory is provided;
- *Implementation:* the goal of this phase is the translation of axioms in a machine readable language, on which automatic inference may be applied. The language must be enough expressive, for representing the domain in a proper way. However, an excessive expressiveness could lead to time-consuming inferential procedures or even to undecidable inferences. For such a reason, a balance between expressiveness and decidability is needed.

Furthermore, at the end of each step the satisfaction of functional and non-functional requirements is verified.

3 KDD Ontology Design

In this Section, we apply the above discussed methodology to the design of KDDONTO, an ontology describing the domain of KDD algorithms. Such an ontology is conceived for supporting the *discovery* of algorithms and their *composition*. Discovery is the activity of searching algorithms on the basis of the KDD goal to achieve, the dataset at hand, and functional and non-functional characteristics, whereas algorithm composition is the activity of linking suitable algorithms in order to build valid and useful knowledge discovery processes.

3.1 Concepts Identification and Definition

The goal of this phase is to identify key concepts in the domain of interest. Being KDD a research field born of preexistent areas (e.g.: Machine Learning, Statistics, Databases), an open issue concerns with the existence of heterogeneous interpretations about several terms used in Data Mining and KDD fields. A homogeneous conceptual synthesis has not been achieved yet, and same concepts are described in different ways by several authors, among others [1, 5, 17, 18]. Furthermore, terms definitions are often incomplete, there exist exceptions and overgeneralizations due to the gap between the world (continuous) and terms used to represent it (discrete). Thus, the first activity we performed has been the analysis and comparison of distinct interpretations, with the aim to identify, if possible, similarities among them and to choose the most shared ones, compatible with KDDONTO goals. Furthermore, since knowledge sharing and reusing represent a central help in knowledge engineering activity, a common practice in ontology building is the reuse of already tested solutions. Hence, we also took into account other KDD ontologies, and in particular the DAMON ontology for concepts regarding the Data Mining field.

In order to minimize the ontological commitment, we used a constructivist approach, identifying the least number of primitive concepts, which will be used in the next phase for defining the others. Definitions reported below are the results of such efforts.

The key concept of KDDONTO is *algorithm*, because it is the basic component of each process. An algorithm can be defined as "a formal finite sequence of basic instructions that, executed in a precise order, leads to solve a class of problems". A Data Mining algorithm is a procedure that, given an input, yelds an output in the form of a model. Uniqueness in descriptions cannot exist, because an algorithm can be described from many points of view: the task it accomplishes, the sequence of instructions, its performance, its input/output models. As suggested in [17], it is needed a form of reductionism, which allows to focus on similarities and differences among algorithms, reducing them to their basic elements. Other fundamental concepts are the following:

- *method*: a methodology, a technique used by an algorithm to extract knowledge from input data; e.g. the C4.5 algorithm uses a "decision tree" method;
- *phase*: a phase of a KDD process. According to CRISP-DM process model [5], a KDD process can be split into the following phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation and Deplyoment. Only some of them are automatable and taken into consideration for the purposes of this work;
- *task*: the goal at which aims who execute a KDD process. Among others there are descriptive tasks like Clustering and Feature Extraction, and predictive ones like Classification and Regression;
- *model*: model in KDD field is used in heterogeneous ways, ranging from a schema representing the discovered knowledge to a way to represent it. Hereafter, we define model as a set of constructs and rules for representing knowledge, whereas a *pattern* is an instance of a model used for representing

specific knowledge. Different kinds of models exist: “decision tree”, “Labeled Vector Quantization”, “neural network”, “Support Vectors” are classification models; in a “tree” model, for example, nodes describe conditions on features, while leaves describe the predicted class. According to feature value of an instance, it is possible to go through the tree to reach the prediction;
- *dataset*: a set of data in a proper format. It describes a set of objects by means of their measurable features;
- *parameter*: any information required in input or produced in output by an algorithm; the whole set of parameters defines its interface;
- *precondition/postcondition*: specific features that an input (or output) must have in order to be used by a method or an algorithm. Such conditions can concern data format (normalized dataset), data type (numeric or literal values), or data quality (missing values, balanced dataset) properties of an input datum. The same elements can describe postconditions, that are features of an output datum. For example, the RemoveMissingValues algorithm yelds in output a dataset with the postcondition “no missing values”;
- *performance*: an index and a value about the way an algorithm works; scalability, complexity, accuracy, entropy are some indexes;
- *optimization function*: the function that an algorithm or a method optimizes with the aim to obtain the best predictive/descriptive model; e.g. C4.5 builds the tree by optimizing, at each iteration, an “information gain”.

3.2 Building of a Conceptual Schema

The aim of this phase is to define a set of classes and relations among them in order to represent the KDD domain. Representing classes through axioms helps to make the model clearer, and also enable to represent the schema with a formal language. In this way deductive inference can be performed on the model, with the aim to extract hidden knowledge and to check consistency and its non-contradictory nature. Hereafter descriptions are given in a Description Logics-like syntax.

Starting from the basic terms identified in the previous paragraph, the top level classes and main relations have been identified. Top level classes are:

- `Algorithm`, `Method`, `Phase` and `Task`, corresponding to concepts with the same name;
- `Data`, which contains as subclasses `Model`, `Dataset` and `GeneralParameter`. The first two correspond to homonymous concepts, whereas the last addresses the *parameter* concept;
- `DataFeature`, corresponding to *precondition/postcondition*;
- `PerformanceIndex` and `PerformanceClass`. The former describes *performance* indexes (e.g. `COMPLEXITY`, `ACCURACY`, ...), the latter aggregates performance values, if possible and needed, in a finite number of classes (e.g. for `COMPLEXITY`: `LINEAR`, `POLYNOMIAL`, ...);
- `ScoreFunction`, corresponding to *optimization function*.

A more formal description of main classes goes beyond the goal of this work, and according to [19] it would need a top level ontology to completely clarify semantics. Main relations are:

- `specifies_phase`, between `Task` and `Phase`. An instance of `Task` can belong to more than one `Phase`;
- `specifies_task`, between `Method` and `Task`. Cardinality can be more than 1; for example `SVM`, a `Method` instance, can be used for two different tasks: `REGRESSION` and `CLASSIFICATION`;
- `uses`, between `Algorithm` and `Method`;
- `has_input`/`has_output`, a n-ary relation with domain `Algorithm`, `Method` or `Task` and codomain `Data`, and optionally `DataFeature`.
  In addition, `has_input` joins the classes to a boolean value that states the optional or mandatory nature of the input datum and to a number that expresses the strength of the precondition: a value equal to 1.0 corresponds to a mandatory precondition, while lower values to optional ones; also inverse properties `input_for`/`output_for` have been introduced;
- `has_performance`, a n-ary relation with domain `Algorithm`, `Method`, or `Task` and codomain `PerformanceIndex` and `PerformanceClass` and a string which represents the specific value of the performance index.

Subclasses are defined by means of existential restrictions on main classes, that can be considered as fundamental bricks for building the ontology. At first some `Phase` instances are introduced, namely `PREPROCESSING`, `MODELING`, `POSTPROCESSING`. They represent the main phases in a KDD process and are used to start the subclassing as follows:

- `Task` specializes in the following subclasses, according to the argument of `specifies_phase`:
  - $\texttt{PreProcessingTask} \sqsubseteq \texttt{Task} \sqcap \exists\texttt{specifies\_phase}\{\texttt{PREPROCESSING}\}$
  - $\texttt{ModelingTask} \sqsubseteq \texttt{Task} \sqcap \exists\texttt{specifies\_phase}\{\texttt{MODELING}\}$
  - $\texttt{PostProcessingTask} \sqsubseteq \texttt{Task} \sqcap \exists\texttt{specifies\_phase}\{\texttt{POSTPROCESSING}\}$
- `Method` is detailed in subclasses according to the tasks that each method specifies by means of `specifies_task` relation. Some examples:
  - $\texttt{ClassificationMethod} \sqsubseteq \texttt{Method} \sqcap \exists\texttt{specifies\_task}\{\texttt{CLASSIFICATION}\}$
  - $\texttt{FeatureExtractionMethod} \sqsubseteq \texttt{Method} \sqcap \exists\texttt{specifies\_task}\{\texttt{FEATURE EXTRACTION}\}$
- `Algorithm` specializes in subclasses according to `uses` and `has_output` relations, in the following way:
  - $\texttt{ClassificationAlgorithm} \sqsubseteq \texttt{Algorithm} \sqcap \exists\texttt{uses.ClassificationMethod} \sqcap \exists\texttt{has\_output.ClassificationModel}$

  while a `ClassificationAlgorithm` subclass can be defined as follows:
  - $\texttt{TreeAlgorithm} \sqsubseteq \texttt{ClassificationAlgorithm} \sqcap \exists\texttt{uses}\{\texttt{DECISION TREE}\} \sqcap \exists\texttt{has\_output.DecisionTreeModel}$

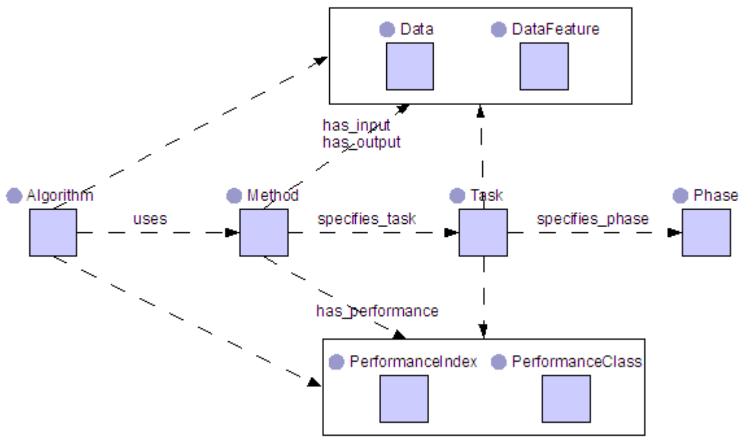


Fig. 1. KDDONTO: main classes and relations

– `Model` is further detailed in subclasses, on the basis of the task which the models are used for. For example:
  - `ClassificationModel`$\sqsubseteq$ `Model` $\sqcap$ $\exists$`output_for{CLASSIFICATION}`

A top-level view of described classes and relations is shown in Figure 1.

Any KDD process is built for achieving a specific goal, namely a KDD task, by means of various manipulations of a given dataset. Hence, the procedure we use for process composition starts from the goal and goes backwards iteratively adding one or more algorithms to a process, until the head of the process is not able to directly elaborate the given dataset. Although such a procedure is mainly based on the use of the `has_input/output` relations for matching algorithms with a common interface, we introduce other relations for composing valid processes. In the rest of this section we discuss the most interesting.

In some cases, two KDD methods, even if they have compatible input and output, cannot be used in the same process, because they may lead to useless processes. In this case the incompatibility can be expressed through the symmetric relation `not_with`. Furthermore, the transitive relation `not_before` means that a method cannot be in a process before another one. In both relations domain and codomain is the class `Method`.

Relations `in_module` and `out_module` allow to connect an instance of algorithm to others, which can be executed respectively before or after it. These relations provide suggestions about process composition, representing in an explicit fashion KDD experts' experiences about process building.

Each `Model` is defined by a structure, that in turn can be recursively defined through substructures, e.g. a Labeled Vector Quantization model (LVQ) is made of a VQ model and a Labeling function. Hence, `part_of` (and its inverse `has_part`) represents the relation between a model and an its component (a generic `Data` instance). Among the many different meanings of parthood relation studied by works in mereology theory, in this work we refer to a *component/integral* part-of, i.e. a configuration of parts within a whole. The parthood relation offers benefits also for algorithm discovery, for example it is possible to retrieve algorithms working on similar models, i.e. having common substructures. Note that subsumption (is-a) and parthood (part-of) can be jointly used to define not only exact but also approximate matches between algorithms interfaces, allowing also to define ranking strategies of generated processes. Due to page limitations, we refer to [12] for details about the composition procedure, algorithm matchmaking and ranking strategies.

Finally, we like to highlight that KDDONTO schema does not limit the mapping among instances of `Algorithm`, `Task` and `Phase`. This choice allows us to overcome a limitation introduced in [4] and [6], where an algorithm belongs to only a phase or a task. Moreover, unlike [4], where in a process "preprocessing precedes induction, which precedes postprocessing", we do not use `Phase` for constraining the sequence of algorithms in a process. Hence, we can obtain processes where a `K-NN` algorithm is used both for removing missing value (i.e. as preprocessing) and for inducing a classification model (i.e. as modeling).

3.3 Implementation

The final step in development process concerns the translation of the axiomatic theory in a machine readable language, that must be enough expressive in order to represent the domain of KDD algorithm in a proper way. However, an excessive expressiveness implies that inferential procedures, applied on the ontology, could result too much complex from a computational point of view. For such a reason it is necessary to find a balance between expressiveness and decidability.

Nowadays OWL is the de-facto standard language for building ontologies, on which automatic inference may be applied. For KDDONTO implementation, among the OWL sublanguages we chose OWL-DL, whose logical model is based on Description Logics and is decidable. Some classes of the axiomatic model can be directly mapped to the OWL-DL *Class* construct without any problem. However, due to expressive restrictions of OWL-DL semantics, some issues arise in translation of properties related to `Data`, `Model` and `PerformanceIndex` classes. As matter of fact, in OWL-DL it is not possible to have a class as a property value: for example it is not feasible to express the property `has_output` between `CLASSIFICATION`, that is an instance of `ModelingTask`, and `Classification-Model` class. An other restriction is the impossibility to define n-ary relations, because only binary ones are allowed in DL logic model, hence the properties `has_performance` and `has_input/output` cannot be mapped to any construct.

According to shared best practices [20, 21], such issues can be solved by means of a different modeling, that keeps unaltered the meaning of the concepts. As regards the first issue, anonymous instances, that are instances with no specified name, have been introduced. They can be used as property values, but they assume the meaning of *Class*. From a practical point of view, an anonymous instance can be implemented as an OWL class with a single instance.

Benefits of such an approach are manyfold. Firstly, instances can be connected to classes, at any abstraction level, e.g. `CLASSIFICATION` task can be linked to the anonymous instance of `ClassificationModel`, while a specific instance of `ClassificationAlgorithm` can have as output a specific subclass of `ClassificationModel`. Secondly, resulting ontology is compatible with OWL-DL. Lastly, DL-based reasoners are able to infer a hierarchy among anonymous instances as well as among classes.

For the issue related to n-ary relations, the solution is similar to the *relationship reification* in the context of database engineering. In this way the n-ary relation is translated into more binary relations between the anonymous instance and other instances. Hence, input/output and performance relations are reified with the introduction of new classes.

At present the KDDONTO implementation is formed of 95 classes, 31 relations and more than 140 instances, representing some algorithms of preprocessing, modeling and postprocessing phases. In particular we described algorithms implementing Feature Extraction, Classification, Clustering, Evaluation and Interpretation tasks. For lack of space we cannot show the whole KDDONTO, which is available at the project website[2].

4 Discussion

Ontology evaluation is an important issue in ontology building field, and various approaches have been proposed in literature. From the survey [22], we derive that an ontology has to be evaluated on the basis of both the domain description it provides, and the advantages it gives to an application task.

According to approaches described in [22], the former evaluation could be achieved by comparing the ontology to a *golden standard*, or to a *source of data*. As already stated in 3.1, both of them are not suitable for KDDONTO, because so far there is neither a wide accepted standard for the KDD domain nor a corpus of documents from which significant terms can be extracted. However, the use of the design methodology discussed in in Section 2 leads KDDONTO to satisfy quality requirements, allowing us to also state that the domain is correctly and effectively described. In particular, following the first and the second steps of the methodology we build a *clear* ontology, where the meaning of each primitive concept is first given in natural language, and then used as bricks for composing complex concepts, by means of logical restrictions. This accurate choice of the basic concepts allows us to also satisfy the *minimal ontological commitment* requirement. The division into phases supports in the *minimization of the encoding bias*, solving implementation issues only after a conceptual schema is built. The *coherence* of KDDONTO has been checked through Pellet[3], one of the most used open source OWL-DL reasoner. Finally, the design choices do not prevent the *extension* of KDDONTO, both introducing new taxonomies and importing KDDONTO as part of other ontologies. It is also to be noted that at each step

[2] http://boole.diiga.univpm.it/kddontology.owl
[3] http://clarkparsia.com/pellet

of the methodology we have a valid ontology represented in a different language; this enables, for instance, the possibility to translate KDDONTO in an other language without performing again the first two steps.

KDDONTO has been designed for giving support both in discovering and composition of KDD algorithms, therefore it has to be evaluated w.r.t. both these application tasks. For lack of space, in this work we cannot detail the advantages KDDONTO gives to these tasks, but refer the interested reader to our previous works. As concerns the discovery task, in [11] we plugged a preliminary version of KDDONTO into the KDDVM Broker. It turns out that the use of relations like `is-a` and `in/out_module` led to increase the accuracy in retrieving algorithms. On the other hand, we proved that KDDONTO enables the semi-automatic building of valid processes, in which reasoning capabilities based mainly on `has_input/output`, `is-a` and `part_of` relations are exploited for linking two algorithms [12].

5 Conclusion

A KDD process is a highly complex, iterative and interactive process, with a goal-driven and domain dependent nature. In order to effectively design such a process the user has 1) to individuate suitable algorithms for achieving her goal starting from the data at hand, and 2) to correctly compose the algorithms for forming a valid process. In order to give support to users in both these activities, within the Knowledge Discovery in Databases Virtual Mart project we are developing discovery and composition support services, that are based on KDDONTO, a formal ontology describing the domain of KDD algorithms.

In this paper we have discussed the design of KDDONTO. We first discussed the necessity to follow a formal methodology, then we use an ontology building methodology aimed to define goal-oriented ontologies, satisfying quality requirements. As results, the glossary of domain characteristic terms, the set of logical axioms, and the OWL-DL implementation of KDDONTO have been obtained.

While KDDONTO-based services for supporting algorithms discovery have been developed and are available at the project web site, we are at present working on the development of a service supporting the semi-automatic KDD process composition.

References

1. Fayyad, U.M., Piatetsky-Shapiro, G. and Smyth, P. In: From data mining to knowledge discovery: an overview. American Association for Artificial Intelligence, Menlo Park, CA, USA (1996) 1–34
2. Morik, K. and Scholz, M.: The MiningMart Approach to Knowledge Discovery in Databases. In Zhong, N. and Liu, J., ed.: Intelligent Technologies for Information Analysis. Springer (2004) 47–65
3. Wirth, R., Shearer, C., Grimmer, U., Reinartz, T., Schlösser, J.J., Breitner, C., Engels, R. and Lindner, G.: Towards Process-Oriented Tool Support for Knowledge Discovery in Databases. In: PKDD '97: Proceedings of the First European

Symposium on Principles of Data Mining and Knowledge Discovery, London, UK, Springer-Verlag (1997) 243–253
4. Bernstein, A., Provost, F. and Hill, S.: Towards Intelligent Assistance for a Data Mining Process: An Ontology Based Approach for Cost-Sensitive Classification. IEEE Transactions on Knowledge and Data Engineering **17**(4) (2005) 503–518
5. CRISP-DM site. http://www.crisp-dm.org
6. Cannataro, M. and Comito, C.: A data mining ontology for grid programming. In: Proc. 1st Int. Workshop on Semantics in Peer-to-Peer and Grid Computing, in conjunction with WWW2003. (2003) 113–134
7. Yu-hua, L., Zheng-ding, L., Xiao-lin, S., Kun-mei, W. and Rui-xuan, L.: Data mining ontology development for high user usability. Wuhan University Journal of Natural Sciences **11**(1) (2006) 51–56
8. Panov, P., Džeroski, S. and Soldatova, L.: OntoDM: An Ontology of Data Mining. In: Proc. of the 8th IEEE Int. Conf. on Data Mining Workshops. 1st Int. Workshop on Semantic Aspects in Data Mining. (2008) 752–760
9. Žáková, M., Křemen, P., Železný F. and Lavrač, N.: Using Ontological Reasoning and Planning for Data Mining Workflow Composition. In: SoKD: ECML/PKDD 2008 workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery. (2008)
10. Gruber, T.: Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum.-Comput. Stud. **43**(5-6) (1995) 907–928
11. Diamantini, C. and Potena, D.: Semantic Annotation and Services For KDD Tools Sharing and Reuse. In: Proc. of the 8th IEEE Int. Conf. on Data Mining Workshops. 1st Int. Workshop on Semantic Aspects in Data Mining. (2008) 761–770
12. Diamantini C., Potena D., and Storti E.: Ontology-driven KDD Process Composition. In Adams, N. et al., ed.: Proc. of the 8th International Symposium on Intelligent Data Analysis. Volume 5772 of LNCS. Springer, Lyon, France (Aug 31 - Sep 2 2009) 285–296
13. Missikoff, M. and Navigli, R.: Applying the Unified Process to largescale Ontology Building. In: Proc. of 16th IFAC World Congress. (2005)
14. Staab, S., Studer, R., Schnurr, H. and Sure, Y.: Knowledge Processes and Ontologies. IEEE Intelligent Systems **16**(1) (2001) 26–34
15. Fernandez, M., Perez, A. and Juristo, N.: METHONTOLOGY: from Ontological Art towards Ontological Engineering. In: Proc. of the AAAI97 Spring Symposium Series on Ontological Engineering, Stanford, USA (March 1997) 33–40
16. Noy, N. and McGuinnes, D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford University (2002)
17. Hand, D.J., Mannila, H. and Smyth, P.: Principles of Data Mining (Adaptive Computation and Machine Learning). The MIT Press (August 2001)
18. Sumathi, S. and Sivanandam, S.N.: Introduction to Data Mining and its Applications. Springer (2006)
19. Guarino, N.: Formal Ontology and Information Systems. In: Proc. of the 1st Int. Conf. on Formal Ontologies in Information Systems (FOIS98), Trento, Italy, IOS Press (1998) 3–15
20. Noy, N., ed.: Representing Classes as Property Values on the Semantic Web. W3C Working Group Note (2005)
21. Noy N. and Rector, A., ed.: Defining N-ary Relations on the Semantic Web: Use with Individuals. W3C Working Group Note (2006)
22. Brank, J., Grobelnik, M. and Mladenic, D.: A survey of ontology evaluation techniques. In: Proc. of the Conference on Data Mining and Data Warehouses (SiKDD). (2005)

Towards a Service-Oriented Knowledge Discovery Platform

Vid Podpečan[1], Matjaž Juršič[1], Monika Žakova[2], Nada Lavrač[1,3]

[1] Jožef Stefan Institute, Ljubljana, Slovenia
[2] Czech Technical University, Prague, Czech Republic
[3] University of Nova Gorica, Nova Gorica, Slovenia

Abstract. The paper addresses a challenge of third-generation data mining systems: the ability to make use of distributed data processing/ mining algorithms and potentially distributed heterogeneous information/knowledge sources. It proposes a Serviceoriented Knowledge Discovery (SoKD) framework and its prototype implementation, based on the Orange toolkit, which implements three main new functionalities: firstly, simple creation of graphical workflow components (Orange widgets) from distributed web services; secondly, the composition of data mining workflows from local and distributed data processing/mining algorithms applied to a combination of local and distributed data/knowledge sources; and thirdly, an implementation of a toolkit for producing new web services from existing data processing/mining algorithms. The proposed approach has been demonstrated on the task of constructing a workflow combining Orange processing components with Weka data mining services, as well as a workflow of text mining services combined with the widely known Pubmed database.

1 Introduction

Knowledge discovery and data mining have a long tradition, and two generations of knowledge discovery and data mining systems have already emerged. While the first-generation systems support a single or a few data mining algorithms designed to mine simple attribute-valued data [26], today's second-generation systems are characterized by high-performance interfaces to databases and data warehouses, the provision for inductive queries and by increased algorithm scalability. However, these second-generation mining systems fail to provide support for the analysis of distributed and poorly structured data/knowledge sources. They also lack more advanced inductive and reasoning capabilities, both of which are required to support the discovery and design processes involved in modern science and engineering.

Primarily focusing on the growing volumes of mainly homogeneous data, second-generation data mining methods and systems have failed to recognize the huge opportunities (and challenges) presented by fast-growing volumes of complex and geographically dispersed information and knowledge sources publicly available on the web. Neither existing data mining nor contemporary information retrieval technologies are designed to handle complex and dispersed knowledge sources which permeate all current science and technology development activities, thereby creating a fundamental bottleneck in scientific discovery and technology development.

A Service-oriented Knowledge Discovery (SoKD) approach, proposed in this paper, aims at developing a thirdgeneration knowledge discovery framework [1,7,10] and its implementation that is intended to address the bottlenecks discussed above. The proposed SoKD framework is targeted to become the enabling technology for scientific discovery, assisting the human scientist by providing access to information fusion and data analysis mechanisms and by enabling the construction of workflows of data processing/mining algorithms implemented as web services.

A practical implementation of the third-generation knowledge discovery platform, named Orange4WS, aims to support scientists in the creation of new scientific hypotheses from the wealth of knowledge and services available on the web. The third-generation data mining paradigm shift implies the need for a substantially different knowledge discovery platform, aimed at supporting human experts in scientific discovery tasks. In comparison with the current publicly available data mining platforms (e. g. Weka [26], KNIME[4], Orange [4], R-Bioconductor[5]), the SoKD platform should give access to open-source data processing and mining tools, designed as SOAP-based web services[6], in our future work to be supported by a suitable meta-data approach[7].

The described paradigm shift can be achieved through the integration of existing and newly developed data mining and knowledge discovery services, using the knowledge retrieved from publicly available sources in different formats (tabular data, text data, rules and other descriptive patterns, rule sets and other predictive models). The SoKD paradigm shift can potentially lead to the development of a novel service-oriented knowledge discovery process model for data mining (extending the current CRISP-DM data mining standard[8]). This will enable the orchestration of web-based data mining services and fusion of information of various formats, as well as simple design of repeatable data mining and information fusion workflows used in novel life science, bioinformatics and e-science applications.

In summary, this paper addresses a challenge of third-generation data mining systems: the ability to make use of distributed data processing (mining) algorithms applied to distributed and heterogeneous information and knowledge sources. It proposes a service-oriented knowledge discovery framework, and the prototype implementation of the SoKD platform which upgrades the Orange data mining toolkit [4].

[4] http://www.knime.org/

[5] http://www.bioconductor.org/

[6] In the future also ReST-based or a combination of both, once WSDL2.0 becomes more widely accepted.

[7] Using technologies such as SA-WSDL, OWL-S, WSDL-S, SWSF-SWSL, WSMO-WSML

[8] http://www.crisp-dm.org/

The paper is structured as follows. Section 2 presents a selection of past and current service-oriented data analysis approaches. Section 3 provides a motivating use case based on data mining services from the Weka environment. Section 4 is the core of this paper, presenting the SoKD framework and its implementation, named Orange4WS, upgrading the Orange data mining toolbox with the following new functionalities: firstly, simple creation of Orange widgets from distributed web services; secondly, the composition of data mining workflows from local and distributed data processing/mining algorithms applied to a combination of local and distributed data/knowledge sources; and thirdly, an implementation of a toolkit for producing new web services. Section 5 shows the application of the proposed approach used in the construction of workflows of text mining services combined with the widely known Pubmed database. Section 6 concludes by discussing the main advantages and shortcomings of the proposed framework.

2 Related work

This section discusses the idea of scientific workflows and introduces the term Service-oriented Computing. Main advantages and shortcoming of web services in the context of knowledge discovery are presented along with some general principles of service-orientation.

2.1 Workflow construction

Construction of analytic workflows has attracted a lot of development in recent years. Our work is mainly concerned with the composition of data processing/mining workflows composed of data mining algorithms available in data mining toolkits, combined with the execution of algorithms available as services on the web. Moreover, we are concerned with the analysis and use of distributed data/knowledge sources. In this sense, our work is related to the goal of providing an adequate workflow editing environment focused on the integration of computational resources and middleware and efficient execution, such as Triana [23], the system for scientific workflows developed in Kepler [9], WCT developed within the K-WF grid[10] and the tools developed within the DiscoveryNet project [16] and projects ADMIRE [11] and Weka4WS [21]. Furthermore, the Taverna [15] environment for workflow development and execution developed within the myGrid[11] project, uses an ontology focused on operations specific to bioinformatics tasks.

Similarly to the FAEHIM [2] project, we concentrate on the subdomain of scientific knowledge discovery connected to data mining. Similar to our approach, the toolkit developed within FAEHIM allows for manual composition of workflows, but the system is not implemented on top of an existing data mining

[9] http://kepler-project.org
[10] http://www.kwfgrid.eu
[11] http://www.mygrid.org.uk

platform, and hence it lacks the basic collection of data mining/knowledge discovery algorithms, visualization techniques and testing routines.

There have been some efforts to provide a systematic description of data and processes for the classical data mining tasks. The systems developed in the MiningMart project [12] and in the DataMiningGrid project [19] focus on mining propositional patterns from data stored in a relational database, each containing a meta-model for representing and structuring information about data and algorithms. The systems CITRUS [25] uses an object oriented schema to model relationships between the algorithms, while CAMLET [20] uses an ontology of algorithms and data structures, making a limited use of planning for process decomposition starting from a manually defined structure. Particularly relevant to our work is the problem of *web service composition*. In [14] BPEL4WS[12] is used for task formulation and workflow representation, but the adaptation of BPEL4WS to scientific workflows is still not standardized [18].

2.2 Service-oriented design and web services for knowledge discovery

Service-oriented architecture (SOA) is a term that represents a model in which automation logic is decomposed into smaller, distinct units of logic [5] (within SOA, these units are known as services). Some of the key principles of service-orientation [5,13] are as follows: loose coupling, autonomy, reusability, statelessness, abstraction, composability, and discoverability.

In general, service-oriented architecture can be implemented using a wide range of technologies including SOAP, REST, RPC, DCOM, CORBA, Web Services or WCF. Most commonly, web services are used as the key technology as they are platform and language independent, simple, standardized, and flexible. It should be noted that web services are not neccesarily inherently service-oriented which means that they can be designed to duplicate the behaviour and functionality found in complex distributed systems instead of being fully SOA-compliant (in this paper we mostly focus on the design based on SOA-compliant, loosely coupled, composable web services). Probably the most important characteristics of web services is that the data exchange is based on open standards. Messages are traveling from one web service to another via globally standardized and accepted protocols. Moreover, messages themselves are standardized and the use of SOAP, WSDL, XML, and XML schema allows for messages to be self-contained. All this, combined with their simplicity, distributed nature, and possibility of accessing large online databases make web service an appealing choice for various knowledge discovery tasks. However, there are several open issues and we will shortly discuss the most important.

In contrast with early machine learning and data mining algorithms current methods are focused on mining knowledge from large data sources (several gigabytes or even terabytes of raw and structured data). While usually this is not

[12] http://www.ibm.com/developerworks/library/specification/ws-bpel

a problem for the mining algorithms themselves, there is a problem of communication as the communication capabilities of computer systems are much worse than their internal data transfer abilities. Obviously, some principles of service-orientation have to be broken in order to apply the concept of web services to such large-scale problems. Most notably, web services will loose statelessness and will also become more tightly connected. For example, only pointers to data are exchanged between services and only pointers to results are returned. Clearly, such services have states and all depend on the central service, which provide storage and a collection of methods for the analysis.

Another, even more serious problem, is the lack of standards for exchanging data and knowledge. There are some advances in this direction, e.g. Predictive Model Markup Language[13] (PMML) for describing various data mining and statistical models, and ExpML language for sharing machine learning information [24], but are mostly unsupported by the general community. Moreover, there is no common and generally accepted XML-based language for describing tabular and other types of data. Old style data formats [4,26] like *csv, tab* or *arff* are not acceptable for the third-generation of knowledge discovery tools as they are not extensible nor self-describing. Consequently, most available knowledge discovery services are still in their infancy (including most services presented in this paper), communicating using data and models encoded in implementation specific representations.

Finally, as in the case of all web-related entities there is a problem of availability, security, and privacy. While little can be done about availability except by providing a certain level of redundancy, there are numerous WS-* extensions [5,13] dealing with security[14].

3 SoKD platform: A motivating example

This section presents a motivating use case for developing and using a service-oriented knowledge discovery platform (SoKD). The use case is built upon Weka [26] web services provided by A. Bosin[15]. Available web services are based on Weka version 3.4.6 and are running on Apache Tomcat web server using Apache Axis2 web service tools. There are 8 available services: *attributeRank*, *attributeSelect*, *datasetFilter*, *datasetDiscretize*, *classifierBuild*, *clustererBuild*, *modelTest*, and *modelApply*. All parameters of all services are implementation specific (Weka models, arff data format, etc), which is not appropriate for the following reasons:

1. Validity of parameter values can be checked only internally (Weka) as all strings are accepted by the web server. This is a waste of resources as Weka methods are invoked even with invalid parameter values. Also, error messages are more difficult to understand.

[13] http://www.dmg.org/pmml-v3-2.html
[14] We will not discuss them here as this is out of scope of this paper.
[15] http://www.dsf.unica.it/∼andrea/webservices.html

2. Models, encoded as hexadecimal strings are completely unportable. Only two exactly the same versions of Weka can exchange such models. Encoding and decoding models in PMML would be a better choice.

Although these services have poor connectivity with outer world processing units, they can be interconnected to produce better than trivial workflows because the major functionality of Weka is available (attribute evaluation, data filtering, model creation and testing).

Firstly, this use case supports the idea of service-oriented knowledge discovery by combining distributed processing components into non-trivial, easily understandable diagrams, which can be stored and executed on demand. Secondly, it demonstrates the general ability of the SoKD platform to import, connect, execute, and combine web services with local components. In our implementation of the SoKD platform, built on top of the Orange data mining toolkit [4] the local components are the Orange data/knowledge processing algorithms (Orange widgets), and simpler components providing integral data types (i.e. numbers, strings, and booleans).

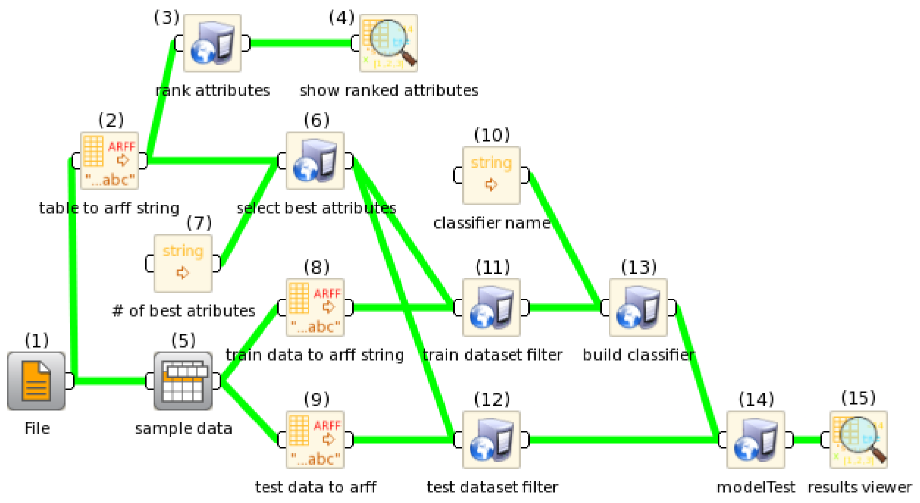


Fig. 1. A workflow of Weka data mining services, combined with Orange's local processing elements constructed within the Orange4WS platform. Components with numbers 3, 6, 11, 12, 13, and 14 are web services, components with numbers 1 and 5 are native Orange widgets. Other components are supporting widgets provided by Orange4WS. Names of parameters on connecting lines are not shown for the sake of simplicity.

Apart from the shortcomings, stated above, there are also several advantages which support the idea of exposing Weka implementations of data mining algorithms as web services. Most notably, Weka web services distribute the actual processing between the user's computer and a remote computer system (or systems). Moreover, implementations of data mining algorithms are now self-contained, platform independent, and do not need the entire Weka environment. Obviousy, this greatly expands the range of potential users as Weka algorithms can now be integrated into any software solution capable of making use of web services. Finally, if these services are maintaned and updated regularly, all of their users always have access to the latest implementations without any effort.

Our use case implements the following processing steps: (1) loading the data from a local file, (2) ranking of attributes to manually select few best, (3) partitioning the data into the training and test set, (4) building a classifier and evaluating it on the test set, (5) reporting results to the user. This is accomplished by connecting 15 processing entities 6 of which are web services, 2 are native Orange Widgets while the rest are supporting widgets provided by Orange4WS (data transformation and production of integral data types). The workflow, created and executed within the Orange4WS platform is shown in Figure 1.

4 The implementation of Orange4WS

This section briefly describes the structure and design of the presented SoKD software platform. We explain and comment our decisions about technologies and software tools that were used. The main part of the section is a description of the design of Orange4WS and the acompanying toolkit for producing new web services.

4.1 Design choices and technological background

The proposed software solution, named Orange4WS, is built on top of two open-source scientific community driven projects:

- Orange[16] data mining framework [4], which provides a range of preprocessing, modelling and data exploration techniques, and
- Python Web Services project[17] [8,9], which provides libraries for the development of web services using the Python programming language[18] by implementing various protocols including SOAP, WSDL, etc.

Among other alternatives the following tools were also considered: Weka, Triana, KNIME, and Taverna. The final decision, however, was Python and Orange due to simplicity and power of the Python scripting language[19], and availability

[16] The Orange data mining toolkit consists of the C++ layer with interfaces to the Python language, the Canvas, which is essentially a large collection of processing elements, called Orange Widgets, and the workflow execution engine.

[17] http://pywebsvcs.sourceforge.net/

[18] Use of the pythonic branch of the WSO2 seb service framework (based on Apache Axis2/C) is planned in the future.

[19] This decision has ruled out Java-based tools, even though they provide many desirable features like large-scale data mining (KNIME), widely accepted implementations (Weka), support for various forms of distributed computing (Triana) and support for numerous biology-oriented web services (Taverna).

of numerous data mining algorithms and powerful visualization techniques in Orange. By using an interpreted language it is possible to avoid the compile-test-recompile development cycle and, what is more important, dynamic languages are a perfect choice for the dynamic world of web-related technologies, especially web services.

Our ultimate goal was to provide a simple, user-friendly software tool, able to seamlessly integrate web services and local components in terms of workflow composition, originating from different communities: data mining, text mining and knowledge discovery in general, systems biology, etc. Therefore, having in mind that the construction of workflows of processing elements will be the central feature of our tool and that the implementation of a user-friendly workflow management is a very time consuming process, the decision was to retain the complete Orange workflow engine at the first stage of development and provide only additional capabilities which will be later able to evolve into a more powerful data flow engine.

4.2 Design and implementation of Orange4WS

Orange4WS consists of two layers: the supporting level which deals with technical details of web services (information extraction, execution, error reporting, data transfer and transformation, code generator) and the upper level, which simply uses low-level functionalities to enable web services as building blocks of the Orange Canvas. This section describes the structure of Orange4WS followed by the description of the accompanying tool for production of new web services in Section 4.3

The key element of programmatic use of web services is the Web Services Description Language (WSDL). Orange4WS does not parse WSDL files directly as this functionality is provided by the tools from the Python for Web Services Project. However, the results of deep parsing of message objects defining web service inputs and outputs are not available directly to the user but through auto-generated clients. Thus, Orange4WS provides two modules to get all the required information: the *extractor* and the *importer* module which handle web service message communication and automatically generate web service client code, respectively.

The actual invocation of the web service is performed by the *executor* module. This component provides threaded execution monitor and custom HTTP connection class with timeout. The latter is essential as there is no guarantee that the called web service will ever return a response message (reliability problems were discussed in Section 2.2). The structure of Orange4WS, described above, is summarized in Figure 2.

As the principal design guideline of Orange4WS is simplicity and automatization, a mapping mechanism is required which is able to provide seamless integration of web services into the Orange Canvas. This mechanism must hide all details of web service communication and provide a widget-like interface to all supported services. The Orange4WS *generator* module, also shown in Figure 2, provides the this functionality by producing valid Python code which is essentially an Orange Widget and uses the same communication mechanisms as other "native" widgets. Thus, all technical details regarding the integration of web services are summarized as one user interface command, namely "*import web service*" which combines all the necessary steps to produce new Orange widgets from specified web services. All details about the communication with a web service[20] are hidden and can be summarized from the user's perspective as a normal widget operation: (1) receiving data, (2) widget internal processing, and (3) outputting processed data.

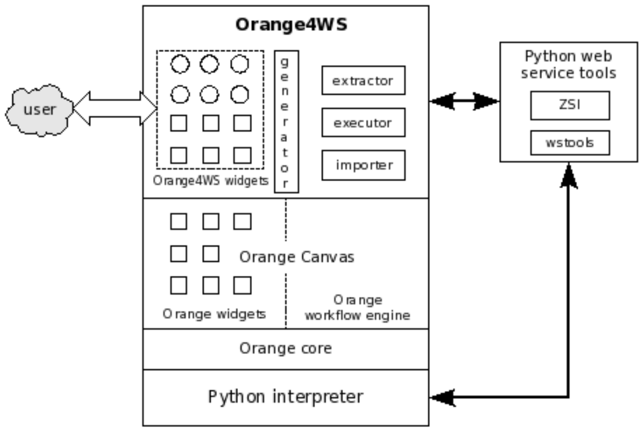


Fig. 2. The structure of Orange4WS platform, built on top of the Orange data mining toolkit.

4.3 Production of new web services

A separate part of our service-oriented knowledge discovery platform is a package of tools which ease the production of new services based on the code in languages, callable from Python (e.g. C/C++, Java, Fortran, C#). These tools closely follow the general *"WSDL first"* design principle [5,13]. This principle promotes clearly designed, interoperable and reusable services by separating the design of interfaces from the actual logic. The package provides a simple-to-use web server, which is based on the server included in the *ZSI* package which itself is based on the server provided by the standard Python distribution. Additionally, multi-threading, logging and address filtering were enabled and the resulting server is lightweight, secure, and easy to use.

[20] Our treatment of web services is deliberately simpified as one of our primary goals is simplicity. Thus, we consider only primitive, synchronous request-resonse MEPs (message exchange patterns).

5 A text mining use case

In Section 3 we presented a motivating use case which employs machine learning (data mining) web services based on Weka's implementations. The second use, which is presented in this section, demonstrates the use of text mining web services, available from the LATINO[21] project.

LATINO is a software library providing a range of data mining and machine learning algorithms with the emphasis on text mining, link analysis, and data visualization. Recently, its functionalities are becoming available as web services to ease the use of the library and consequently broaden its user community.

In our use case five LATINO services were used in combination with the Pubmed search service, local Orange graph visualization tool and supporting services from Orange4WS.

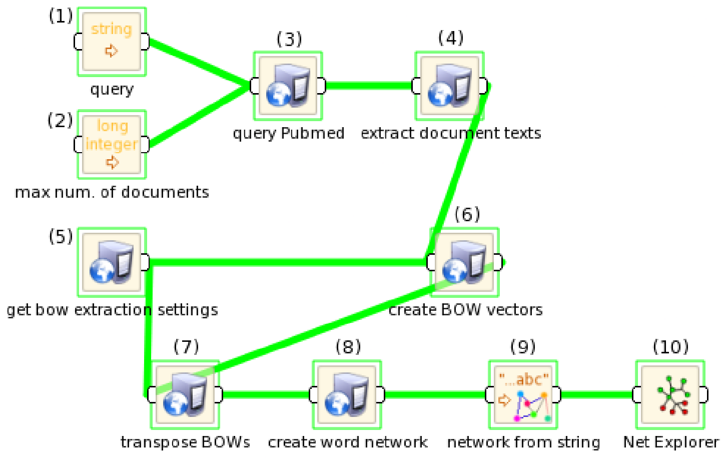


Fig. 3. A worflow of text mining services. Components numbered 3, 4, 5, 6, 7, and 8 are web services, components 1, 2 and 9 are Orange4WS supporting widgets, component 10 is a native Orange graph visualizer.

The goal of the use case is to produce a term graph which could potentially give insight into relations between biological, medical and chemical terms, relevant to the subject of the query. Also, it shows the ability of Orange4WS to combine publicly available data repositories (Pubmed) with third-party data analysis tools (LATINO) and powerful local visualization components (Orange graph visualizer). The workflow of processing components is shown in Figure 3.

Firstly, Pubmed is queried with a query string and maximal number of documents (components 1, 2, and 3). It returns a collection of IDs of relevant documents. Then, obtained IDs are used to collect titles, abstracts and keyword of

[21] http://sourceforge.net/projects/latino

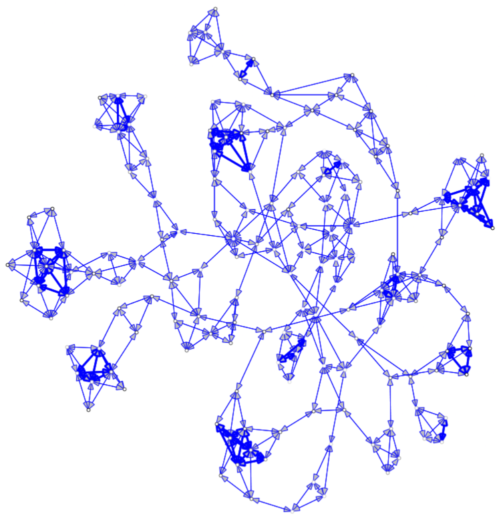


Fig. 4. A term graph obtained by querying Pubmed. Querry: "stem+cell", max. number of documents: 15. Because of the space limitations names of vertices and values of lines are not shown.

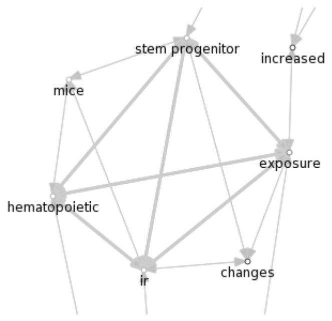


Fig. 5. A zoomed component from a term graph in Figure 4.

these documents (component 4). Afterwards, bag-of-words (BoW) sparse vectors are created from the collection of words (component 6). To simplify setting the parameters for unexperienced users there is a service providing a suitable set of default values which can be used as an input to the web service which constructs BoW vectors (component 6). Then, BoW vectors are transposed and a term network is created (we omit the details how this graph is produced as it is out of the scope of this paper). Component 8 produces a network in the widely used Pajek's .net format [3], which is finally loaded into Orange's native graph structure and visualized with the *Net explorer* widget.

For example, the query string "stem+cell" with the limit of 15 documents produced a graph which is shown in Figure 4. A zoomed component is shown in Figure 5.

6 Discussion

This paper proposes a third-generation knowledge discovery framework and its implementation named Orange4WS. Based on a second-generation data mining toolkit (Orange) that supports execution of workflows of processing components, our new platform upgrades its capabilities by transparent integration of web services. As web services are an extremely versatile and powerful concept which is becoming more and more popular, we believe their use in knowledge discovery will increase rapidly.

There are, however, many issues and shortcomings which still need to be resolved. As mentioned in Section 2.2, the most important issues are the standardization of knowledge and data representation and intelligent handling (processing and transfer) of large amounts of data. Our proposed platform is already targeted at semantics of web services although we did not explicitly address this issue[22]. Also, we took into consideration intelligent data handling, especially in connection with text mining services offered by the LATINO project where large amounts of data are being processed.

Our future work will be mainly focused on the semantic level of service-oriented knowledge discovery trying to reach the ultimate goal: a third-generation knowledge discovery platform. Several technologies, many of which are still under development will have to be considered for use and/or integration, e.g. SA-WSDL[23], WSDL-S[24], SWSF-SWSL[25], OWL-S[26], WSMO-WSML[27] etc. Finally, as one of the main goals of our platform is to be used by researchers with different background knowledge and different knowledge discovery tasks and a user

[22] Semantic annotations of algorithms and annotated data mining workflow construction is addressed in [27].
[23] http://www.w3.org/TR/sawsdl/
[24] http://www.w3.org/Submission/WSDL-S/
[25] http://www.w3.org/Submission/SWSF-SWSL/
[26] http://www.w3.org/Submission/OWL-S/
[27] http://www.w3.org/Submission/WSML/

community is expected to be formed, the platform will have to provide capabilities of sharing solutions of various knowledge discovery tasks, provided by its users.

Acknowledgments

The work presented here was partially supported by the European Commission under the 7th Framework Programme FP7-ICT-2007-C FET-Open, contract no. BISON-211898. The authors are grateful to Andrea Bosin for providing a repository of web services of Weka algorithms, and Miha Grčar for his work on the LATINO library.

References

1. R. Agrawal, J. C. Freytag, R. Ramakrishnan (eds.) Data Mining: The Next Generation. *Dagstuhl Seminar Proceedings* 04292, 2004.
2. A. Ali, O. Rana, and I. Taylor, Web services composition for distributed data mining. In *Proceedings of the 2005 IEEE International Conference on Parallel Processing Workshops*, ICPPW, 2005.
3. V. Batagelj, A. Mrvar, Pajek - Analysis and Visualization of Large Networks. Graph Drawing Software, Springer, 77-103, 2003.
4. J. Demšar, B. Zupan, and G. Leban, Orange: From experimental machine learning to interactive data mining, White Paper, 2004.
5. T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design.* Prentice-Hall. 2006.
6. U. Fayyad, G. Piatetsky-Shapiro and P. Smyth. From Data Mining to Knowledge Discovery in Databases, AI Magazine, 37–54, Fall 1996.
7. T. Finin et al. (eds.) National Science Foundation Symposium on Next Generation of Data Mining and Cyber-Enabled Discovery for Innovation (NGDM'07): Final report. http://www.cs.umbc.edu/~hillol/NGDM07/abstracts/NGDM07-Report.pdf
8. C. Hobbs: Using ZSI. Technical report. Nortel Advanced Technology Research, 2007.
9. H. Joukl: Interoperable Python ZSI WSDL/SOAP Web Services tutorial. Technical report. LBBW Financial Market Technologies, 2008.
10. H. Kargupta, A. Joshi, K. Sivakumar and Y. Yesha (eds.). *Data Mining: Next Generation Challenges and Future Directions.* AAAI Press, 2004.
11. N.L. Khac, M.T. Kechadi, and J. Carthy. Admire framework: Distributed data mining on data grid platforms. In *Proceedings of the First Int. Conf. on Software and Data Technologies*, vol. 2, pp. 67–72, 2006.
12. K. Morik and M. Scholz: The MiningMart approach to knowledge discovery in databases. In *Proc. of the International Conference on Machine Learning*, pp. 47–65, 2004.
13. E. Newcomer, G. Lomow. *Understanding SOA with Web Services.* Addison Wesley, 2005.
14. M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso. Planning and monitoring web service composition. In *Proceedings of AIMSA*, pp. 106–115, 2004.

15. D. D. Roure, C. Goble, and R. Stevens. The design and realisation of the my-experiment virtual research environment for social sharing of workflows. *Future Generation Computer Systems*, vol. 25, pp. 561–567, 2008.
16. A. Rowe, D. Kalaitzopoulos, M. Osmond, M. Ghanem, and Y. Guo. The discovery net system for high throughput bioinformatics. *Bioinformatics*, vol. 19, pp. 225–231, 2003.
17. J. C. Schlimmer: *Concept Acquisition Through Representational Adjustment.* Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine, CA, 1987.
18. A. Slominski, Adapting BPEL to scientific workflows. In I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds., *Workflows for e-Science*, pp. 208–226, Springer, 2007.
19. V. Stankovski, M. Swain, V. Kravtsov, T. Niessen, D. Wegener, J. Kindermann, and W. Dubitzky, Grid-enabling data mining applications with DataMiningGrid: An architectural perspective, *Future Generation Computer Systems*, vol. 24(4), pp. 259–279, 2008.
20. A. Suyama, N. Negishi, and T. Yamagchi, Composing inductive applications using ontologies for machine learning. In *Proceedings of the First International Conference on Discovery Science*, pp. 429–431, 1998.
21. D. Talia, P. Trunfio, O. Verta, Weka4WS: A WSRF-enabled Weka Toolkit for Distributed Data Mining on Grids. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, vol. 3721, pp. 309-320, 2005.
22. I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds., *Workflows for e-Science, Scientific Workflows for Grids.* Springer, 2007.
23. I. Taylor, M. Shields, I. Wang, and A. Harrison, The Triana workflow environment: Architecture and applications. In *Workflows for e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds. Springer, pp. 320–339, 2007.
24. J. Vanschoren, B. Pfahringer, and G. Holmes: Learning From The Past with Experiment Databases. Working Paper Series 08/2008, Computer Science Department, University of Waikato, 2008.
25. R. Wirth, C. Shearer, U. Grimmer, T. P. Reinartz, J. Schloesser, C. Breitner, R. Engels, and G. Lindner, Towards process-oriented tool support for knowledge discovery in databases. In *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, vol. 1263, pp. 243–253, 1997.
26. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* 2nd Edition, Morgan Kaufmann, 2005.
27. M. Žakova, Petr Kremen, Filip Železny and Nada Lavrač. Using Ontological Reasoning and Planning for Data Mining Workflow Composition. SoKD: ECML/PKDD 2008 workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery, 2008.

Advancing Data Mining Workflow Construction: A Framework and Cases using the Orange Toolkit

Monika Žáková, Vid Podpečan, Filip Železný, and Nada Lavrač

Czech Technical University in Prague, Czech Republic
{zakovm1,zelezny}@fel.cvut.cz
Jožef Stefan Institute, Ljubljana, Slovenia
{nada.lavrac,vid.podpecan}@ijs.si

Abstract. The paper presents a framework for automatic construction of data mining workflows based on input and output specification of the data mining task. An ontology of data mining components is used for the formalization of the data mining task, knowledge types and algorithms. The ontology is then used for automated workflow construction using an algorithm that combines planning and ontological reasoning. The paper demonstrates how enhancing the classical planning with ontological reasoning can address some of the challenges of data mining workflow construction, including complex objects passed between the algorithms, constraints formulation and workflow presentation. The proposed methodology was tested in a case study annotating algorithms available in the Orange data mining toolkit and extending Orange to enable the execution of the generated data mining workflows consisting of algorithms available in Orange.

Key words: data mining workflows, ontology, planning

1 Introduction

Integration of heterogeneous data sources and inferring new knowledge from such combined information is one of the key challenges in present-day science. Consider e.g. bioinformatics, where for virtually any biological entity (a gene, for example) vast amounts of relevant background information are available from public web resources. A principled fusion of such relevant data requires the interplay of diverse specialized algorithms resulting in highly intricate workflows.

While the mutual relations of such algorithms and principles of their applicability may be mastered by computer scientists, their command cannot be expected from the end user, e.g. a life scientist. A formal capture of this knowledge is thus needed, e.g. in the form of ontologies of relevant services and knowledge/data types, to serve as a basis for intelligent computational support of knowledge discovery workflow composition. A formal capture of the knowledge discovery task can be used to improve repeatability of experiments and to enable reasoning on the results to facilitate reuse of workflows and results.

In our work we define *knowledge discovery workflow* as a *progression of steps (inductive, deductive, format-conversion procedures etc.) involved in generalizing specific data (e.g. measurements) into models and patterns, which, under appropriate interpretation, may represent novel knowledge about the problem domain under investigation.* Therefore it can be viewed as a special form of scientific workflows [28], covering the data preparation and modeling stages of the standard CRISP-DM data mining methodology[1].

This work was originally motivated by the complex knowledge discovery workflow of interleaving inductive, deductive and format-conversion procedures which had to be manually constructed in our previous study in bioinformatics [30]. A methodology combining planning and ontological reasoning to construct this type of workflows is described in [32],[33]. Its key ingredients are an ontology of knowledge discovery algorithms, named the KD ontology, and a planning algorithm.

To evaluate the generality of our methodology and to identify additional challenges, this paper concentrates automatic generation of workflows of data processing and data mining algorithms available within a single selected toolkit. Algorithms available in the Orange [7] data mining toolkit were annotated using the KD ontology. Two challenges were identified. Firstly, there are collections of algorithms, which are equivalent from the point of view of input/output description (e.g. set of ranking algorithms resulting in redundant search and a large number of generated workflows). Secondly, indefinitely long graphs can be constructed from the algorithms (e.g. using a sequence of preprocessing steps). These two problems were addressed by imposing additional constraints, implemented in the prototype solution described in this paper.

In this paper we present an enhanced version of the planning algorithm (described in [33]), which is able to exploit the hierarchy of algorithms to reduce the search space and also for filtering and a more user-friendly presentation of the discovered workflows. Moreover we present a framework for integrating our methodology into the Orange toolkit based on an ontology mapping between the representation of algorithms in Orange and the representation using KD ontology.

The paper is structured as follows: Section 2 describes related work in area of data mining ontologies and planning, Section 3 presents an overview of the KD ontology and an example annotation of an Orange algorithm using the KD ontology. Section 4 describes the algorithm used for automatic workflow construction and Section 5 presents the prototype implementation of our framework.

2 Related Work

Our work is mainly concerned with automatic *composition* of data mining and knowledge discovery workflows and we view this problem in the context of planning. We currently focus on generating abstract workflows, i.e. workflows without

[1] http://www.crisp-dm.org

mapping to concrete computational resources, rather than providing a workflow editing environment focused on the integration of computational resources and middleware and efficient workflow execution, such as Triana [29].

The most relevant for our work is the IDA system described in [3]. The system uses an ontology, which provides a relatively detailed structure of the propositional DM algorithms and is built on OWL-S [17]. Workflow construction focuses on classical DM processes, which contain three subsequent steps: pre-processing, model induction and post-processing. In contrast, we address more complex, relational DM workflows with possibly multiple interleaved occurrences of steps pertaining to the three categories. Furthermore, the workflows generated by the IDA system are linear, whereas our workflows are directed acyclic graphs. Another system for automatic workflow construction using a knowledge discovery ontology is described in [4], however this work is focused only on automatic formation of linear sequences of tasks.

The systems CITRUS [31] and CAMLET [27] aim at providing a systematic description of data and processes for the propositional DM tasks and make a limited use of planning for process decomposition starting from a manually defined structure. [31] uses an object oriented schema to model relationships between the algorithms, while [27] uses an ontology of algorithms and data structures. The FAEHIM [1] project also focused on creating DM workflows. In contrast to our approach, the toolkit developed within FAEHIM supports only manual workflow composition and does not use any formally defined conceptualization of the domain.

Other efforts to provide a systematic formalization of the DM tasks include projects MiningMart [16], DataMiningGrid [26] and systems described in [5] and [14]. The systems [16] and [26] focus on mining propositional patterns from data stored in a relational database. Each contains a meta-model for representing and structuring information about data and algorithms, however, none of the meta-models is expressed in an ontology language. Also, the systems do not provide means for automatic workflow creation.

In parallel to our work, the OntoDM [18] ontology is being developed on the basis of [8]. A principled top-down approach was adopted to the development of OntoDM aiming at its maximum generality. Given the complexity of the domain subject to modeling, the ontology is currently not sufficiently refined for purposes of workflow construction [19]. Also, unlike our ontology, OntoDM is not compatible with OWL-S.

Several previous works have explored planning in the context of workflows outside of the DM domain. Notably, within the Pegasus project [6] a planner is used to construct a concrete workflow given an abstract workflow. We tackle a related yet different goal; given an ontology and a task description, we use a planner to construct an abstract workflow.

Also relevant are solutions to the problem of *web service composition* in the framework of planning. The work of [13] relies on computing a *causal link matrix* for all the available services. However we work with a more general, non-linear notion of a plan, where the inputs of an algorithm (action) combine the outputs of multiple other algorithms.

Work reported in [23], [12] and [15] translate an OWL description to a planning formalism based on PDDL. While the work presented in [12] and [15] use classical STRIPS [9] planning, in [23], Hierarchical Task Network (HTN) planning [22] is employed. HTN is not applicable in our framework not constrained to tree-based task decomposition. The approach presented in [15] and [12] uses a reasoner in the pre-processing phase; we make a step beyond by integrating a reasoning engine directly with the planner. Planning directly in description logics is addressed in [11]. Currently the algorithm can only deal with DL-Lite descriptions with reasonable efficiency.

3 Knowledge Discovery Ontology

We have provided a formal conceptualization of the knowledge discovery domain by developing the *Knowledge Discovery Ontology* (KD ontology, for short). The ontology defines relationships among the ingredients of knowledge discovery scenarios, both declarative (various knowledge representations) and algorithmic. The primary purpose of the ontology is to enable the workflow planner to reason about which algorithms can be used to produce the results required by a specified knowledge discovery task and to query the results of the knowledge discovery tasks.

The framework for data mining proposed in [8] (to be implemented in OntoDM [18]) identifies three basic concepts of data mining: 'data', 'patterns and models' and 'data mining task'. In contrast our three core concepts are: *knowledge*, capturing the declarative elements in knowledge discovery, *algorithms*, which serve to transform knowledge into (another form of) knowledge, and *knowledge discovery task*, which we have extended to involve workflows.

The ontology is implemented in the description logic variant of the semantic web language OWL-DL [20]. Our primary reasons for this choice were OWL's sufficient expressiveness, modularity, availability of ontology authoring tools and optimized reasoners. The core part of the KD ontology currently contains around 150 concepts and is available online.[2] More details on the ontology are presented in [33]. The structure of workflows is described using OWL-S [17].

In the following subsections we focus only on the description of the `Algorithm` class of the KD ontology and provide details of the annotation of algorithms available in the Orange toolkit.

3.1 Algorithms

The concept of algorithm is central to the work presented in this paper. The `Algorithm` class is a base class for all algorithms, like the Apriori (algorithm for association rule induction implemented in Orange [7]), in the example below. For this work we have refined the hierarchy of fully defined classes, like

[2] http://krizik.felk.cvut.cz/ontologies/2008/kd.owl

`DecisionTreeAlgorithm` or `DataPreprocessingAlgorithm` for fine-grained categorization of DM algorithms according to their functionality. The hierarchy of algorithms allows for the formulation of additional constraints on the workflows. E.g. there should be at most two preprocessing algorithms on each branch of the workflow.

Each algorithm configuration is defined by its input and output knowledge specifications and by its parameters. The `Algorithm` class is defined as a specialization of the OWL-S class `Process` and an algorithm configuration is an instance of its subclass `NamedAlgorithm`. Both the input knowledge and the parameters are instances of `AlgorithmParameter` and defined using the `input` property. The output knowledge specifications are instances of `AlgorithmParameter` and defined using the `output` property. The parameter instances are then mapped to the appropriate `Knowledge` subclasses using the `isRangeOf` property.

3.2 Annotating Orange Algorithms

The KD ontology was used to annotate most of the algorithms available in the Orange toolkit. More than 60 algorithms have been annotated so far. As an example we present a definition of the Apriori algorithm in the description logic notation using the extended ABox syntax [2]:

$$\begin{aligned}\{\texttt{Apriori}\} \sqsubseteq\ & \texttt{NamedAlgorithm}\\ & \sqcap\ \exists\,\texttt{output}\cdot\{\texttt{Apriori-O-Rules}\}\\ & \sqcap\ \exists\,\texttt{input}\cdot\{\texttt{Apriori-I-Dataset}\}\\ & \sqcap\ \exists\,\texttt{input}\cdot\{\texttt{Apriori-I-MinSupport}\}\\ & \sqcap\ \exists\,\texttt{input}\cdot\{\texttt{Apriori-I-MinConfidence}\}\end{aligned}$$

$$\begin{aligned}\{\texttt{Apriori-I-Dataset-Range}\} & \equiv \texttt{isRangeOf}\cdot\{\texttt{Apriori-I-Dataset}\}\\ & \equiv \texttt{Dataset} \sqcap \forall\,\texttt{hasFormat}\cdot\{\texttt{TAB}\}\\ & \quad \sqcap \forall\,\texttt{hasExpressivity}\cdot\texttt{SingleRelationStructure}\\ & \quad \sqcap \forall\,\texttt{hasAttributesType}\cdot\{\texttt{dDiscrete}\}\end{aligned}$$

$$\begin{aligned}\{\texttt{Apriori-O-Rules-Range}\} & \equiv \texttt{isRangeOf}\cdot\{\texttt{Apriori-O-Rules}\}\\ & \equiv \texttt{Patternset} \sqcap \forall\,\texttt{contains}\cdot\texttt{AssociationRule}\end{aligned}$$

The Apriori algorithm is defined as an algorithm that can be applied to a single relation dataset in the TAB format containing only discrete attributes and produces a result in the form of a set of association rules. It has two parameters: minimal support and minimal confidence of the rule. All three parameters are specified by integer values. The other input parameters were omitted from this example.

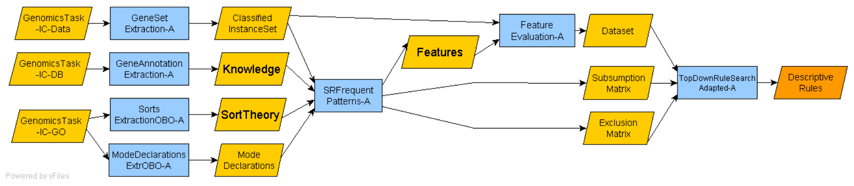


Fig. 1. An example of a workflow generated for the task of producing rules distinguishing between two types of leukemia on the basis of gene expression and annotations of genes using the GO ontology.

The algorithms were annotated manually, since no systematic description of these algorithms e.g. in PMML[3] or WSDL[4] was available. The annotated algorithms also served as case studies to validate and extend the KD ontology, therefore developing a procedure for semi-automatic annotation is a subject for future work.

4 Automated workflow construction

Our methodology focuses on automatic construction of abstract workflows of DM algorithms. The mapping to concrete computational resources, particular data sets and algorithm parameters are not taken into account during workflow construction. Each generated workflow is stored as an instance of the `Workflow` class and can be instantiated with a specific algorithm configuration either manually or using a predefined default configuration. We treat the automatic workflow construction as a planning task, in which algorithms represent operators and their input and output knowledge types represent preconditions and effects. However since the information about the algorithms, knowledge types and the specification of the knowledge discovery task is encoded through an ontology, we implemented a planning algorithm capable of directly querying the KD ontology using a reasoner [33]. The Pellet [24] reasoner was used. The main motivation for using Pellet was its ability to deal with literals, availability in Protégé[5], which we used for ontology development, and processing of SPARQL-DL [25] queries.

Our original work was motivated mainly by complex relational DM tasks such as discovery of rules to distinguish between two types of leukemia based on gene expression and gene annotations using terms of the Gene Ontology[6]. An example of a workflow generated for this task is shown in Figure 1. Since complex algorithms are needed for this task, the number of alternative workflows, which can be produced, is quite small.

[3] http://www.dmg.org/pmml-v4-0.html
[4] www.w3.org/TR/wsdl
[5] http://protege.stanford.edu/
[6] www.geneontology.org

When we extended our ontology with annotations of algorithms available in the Orange toolkit, we encountered the problem of having sets of algorithms, which on the basis of their inputs and outputs subsume each other or are even equivalent. For tasks such as inducing association rules from a propositional dataset, this led to producing a large number of workflows, a lot of which were very similar. In this work we alleviate this problem by exploiting the algorithm subsumption hierarchy.

In the next sections we present an enhanced version of the algorithm described in [33], which exploits the algorithm hierarchy for planning at multiple abstraction levels. Furthermore, a post-processing step using the ontology was added for a more user-friendly presentation of the KD workflows, in case the task is weakly constrained and thus a relatively large number of workflows is generated.

4.1 Exploiting algorithm hierarchy

The planning algorithm used to generate abstract workflows automatically is based on the Fast-Forward (FF) planning system [10]. We have implemented the basic architecture of the FF planning system consisting of the enforced hill climbing algorithm and the relaxed GRAPHPLAN. Since the planning problem in workflow construction contains no goal ordering, no mechanisms for exploiting goal ordering were implemented.

The planner obtains neighboring states during enforced hill-climbing by matching preconditions of available algorithms with currently satisfied conditions. Each matching is conducted in the planning time via posing an appropriate SPARQL-DL query to the KD ontology. In the original version of the planner presented in [33], there are no mechanisms for exploiting the algorithms hierarchy. We have enhanced the algorithm in two ways: a hierarchy of algorithms based on defined classes and input/output specifications computed and in searching for neighboring states the planner exploits the algorithm hierarchy.

A hierarchy of algorithms is inferred before the actual planning. It needs to be recomputed only when a new algorithm is added to the ontology. The hierarchy of algorithms is based on the inputs and outputs of the algorithms and on the defined algorithm classes such as `PreprocessingAlgorithm`. An algorithm $A_j \sqsubseteq A_i$, if for every input of I_{ik} A_i there is an input I_{jl} of algorithm A_j such that range of $I_{ik} \sqsubseteq I_{jl}$. An algorithm $A_i \equiv A_j$, if $A_j \sqsubseteq A_i$ and $A_i \sqsubseteq A_j$. The subsumption relation on algorithms is used to construct a forest of algorithms with roots given by the explicitly defined top-level algorithm classes e.g. `DataPreprocessingAlgorithm`.

The planning algorithm was adapted so that in the search for the next possible algorithm it traverses the forest structure instead of only a list of algorithms and considers a set of equivalent algorithms as a single algorithm. Currently, only constraints on repetition of some kind of algorithms in a linear part of the workflow are built into the planner. The additional constraints on workflows are used only in filtering of workflows during post-processing (procedure **filterWorkflows**). Workflows for all the members of an equivalence set are generated using

```
$task$ - instance of KnowledgeDiscoveryTask, $maxSteps$ - max length of the workflow,
$constr$ - additional constraints on the workflows
generateWorkflows($task$, $maxSteps$, $constr$):
  classify KD ontology;
  $algs$ := {instances of NamedAlgorithm};
  $algforest$ := inferAlgorithmHierarchy($algs$);
  $workflows$ := runPlanner($task$, $algforest$, $maxSteps$);
  $atomicW$ := expandWorkflows($workflows$, $algforest$);
  $filteredW$ := filterWorkflows($atomicW$, $constr$);
```

Fig. 2. A skeleton of the procedure for workflow composition using the KD ontology.

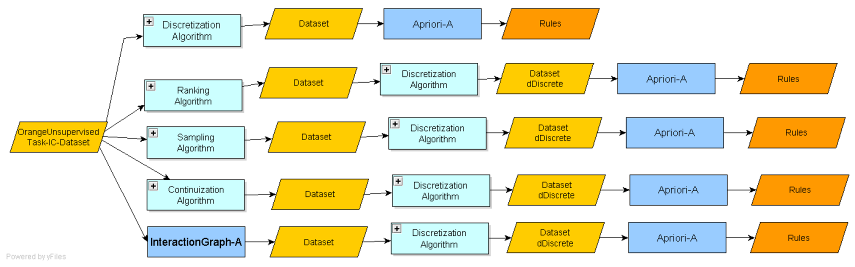


Fig. 3. An example of workflows for discovering association rules in Orange.

the procedure **expandWorfklows**. The information about algorithms subsumption is also used in workflow presentation. An overview of the whole procedure for workflow generation is shown in Figure 2.

The generated workflows are presented to the user using an interactive visualization, which enables the user to browse the workflows from the most abstract level to specific combination of algorithm instances. The workflows with the smallest number of steps are presented first. An example of a set of workflows generated for discovering association rules in Orange is in Figure 3.

5 A framework for workflow execution in Orange

We have developed a framework for integrating our methodology into the Orange data mining platform, so that workflows, which were constructed manually using the Orange GUI, can be automatically annotated using the KD ontology. The annotated workflows can then be used for querying and reasoning. All the information required for the Orange representation is preserved in the annotation; therefore Orange workflows can be recreated from the annotations and executed again in the Orange toolkit. On the other hand, workflows generated by the planner using KD annotations of Orange algorithms can be converted to the Orange representation and executed in Orange. An overview of the framework is shown in Figure 4. The module Orange2Onto, which acts as an interface

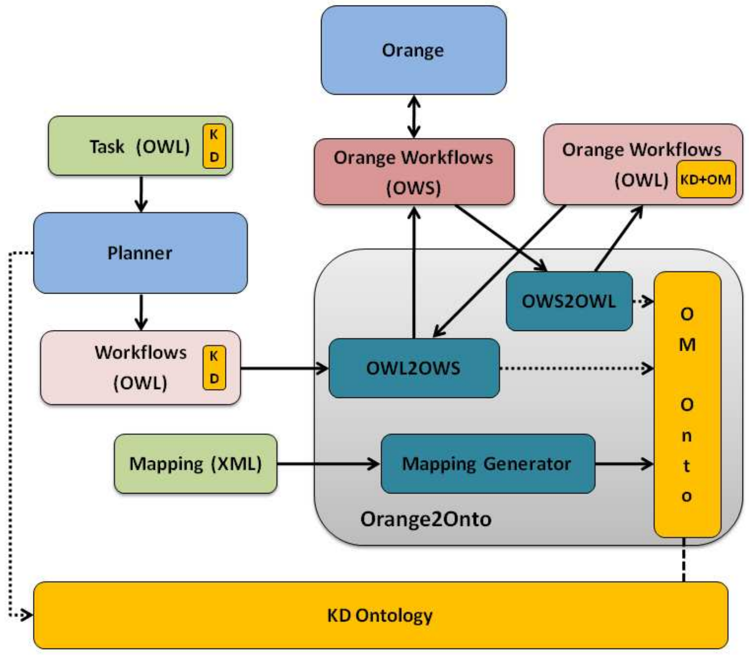


Fig. 4. An overview of the framework for integration annotations and planning into Orange.

between Orange and ontology representation does not work directly with internal representation of Orange, but it works with the OWS format used in the standard Orange distribution to store workflows in XML format.

In order to capture formally the mapping between the internal Orange representation and the representation of algorithms using the KD ontology, the Orange-Map (OM) ontology was developed defining templates for mapping of algorithms, data and parameters. The template for a parameter represented using a set of radio buttons in Orange is shown below:

$$\begin{aligned}\texttt{OrangeRadioParamMapping} \sqsubseteq\ & \exists\,\texttt{parameter} \cdot \{\texttt{kd:AlgorithmParameter}\}\\ \sqcap\ & \exists\,\texttt{radioValue} \cdot \texttt{OrangeRadioValue}\\ \sqcap\ & \exists\,\texttt{orangeAlias} \cdot \texttt{string}\end{aligned}$$

$$\begin{aligned}\texttt{OrangeRadioValue} \sqsubseteq\ & \exists\,\texttt{paramURI} \cdot \{\texttt{anyURI}\}\\ \sqcap\ & \exists\,\texttt{rbNumber} \cdot \texttt{int}\end{aligned}$$

Currently in the Orange format for storing the workflows, a parameter value is represented only by the number of the selected radio button. It is specified in the mapping using the property `rbNumber`. In the KD ontology it is mapped into an instance with URI specified using `paramURI`.

The OM ontology is then used for converting the automatically generated workflows into the Orange representation. In order to facilitate the creation of the mapping for new algorithms, the mapping can be specified using an XML file. The corresponding instances in the ontology are then generated automatically.

Annotation of a new algorithm available in Orange thus requires the following steps:

1. create instances of `AlgorithmParameter` for all inputs and outputs
2. create an instance of `NamedAlgorithm`
3. for each instance of `AlgorithmParameter` create a class defining its range (if not yet defined, add the necessary subclasses of `Knowledge` - this should be required only when a new type of algorithm is added)
4. create an XML file defining a mapping between the algorithm representation in Orange and in the KD ontology
5. run a script for generating a mapping using the OM ontology

Annotations of Orange workflows containing algorithms not annotated using the KD ontology can also be created automatically. The missing information about input/output types of the algorithms is then either deduced from the links with annotated algorithms or considered to be some `Knowledge` expressed as string. The annotations of such workflows can therefore be used for some querying and repeating of experiments, however the generated annotation of the unknown algorithm is not suitable for planning.

The procedures for converting Orange representation to OWL and vice versa are implemented in Python using JPype[7] to call the Jena[8] ontology API implemented in Java.

6 Evaluation

We carried out experiments comparing the enhanced planner exploiting the algorithm hierarchy with the original classical planner. We used each planner for two tasks. The first task was the complex and specialized task of discovering descriptive rules in the genomics domain. The second task was a simple task of discovering association rules. The KD ontology including the subontologies for annotation of the individual algorithms contains about 500 classes and 500 individuals.

The results summarized in Table 1 indicate that the HierarchyPlanner exploiting the algorithm hierarchy needs shorter time for planning for all the tested settings. With increasing number of equivalent algorithms the time taken for

[7] http://jpype.sourceforge.net/
[8] http://jena.sourceforge.net/

Task	No. of algorithms	Planner		HierarchyPlanner	
		Prep.	Plan	Prep.	Plan
GEN	71	72	0.854	115	0.560
GEN	99	104	1.123	155	0.568
ASSOC	71	94	27.291	125	25.154
ASSOC	99	98	107.549	153	24.354

Table 1. Planner performance results, with respect to the task, the number of algorithms available. The time for preprocessing (Prep.) and planning (Plan) is shown in seconds.

planning rises less rapidly for the HierarchyPlanner. The preprocessing stage lasts longer for the HierarchyPlanner due to the construction of algorithms hierarchy, however this task can be performed offline and repeated only when the ontology changes.

The example of a set of generated workflows shown in Figure 3 illustrates the use of algorithm hierarchy for workflow presentation. Since there are 4 discretization, 4 sampling, 5 ranking and 6 continuization algorithms, it would be infeasible to present all the generated workflows without using the algorithm hierarchy. The automatic selection of some relevant subset of workflows is nontrivial and will be a subject of future work.

7 Conclusions and Future Work

The primary objective of this study was to investigate challenges of data mining workflow construction resulting mainly from sets of similar or equivalent algorithms, which are typically available in a data mining toolkit and to develop a methodology for integrating our approach to automatic workflow composition to a data mining toolkit, which contains means for manual workflow creation and execution.

We have developed a planner, which exploits the hierarchy of algorithms annotated using the KD ontology, and shown that during the planning stage, this planner is faster than the classical planner. The construction of algorithm hierarchy is time consuming, however it needs to be recomputed only when a new algorithm is added to the ontology. Moreover the hierarchy can also be exploited in the presentation of the workflows to the user.

We have also proposed a methodology for integrating annotation and planning into a data mining platform by means of an ontology describing a mapping between KD representation and native representation of the data mining platform. The methodology was implemented in the Orange toolkit. It could be modified to other data mining platform, however it is less feasible if the data mining platform does not have any structured form of external representation of algorithms e.g. as XML.

A new version of the Orange data mining toolkit, named Orange4WS, which is able to import algorithms available as web services into Orange, is currently being developed [21]. In future work we are planning to integrate semantic annotations and automatic workflow construction into the new toolkit Orange4WS, containing a wider range of data mining algorithms. We also plan to include additional constraints and user preferences into the planner.

Acknowledgments

MZ was supported by the 1ET101210513 project Relational Machine Learning for Biomedical Data Analysis. FZ is supported by the Czech Science Foundation project number 201/08/0509. NL and VP are supported by the Knowledge Technologies project funded by the Slovenain Research and Technology Agency.

References

1. A. Ali, O. Rana, and I. Taylor, "Web services composition for distributed data mining," in *Proc. of the IEEE Int. Conf. on Parallel Processing Workshops*, 2005.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, Eds., *The Description Logic Handbook, Theory, Implementation and Applications.* Cambridge University Press, 2003.
3. A. Bernstein, F. Provost and S. Hill, "Toward Intelligent Assistance for a Data Mining Process: An Ontology-Based Approach for Cost-Sensitive Classification, " in IEEE Trans. on Knowledge and Data Engineering, vol. 17, pp. 503-518, 2005.
4. P. Brezany, I. Janciak and A. M. Tjoa, "Ontology-based construction of grid data mining workflows," in *Data Mining with Ontologies: Implementations, Findings and Frameworks.* IGI Global, 2007.
5. D. T. A. Congiusta and P. Trunfio, "Distributed data mining services leveraging WSRF," *Future Generation Computer Systems*, vol. 23(1), p. 2007, 34-41.
6. E. Deelman, J. Blythe, G. Yolanda, C. Kesselman, S. Koranda, A. Lazzarini, G. Mehta, M. A. Papa, and K. Vahi, "Pegasus and the pulsar search: From metadata to execution on the grid," in *Parallel Processing and Applied Mathematics*, 2004.
7. J. Demsar, B. Zupan, and G. Leban, "Orange: From experimental machine learning to interactive data mining," White Paper, 2004. Available: `www.ailab.si/orange`
8. S. Džeroski, "Towards a general framework for data mining," in *Knowledge Discovery in Inductive Databases KDID'06*, LNCS, vol. 4747. Springer, 2007, pp. 259–300.
9. R. Fikes and N. Nilsson, "STRIPS: a new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.
10. J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence research*, vol. 14, p. 2001, 253-302.
11. J. Hoffmann, "Towards efficient belief update for planning-based web service composition," in *Proc. of ECAI 2008*, 2008, pp. 558–562.
12. M.Klusch, A. Gerber, and M. Schmidt, "Semantic web service composition planning with owls-xplan," in *Procs of 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web*, 2005.
13. A. D. F. Lécué and A. Léger, "Applying abduction in semantic web service composition," in *Proc. of the ICWS 2007*, 2007, pp. 94–101.

14. Y. Li and Z. Lu, "Ontology-based universal knowledge grid: Enabling knowledge discovery and integration on the grid," in *Proc. of the 2004 IEEE Int. Conf. on Services Computing*, 2004.
15. Z. Liu, A. Ranganathan, and A. Riabov, "A planning approach for message-oriented semantic web service composition," in *Proc. of the Nat. Conf. on AI*, vol. 5(2), 2007, pp. 1389–1394.
16. K. Morik and M. Scholz, "The MiningMart approach to knowledge discovery in databases," in *Proc. of the ICML 2004*, 2004, pp. 47–65.
17. D. Martin, Ed., "OWL-S: Semantic markup for web services," W3C Member Submission, 2004. [Online]. Available: `http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/`
18. P. Panov, S. Džeroski, and L. N. Soldatova, "OntoDM: An ontology of data mining," in *IEEE ICDM Workshops 2008*, 2008, pp. 752–760.
19. P. Panov and S. Džeroski, 2009, Personal communication.
20. P. Patel-Schneider, P. Hayes, and I. Horrocks, "OWL web ontology language semantics and abstract syntax," W3C Recommendation, 2004. [Online]. Available: `http://www.w3.org/TR/owl-semantics/`
21. V. Podpečan, Monika Žáková and N. Lavrač, "Towards a Service-Oriented Knowledge Discovery Platform," accepted to SOKD-2009 in July 2009.
22. E. D. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artif. Intell.*, vol. 5(2), pp. 115–135, 1974.
23. E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau, "HTN planning for web service composition using shop2," *Journal of Web Semantics*, vol. 1(4), pp. 377–396, 2004.
24. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Journal of Web Semantics*, vol. 5(2), p. 2007, 51-53.
25. E. Sirin and B. Parsia, "SPARQL-DL: SPARQL query for OWL-DL," in *Proc. of the OWLED 2007 Workshop on OWL: Experiences and Directions*, 2007.
26. V. Stankovski, M. Swain, V. Kravtsov, T. Niessen, D. Wegener, J. Kindermann, and W. Dubitzky, "Grid-enabling data mining applications with dataminingrid: An architectural perspective," *Future Generation Computer Systems*, vol. 24(4), pp. 259–279, 2008.
27. A. Suyama, N. Negishi, and T. Yamagchi, "Composing inductive applications using ontologies for machine learning," in *Proc. of the First Int. Conf. on Discovery Science*, 1998, pp. 429–431.
28. I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds., *Workflows for e-Science, Scientific Workflows for Grids.* Springer, 2007.
29. I. Taylor, M. Shields, I. Wang, and A. Harrison, "The Triana workflow environment: Architecture and applications," in *Workflows for e-Science*, Springer, 2007, pp. 320–339.
30. I. Trajkovski, F. Železný, N. Lavrač, and J. Tolar, "Learning relational descriptions of differentially expressed gene groups," *IEEE Trans. Sys Man Cyb C*, vol. 38(1), pp. 16–25, 2008.
31. R. Wirth, C. Shearer, U. Grimmer, T. P. Reinartz, J. Schloesser, C. Breitner, R. Engels, and G. Lindner, "Towards process-oriented tool support for knowledge discovery in databases," in *Proc. of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery*, vol. 1263, 1997, pp. 243–253.
32. M. Žáková, P. Křemen, Železný and N. Lavrač, "Planning to Learn with a Knowledge Discovery Ontology," in *Proc. of PlanLearn 2008*, 2008.
33. M. Žáková, P. Křemen, Železný and N. Lavrač, "Automating Knowledge Discovery Workflow Composition through Ontology-based Planning," submitted to IEEE TASE, March 2009.

The Fantom Service for Subgroup Discovery in Score Lists

Jeroen de Bruin[1], Nada Lavrac[2], Joost N. Kok[1]

[1] LIACS, Leiden University, Leiden, The Netherlands
[2] Jozef Stefan Institute, Ljubljana, Slovenia

Abstract. We describe a Subgroup Discovery Service called Fantom that finds subgroups given a set of elements with scores. The subgroups are described by conjunctions of predicates and are given a measure of interestingness based on an idea from Bioinformatics. For the generation of interesting subgroups we use frequent structure mining, which exhaustively searches for all relevant subgroups above a minimal interestingness. As a use case we apply Fantom to data from microarray experiments.

1 Introduction

Consider the following generic problem in data mining: As input we have a number of data elements with a score for each element. We want to find all subgroups that

– have a high score for the subgroup (i.e. a score for a set of elements based on the score of the individual scores of the elements in the set) indicating that this subgroup is very interesting.
– can be described by a conjunction of predicates (rules) which all elements of the subgroup have in common.

Hence we need a score not just for individual data elements, but also for sets of elements. We propose to use a score on set of elements that is inspired by a scoring function in Bioinformatics (Gene Enrichment). Furthermore, we need to make a choice for the predicates that can be used. A natural choice is to use terms of ontologies in the domain. Moreover, we want to do some pruning on the output (predicates can imply other predicates and we strive for most specific rules as well as the highest scores).

We propose a service called *Fantom* that finds all subgroups above a certain size and above a minimum score (both conditions are provided as configurable parameters). It is built in a generic way so that we can apply it in a variety of fields. As input, it uses a set of identifiers coupled to a set of scores and allowed predicates. As an output, it presents the user with a set of rules and an appreciation of those rules in the form of the score.

We apply this service to a specific field of research, namely to data from microarray experiments. A natural outcome of a microarray experiment is a set of genes together with their t-values. There are a number of ontologies available about genes, their function and in which pathways they play a role. Moreover, there is information about interaction with other genes. We experimented on a standard data set of genes discussed in [GST$^+$99].

The rest of the paper is partitioned as follows. In Section 2, we will discuss work related to our Fantom approach, and discuss various knowledge sources that are used in Fantom as well. In Section 3, we will discuss Fantom itself, as well as specific algorithms and technologies used to implement the case study. In Section 4 we discuss the case study and its results, together with some statistics. Finally, in Section 5, we will make some preliminary conclusions, and discuss research and improvements that can be done in the future work.

2 Related Work

In this section we present some work related to the Fantom algorithm, as well as work related to structured knowledge sources, knowledge mappings and other sources of information used in Fantom.

2.1 Ontologies

An ontology, as seen in information science, is the hierarchical structuring of knowledge about things by subcategorising them according to their essential (or at least relevant and/or cognitive) qualities [Ont]. Over time, many efforts have been done by the computer science community together with field experts to create and reason about ontologies for diverse scientific fields such as chemistry [OAM$^+$03], web-mining and the semantic web [Dav06] and Bioinformatics [ABB$^+$00,OGS$^+$99].

Due to the increased attention in data mining with ontologies, related technologies such as representations of ontologies, description logic and ontology reasoning have been given alot of attention as well. Currently, there is a wide range of (ontology) description languages available, and each of them has their own specific role. For representation of ontology elements and data, usually a form of the eXtended Markup Language (XML) [XML] is applied, sometimes together with the Resource Description Format (RDF) [RDF]. For representation of relations among the data elements and extensions to allow reasoning over these entity-relationship models, currently the Web Ontology Language (OWL) [OWL] and the older F-Logic [Bal93] are commonly used. A good overview of ontology languages is provided in [TS06].

2.2 Annotations and Mappings

Within Fantom we seek to generate knowledge in terms of conjunctions of ontological terms by using a ranked list. This would not be possible if there was no mapping that asssociates or correlates an identifier with one or more ontology terms. Considering the field of BioInformatics, ther are many identifiers that can be used in genomics and proteomics [MOPT05,PTM07,MCOW05,Wai] and they are usually accompanied by mappings between those identifiers and ontologies, or those identifiers and other identifiers. For example, for GO terms there are several mappings (their default identifier is ENTREZ) that are updated either daily or monthly [go-]. KEGG maps work with KEGG orthologies, but also to HUGO gene symbols. [keg].

Another crucial mapping is the mapping of interactions between data elements. Fantom provides an option to mine for knowledge not based on direct association, but through degrees of interaction, assuming a transitive closure between elements; If A acts on B, and B acts on C, then A acts on C. By using a data source that states interactions between identifiers, Fantom can uncover knowledge that describes these indirect relations. In BioInformatics, interactions are being monitored in the GeneRIF project [Gen] and Reactome [VDS+07].

2.3 Related Algorithms

The Fantom algorithm was originally based on [TZTL06] and [TLT07]. While SEGS uses a simliar method to Fantom, it is restricted to only one entry per (sub)ontology, is tailored specifically to microarray experiments, and does not seem to prune rules that provide redundant information. In [LRS+08] subgroups are matched to a subset of GO terms in a probibalistic way, which induces a greater portion of error and false discoveries than an exhaustive search through the search-space. The GOEAST [ZW08] algorithm also checks for gene enrichment in GO terms, but only checks for a single go term, and takes as input raw microarray data, which again restricts its apllicability.

Alot of work has also been done on scoring functions. Typically, there is not just one scoring function that is considered the best, it all depends on what the someone is researching and what properties are considered interesting. In the case of Fantom the aim is to have a score for a subset of elements from a list of identifiers and scores; the group score is thus dependent on the score of individual elements. In Bioinformatics, well-known algorithms that perform this kind of scoring are [GST+99,LB06].

3 The FANTOM Service

The Frequent pAtterN Tree-based Ontology Miner algorithm, or FANTOM, is a service that takes as input a set of identifiers and their weights (scores), background knowledge in the form of ontologies, mappings and interaction data, a preferred scoring function that is applicable to the experiment domain, and specific thresholds for rule generation, and through rule generation and pruning delivers a non-redundant set of rules that describe subgroups of the input set. In this section we will discuss the inputs, internal mechanics and outputs of Fantom.

3.1 Inputs

Context The context parameters are used to define the experimental context needed for the algorithm to function correctly. Based on these context parameters the correct versions of mappings and ontologies will be presented.

Set of Identifiers The set of identifiers is a table with two columns. The first column contains the identifiers, which have to be unique and each one has to correspond to an identifier in the provided mapping. The second column is the score, or weight, of the identifier. There are no limits to these weights, but Fantom assumes that a higher weight means a higher importance.

Ontologies The ontology format is a table with six columns. The first one is the element identifier, which has to be unique across all ontologies. The second column represents the predicate. Usually each ontology has only one predicate, but in the case of GO, for example, there are three predicates (three sub-ontologies as it were). The third column is a natural language description for the element, basically the elements normal name. Columns four and five are collections of identifiers, the fourth column specifying aliases of this element, the fifth one specifying its parents. The sixth column is reserved as an indicator, indicating the element is deprecated, in which case the collection of aliases specifies the alternative identifiers.

Mappings and Interactions Both mapping and interaction files have the same structure, a table with two columns. The first column specifies the identifier to map, and the second one specifies one or more identifiers that they map to, either as a different ID type in case of mappings, or a set of other identifiers that this identifier interacts with.

Scoring Functions The user can select a list of predefined scoring functions. By default, the Enrichment Score (ES) function [GST$^+$99] is selected, which calculates the score of a subgroups based on the score of the individual members as well as the members not in the subgroup:

- Sort the list of N identifiers according to their score with score function s, whereby $s_j = s(id_j)$. For the resulting list $L = (id_1, id_2, ..., id_N)$ it holds that $s_1 >= s_2) >= .... >= s_N$.
- For each position i in L, evaluate the identifiers in the subgroup S conforming to the conjunction of ontology terms weighed by their score, and the identifiers not in S but still present in L at a higher position than i:

  $P_{included}(S, i) = \sum_{id_j \in S j \leq i} \frac{|s_j|}{\sum_{id_j \in S} s_j}$
  $P_{excluded}(S, i) = \sum_{id_j \notin S j \leq i} \frac{1}{N - N_S}$

– The ES is the maximum deviation from zero of $P_{included} - P_{excluded}$

Note that Fantom only uses this score function, and is not tailored to it. Fantom is generic enough to allow any kind of scoring function that adheres to the follwoing principles:

– A higher score indicates a higher interestingness.
– The score conforms to the provided .Net framework architecture and interface specified in Fantom.

Thresholds Fantom allows the user to work with two thresholds: minimum support and minimum score. Minimum support indicates how much genes a subgroup should contain at least, and heavily influences the duration of an experiment. The second threshold indicates the minimum score a rule should have. It is primarily used for pruning rules in a postprocessing step, but can sometimes also be used to help decrese the number of generated rules.

3.2 Algorithm and Structures

The core of the Fantom algorithm is based on the Apriori principle freqently used in itemset mining [AIS93] and refined many times since its conception [HPY00,Bod03]. Given that it is a proven technique that has been used in many fields, it was considered generic enough to be used in Fantom. Within Fantom, the algorithm repeatedly executes three stages: rule generation, rule appreciation and rule pruning.

Rule Generation Rule generation proceeds on the basis of subset combination. Let us illustrate this by an example: Suppose we have two rules which both contain n ontology elements. Then for those rules to make up a new rule together of $n+1$ elements, they have to share $n-1$ elements, while the other two should not be related to each other; the unshared element in the first rule should not be a descendant or ancestor of the unshared element in the second rule.

Suppose that there are m rules of n ontology elements. That means that there are at maximum $m * n$ subsets. Let S_i be the ith subset, and the number of elements in that subset k_{S_i}. Then the number of rule generations would be at maximum:

$\sum_{i=0}^{(m*n)}(k_{S_i} - 1)!$

It is easy to see that this number is equal to, but usually alot smaller than the brute force method, which consider $(m-1)!$ possibilities.

Rule Appreciation The rule appreciaton phase is the phase where all rules generated in the rule generation phase are being assigned a score on the basis of the selected score function. Two scores are being calculated; the score of the rule and the maximum score that the minimum number of elements involved in this rule could possibly have (note that this minimum is equal to the minimum support parameter). If this maximum score is below the score threshold, that means that this rule can be pruned in the next phase.

Rule Pruning Within each stage, rule pruning is done in three phases. First, rules generated and appreciated in the previous two phases are now compared to the input constraints; all rules not satisfying the support or minimum score constraint are pruned, leaving the rest as possible candidates for the next rule generation epoch.

The second phase identifies redundant knowledge within rules of the same amount of ontology elements, also called the rule dimensionality. All rules of the same dimensionality are compared to each other (in a similar manner as rule generation, through subgroups) and the ontologies to identify if one or more ontology terms in one rule are more general than the terms in another. If this is the case, and the score of the more specific rule is higher or equal than the score of the more general rule, then the more general rule is discarded as a rule (but still be kept for the next epoch).

Finally, in the third phase, subgroups of $1...i - i$ ontology elements are generated for each remaining rule of dimensionality i, and these are subgroups of rule elements are compared in the same way as phase two, only now to rules generated in past epochs.

3.3 Output

As an output, Fantom generates a textfile that contains all the remaining rules after pruning, and a summary of ontology terms in those rules, and how often they appeared in all the rules together. Furthermore, rules with the same subset of identifiers are clustered together, improving readability. An example rule from the use case looks like this:

```
Rule 1
Score: 0,761302007553004
A participants: [Epha1, Epha2, Epha8, Ephb2, Ephb3, Ephb4, Ptk2, Met, Ephb1]

All genes in the subgroup
  have the following properties:
    molecular_function(protein tyrosine kinase activity),
    molecular_function(ATP binding),
    biological_process(protein amino acid phosphorylation),
    biological_process(receptor protein tyrosine kinase signaling pathway),
    KEGG_pathway(Axon guidance)
```

Fantom can also generate rules that take into account interaction genes with other genes. When interaction rules are generated, the conjunctions of ontology terms do not describe the subgroup mentioned in the Participants header, but rather an anonymous subgroup that all the subroup participants interact with. Below is an example of such an interaction rule.

```
Rule 1
Score: 0,915794797276831
Participants: [Acat1, Acat2, Cycs, Cyct, Dld, Mdh2, Ogdh, Uqcrq, Uqcrc1]

All genes in the subgroup
  have interaction with a gene (or genes) that has the following properties:
    molecular_function(ATP binding),
    cellular_component(mitochondrial envelope),
    biological_process(tricarboxylic acid cycle),
    KEGG_pathway(Citrate cycle (TCA cycle)),
    KEGG_pathway(Alzheimer's disease)
```

3.4 Implementation

Implementation of the ontology and mapping generation as well as the interaction files was done in the Python language and run on Python 2.5.2. The web service implementations were done in Microsoft C++ .Net 2008 and Microsoft C-Sharp 2008, both using the .Net Framework wersion 2.0 and 3.5.

4 Use Case

In this section we present the use case, where the FANTOM algorithm was applied to one of the microarray data sets used in [GST+99]. We discuss the dataset used, what transformations we applied to get a ranking of genes, and what parameters we supplied to Fantom. We also present some performance statistics that address both speed and pruning.

4.1 Dataset and Inputs

In this use case we used a well know publicly available dataset that compares gene expression profiles of Acute Lymphoblastic Leukemia (ALL) and Acute Myelogenous Leukemia (AML) [ASS+02]. In this dataset, gene expression profiles were taken from 24 patients suffering from ALL and 24 patients suffering from AML. We first normalized the raw data using Quantile normalization [BIAS03]. After that, we performed mapping of the probes to entrez genes using the Hu6800 annotations supplied by Affymetrix. We discarded any entries that could not be mapped succesfully to a single identifier, to reduce the uncertainty error. Finally, we performed a t-value calculation with the Student's T-test between those two groups, thus researching what all overexpressed genes have in common. If a gene had multiple T-values, the average of those values was taken.

As ontology inputs the GO and KEGG ontologies were used, combined with GSEA score metric. Context inputs were set to homo sapiens, and identifier was kept default to Entrez.

4.2 Embedded Fantom Service

As can be seen in Figure 1, FANTOM can be easily embedded as a service in a larger workflow. In this experiment we used the Taverna [MyG] workbench to create and run a workflow where we automatically collect the newest ontology definitions from GO and KEGG and their newest mappings (all happens through webservices made available by ECBI and GenomeNet), and then run the FANTOM algorithm (the run function below).

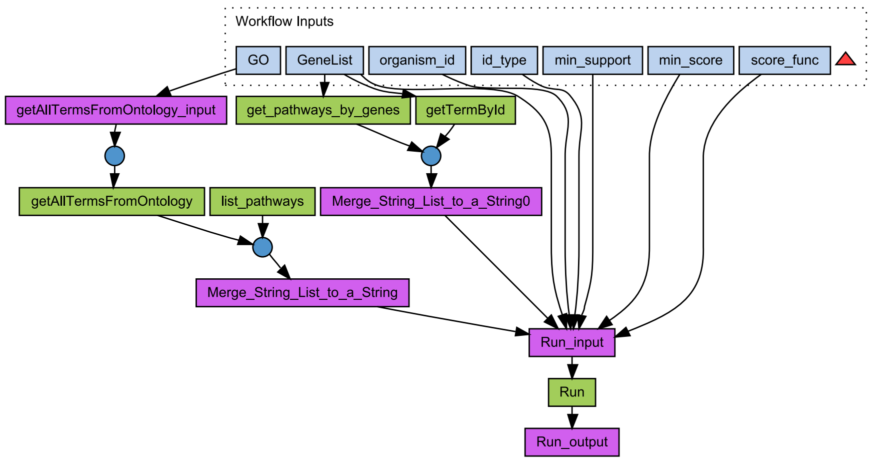


Fig. 1. Taverna workflow of FANTOM

Statistics concerning speed and pruning with various thresholds are presented below. In Figure 2, performance measurements are shown for different minimal support sizes S (the graph on the left) and different minimum score thresholds C.
As can be seen in Figure 2(a), the increase of the minimum participants has a profound effect on the performance of the algorithm. The same effect can be seen in Figure 2(b) where the minimum score was increased, though at a lesser extend with bigger subgroups. Still, if we extrapolate the lines in Figure 2(b), it is still obvious that pruning based on the ES measurement improves performance greatly (one can simulate the lack of ES pruning by taking a minimum score of 0).

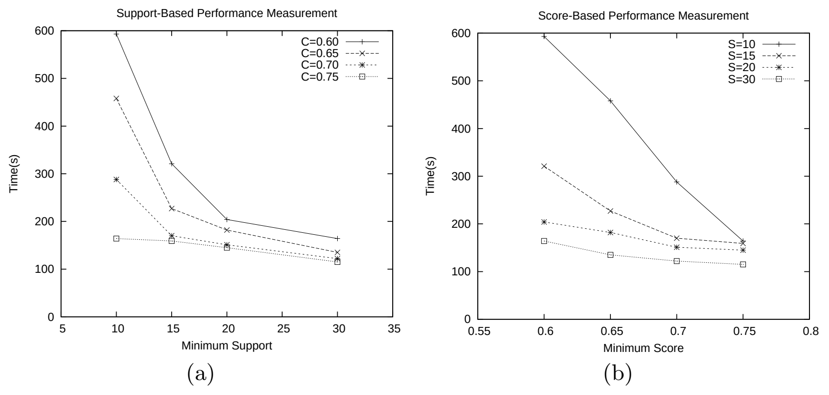


Fig. 2. Performance Measurements

Another interesting question is how these two thresholds affect pruning. Intuitively, lower smaller subgroups and lower minimum scores cause more rule generation, and therefor more rules pruned, but if we examine the percentages of rules pruned we see that with both thresholds it is fairly stable aroung 99.8%. These results are shown in Figure 3. As can be seen, the pruning algorithm is

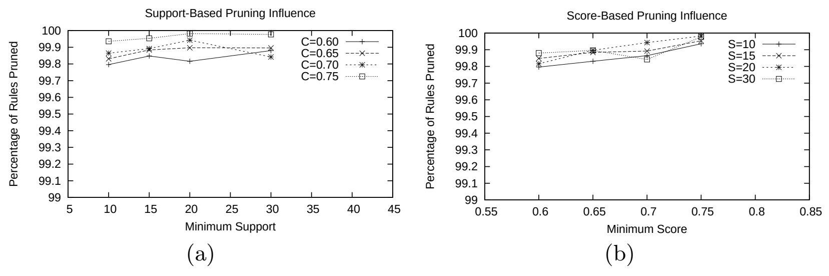


Fig. 3. Pruning Measurements

slightly more eradic in the support cutoff seen in Figure 3(a) than in the score cutoff shown in Figure 3(b), but overall both are monotonically increasing.

5 Conclusions and Future Work

In this paper we discussed a Subgroup Discovery Service called Fantom that finds subgroups given a set of weighed elements. We explained the technologies behind the algorithm, its data sources, and its way of combining that data to generate comprehensive pieces of knowledge that are tailored to the expert knowledge of the researcher.

In our use case, we have shown several statistics on the Golub et al. dataset, which we normalized and then extracted the participating genes and their scores. We have shown that with pruning can be done with both a monotonical constraint as support, but also by adapting a non-monotonical constraint such as GSEA using the support. This yielded in less rule generation and increased pruning, which rendered at least 99.85% of all the rules generated useless. For future work, efforts have to be made to increase rule statistics, not only with GSEA scores, but also p-values for confidence. Furthermore, more score funtion than just GSEA should be present. A wide overview is presented in [AS09], and plans are to at least support a few of them. Of course, a qualitative re-assessment of the rules with different score measures will have to be made then, as well as research into the performance / Quality tradeoff.
For the immediate future, research will be focussed on adapting more ontologies, and adapting FANTOM to more closely resemble related work discussed in [TLT07], [LRS+08] and [ZW08]. Currenty the rules generated in this paper are under reevision by BioInformaticist, who will compare them to rules generated on the same dataset by [TLT07], so that we can also give a qualitative comparison of FANTOM with SEGS.

References

[ABB+00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–29, May 2000.

[AIS93] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.

[AS09] M. Ackermann and K. Strimmer. A general modular framework for gene set enrichment analysis. *BMC Bioinformatics*, 10:47+, February 2009.

[ASS+02] S. A. Armstrong, J. E. Staunton, L. B. Silverman, R. Pieters, M. L. den Boer, M. D. Minden, S. E. Sallan, E. S. Lander, T. R. Golub, and S. J. Korsmeyer. Mll translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nat Genet*, 30:41–47, Jan 2002.

[Bal93] Mira Balaban. The f-logic approach for description languages. *Annals of Mathematics and Artificial Intelligence*, 15:15–19, 1993.

[BIAS03] B. M. Bolstad, R. A. Irizarry, M. Astrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, January 2003.

[Bod03] Ferenc Bodon. A fast apriori implementation. In *FIMI*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[Dav06] J. Davies. *Semantic Web Technologies: Trends and Research in Ontology-based Systems.* John Wiley & Sons, July 2006.

[Gen] Generif – gene reference into functions. http://www.ncbi.nlm.nih.gov/projects/GeneRIF/.

[go-] Mappings of external classification systems to go. http://www.geneontology.org/GO.indices.shtml.

[GST+99] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, October 1999.

[HPY00] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD Conference*, pages 1–12. ACM, 2000.

[keg] Kegg patway mappings. ftp://ftp.genome.jp/pub/kegg/pathway/map.

[LB06] Elo Leung and Pierre R. Bushel. Page: phase-shifted analysis of gene expression. *Bioinformatics*, 22(3):367–368, 2006.

[LRS+08] Y. Lu, R. Rosenfeld, I. Simon, G. J. Nau, and Z. Bar-Joseph. A probabilistic generative model for go enrichment analysis. *Nucl. Acids Res.*, pages gkn434+, August 2008.

[MCOW05] X. Mao, T. Cai, J. G. G. Olyarchuk, and L. Wei. Automated genome annotation and pathway identification using the kegg orthology (ko) as a controlled vocabulary. *Bioinformatics*, April 2005.

[MOPT05] D. Maglott, J. Ostell, K. D. Pruitt, and T. Tatusova. Entrez gene: gene-centered information at ncbi. *Nucleic Acids Res*, 33(Database issue), January 2005.

[MyG] MyGrid. Taverna workbench 2.0. http://taverna.sourceforge.net/.

[OAM+03] Angele Moench Oppermann, J. Angele, E. Moench, S. Staab, and D. Wenke. Ontonova @ project halo. In *in Proceedings of the Second International Semantic Web Conference (ISWC2003). 2003*, pages 913–928. Springer Verlag, 2003.

[OGS+99] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*, 27(1):29–34, January 1999.

[Ont] Ontology definition in information science. http://www.computer-dictionary-online.org/ontology.htm?q=ontology.

[OWL] Web ontology language (owl). http://www.w3.org/2004/OWL/.

[PTM07] K. D. Pruitt, T. Tatusova, and D. R. Maglott. Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res*, 35(Database issue):D61–D65, January 2007.

[RDF] Resource description framework (rdf). http://www.w3.org/RDF/.

[TLT07] I. Trajkovski, N. Lavrač, and J. Tolar. Segs: Search for enriched gene sets in microarray data. *J Biomed Inform*, December 2007.

[TS06] C. Todorova and K. Stefanov. Selection and use of domain ontologies in learning networks for lifelong competence development. In *Proceedings of the 2006 International Workshop on Learning Networks for Lifelong Competence Development*, pages 11–17. Springer Verlag, 2006.

[TZTL06] I. Trajkovski, F. Zelezný, J. Tolar, and N. Lavrac. Relational subgroup discovery for descriptive analysis of microarray data. In *CompLife*, volume 4216 of *Lecture Notes in Computer Science*, pages 86–96. Springer, 2006.

[VDS+07] I. Vastrik, P. D'Eustachio, E. Schmidt, G. Joshi-Tope, G. Gopinath, D. Croft, B. de Bono, M. Gillespie, B. Jassal, S. Lewis, L. Matthews, G. Wu,

E. Birney, and L. Stein. Reactome: a knowledgebase of biological pathways and processes. *Genome Biology*, 8:R39+, March 2007.

[Wai] Lush M. Ducluzeau F. Povey S. Wain, H. M.

[XML] Extensible markup language (xml) 1.0 (fifth edition). http://www.w3.org/TR/REC-xml/.

[ZW08] Qi Zheng and Xiu-Jie J. Wang. Goeast: a web-based software toolkit for gene ontology enrichment analysis. *Nucleic acids research*, May 2008.

Definition of a Metadata Schema for Describing Data Preparation Tasks

Miguel Hidalgo, Ernestina Menasalvas[⋆] and Santiago Eibe[⋆⋆]

Facultad de Informática, Universidad Politécnica de Madrid, Spain
mahidalgo@alumnos.upm.es, emenasalvas@fi.upm.es, seibe@fi.upm.es

Abstract. The effort involved during data preparation phase needs to be minimized through the automation of preparation tasks as possible. Within the mobile and dynamic environment we interact, the changing nature of data and the speed it is generated, demands that autonomy be a top priority. Based on this idea, we aim to address the automation of data preparation tasks in ubiquitous environments. With this goal in mind, we think it's necessary to identify requirements for a data preparation task, and the semantics related to the expertise of a user as well. In this paper, we propose a number of steps to gather the requirements for a documented data preparation task and we define a metadata schema that would allow the creation of semantics to support this process. At the end, a case study demonstrates the application of such metadata schema, however its continuous application to complex tasks, will determine if an extension would be needed.

1 Introduction

Although data miners utilize the well-known CRISP-DM[1] process model, they know that data preparation phase is not trivial because it demands a lot of expertise. The number of tasks in the data preparation phase varies according to the objective of a particular data mining project, but the tasks are usually group into cleaning, construction, integration and formatting tasks[2]. Due to data preparation phase represents the previous step when applying mining algorithms, the interaction with modeling phase occurs frequently.

On one hand, CRISP-DM doesn't indicate how to perform a data preparation task, and on the other hand, it's necessary to minimize the human participation when executing such preparation tasks. In order to benefit both, expert and non-expert users, the automation of data preparation tasks will reduce the time and effort involved. For accomplish this, we think it's necessary to indicate how to proceed with the execution of a data preparation task, in particular, or any other data mining task, in general.

Nowadays, most of the people interact in a mobile environment. The mobile devices like sensors or wireless networks produce huge amounts of data that can

[⋆] Project partially financed by Project TIN2008-05873
[⋆⋆] Project partially financed by Project CCG06-UPM/ESP-0259

be analyzed with the aim of enriching the patterns obtained. In turn, mobile devices like cell phones or personal digital assitants (PDA) have processing capabilities that represent a new framework to achieve some tasks. These facts represent an interesting opportunity for data mining, and the relation between data mining & mobile devices, has derived in a scenario called *ubiquitous data mining*[3]. The realization of the data preparation phase within ubiquitous environments, forces us to re-think how to achieve the preparation tasks with certain degree of autonomy.

In order to achieve it, we think that the utilization of annotations can greatly support the automation of a data preparation task. The benefits can be double: the automation of a data preparation task in ubiquitous environments, as the main goal, and the assistance to users during the realization of such tasks, as a secondary matter. With the main goal in mind, it's necessary to find out what sort of information will adequately describe a data preparation task. We propose a categorization of the information needs for a data preparation task with the aim of classifying every requirement accordingly.

In this paper, we also specify a number of steps to gather the information needs, in the form of metadata, for a data preparation task. The steps utilize a modeling language and a XML-based metadata schema, in order to describe the actions performed by an expert when executing a data preparation task. In consequence, the experience obtained in the data preparation phase could motivate the extension to other parts of the data mining process.

The rest of the paper is organized as follows: In section 2 we present related work with the description of the data mining process and we briefly stress what PMML elements are related to data preparation phase. Section 3 explains the proposed steps to model a data preparation task, as well as the XML-based metadata schema that supports its description. Then in section 4, a case study shows the resulting metadata for an already documented data preparation task. Finally, section 5 presents conclusions and ongoing work.

2 Related Work

Krishnaswamy et al.[4] propose the creation of services among data mining federation systems. The DDMS-ML language defines the functionality and characteristics of a data mining system, and its section *Specializations and Features* is used to describe services. However, the descriptions emphasize general categories like pre-processing, visualization or algorithms. The preprocessing service is not fully described, because the architecture focuses on the distributed nature of data mining systems. As a result, there are not enough elements to address a data preparation task.

MiningMart[5] is a framework based on KDD test-cases that defines the needed steps for a pre-processing chain through a metadata model. Metadata represent pre-processing steps that utilize operators that are executed by a compiler. To translate a pre-processing chain, the compiler relates each conceptual entity (operators for a task) to its relational entity (business data), establishing the parameters and the expected output. A similarity of MiningMart with our research is the fact that we consider metadata as a mechanism to guide the achievement of data mining tasks.

KDDML[6] is a XML-based language used to represent data, models and queries in high level data mining systems. The data mining process is seen as a query, where syntactical elements of a KDDML document represent operations on data or passing parameters among those operations. For data pre-processing, KDDML utilize a fixed group of operators already defined, such as: *data selection, filtering or aggregation*. The operators use a function that defines input and output parameters in the form of attributes. Due to operator's number is low, creation of new ones requires to extend DTD in order to use them.

DMPML[7] is another XML-based framework that establishes a language for the data preparation phase. To describe a data mining task, it proposes the creation of several XML files. This separation corresponds to original data, data preparation project directives and transformed data that will be submitted to a data mining technique. The framework uses XSLT transformation rules to map original data to transformed data, in order to be utilized according to different data mining algorithms. We agree to the use of XML for description purposes. However, we extend such descriptions to consider the execution flow of a preparation task.

Othman et al.[8] propose a framework for data preparation based on agents. The main advantages of such implementation are the decrease of expert's dependency and the associated cost of the data mining model update. The agents propose the best preprocessing technique according to an appropriate domain. We stress the requirements identification achieved by the authors for pre-processing phase, because it facilitates agent's coordination for a complete task execution.

Within an autonomous agents environment, Rajan et al.[9] identify a group of steps related to data selection and transformation. The framework achieves the identification of data, proper selection of mining algorithm and presentation of results. In particular, the *user interface intelligent agent*, performs tasks like selection of attributes or identification of attribute data type, but complex tasks are not specified because the authors focus on a theoretical methodology.

With regard to data mining standards, PMML[10] supports interoperability of data mining models. However, PMML has several elements related to data preparation. For example, the *data dictionary* element focuses on data fields types and value ranges. The *transformation dictionary* defines various types of simple transformations, e.g. normalization, discretization, aggregation, etc. Other important elements are: *statistics, output, functions and built-in* functions.

3 Steps for Gathering Metadata for Data Preparation

3.1 Objective

The goal of this paper is twofold. On one hand, we present a number of steps that model a data preparation task with the purpose of obtaining its information needs for deriving metadata from them. On the other hand, we explain the development of a XML-based metadata schema, necessary to adapt the task's requirements with the elements defined in this schema. The proposed steps are presented in figure 1.

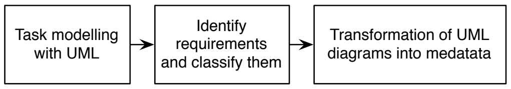

Task modelling with UML
Identify requirements and classify them
Transformation of UML diagrams into medatata

Fig. 1. Steps for identifying metadata for a data preparation task.

3.2 Relevant aspects

Step 1. Task Modeling with UML

- **Utilization of terms: task and activity**. The data mining process is organized in phases according to CRISP-DM. Each phase identifies a group of tasks to perform. Therefore, information needs must focus on a task, because it represents a need in the data mining process. Due to task concept is broaden, it's necessary to break it down into small units called *activities*. Thus, execution of task's activities achieves a task's goal. CRISP-DM establishes a four level hierarchy to carry out tasks in a data mining project. The levels are: phase, generic task, specialized task and process instance[11]. With the purpose of being coherent with CRISP-DM, task and activity terms are related to generic task and specialized task, respectively.
- **Modeling data preparation tasks with UML**. The UML diagrams [12] represent an excellent mechanism to specify inherent requirements for a system, a process or a task. Due to it's feasible to model information needs for a business process with UML, we consider this approach could be equally applied to the data mining process, in particular, to the tasks in each phase[13]. The UML modeling for a data preparation task contributes to the identification of its requirements with the aim of describing it. This will avoid to forget any requirement, or to cause any inconsistency when executing a data preparation task automatically. We propose to utilize use case and activity diagrams. These diagrams represent our starting point and they will model different requirements. For each data preparation task, a pair of UML diagrams will be obtained.
  The use case diagram will model a data preparation task using the elements: *actors, use cases and a template*. The use case diagram will provide a general view of the task. Every use case will have an associated template, and each template will provide the following information: use case name, actor name, preconditions and basic flow. However, a particular template for identifying a preparation task will be created.
  The activity diagram will model the data preparation task through activities, which are mapped from the basic flow described in the templates of the

use case diagram. For example, if a task called *“remove zero values from an attribute”* has one template with eight steps in its basic flow, then its activity diagram will be built with these eight “steps”.

Step 2. Requirements Identification and Classification

- **Requirements for a task**. It’s important to identify what information is needed by a task, with the aim of executing it and obtaining a result. We have to follow an order when working with data preparation tasks and, we need to verify whether a result satisfies the need of another preparation task or not. Every task in the data mining process responds to different and particular needs according to its objective. However, it’s possible to identify general requirements, such as: how to execute a task?, what data it needs?, what is the result of the task?, and what resources do we need to perform it?. To sum up, the information collected through requirements will allow the configuration and execution of a task.
- **Requirements categorization**. An adequate task documentation will contribute to requirements identification, but in fact, there is a lack of documentation for the data preparation tasks. Hence, it’s of paramount importance to establish a categorization of all needed requirements. In this paper, we propose the following categories:

  - Objective category: Indicates the purpose of a data preparation task.
  - Output category: It corresponds to the result of a data mining task execution. It must indicate the type of output obtained.
  - Definition category: Represents identification information for a preparation task, i.e. task name, task identifier and task executor. It’s important to establish the task priority and task restrictions as well. For example, if a task has high priority over another task, it could be executed first.
  - Control category: It corresponds to control elements, e.g. information about input & output parameters for a task and its activities.
  - Flow category: It establishes the execution mode for a task among the rest of tasks in the same phase, e.g. executing two or more tasks in parallel. Besides, it’s necessary to establish an order of how the task’s activities will be executed. For this, XPDL[1] will be used, in particular, its workflow process element[14].
  - Content category: It establishes information of the data sources accessed by a task. It’s important for a task to know how the data source is structured, because in this way, it will possible to determine the attribute’s type or its role in the dataset.
  - Composition category: It describes information about the number and the name of the activities that built a data preparation task.
  - Execution category: It describes information related to the verification of a correct task execution. That is, to indicate if a task’s execution performed correctly or not, and the errors produced in case of failure.

[1] XML Process Definition Language

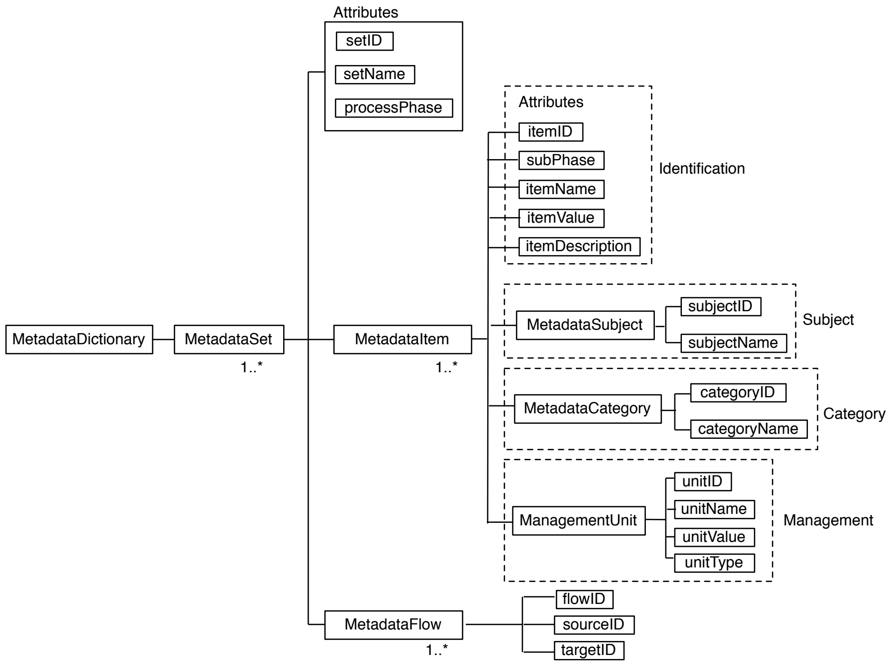


Fig. 2. Metadata schema.

Step 3. Transformation of UML diagrams into metadata

– **Metadata schema definition**. The metadata schema has elements (figure 2) that will represent the requirements of a data preparation task. Every element of the schema has a particular purpose and in general, all of them are related to the semantics of a data preparation task. A special feature of this schema is about *life cycle* elements related to a metadata instance. Such life cycle has the purpose of providing information to update each metadata. The explanation of the schema elements is presented as follows.

  • **MetadataDictionary:** This element is the root of the metadata schema and consists of one or more *MetadataSet* elements.
  • **MetadataSet:** It has been determined that a MetadataSet element is related to the data preparation phase. However, other phases of the data mining process can be considered. A *MetadataSet* consists of three attributes and two elements. The information about the identifier (setID), name (setName) and phase (processPhase) for a *MetadataSet*, is managed by the attributes. With regard to the elements *MetadataItem* and *MetadataFlow*, a MetadataSet can have one or more of them.
  • **MetadataItem:** Each metadata used to describe information for a task or activity is represented with a *MetadataItem* instance. A MetadataItem

  has four sections: *Identification, Subject, Category and Management. Identification* section has five attributes and they refer to an identifier (itemID), the data preparation subphase it belongs (subPhase), its name (itemName), its value (itemValue) and its purpose (itemDescription) *Subject* section is represented by *MetadataSubject.* Its purpose is to indicate the object to be described, e.g. a task or an activity. For each object, an identifier (subjectID) and a name (subjectName) are needed. *Category* section associates each *MetadataItem* with a requirement category. For each category, an identifier (categoryID) and a name (categoryName) are needed. *Management* section relates to the life cycle for a MetadataItem through the *ManagementUnits* element. Each information unit needs an identifier (unitID), a name (unitName), a value (unitValue) and a type (unitType). Some basic ManagementUnits are: creationDate, creationTime, actualValue, previousValue, createdBy, etc.
  - **MetadataFlow:** This element is related to the flow transitions between data preparation tasks or activities in a single preparation task. For each transition, it's necessary to use a global identifier (flowID), a source identifier (sourceID) and a target identifier (targetID).

– **Transformation of diagrams**. According to the steps defined in figure 1, it's necessary to perform a transformation of UML diagrams into metadata with the aim of describing a preparation task. The significance of metadata is demonstrated with the usage[15], content[16] and management of data[17]. In data mining, metadata represent two important aspects:
  - Metadata is considered another type of data susceptible of extracting knowledge[18].
  - Metadata can guide the data mining process[18],[19].

  Once categories for requirements were established, it's possible to indicate what categories will be mapped to use case and activity diagrams.
  The basic template of a use case diagram represents standard information for any use case. However, according to section 3.2 - step 1, we need to create a particular template for identifying a data preparation task. This additional template will collect the information mentioned in the *definition* category. Thus, use case diagram will fulfill the requirements for *definition, flow, objective and output* categories.
  The activity diagram will detail how a task is performed. As mentioned in section 3.2 - step 1, the basic flow described in each use case template will help to develop the activity diagram for the current task. Thus, activity diagram will fulfill the requirements for *composition, control, content, execution and flow* categories.
  So far, transformation process is done manually. With regard to how to name a metadata, the process is the following: we put appropriate words together that semantically have a meaning. For example, for creating a metadata to give a name to a data preparation task, we create *taskName.*

With regard to which elements from the use case and activity diagrams are mapped to the schema of figure 2, the explanation is as follows: Each requirement extracted from the templates of the use case diagram is linked to a category that is mapped to the *MetadataCategory* element. Due to the basic flow is detailed in the activity diagram, each oval from the diagram is mapped to the *MetadataItem* element. The arrows in such diagram, are mapped to the *MetadataFlow* element. The initial node is related to the attribute *sourceID* and the final node is related to the attribute *targetID* (see section 3.2 - step 3 & figure 2 for further details).
With the aim of offering a small set of metadata instances to the user, we consider that the PMML elements related to data preparation phase can represent a basic support.

4 Case Study

In order to apply the steps defined in the previous section, we present a data preparation task. This task is taken from[20] and represents a well-explained and documented example. A dataset of patients is used to predict diabetes, based on the following measurements: age, systolic blood pressure, diastolic blood pressure and weight. The data preparation task consists in binning a numerical variable called *weight*, according to: if weight is less than 60kg, category is *low*; a weight between 60 and 100kg, category is *medium*, and if weight is more than 100kg, category is *high*. The attribute is located in a table called *Patients* and it's stored in a database called *Hospital*. We will refer to this task as *binatt*.

4.1 Example development

We consider that *binatt* has already been executed and the new attribute has been obtained. In a mobile scenario, the device needs the metadata that *binatt* requires, with the purpose that the device conducts the execution of *binatt* in an autonomous way. This is necessary, because any model installed in the device can demand new data to be pre-processed, regardless of its characteristics of memory, battery or CPU.

Step 1. Task modeling with UML. With regard to the identification information of *binatt*, we create a particular template (Table 1). *Binatt* needs the use case and activity diagrams as well. The use case diagram has one use case called *Binning an Attribute*, related to the actor *"Expert"*. However, this use case includes a second use case called *Access Data Source* and a second actor appears: *Database*. With the previous information and the basic flow of each use case, the templates are created (Tables 2 and 3).

Taking into account the basic flow of both use cases, the activity diagram is derived (figure 3). The arrows of the activity diagram represent the transitions among the nine activities.

Table 1. Template for task identification.

Item	**Value**
Identifier	DT1
Name	Binning an attribute
Executor	Expert
Priority	1
Restrictions	None

Table 2. Template for Binning an Attribute use case.

Use case name	Binning an Attribute
Actor	Expert
Preconditions	Execute *Access Data Source* use case
Basic flow	1. Ask attribute to transform 2. Read value 3. Assign it a category 4. Write result

Table 3. Template for Access Data Source use case.

Use case name	Access Data Source
Actor	Database
Preconditions	None
Basic flow	1. Ask data source path 2. Ask database name 3. Ask user ID 4. Ask user password 5. Ask table name

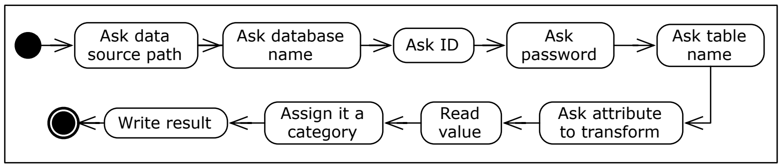


Fig. 3. Activity diagram for *binatt* task.

Step 2. Identify and categorize requirements. All requirements for *binatt* are assigned according to the categories defined in section 3. Hence, the result is as follows:

– Objective category: The requirement is to obtain a new the attribute.
– Output category: The name of the new attribute will be *weightCategorized*.
– Definition category: From the additional template, five requirements are identified. Task identifier is *"DT1"*, task name is *binning an attribute*, task executor is *Expert*, task priority is *1* and the task has no restrictions.
– Control category: Three requirements are identified. The access path is *"/users/desktop"*, the user name is *"us023"* and the password is *"default"*.
– Flow category: The execution mode of the activities is *sequential*.
– Content category: Three requirements are identified. The database name is *Hospital*, table name is *Patients* and the attribute to categorize is *weight*.
– Composition category: The number of activities is nine. These are represented in the activity diagram through ovals.
– Execution category: The result of the execution will have a value of 1 (failure) or 0 (normal).

Step 3. UML modeling transformation into metadata. In this step, the naming process for each requirement takes place. Table 3 shows the MetadataItem instances obtained from this process. Each row in the table represents a MetadataItem with its name and its value. In order to show how a particular MetadataItem fits to the four sections of the schema (figure 2), we detail the MetadataItem called *taskName* (Table 4).

Table 4. Metadata derived from templates and activity diagram.

Category	MetadataItem .itemID	MetadataItem .itemName	MetadataItem .itemValue
Objective	1	taskGoal	Create new attribute
Output	2	taskOutput	weightCategorized
Definition	3	taskName	Binning an attribute
	4	taskID	DP1
	5	taskExecutor	Expert
	6	taskPriority	1
	7	taskRestriction	none
Control	8	accessPath	/users/desktop
	9	userID	us023
	10	userPassword	default
Content	11	DBName	Hospital
	12	tableName	Patients
	13	attributeToCategorize	weight
Composition	14	activitiesNumber	9
Flow	15	executionMode	sequential

Table 5. Further details for the MetadataItem *taskName*.

Section	Prefix: MetadataItem.	Value
Identification	itemID	met-dt-01
	subPhase	data transformation
	itemValue	Binning an attribute
	itemDescription	Name given to a particular task
Subject	MetadataSubject.subjectID	subject1
	MetadataSubject.subjectName	task
Category	MetadataCategory.categoryID	category3
	MetadataCategory.categoryName	Definition
Management	ManagementUnit.unitID	met-mu-01
	ManagementUnit.unitName	creationDate
	ManagementUnit.unitValue	14-01-2009
	ManagementUnit.unitType	static

As a result of the previous steps, the metadata required for *binatt* were generated with the aim of deploy them to the mobile device. This would allow the execution of *binatt* at any time in an autonomous way.

The metadata can be structured in XML-based documents, according to the sections defined in the metadata schema. Each XML document will have all the *MetadataItem* and *MetadataFlow* elements for each data preparation task.

5 Conclusions

The interaction between people and mobile devices produces a lot of data that would be interesting for applying data mining techniques. The data preparation phase in an ubiquitous scenario demands an automation of such tasks. In order to perform it, we need to identify those requirements for each preparation task. In this paper, we have established a set of categories to map such requirements and we have defined a metadata schema that supports the semantics for that sort of tasks. Our proposal has to be applied to more data preparation tasks and some issues require a thorough analysis. The incorporation of a life cycle for metadata represents an interesting approach due to the changing nature of data. The activities for further analysis are: the creation of a relational schema to store MetadataItem instances; the identification of ManagementUnits elements; the identification of XPDL standard execution modes for data preparation tasks, and last but not least, the utilization of a service-oriented approach to offer the execution of a data preparation task in an automatic way. Finally, we have started the development of a software application that checks the validity of a document, according to the metadata schema, and we are working in the incorporation of PMML data preparation elements.

References

1. CRISP-DM. http://www.crisp-dm.org (2008)
2. Han. J, K.M.: Data Mining:Concepts & Techniques. Morgan Kaufmann (2001)
3. Hsu, J.: Data mining trends and developments: The key data mining technologies and applications for the 21st century. The Proceedings of the 19th Annual Conference for Information Systems Educators (ISECON 2002)
4. Krishnaswamy, S., Zaslavsky, A., Loke, S.W.: Federated data mining services and a supporting xml-based language. Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34) **3** (2001)
5. Zücker, R., Kietz, J., Vaduva, A.: Miningmart: Metadata-driven preprocessing. Proceedings of the ECML/PKDD Workshop on Database Support for KDD (2001)
6. Romei, A., Ruggieri, S., Turini, F.: Kddml: A middleware language and system for knowledge discovery in databases. Data & Knowledge Engineering **57**(2) (2006)
7. Goncalves, P., Arnaud, A., Barros, R.: Data mining preparation markup language. AICCSA. IEEE/ACS International Conference on Computer Systems and Applications (2008) 116–125
8. Othman, Z., Hamdan, A., Omar, A., Shuib, K., Liyana, N.: Agent based preprocessing. ICIAS. International Conference on Intelligent and Advanced Systems (2007) 219–223
9. Rajan, J., Saravanan, V.: A framework of an automated data mining system using autonomous intelligent agents. ICCSIT. International Conference on Computer Science and Information Technology (2008) 700–704
10. DMG: *Predictive Modeling Markup Language.* http://dmg.org/v4-0/GeneralStructure.html (2009)
11. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R.: Crisp-dm:step-by-step data mining guide. CRISP-DM (2000)
12. Miles, R., Hamilton, K.: Learning UML 2.0. O'Reilly Media, Inc. (2006)
13. Marban, O., Mariscal, G., Menasalvas, E., Segovia, J.: An engineering approach to data mining projects. LNCS **4881** (2007) 578
14. Workflow Management Coalition: Process Definition InterfaceXML Process Definition Language, 2.1. Wfmc-tc-1025 edn. (2008) http://www.wfmc.org/.

15. Stephens, R.: Utilizing metadata as a knowledge communication tool. Proceedings IPCC. International Professional Communication Conference (2004) 55–60
16. Orso, A., Harrold, M., Rosenblum, D.: Component metadata for software engineering tasks. 2nd Int. Workshop on Engineering Distributed Objects (2001) 129–144
17. Marco, D.: Building and managing the Meta Data Repository: A Full Life-Cycle Guide. John Wiley & Sons, Inc. New York, NY, USA (2000)
18. Thuraisingham, B.M.: Xml Databases and the Semantic Web. CRC Press (2002)
19. Nocke, T., Schumann, H.: Meta data for visual data mining. Proceedings Computer Graphics and Imaging, CGIM **2** (2002)
20. Myatt, G.: Making Sense of Data: A Practical Guide to Exploratory Data Analysis and Data Mining. Wiley-Interscience (2007)

A Data Mining Ontology for Algorithm Selection and Meta-Mining

Melanie Hilario, Alexandros Kalousis, Phong Nguyen, Adam Woznica

University of Geneva, Department of Computer Science
Artificial Intelligence Laboratory
CUI - 7, route de Drize, CH-1227 Carouge, Switzerland
{Melanie.Hilario, Alexandros.Kalousis, Phong.Nguyen, Adam.Woznica}@unige.ch

Abstract. Given a learning task, the standard approach is to experiment with a broad range of algorithms and parameter settings, and select the model which performs best according to some performance criterion. One of the aims of meta-learning is to at least restrict the space of candidate models by exploiting insights gained from previous experiments. This has been done over the years by correlating dataset characteristics with the observed performance of algorithms viewed as black boxes. We have started to pry open these black boxes to sort out salient algorithm features such as the structure and parameters of the models built, the data partitions effected in data space, the cost function used and the optimization strategy adopted to minimize this cost function. The immediate goal is to build a data mining ontology formalizing the key components that together compose an algorithm's inductive bias. Based on this ontology, a meta-learner could infer algorithm selection guidelines by correlating an algorithm's intrinsic bias with empirical evidence of its performance.

1 Introduction

The medium-term goal of the work reported in this paper is to build an e-Laboratory for Interdisciplinary COllaborative research (e-LICO) in data mining and data-intensive sciences. The proposed e-lab comprises three layers: the e-science and data mining layers form a generic research environment that can be adapted to different scientific domains by customizing the application layer. The e-science infrastructure integrates semantic web technologies for resource sharing and integration to support collaborative scientific research. The main innovation of the data mining (DM) layer is a self-improving, planner-based DM assistant. Improvement with experience is ensured by a meta-miner that thrives on data and meta-data collected from groups of committed scientists. The term meta-mining specifically designates meta-learning applied not only to the learning phase, but to the complete knowledge discovery process, in particular to all tasks that require search in the space of applicable methods. The DM assistant draws its intelligence not only from its planning and meta-mining capabilities but also from the wealth of domain-specific and domain-independent knowledge at its disposal.

A key source of domain-independent knowledge is the data mining ontology (DMO), which can be viewed as the repository of the intelligent assistant's data mining expertise. The DMO plays a major role throughout the lifecycle of the DM e-lab. As a compendium of knowledge about DM tasks, algorithms, data and models, it will be used: 1) to plan the DM process using hierarchical task networks and generate alternative workflows; 2) to guide algorithm and model selection for critical tasks such as learning and dimensionality reduction; 3) to meta-mine experimentation records in order to improve algorithm and model selection; 4) to provide a controlled vocabulary for semantic annotation of DM tools and services offered in e-LICO. Use of the DMO to plan the knowledge discovery process is discussed in [17]. This paper describes how the DMO has been designed to support algorithm selection and meta-learning. Section 2 proposes a novel approach to the algorithm selection problem; Section 3 discusses related work concerning algorithm selection, meta-learning, and DM ontologies. Sections 4 and 5 give an overview of the DMO and its conceptualization of tasks and methods to support algorithm selection and meta-learning. Section 6 concludes with a discussion of major open issues and future work.

2 Algorithm selection and meta-learning

It is now a matter of consensus that no learning algorithm can outperform all others across broad classes of problems and domains [26]. Thus an essential step in any machine learning experiment is selecting the algorithm that will perform best for a given task and data set. As pointed out in a recent survey [24], research on algorithm selection finds its origins outside machine learning, in a broader framework that cuts across diverse areas of mathematics and computer science. In 1976 a seminal paper by John Rice [23] proposed a formal model comprising four components: a problem space $\mathcal{X}$ or collection of problem instances describable in terms of features defined in feature space $\mathcal{F}$, an algorithm space $\mathcal{A}$ or set of algorithms considered to address problems in $\mathcal{X}$, and a performance space $\mathcal{P}$ representing metrics of algorithm efficacy in solving a problem. Algorithm selection can then be formulated as follows: Given a problem $x \in \mathcal{X}$ characterized by $f(x) \in \mathcal{F}$, find an algorithm $\alpha \in \mathcal{A}$ via the selection mapping $S(f(x))$ such that the performance mapping $p(\alpha(x)) \in \mathcal{P}$ is maximized. A schematic diagram of the abstract model is given in Fig. 1.

In Rice's model, selection mapping from problem space $\mathcal{X}$ onto algorithm space $\mathcal{A}$ is based solely on features $f \in \mathcal{F}$ over the problem instances. In machine learning terms, the choice of the appropriate induction algorithm is conditioned solely on the characteristics of the learning problem and data. Strangely, meta-learning research has independently abided by the same restriction from its inception to

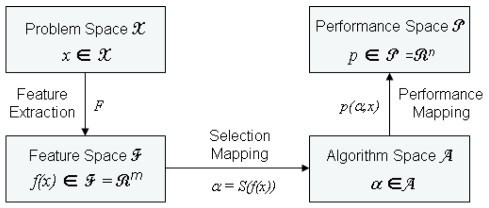

Problem Space $\mathcal{X}$
$x \in \mathcal{X}$
Feature Extraction
F
Feature Space $\mathcal{F}$
$f(x) \in \mathcal{F} = \mathfrak{R}^m$
Selection Mapping
$\alpha = S(f(x))$
Algorithm Space $\mathcal{A}$
$\alpha \in \mathcal{A}$
$p(\alpha,x)$
Performance Mapping
Performance Space $\mathcal{P}$
$p \in \mathcal{P} = \mathfrak{R}^n$

Fig. 1. Rice's model of the algorithm selection problem. Adapted from [23,24]

the present. Learned meta-rules are generally of the form: *if the given dataset has characteristics C_1, C_2, ..., C_n, then use algorithm A_1*. Sometimes the conclusion can take other forms such as *"don't use algorithm A_2" or "prefer A_1 to A_2"*; in all cases, however, these rules represent mappings from data set features to algorithms viewed essentially as black boxes.

So far no attempt has been made to correlate dataset and algorithm characteristics, in other words to understand which aspects of a given algorithm explain its expected performance given the features of the data to be modelled. As a consequence, current meta-learners cannot generalize over algorithms as they do over data sets. To illustrate this problem, suppose that three algorithms are observed to achieve equivalent performance on a collection of datasets representing a task family. Meta-learning would yield three disjunctive rules with identical conditions and distinct recommendations. There would be no way of characterizing in more abstract terms the class of algorithms that would perform well on the given task domain. In short, no amount of meta-learning would reap fresh insights into the commonalities underlying the disconcerting variety of algorithms.

To overcome this difficulty, we propose to extend the Rice framework and pry open the black box of algorithms. To be able to differentiate similar algorithms as well as detect deeper commonalities among apparently unrelated ones, we propose to characterize them in terms of components such as the model structure built, the objective functions and search strategies used, or the type of data partitions produced. This compositional approach is expected to have two far-reaching consequences. Through a systematic analysis of all the ingredients that constitute an algorithm's inductive bias, meta-learning systems (and data miners in the first instance) will be able to infer not only which algorithms work for specific data/task classes but—more importantly—why. In the long term, they should be able to operationalize the insights thus gained in order to combine algorithms purposefully and perhaps design new algorithms. This novel approach to algorithm selection is not limited to the induction phase; it should be applicable to other data and model processing tasks that require search in the space of candidate algorithms. The proposed approach will also be adapted to model selection, i.e., finding the specific parameter setting that will allow a given algorithm to achieve acceptable performance on a given task. This will require an extensive study of the parameters involved in a given class of algo-

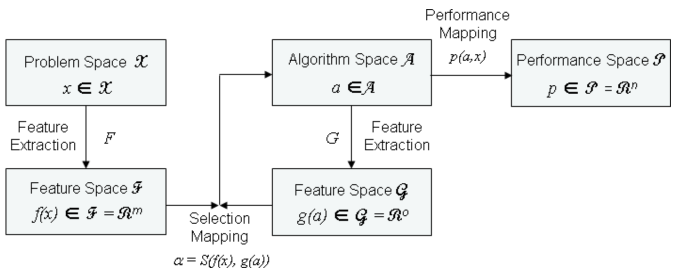


Fig. 2. Proposed model for algorithm selection

rithms, their role in the learning process or their impact on the expected results (e.g., on the complexity of the learned model for induction algorithms), and their formalization in the data mining ontology.

The proposed revision of Rice's model for algorithm selection is visualized in Fig. 2. It includes an additional feature space $\mathcal{G}$ representing the space of features extracted to characterize algorithms; selection mapping is now a function of both problem and algorithm features. The revised problem formulation now is: Given a problem $x \in \mathcal{X}$ characterized by $f(x) \in \mathcal{F}$ and algorithms $a \in \mathcal{A}$ characterized by $g(a) \in \mathcal{G}$, find an algorithm $\alpha \in \mathcal{A}$ via the selection mapping $S(f(x), g(a))$ such that the performance mapping $p(a(x)) \in \mathcal{P}$ is maximized.

3 Related work

Of the few **data mining ontologies** reported in the literature, the majority focus on planning the DM process and building workflows [4,28,27], sometimes in the specific context of Grid computing [8,7]. We shall not delve into their content which is not directly relevant to the focus of this paper. A recent paper proposes a data mining ontology aimed at "the unification of the field of data mining" [20] but defines no specific use case that it is intended to support. To our knowledge, the DMO is the first data mining ontology that has been designed to support, among other tasks, algorithm/model selection and meta-learning.

Algorithm and model selection in data mining has been the object of intensive experimentation and large-scale comparative studies, a comprehensive review of which is outside the scope of this paper. (e.g., [19,14,18,9,12]). More interestingly, choosing the right algorithm and parameter setting has been cast as a learning problem in itself: **meta-learning** for algorithm and model selection has been an active area of investigation for the past two decades [22,1,6,16,13,2]. As pointed out in Section 2, most research on this topic has been implicitly done within the bounds of Rice's framework, where black-box algorithms are selected based solely on problem/data descriptions. An important body of meta-learning research has been devoted to dataset characterization. The Statlog project [19] yielded several dozen dataset features grouped into three categories: simple counts (e.g., number of instances, features or classes), statistical measures (e.g., feature covariance) and information-theoretic measures (e.g., feature entropy). Thereafter, other researchers have tried to expand this set by exploring new features that might yield clues on which algorithms work best for which dataset characteristics [11,10].

The use of landmarking [21,3] in the METAL project gave new impetus to the study of algorithm performance on datasets. This approach uses two sets of algorithms: so-called landmarkers and the actual candidate algorithms. Landmarkers are simple and fast learners (preferably with different inductive biases) whose performance on a set of different learning tasks serve to chart the space of learning problems. To generate meta-rules, both landmarkers and candidates are trained and evaluated on a given set of datasets. Each learning task/dataset then becomes a meta-learning instance which is characterized, in addition to standard predictive features, by the different landmarkers' performance scores. The label of each meta-instance is the candidate algorithm with the best performance measure. The meta-learner is then trained to predict the winning algorithm by identifying tasks in which landmarkers' performance correlate with that of a particular candidate. An example of a learned meta-rule is: *If error-LINEAR-DISCR* $\leq$ *0.0652* $\vee$ *(num-inst* $\geq$ *10* $\wedge$ *num-classes* $\geq$ *5* $\wedge$ *maxclass* ≥ 0.547*) then choose LTREE, else choose RIPPER* [21]. However, despite the use of learners to landmark areas of expertise of other learners, no attempt is made to explain observed performance of algorithms on the basis of landmarkers' or their own characteristics. In landmarking, as before, learners remain black boxes.

4 The data mining ontology

As indicated in Section 1, the DMO is meant to support a number of use cases. This section presents a specific view of DMO based on the algorithm selection use case. The most important competency questions that the ontology should be able to answer include the following: Given a data mining task/data set, what is the set of potentially applicable methods/algorithms? Given a set of candidate methods/algorithms for a given task/data set, which data set characteristics should be taken into account in order to select the most appropriate one? Given a set of candidate methods/algorithms for a given task/data set, which method/algorithm characteristics should be taken into account in order to select the most appropriate one?

The DMO is currently being developed in OWL2 using the Protégé 4 editor. To support algorithm selection, it provides a conceptualization of data mining tasks, methods/algorithms and datasets. The task hierarchy is divided into two major subtrees: the first represents the user task which is more relevant to the planning use case described in [17], while the concept of GenericDMTask subsumes four major task classes: data processing, modelling, model transformation, and model evaluation. Since the focus of this paper is on algorithm selection for classification, Fig. 3 shows an extract of the ModellingTask hierarchy where PredictiveModellingTask subsumes three subclasses distinguished by the data type of their output: categories for classification, scalars for regression, and complex objects (e.g., tuples, trees) for structured prediction.

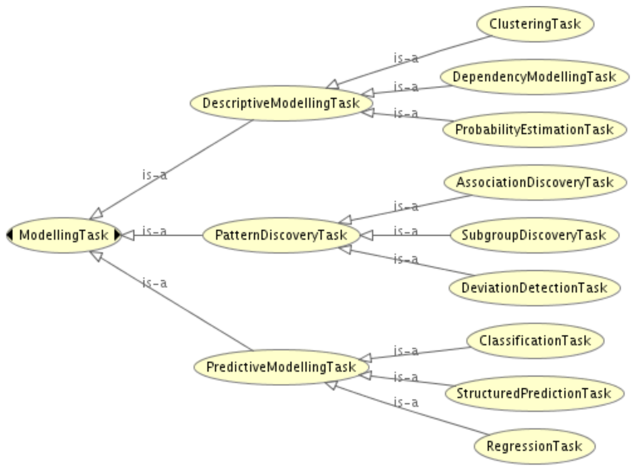


Fig. 3. The ModellingTask subtree

For each leaf class of the task hierarchy, there is a corresponding Method subtree whose branches represent broad classes of methods that address the task. For instance, classification methods can be divided into three broad categories [5] that form the main branches of the ClassificationMethod subtree (Fig. 4). *Generative methods* compute the class-conditional densities $p(\mathbf{x}|C_k)$ and the priors $p(C_k)$ for each class C_k, then use Bayes' theorem to find posterior class probabilities $p(C_k|\mathbf{x})$. They can also model the joint distribution $p(\mathbf{x}, C_k)$ directly and then normalize to obtain the posteriors. In both cases, they use statistical decision theory to determine the class for each new input. Examples of generative methods are normal discriminant analysis and Naive Bayes. *Discriminative methods* such as logistic regression compute posteriors $p(C_k|\mathbf{x})$ directly to determine class membership. *Discriminant functions* build a direct mapping $f(\mathbf{x})$ from input $\mathbf{x}$ onto a class label; neural networks and support vector machines (SVMs) are examples of discriminative methods.

5 Algorithm characterization in the DMO

The DMO's conceptualization of learning algorithms hinges on the 4-tuple of concepts (Task, DataSet, Method, Model). For instance, a ClassificationTask is

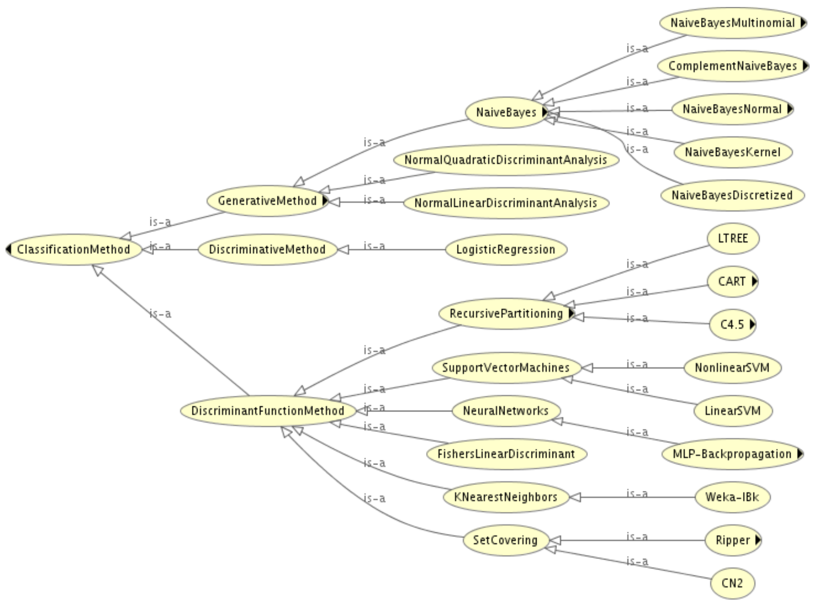


Fig. 4. The ClassificationMethod subtree

achieved by applying a ClassificationMethod to a LabelledDataSet, producing a ClassificationModel. As we go down the classification method subtree in Fig.4, the broad approaches described in the previous section split into more specialized methods which in turn give rise to formally specified algorithms such as those on the right side of the figure. In ontological terms, these specific method subclasses are simultaneously declared as instances of the Algorithm meta-class; their subclasses represent operators, defined as concrete software implementations of algorithms. In the same vein, these subclasses are themselves instances of the Operator meta-class. For example, DiscriminantFunctionMethod subsumes RecursivePartitioning which in turn subsumes algorithms LTREE, CART and C4.5. The black triangle to the right of C4.5 depicts its (hidden) subclasses, operators Weka-J48 and RapidMiner-DecisionTree.

We now zoom in on the key components of classification algorithms; these are represented by datatype and object properties of Algorithm instances. For the purposes of this paper, we focus on characterizing how algorithms work and ignore shallow algorithm characteristics such as ease of implementation, computational cost or readability. To do this, we must first characterize the models they were designed to produce.

A ClassificationModel is defined by its ModelStructure and by the ModelParameters that instantiate this basic structure. It is the ModelStructure that distinguishes the major classification models: a GenerativeModel's basic structure is a JointProbabilityDistribution, that of a DiscriminativeModel is a PosteriorProbabilityDistribution. DiscriminantFunctionMethods produce diverse model structures such as decision trees and neural networks, depending on the nature of the mapping function. Within each model family, a variety of models are produced by coupling the model structure with different types/values of model parameters. To see this, consider the difference between linear and quadratic discriminant analysis under the Gaussian assumption. The NormalQuadraticDiscriminantModel has the same model structure and first model parameter as NormalLinearDiscriminantModel shown in Fig. 5. However, its second model parameter is not a single SharedCovarianceMatrix, but as many class-specific covariance matrices as there are classes in the given dataset. The outcome is a major difference in the geometry of the resulting models: one draws a linear (value of the doesDataSplit property, Fig. 5) and the other a quadratic boundary between the classes.

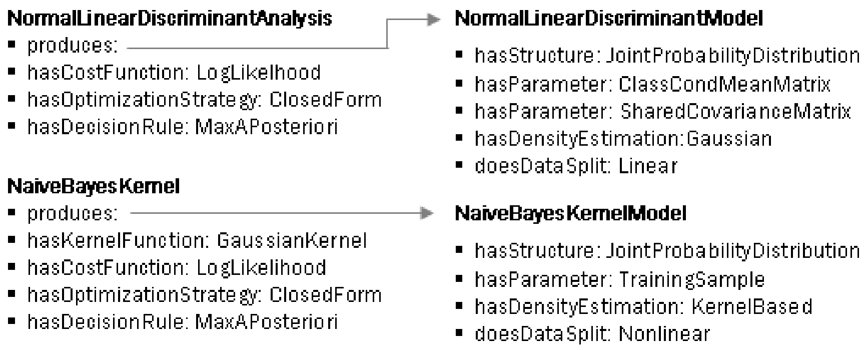

NormalLinearDiscriminantAnalysis
produces:
hasCostFunction: LogLikelihood
hasOptimizationStrategy: ClosedForm
hasDecisionRule: MaxAPosteriori
NormalLinearDiscriminantModel
hasStructure: JointProbabilityDistribution
hasParameter: ClassCondMeanMatrix
hasParameter: SharedCovarianceMatrix
hasDensityEstimation:Gaussian
doesDataSplit: Linear
NaiveBayesKernel
produces:
hasKernelFunction: GaussianKernel
hasCostFunction: LogLikelihood
hasOptimizationStrategy: ClosedForm
hasDecisionRule: MaxAPosteriori
NaiveBayesKernelModel
hasStructure: JointProbabilityDistribution
hasParameter: TrainingSample
hasDensityEstimation: KernelBased
doesDataSplit: Nonlinear

Fig. 5. Characterization of two generative algorithms and models

In probabilistic (generative and discriminative) models, another property that further specifies the model structure to yield diverse models is the DensityEstimationMethod used. For instance, although the two generative models in Fig. 5 use a JointProbabilityDistribution structure, the NormalLinearDiscriminantModel uses a Gaussian distribution whereas NaiveBayesKernel [15] estimates a non-parametric distribution by fitting a Gaussian kernel around each training instance. This entails clear differences in the type of model parameters: the sufficient statistics of the estimated Gaussian distribution are the NormalLinearDiscriminantModel's mean and covariance matrices, whereas those of NaiveBayesKernelModel are all the training instances.

Given a model structure and its parameters, the learning process is nothing more or less than the automated adjustment of these parameters to produce a fully specified, operational model. This is the task of the learning algorithm. The goal is to determine the set of parameter values that will maximize classification performance as gauged by some criterion. Independently of the manner in which the learned model will be evaluated after the learning process, the learner should define a cost (or objective) function, which quantifies how close the current parameter values are to the optimum. Learning stops when the cost function is minimzed. In its simplest version, the cost function can be simply some measure of error or more generally of loss (e.g. misclassification rate or sum of squared errors). However, minimizing training set error can lead to overfitting and generalization failure. The more general concept of CostFunction used in the DMO can be formalized as $F = \epsilon + \lambda c$, where ϵ is a measure of loss, c is a measure of model complexity, and λ is a regularization parameter which controls the trade-off between loss and complexity. The components of the cost function used in SVM learning are shown in Fig. 6.

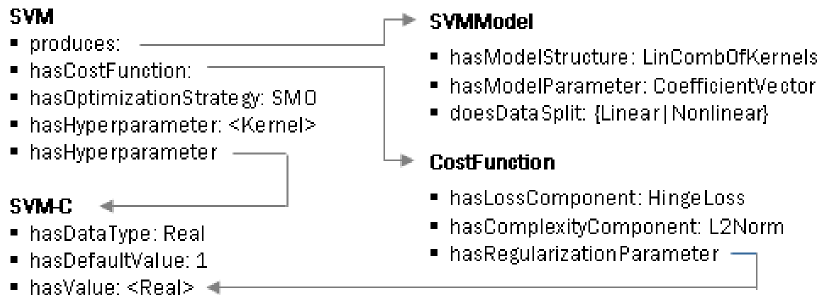

SVM
produces:
hasCostFunction:
hasOptimizationStrategy: SMO
hasHyperparameter: <Kernel>
hasHyperparameter
SVM-C
hasDataType: Real
hasDefaultValue: 1
hasValue: <Real>
SVMModel
hasModelStructure: LinCombOfKernels
hasModelParameter: CoefficientVector
doesDataSplit: {Linear|Nonlinear}
CostFunction
hasLossComponent: HingeLoss
hasComplexityComponent: L2Norm
hasRegularizationParameter

Fig. 6. Characterization of a discriminant function algorithm and model

The search for the right setting can be cast as an optimization problem that consists in minimizing the cost function. Hence an OptimizationStrategy is another essential component of a learning algorithm. In certain cases, optimization is straightforward. This is the case of NormalLinearDiscriminantAnalysis (Fig. 5), where the cost function is the log likelihood, and the maximum likelihood estimates of the model parameters have a closed form solution: it suffices to take the derivatives of the log likelihood with respect to the different parameters, set them to 0, and solve for the parameters. Logistic regression, on the other hand, estimates the maximum likelihood parameters using methods such as Newton-Raphson. SVMs use Sequential Minimal Optimization (SMO), a quadratic programming method rendered necessary by the quadratic complexity component of the cost function (L_2 norm in Fig. 6).

A learning algorithm's model structure and its strategy for finding the optimal model parameters are essential ingredients of its inductive bias, without which no generalization is possible. Despite such design options that restrict the space of target functions that a learning algorithm can explore, the combinatorics of search remains daunting. Thus many algorithms allow the user to restrict further the space of considered models or steer the search in regions deemed promising. This is the role of hyperparameters: they allow the user to reinforce an algorithm's built-in inductive bias by specifying choices that might be informed by prior knowledge. In SVMs, for instance, a single generic algorithm can give rise to a number of different models based on the hyperparameter values selected by users. One such hyperparameter is the kernel function, which is defined by the kernel type (e.g., polynomial, Gaussian) and its associated parameters: the order or degree of a polynomial kernel, or the bandwidth of a Gaussian kernel. The kernel function selected by the user (depicted as <Kernel> in Fig.6) specifies the LinearCombinationOfKernels that comprises the model structure. Adjustment of the model parameters (the kernel coefficients) is controlled by yet another hyperparameter called C. As shown in the figure, the value of C becomes the regularization parameter that controls the trade-off between error (measured by Hinge Loss) and model complexity (quantified by the L_2 norm of the kernel coefficients). This is expressed in OWL through the SWRL rule: *If SVM(?x) ∧ hasCostFunction(?x, ?y) ∧ hasHyperparameter(?x, ?z) ∧ hasValue(?z, ?c) -> hasRegularizationParameter(?y, ?c).*

6 Conclusion

In this paper we presented our vision of a data mining ontology designed to support meta-learning for algorithm (and subsequently model) selection. Previous research has focused obsessively on aligning experiments and performance metrics while little effort has gone into explaining observations in terms of the internal logic and mechanisms of learning algorithms. In this sense, meta-learning research has remained within the strict bounds of the Rice framework, which relates dataset descriptions to performance of algorithms viewed mainly as black boxes. We propose to extend the Rice model by adding algorithm features to dataset features as parameters of the algorithm selection function. To do this, we need to investigate the building blocks that comprise algorithms in order to reveal commonalities underlying their apparent diversity; more ambitiously, the goal is to identify the components of inductive bias that characterize each algorithm and algorithm family. Key components are: the structure and parameters of the models produced, the cost function used to quantify the appropriateness of a model, and the optimization strategy adopted to find the model parameter values that minimize this cost function.

Ongoing work involves two broad groups of issues. First, we should sort out a number of *ontology engineering* problems. The main hurdle we face concerns the limitations of description logic; we need the power of first-order logic to formulate the underlying mathematics of learning in an ontological framework. However, we must weigh the trade-off between expressive power and interoperability with OWL-based e-science platforms. Collaboration with specialists in formal ontologies is crucial at this point. Second, the priority *data mining* issue is identifying other components of bias for learning algorithms, in addition to those described in this paper. This task concerns classification in the first instance, but could be fruitfully extended to other predictive and descriptive data mining tasks.

This two-pronged research agenda is clearly beyond the reach of a single research group or even of a small-scale European project. The short-term goal is to gather interested data miners and ontology engineers to consolidate the core concepts and orientation of the DMO. The next step will be to show how the DMO can be used to improve algorithm selection through meta-learning. Here again, it is indispensable to establish broad collaborations and leverage the results of teams working actively in the area. For instance, the wealth of meta-data gathered in extensive empirical comparisons [9] and community-based experimentation platforms [25] will certainly help to overcome the well-known bottleneck of meta-data sparsity that has always hindered meta-learning research.

Acknowledgements

This work was supported by the European Union within FP7 ICT project e-LICO (Grant No 231519).

References

1. D. W. Aha. Generalizing from case studies: a case study. In D. Sleeman and P. Edwards, editors, *Proc. of the 9th International Workshop on Machine Learning*, pages 1–10. Morgan Kaufmann, 1992.
2. S. Ali and K. Smith-Miles. A meta-learning approach to automatic kernel selection for support vector machines. *Neurocomputing*, 70(1-3):173–186, 2006.
3. H. Bensusan and C. Giraud-Carrier. Discovering task neighbourhoods through landmark learning performances. In *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 325–330, 2000.
4. A. Bernstein, F. Provost, and S. Hill. Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):503–518, 2005.
5. C. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.
6. P. B. Brazdil and R. J. Henery. Analysis of results. In Michie et al. [19], chapter 10, pages 175–212.
7. P. Brezany, I. Janciak, and A. Min Tjoa. Ontology-based construction of grid data mining workflows. In H. O. Nigro, S. E. Gonzalez Cisaro, and D. H. Xodo, editors, *Data Mining with Ontologies: Implementations, Findings and Frameworks.* IGI Global, 2008.
8. M. Cannataro and C. Comito. A data mining ontology for grid programming. In *Proc. 1st Int. Workshop on Semantics in Peer-to-Peer and Grid Computing, in conjunction with WWW2003*, pages 113–134, 2003.
9. R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.
10. Christian Kopf Charles, Charles Taylor, and Jorg Keller. Meta-analysis: From data characterisation for meta-learning to meta-regression. In *Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support,Meta-Learning and ILP*, 2000.

11. S. J. Cunningham. Dataset cataloging metadata for machine learning applications and research. In *Proceedings of the Sixth International Workshop on AI and Statistics 1997*, Fort Lauerdale, FL, 1997.
12. J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
13. W. Duch and K. Grudzinski. Meta-learning: Searching in the model space. In *Proc. of the Int. Conf. on Neural Information Processing (ICONIP), Shanghai 2001*, pages 235–240, 2001.
14. A. Feelders and W. Verkooijen. *On the statistical comparison of inductive learning methods*, chapter 26, pages 271–279. Springer, 1996.
15. G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In P. Besnard and S. Hanks, editors, *Procs. Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann, 1995.
16. A. Kalousis and M. Hilario. Model selection via meta-learning. *International Journal on Artificial Intelligence Tools*, 10(4), 2001.
17. J. U. Kietz, F. Serban, A. Bernstein, and S. Fischer. Towards cooperative planning of data mining workflows. In *Submitted to the ECML/PKDD-2009 Workshop on Third Generation Data Mining: Service Oriented Knowledge Discovery*, 2009.
18. T. Lim, W. Loh, and Y. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40:35–75, 2000.
19. D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine learning, neural and statistical classification*. Ellis-Horwood, 1994.
20. P. Panov, S. Dzeroski, and L. Soldatova. Ontodm: An ontology of data mining. In *Proceedings of the 2008 IEEE International Conference on Data Mining Workshops*, pages 752–760, 2008.
21. B. Pfahringer, H. Bensusan, and C. Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *Proc. Seventeenth International Conference on Machine Learning, ICML'2000*, pages 743–750, San Francisco, California, June 2000. Morgan Kaufmann.
22. L. Rendell and E. Cho. Empirical learning as a function of concept character. *Machine Learning*, 5:267–298, 1990.
23. J. Rice. The algorithm selection problem. *Advances in Computing*, 15:65–118, 1976.
24. K. A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41(1), 2008.
25. Joaquin Vanschoren, Hendrik Blockeel, Bernhard Pfahringer, and Geoff Holmes. Experiment databases: Creating a new platform for meta-learning research. In *Planning to Learn Workshop, ICML/COLT/UAI-2008*, pages 10–15, Helsinki, July 2008.
26. D. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1381–1390, 1996.
27. Monika Žáková, Petr Křemen, F. Železný, and N. Lavrač. Using ontological reasoning and planning for data mining workflow composition. In *SoKD: ECML/PKDD 2008 workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery*, 2008.
28. Monika Žáková, Petr Křemen, Filip Železný, and Nada Lavrač. Planning to learn with a knowledge discovery ontology. In *Planning to Learn Workshop (PlanLearn 2008) at ICML 2008*, 2008.

Stand on the Shoulders of Giants: Towards a Portal for Collaborative Experimentation in Data Mining

Joaquin Vanschoren and Hendrik Blockeel

Computer Science Dept., K.U.Leuven, Leuven, Belgium

Abstract. Data mining is a science guided by experimentation. Still, there exists only limited support to automate such experimentation, making it a laborious and error-prone process. In this paper, we plead for a more collaborative approach in which researchers build upon prior efforts by other researchers in setting up and running experiments, and in turn share the fruits of their experimentation with the community. To this aim, we propose an online *portal*: a web service which collects annotated datasets, algorithms, experimental setups and millions of empirical evaluations sent to it by data mining tools and individual users, organizes them in one searchable central resource, and assists researchers in setting up and running new experiments. Such a service could drive many useful applications, from online platforms where all stored experiments can be explored in great depth and discovered trends can be discussed, to advanced data mining tools that interact with the service to speed up the setup and execution of experiments and upload new experiments to allow a more in-depth exploration of the acquired results.

1 Introduction

Data mining (DM) is a science guided by the interpretation of experimental results. Whether or not a new technique is noteworthy is assessed by empirical evaluation. Yet, while advances in high throughput computing have made it easier to run ever larger numbers of experiments, this has not been equally matched by tools that facilitate the setup of extensive experimental evaluations. Setting up and running experiments and interpreting their results is still a very labour-intensive and error-prone process, which not only slows down the thorough evaluation of new ideas, but in practice also limits the depth of many evaluations. For instance, algorithms are typically run on only a portion of the available datasets, or important effects such as the impact of algorithm parameters, data properties or preprocessing techniques are not investigated.

Obviously, this raises question marks over the generality of conclusions in many papers. First, a study by Hand [5] has highlighted some of the issues that many comparisons fail to take into account, to the extent that the purported superiority of certain algorithms may well be illusory. A lot depends on which datasets were selected, which evaluation metrics were chosen, whether or not cost functions are involved and last but not least, the researcher's experience with the involved algorithms and her ability to squeeze the best performance from them. Another study by Perlich et al. [9] showed that the number of data points selected from each dataset has a marked effect on the relative superiority of many algorithms and Hoste and Daelemans [6] showed how parameter optimization, data sampling, feature selection and their interaction all have a profound effect on the relative superiority of algorithms. Still, many comparisons do not take these factors into account.

Furthermore, the lack of standardized experimentation also makes it very hard for researchers to disseminate their original experiments, which have much more value beyond their initial interpretation presented in papers. This also holds for the many experiments that do not get published. Indeed, experiments are also used for exploring new ideas, searching for relationships, comparing alternatives or trying to understand the behavior of algorithms [3]. Experiments are the lifeblood of data mining research, and it is unfortunate that these cannot be easily shared.

First, because descriptions of experiments in papers are usually quite vague, making it hard to verify, revise or refine previous conclusions. Different algorithm comparisons may use very different algorithm implementations, datasets may also have been preprocessed without it being clear how, and different scripts to split the data or compute evaluation metrics may yield different results. All this makes it quite easy to find different answers to the same questions, without it being clear what causes the deviating results. Second, even simple questions about the behavior of certain algorithms require the setup of new experiments, while they could perhaps be answered by simply exploring the combined results of many different studies. Third, especially in benchmark studies, the same algorithms are run on the same datasets over and over again, using time and resources that could have been put to better use. And finally, it impedes interaction between different subfields in data mining. For instance, new data preprocessing techniques may greatly enhance the utility of certain learning algorithms, yet, since the preprocessed datasets are not available, the algorithm designer may be hesitant to investigate this because it requires the execution and fine-tuning of techniques she may not be familiar with.

In the remainder of this paper, we motivate how data mining research could benefit greatly from a more collaborative approach to experimentation in Section 2. Next, Section 3 describes how we plan to implement this portal. We propose two ways in which to do this: the first is to provide interfaces for toolboxes and individual algorithm implementations, the second is to design DM techniques as web services which automatically interact with the portal. Finally, Section 4 illustrates the benefits of such a portal in a typical use case.

2 A Collaborative Approach

In the words of Isaac Newton, "If I have seen further it is by standing on the shoulders of giants." Indeed, scientific progress always builds on existing ideas, data and methods, and the fields of machine learning (ML) and data mining are no exception [11]. To accelerate the progress (and reduce the cost) of research, many sciences, such as astronomy, bio-informatics and high-energy physics have started to use shared databases to store very detailed descriptions of various kinds of data and have turned to modern web technologies to use the web as a large, user-driven collaborative workspace [7, 12].

Similarly, we believe that the best way to solve the aforementioned issues in DM and ML is to take a more *collaborative* approach to experimentation, in which experimental procedures, algorithms, datasets and all obtained evaluations are freely exchanged between large groups of researchers, and subsequently reused to speed up and automate this process as much as possible. We plan to realize this by implementing an online community 'portal' for data mining experimentation which interacts with data mining tools to automatically exchange experimental details and resources, and which organizes all this information so it can be easily explored by researchers.

This portal can take the form of a web service that collects algorithms, datasets, experimental setups and results sent to it by data mining tools, organizes them in one searchable central resource, and, upon request, provides the necessary components to quickly setup and run new experiments. Its core consists of a database that stores experimental results in an organized fashion, as well as detailed descriptions of the used algorithms, datasets and experimental setups. Such databases have already been described [2], and allow a thorough exploration of all stored results and meta-information by means of very flexible querying capabilities.

However, when implemented as a web service, it can be expanded into a true platform for collaboration, offering many more services and interacting automatically with other data mining tools and services, so that researchers don't need to manually describe their experiments in order to share them. As such, it should offer interfaces for clients to upload new experiments, resources (such as datasets and algorithms) and meta-information (such as characterizations of datasets), as well as interfaces to answer specific queries over all stored experimental results and meta-information. Moreover, it should also be able to run new experiments, either by providing data mining tools with all information and resources necessary to run the experiments locally, or by automatically calling on algorithms implemented as web services.

Such a portal essentially acts as a window on the behavior of data mining techniques. It surveys all data sources, algorithms and experiments ever submitted, as well as specific domain knowledge such as properties of algorithms or datasets, procedures used for model evaluation, general knowledge about how to set up experiments and so on. All this information can be accessed automatically, by sending requests, or manually by writing queries to it, for instance to test hypotheses, to rearrange the stored data to gain new insights, or to retrieve specific datasets and algorithm implementations.

The rest of this paper is organized as follows. First, we motivate how data mining research could benefit greatly from a collaborative approach to experi-

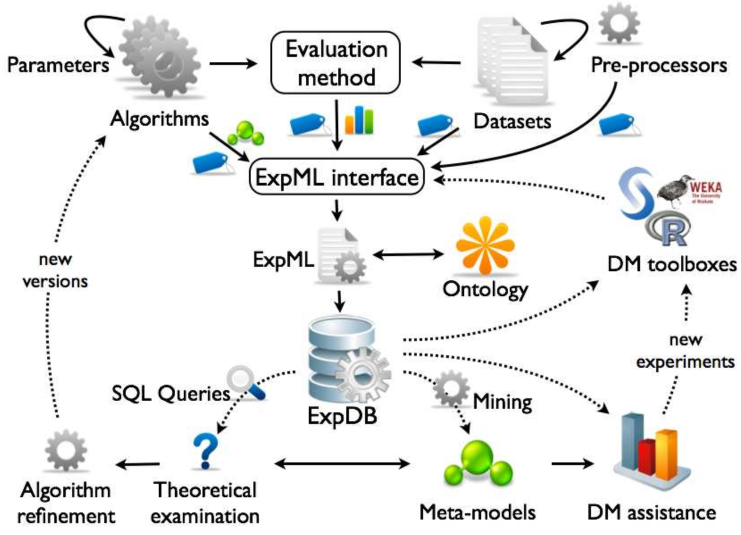


Fig. 1. The architecture of Experiment Databases.

mentation in Section 2. Next, Section 3 describes how we plan to implement this portal. We propose two ways in which to do this: the first is to provide interfaces for toolboxes and individual algorithm implementations, the second is to design DM techniques as web services which automatically interact with the portal. Finally, Section 4 illustrates the benefits of such a portal in a typical use case.

3 Implementation

In this section, we discuss some of the ways in which the portal described above could be implemented. Still, there are many open issues which warrant further investigation.

3.1 Experiment Databases

The implementation we envision builds upon *experiment databases* [2], a platform designed to collect large numbers of data mining experiments. The components of this platform are shown in Fig. 1. First of all, it contains an XML-based language to describe experiments, dubbed ExpML, which is formally described in Vanschoren et al. [15]. It is meant to allow experiments to be exchanged between the database and external data mining tools and web services. The ExpML format describes complete experiment setups, including pointers to the necessary data sources and executables. We show an example in Sect. 3.3. To facilitate the generation of ExpML descriptions by external tools, a Java API is provided to compose the descriptions programmatically and submit them to the system.

The experiments are then stored in a relational database designed to be extensible, scale easily to large numbers of experiments and allow queries on practically any aspect of the experimental setup and outcome. It is implemented in MySQL and currently contains over 600,000 experiments on 67 classification and regression algorithms, 149 different datasets and 2 data preprocessing techniques. Access is provided through web services for submitting experiments and launching queries.

There are several ways to use the database (illustrated at the bottom of Fig.1). First, the platform offers two query interfaces to explore all the stored information. Both are available at `http://expdb.cs.kuleuven.be`, including example queries and video tutorials. The first is a website which offers a query interface and basic online visualizations of the returned results. The second is a stand-alone explorer tool offering more advanced querying and visualization features. In the latter case, the database returns the requested experimental data in the JSON format. SQL queries can be written to test a wide range of hypotheses, provide overviews, or even download lots of experimental data and look for patterns in algorithm performance or to provide advise to users of particular algorithms. The querying abilities have been illustrated in great detail elsewhere [2, 16], so we will not discuss them any further here.

Still, this platform needs to be extended in various ways to offer the functionalities described in Section 1.

3.2 An Open Discussion Platform

A first, simple, extension is to provide a platform where researchers can discuss certain experimental results (whether they were published in papers or not). The result of a query, or any other selection of experiments, can be visualized and posted online to instigate discussion about the observed behavior. This is an easy way for researchers to work together or start discussions about the conclusions of previous studies.

3.3 Ontological Descriptions

An important prerequisite for automated experimentation is an ontology that provides a formal, controlled vocabulary for the setup and results of data mining experiments[1]. They provide clear meaning to various concepts in data mining experimentation, refine certain concepts where necessary and describe their relations with other concepts. For instance, the term 'inductive algorithm' can be either a *specification* (abstract description), an *implementation* (an encoding in a certain programming language) or an *application* (in which the algorithm is given specific parameter settings and run on a training set). Furthermore, algorithm specifications clearly state the structure of those algorithms, consisting of base-learners, kernels, distance functions, internal preprocessing steps, splitting criteria, to name a few. In this work, we have developed an ontology in the

[1] Such ontologies have been an important research goal in bio-informatics[12, 10].

OWL-DL format, dubbed Exposé, built upon previous proposals for ontologies for data mining [4, 8, 18]. It is used to design flexible database schema, provide clear structure and descriptors for ExpML descriptions (in which the meaning of each tag is clearly defined), to allow independently developed systems (e.g. web services) to work together to exchange information and to build clear interfaces for users who wish to query every aspect of the stored data or experimental setups.

An example description of an experiment in the ExpML format is shown below. It this case, we use an implementation of the 'Bagging' algorithm from the WEKA toolbox [17], supply it with parameter settings and a base-learner, and run it on the 'pendigits' datasets from the UCI repository, which is first preprocessed by removing 10% of the data points. Data transformation applications are given a local id so that data transformation workflows can be described using a 'preceeds' relation (which may have multiple targets). The evaluation will happen with a cross-validation procedure, also implemented in WEKA, with the given number of folds and the given random seed used to split the data into train and test sets. It also enlists all evaluation metrics that should be computed.

```
<experiment>
 <learner_evaluation>
  <task>
   <single_target_classification target_attribute="-1" />
  </task>
  <algorithm_appl>
   <algorithm_impl name="Bagging" version="1.31.2.2" libname="weka"/>
   <parameter name="bagsize_percentage" value="90"/>
   <parameter name="nr_iterations" value="40"/>
   <component role="base-learner">
    <algorithm_appl>
     <algorithm_impl name="NaiveBayes" version="1.16" libname="weka"/>
    </algorithm_appl>
   </component>
  </algorithm_appl>
  <input_data>
   <dataset name="pendigits" preceeds="pp1">
    <url>http://archive.ics.uci.edu/ml/</url>
   </dataset>
   <preprocessor_appl id="pp1">
    <preprocessor_impl name="RemovePercentage" version="1.3" libname="weka" />
    <parameter name="downsampling_percentage" value="10"/>
   </preprocessor_appl>
  </input_data>
  <evaluation_method_appl>
   <evaluation_method_impl name="CrossValidation" version="1.53" libname="weka"
        libversion="3.4.8" />
   <parameter name="nbfolds" value="10"/>
   <parameter name="randomseed" value="1"/>
  </evaluation_method_appl>
  <evaluation_metric_appl name="build_cputime">
   <evaluation_metric_impl name="build_cputime" version="1" libname="expdb" />
  </evaluation_metric_appl>
  <evaluation_metric_appl name="predictive_accuracy">
   <evaluation_metric_impl name="pctCorrect" version="1.88" libname="weka" />
  </evaluation_metric_appl>
   ...
 </learner_evaluation>
</experiment>
```

The previous example only shows part of an experiment description. After the experiment has run, it can be extended with the outputs required for that type of experiment, such as evaluation metric results and (probabilistic) predictions[2]. Next to experiments, which consists of *applications*, the format also allows the definition of new *specifications* and *implementations* of any entity (e.g. algorithms, kernels, datasets, evaluation metrics,...), each of which can be described in great detail and 'tagged' with properties such as dataset and algorithm characteristics. These definitions require the minimal amount of information needed to make ensuing experiments reproducible [2].

3.4 Algorithms, Scripts and Datasets

For the portal to supply researchers with the necessary resources to run experiments, it needs to store, or know where to find certain datasets, algorithms and scripts (for instance, to calculate evaluation metrics). Given enough resources, such a portal could actually store datasets locally, and return them in a suitable format upon request. The portal can even store many versions of datasets, for instance, being preprocessed in many different ways. If the preprocessing methods can be executed by the portal itself, it could also preprocess the datasets on the fly. However, many modern data mining techniques operate directly on databases, and many data sources are implemented as web services. In this case, the portal only needs to store pointers to those databases and services and return the necessary information. In either case, to provide users with enough information to interpret the outcome of experiments run on those datasets, the portal should automatically tag these data sources by computing various important data characteristics.

When it comes to algorithms and scripts, the portal can again store executables, as far as they are available[3]. In case the algorithms are part of publicly available toolboxes, we only need to store (several versions of) those toolboxes and pointers to the algorithms inside that toolbox. In case they are implemented as web services, we can again simply store pointers to those services.

Ideally, the portal should also allow contributing users to define who can download their datasets and implementations. Preliminary experiments could also be kept private until all bugs are fixed.

3.5 Running Experiments

Finally, we also want the portal to assist us when running experiments. We propose two ways in which to do this, the first method requires that algorithms (or the toolboxes in which they feature) can read and write ExpML descriptions, the latter that they are implemented as web services which interact with the portal.

[2] Generated models are not yet included (nor stored in the database), but future extensions should allow, for instance, PMML descriptions of the models (see `http://www.dmg.org/pmml-v3-2.html`).

[3] As discussed in Sonnenburg et al. [11], there are many benefits to publishing even preliminary versions of algorithms, and researchers should receive formal credit for publishing good implementations.

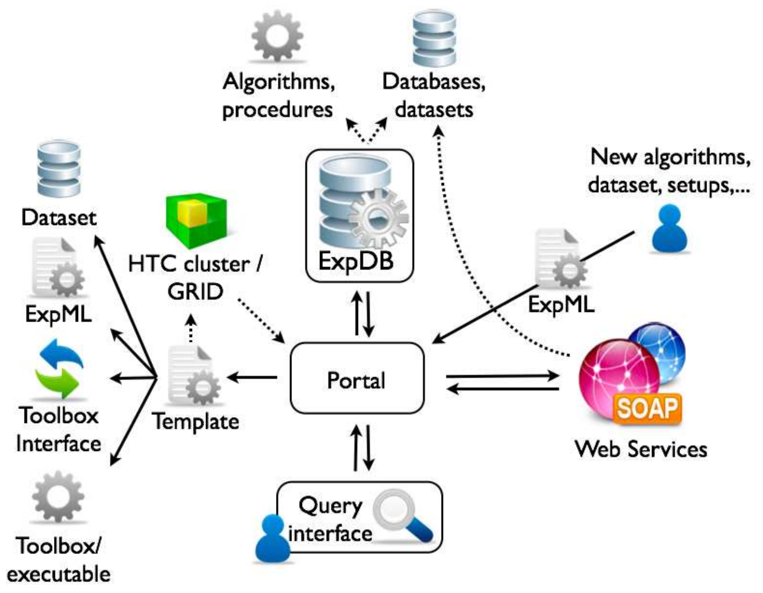


Fig. 2. Automated experiment execution in data mining.

Using Interfaces A first way to generate experiments and execute them, is to write interfaces that take ExpML descriptions as input, execute the experiments and return the final results in the ExpML format. This is illustrated in Fig.2.

In this scenario, a user makes sure her algorithm can read and write ExpML descriptions, and uploads a description of her new algorithm to the portal. Next, she writes a query that asks the portal to generate a number of experiments using a previously stored experimental setup. The portal extracts all the necessary information from the experiment database, checks which experiments still need to be run, and generates a number of templates for experiment execution. The templates are coupled with the necessary datasets, ExpML descriptions of the experiments, and, if necessary, the executables of other algorithms or procedures (e.g. a cross-validation procedure). The algorithm interfaces read the ExpML description, run the experiments with the stated parameters on the stated datasets, using the stated experimental methodology. The results are then again written in the ExpML format and submitted to the portal.

This is especially interesting when using algorithms and evaluation procedures implemented in toolboxes, since then only one interface has to be written per toolbox. In the currently implemented system, interfaces are provided only for the WEKA toolbox, though new interfaces for other popular toolboxes could be written quite easily. The templates it generates can be used for execution on several HPC systems or simply on a local machine. While in this scenario, the process has started from the query interface, this need not always be the case.

Future toolboxes could use their own interfaces to download prior experiments from the portal, run new experiments locally and stream results back to the portal.

Using Web Services Because of the highly diverse nature of data mining resources, a more fundamental approach would be to implement algorithms and other resources as web services which could then be called on by the portal, or vice versa. Illustrations of how such web services for data mining could be implemented can be found in de Bruin et al. [1], and illustrations of workflows of web services can be found in the Taverna project [13]. Weka4WS[14] offers some WEKA algorithms implemented as web services. This approach would considerably facilitate the implementation of many aspects of the proposed portal, such as making algorithms and datasets more easily available, making the design and execution of KDD workflows easier and more intuitive, and improving performance. Indeed, all these web services could automatically interact with the portal to submit experimental results and download any information needed. The ExpML language could be used to exchange details of experiments and new definitions (algorithms, datasets, etc.) in SOAP messages. Moreover, the portal could call on algorithms implemented as web services to automatically run experiments. The simplest setting would be to just send them an ExpML experiment description and leave it to the services to execute it and return the results. This would require services to support such requests. Alternatively, the portal could store some meta-information for each service detailing how to set parameters and run the algorithms. In both cases, the data could be sent along, or the web services could request it from the portal when they are ready to execute the experiment.

4 A Use Case

We finish with a description of a use case illustrating how our portal could be used to speed up experimentation in data mining. Of course, there are many different kinds of experiments in data mining research. A very common scenario, however, is that of evaluating a new technique by running it on a collection of datasets, employing a certain evaluation metric to test the performance of the new technique and to compare it with the performance of other algorithms previously proposed to perform the same task. We follow Mary as she aims to compare her new algorithm against the state-of-the-art in her field.

4.1 Collecting and Preparing Datasets

The first problem she encounters is to collect a number of datasets and to reformat them so that her algorithms, and the ones she wishes to compare with, can run on them. For instance, some algorithms may require missing values to be removed or filled in.

Using the community portal, Mary can query for a list of all known datasets, or for datasets with certain properties (e.g. relational or very large ones) or ones that represent certain tasks (e.g. multi-label prediction). She can even query for versions of those datasets which are already formatted by previous researchers to work with certain algorithms. In case those datasets do not yet exist, she can query for the data requirements for all the algorithms and for a list of useful preprocessing implementations. If Mary has to select a subset of algorithms, the portal could show her how often each dataset has been used in previous comparisons, show all known dataset characteristics (size, skewness, landmarking results,...), or directly return a balanced subset of various kinds of datasets. As such, the portal can significantly facilitate the search for a good collection of datasets.

4.2 Setting up Experiments

Mary now has to decide which experiments she needs to run in order to obtain a clear result, and exactly how to evaluate the algorithms.

Again, the portal can help Mary to obtain more trustworthy results. First of all, she can reuse the experiments on other algorithms, probably run by the original authors. Since the original authors know best how to use their algorithms, and how the parameters of those algorithms should be varied, it makes sense to just reuse them. Of course, to compare with them, the exact same procedures (e.g. data splits) should be employed to evaluate all algorithms. Luckily, the portal has kept track of this, so it will be able to use the exact same procedure for Mary's experiments. Mary can thus browse the used setups, choose one that fits her goal, and download all known results obtained with that setup. Issues with certain setups may also be reported and a number of exemplary setups can be proposed that can easily be adopted by others.

The portal will also show sensible ranges of parameter settings for all algorithms (entered by their authors), preprocessing techniques that should be considered, and a wide range of evaluation metrics to be computed so that the results are useful for many different researchers. Yet, including many parameter settings and preprocessing steps means that a lot more experiments have to be computed to perform algorithm comparisons than is currently customary in data mining research. However, the portal can make such large-scale studies much more attainable by eliminating the labor-intensive step of setting up all experiments by hand, by reusing many experiments run before and by making experiment execution more efficient where possible. For instance, active learning could be used to generate only the most interesting experiments (skipping, for instance, useless parameter ranges), or experiments could be generated to run on dedicate web services or high performance computing systems.

4.3 Exploring the Results

After she runs her experiments and her results are automatically uploaded, the portal automatically links the new results to the properties of datasets and algorithms. Instead of just ranking all algorithms on their average performance over the selected datasets, Mary can use queries to investigate on which kinds of datasets her algorithm has a clear advantage, and on which it doesn't work that well. This may help her to develop it further. To investigate her algorithm's robustness against certain data characteristics, she can run more experiments on other dataset versions which were, for instance, downsampled, or in which noise or random features were added, and query the ensuing results. Similarly, she can also check whether other types of preprocessing (e.g. feature selection or construction) are particularly useful for her algorithm. Many illustrations of the power of such queries when analyzing algorithm behavior can be found elsewere [16].

5 Conclusions

In this work, we propose an online portal to engender a more collaborative approach to data mining research. This portal can be implemented as a web service which collects annotated datasets, algorithms, experimental setups and millions of empirical evaluations sent to it by data mining tools and individual users, organizes them in one searchable central resource, and assists researchers in setting up and running new experiments. As illustrated, such a service could drive many useful applications, from online platforms where all stored experiments can be explored in great depth and discovered trends can be discussed, to advanced data mining tools that interact with the service to speed up the setup and execution of experiments and upload new experiments to allow a more in-depth exploration of the acquired results. However, some obstacles remain to support the automatic setup and execution of experiments with arbitrary algorithms. Two possible solutions are outlined based on a proposed XML-based standard for experiment descriptions, linked to an ontology of DM experiments. They require that algorithms (or the toolboxes in which they feature) can read and write to this format, or that they are implemented as web services which interact with the portal.

Acknowledgements

This research is supported by GOA 2003/08 'Inductive Knowledge Bases' and F.W.O.-Vlaanderen 'Foundations of Inductive Databases for Data Mining'.

References

1. de Bruin, J., Kok, J., Lavrač, N., and Trajkovski, I.: Towards Service-Oriented Knowledge Discovery: A Case Study. ECML 2008 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (2008).
2. Blockeel, H. and Vanschoren, J.: Experiment databases: Towards an improved experimental methodology in machine learning. PKDD '07: Proceedings. Lecture Notes in Computer Science **4702** (2007) 6-17

3. Drummond, C.: Finding a Balance between Anarchy and Orthodoxy. Proc. of the Evaluation Methods for Machine Learning Workshop at the 25th ICML (2008)
4. Džeroski S.: Towards a General Framework for Data Mining. Lecture Notes in Computer Science **4747** (2007) 259–300
5. Hand, D., J.: Classifier Technology and the Illusion of Progress. Statist. Sci. **21(1)** (2006), 1–14
6. Hoste, V. and Daelemans, W.: Comparing Learning Approaches to Coreference Resolution. There is More to it Than 'Bias'. Proceedings of the Workshop on Meta-Learning (ICML-2005) (2005) 20–27
7. Nature. Let data speak to data. Nature, **438(7068)** (2005), 531
8. Panov, P., Džeroski, S. and Soldatova, L.: OntoDM: An Ontology of Data Mining. IEEE International Conference on Data Mining Workshops (2008) 752–760
9. Perlich, C. and Provost, F. and Siminoff, J.: Tree induction vs. logistic regression: A learning curve analysis. Journal of Machine Learning Research **4** (2003) 211–255
10. Soldatova L. N., Clare A., Sparkes A., King R. D.: An ontology for a Robot Scientist, Bioinformatics **22(14)** (2006) 464–471
11. Sonnenburg, S., Braun, M. L., Ong, C. S., Bengio, S., Bottou, L., Holmes, G., LeCun, Y., Müller, K.-R., Pereira, F., Rasmussen, C. E., Rätsch, G., Schölkopf,B., Smola, A., Vincent, P., Weston, J., and Williamson, R. The need for open source software in machine learning. Journal of Machine Learning Research **8** (2007) 2443-2466.
12. Stoeckert,C., Causton,H. and Ball,C.: Microarray databases: standards and ontologies. Nature Genetics **32** (2002) 469–473.
13. The Taverna workbench. `http://taverna.sourceforge.net`
14. Talia, D., Trunfio, P., Verta, O., Weka4WS: a WSRF-enabled Weka Toolkit for Distributed Data Mining on Grids. Lecture Notes in Artificial Intelligence **3721** (2005) 309–320.
15. Vanschoren, J. and Blockeel, H. and Pfahringer, B. and Holmes, G.: Organizing the world's machine learning information. Communications in Computer and Information Science **17** (2008) 693–708
16. Vanschoren J. and Pfahringer B. and Holmes G.: Learning From The Past with Experiment Databases. Lecture Notes in Artificial Intelligence **5351** (2008) 485–496.
17. Witten, I.H. and Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques (2nd edition). Morgan Kaufmann (2005)
18. Žakova, M. and Kremen, P. and Železny, F. and Lavrač,N.: Using Ontological Reasoning and Planning for Data Mining Workflow Composition. ECML 2008 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (2008).

Towards service-oriented knowledge discovery in biomedical research

Tu Bao Ho[1,4], Katsuhiko Takabayashi[2], Pham Tho Hoan[3], Nguyen Thanh Phuong[5], Saori Kawasaki[1], Tran Dang Hung[1]

[1]Japan Advanced Institute of Science and Technology, Japan ⋆
[2]Chiba University Hospital, Chiba, Japan
[3]Hanoi University of Education, Hanoi, Vietnam
[4]Vietnamese Academy of Science and Technology, Vietnam
[5]The Microsoft Research, University of Trento, Italy

Abstract. Service-oriented knowledge discovery has the potential to change the way work can be performed in knowledge discovery. Aiming at analyzing biomedical data by computation programs to discover new and useful knowledge, bioinformatics can become more useful if developed within a service-oriented environment. By analyzing the service-oriented architecture, the features of service-oriented knowledge discovery and current service-oriented systems in biomedicine, this paper proposes five main services in service-oriented bioinformatics: data integration, model selection, workflow design and creation, interpretation of mining results, and literature review.

1 Introduction

Bioinformatics is now becoming a key field to study the functionality of life science. While the extremely complex nature of living organisms makes almost all studies of life science *in vitro* expensive and laborious, the already available large volumes of rapidly expanding and ever-changing biomedical databases allow biomedical scientists to effectively perform various additional studies *in silico*. Though bioinformatics is a relatively young science, its basic methods and tools have become familiar to biomedical scientists. However, the understanding of appropriateness between available bioinformatics tools and fundamental biological principles, and the knowledge to exploit these tools and data correctly to generate useful results still remains a challenge.

As modern services industry is now contributing for more than half of most developed economy, services computing has become an emerging cross discipline that covers the science and technology of leveraging computing and information technology to model, create, operate, and manage business services. The service-oriented approach to science is also being emerged and recognized. Service-oriented science has the potential to increase scientific productivity by making

⋆ bao@jaist.ac.jp, 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan

powerful information tools available to all, and thus enabling the widespread automation of data analysis and computation [8].

In this trend, a major view on the next generation of knowledge discovery and data mining (KDD) is service-oriented KDD that aims at providing integration of different data/knowledge resources and computer systems which are distributed across the network [2].

As the main objective of bioinformatics is to discover biomedical knowledge from biomedical data, it is natural to think of service-oriented knowledge discovery in bioinformatics [4, 9]. A typical work on service-oriented biomedical knowledge discovery is @neuLink [9], a component of the European Integrated Project @neurIST, that aims at providing services for linking genetics to disease with four concrete tasks. There have been also several proposals for architecture of service-oriented bioinformatics systems or trials on service-oriented biomedical systems. In [18], the authors developed Bio-jETI, a service integration, design, and provisioning platform for orchestrated bioinformatics processes. In [16], the authors presents Grendel, a bioinformatics Web service-based architecture for accessing high performance computing resources. In [34], the authors developed a service-oriented data integration and analysis environment for *in silico* experiments and bioinformatics research. The target of various researches in this direction is to support intelligent client for integrating bioinformatics services [19] or Grid service oriented virtual bioinformatics laboratory [20] or for Grid-based high-throughput genome analysis [28]. The key idea of why and how the cyber-infrastructure with service orientation can empower the 'third way' in biomedical research can be found in [4].

Based on our experience and lessons learned from knowledge discovery and biomedical research and practice [10–12], this paper presents another view towards a service-oriented environment for data mining in biomedical research that is expected to fulfill the needs of biologists/scientists. By analyzing the features of service-oriented architecture and modeling [1, 6, 7], the current bioinformatics research and service-oriented bioinformatics systems, we argue that biomedical applications can be developed and used more effectively and largely within a service-oriented environment. In particular, we emphasize five important services in the development of a service-oriented environment for bioinformatics: data integration, model selection, workflow design and creation, interpretation of mining results, and literature review. The main ideas as well as some initialized work and experience for each of these services will also be addressed.

The paper is organized as follows. In section 2, we discuss the main features of a service-oriented architecture and why knowledge discovery systems can be beneficial when developed in the service-oriented architecture. In section 3, we state that bioinformatics essentially employs data mining techniques and it meets most challenging problems in data mining. Based on our past experience, on issues in service-oriented knowledge discovery and on the current service-oriented systems in bioinformatics, we argue and propose the features to be developed for service-oriented environment for the bioinformatics research. Section 4 concludes this investigation and proposal.

2 Service-oriented knowledge discovery

2.1 Service-oriented architecture

Intuitively, services are work done by someone for others to have the benefit. So, there are at least two parties involved in the services, the service provider and the service consumer. In terms of the service process, there are so-called front-stage and back-stage activities in any business transaction – front-stage activities are that come in contact with the consumer and back-stage ones are that do not. In all cases, service requires substantial input from the consumer, and thus service depends on having a high degree of front-stage activities to interact with the consumer, and that improves consumer input to the production process depending almost entirely on back-stage activities. For this reason and as certain service process can be complicated, a third party – the service broker who plays the intermediate role between these other two parties, can improve the service process.

Generally speaking, a *service-oriented architecture* (SOA) is a framework that allows us to easily reuse and combine the business process and services to make up the business [1, 7]. The conceptual service-oriented architecture in [1] defines an interaction model among three above-mentioned parties: the service provider, who publishes a service description and provides the service implementation; the service consumer, who can either invoke or use services via service description in a service registry; and the service broker, who provides and maintains the service registry. Figure 1 (adapted from [1]) depicts the relation among these parties, and the link between the front-stage and the back-stage in the service-oriented architecture.

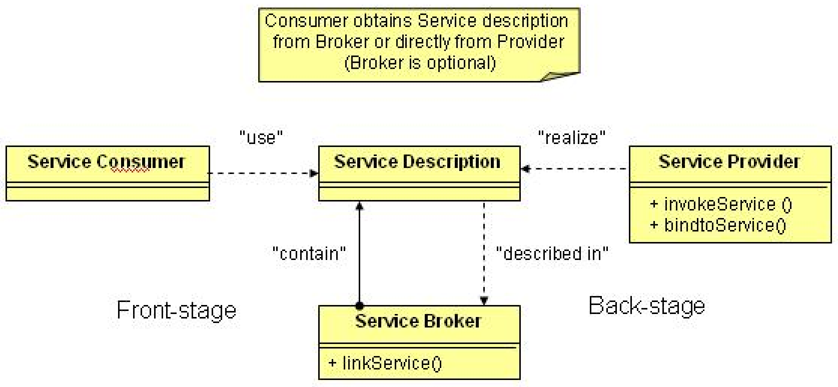


Fig. 1. Conceptual model of service-oriented architecture ([1]).

A SOA usually contains several layers [1, 3]. For example, they are operational systems layer, enterprise component layer, services layer, business process and composition layer, access and presentation layer, and integration layer. Importantly, SOA should separate functions into distinct units or services, which developers make accessible over a network in order that users can combine and reuse them in the production of applications [7].

Another important feature of SOA is the *loose coupling* among the services, in contrast to the functions that a linker binds together to form an executable, i.e. services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other. SOA services also run in 'safe' wrappers such as Java or .NET, and other programming languages that manage memory allocation and reclamation, allow ad hoc and late binding, and provide some degree of indeterminate data typing.

2.2 Why service-oriented knowledge discovery?

There are at least two reasons for developing service-oriented knowledge discovery. One is the most challenging data to be mined are distributed and highly heterogeneous and the other is knowledge discovery technologies should be made and served for people who owns the data but do not know well how to exploit them. We focus our discussion here on the second issue that we have had a number of lessons learned in the last ten years [11]. It can be seen that the following aspects of knowledge discovery often cause difficulty for the user.

The *knowledge discovery process* is complicated and inherently contains several steps. These steps are inherently iterative and interactive, i.e., one cannot expect to extract useful knowledge by just pushing one time a large amount of data into a black box without the user's participation.

It is known that there is no inherent superior method/model in terms of generalization performance. In the absence of prior information about the problem there are no reasons to prefer one learning algorithm or classifier model than another. *Model selection* is difficult and non-trivial because it requires empirical comparative evaluation of discovered models and meta-knowledge on models/algorithms. The user has often to do a trial-and-error process to select the most suitable models from competing ones. Clearly, trying all possible options is impractical and an informed search process is needed to attain expected models. Informed search requires both performance metrics and model characteristics that are often not available for the user. In previous years we have developed D2MS (Data Mining with Model Selection)–a visual data mining environment that provides support for model selection [12].

Moreover, *the user's interest* in discovered models is a subjective matter that depends much on his/her domain knowledge and sometimes is very independent of performance metrics provided by the system. Current data mining provides multiple algorithms within a single system, but the selection and combination of these algorithms are external to the system and specified by the user. This makes the KDD process difficult and possibly less efficient in practice.

Unlike the major research tendency that aims to provide the user with meta-knowledge for an automatic model selection as described in the next section, in our view, model selection would be semi-automatic and it requires an effective collaboration between the user and the discovery system. In such collaboration, *visualization* has an indispensable role because it can give a deep understanding of complicated models that the user cannot have if using only performance metrics. The research on visualization integrated with model selection is significant because there is currently very limited visualization support for the process of building and selecting models in knowledge discovery.

The *literature review* is likely to be very important for biomedical scientists whose work is heavily based on literature. The medical literature is one of the sources of background and domain knowledge that we need to exploit. In particular, MEDLINE – the source of life sciences and biomedical information with nearly eleven million records – is very important and can be exploited with text mining methods. MEDLINE has a special potential for any of biomedical applications. Services for help of extracting information/domain knowledge from MEDLINE abstracts for biomedical scientists are certainly very useful and expected.

For all of these reasons, services for supporting model selection in knowledge discovery should be considered essential in the next KDD generation. In the other words, it is service-oriented knowledge discovery. It is worth noting that among the views on next generation of data mining [17, 5], the above two reasons for service-oriented knowledge discovery also play important roles. The most popular KDD tools Weka has been expanded to Weka4WS for the Web services [29].

3 Service-oriented knowledge discovery in bioinformatics

In this section we first address our view on the knowledge discovery aspects in bioinformatics, then discuss a possible service-oriented environment for bioinfrmatics.

3.1 Knowledge discovery in bioinformatics

It is well known that in ICDM 2005, the KDD community identified the 10 challenging problems in data mining [35]:

1. Developing a unifying theory of data mining
2. Scaling up for high dimensional data/high speed streams
3. Mining sequence data and time series data
4. Mining complex knowledge from complex data
5. Data mining in a network setting
6. Distributed data mining and mining multi-agent data
7. Data mining for biological and environmental problems
8. Data mining-process related problems
9. Security, privacy and data integrity

10. Dealing with non-static, unbalanced and cost-sensitive data

Bioinformatics is the field of science in which molecular biology, computer science and information technology merge together to increase our understanding of biological processes. These biological processes are essentially uncovered by the analysis of biological data acquired by human, i.e. knowledge discovery [33].

It is worth noting that biological data and its analysis meet most of the above challenging problems in data mining. First, the biological data is huge and of high dimensional (e.g, microarray data) and in many cases are stream data (e.g., the simulated data of water molecules that continuously move around a protein). Second, the biological data is typically sequential (e.g., DNA sequences) and temporal (e.g., the relational temporal database on hepatitis given to PKDD challenge in 2004-2005, [10]). Third, the biological data is complex (e.g., they are represented in all kinds of complexly unstructured data such as sequence, temporal, graph, text, etc.) and the mining usually aims to discover the rich structure of relations among objects). Four, the biological data belong to various communities who organize their changing data in different ways and mining them strongly depends on the network development. Five, the biological databases are distributed over the world, and discover biological knowledge always requires integrating many of them together. Six, mining biological data is hard as it relates to the most complicated knowledge of human–knowledge about the life. Seven, mining biological data means mining the biological processes. Eight, the biological data is highly private requiring secure and integrity. Nine, the biological data is heavily non-statistic, unbalance and cost-sensitive.

Facing with such challenges a biomedical scientist have to deal with when doing a study/research in bioinformatics, the question that motivates us is – as service providers and service-oriented system developers – what are the most needs of such consumers that we have to provide services? From our experience they mainly are the followings:

- Tools and guidelines to efficiently access, review, visualise, integrate and manage of various types of biomedical data so that the combination/compostion of such services is the most appropriate and ready to be used by some mining programs.
- Programs that are suitable to the mining target that enable to find significant and useful knowledge in the fields of molecular biology (genomics, proteomics, metabolomics) and its relation to diseases and drugs.
- Guidelines and tools to establish a workflow consisting various ordered steps that the biomedical scientist has to do to reach the goals.
- Support for interpretation of the computed finding and express them in the common language in biomedicine.
- Support to reuse and composite of available bioinformatics services.
- Support for the simulation and perturbation to verify the findings instead of doing in wet-lab, i.e., for systems biology.
- Support for the re-design of existing, natural biological systems for useful purposes, i.e., for synthetic biology.

As noticed, one of the most important issues for biomedical scientists is to understand the appropriateness between fundamental biological principles and bioinformatics tools available. In order to provide appropriate services in bioinformatics research, we consider the following factors are essential.

1. *View the biology problem in terms of computation.* Many existing biomedical problems can have computational solutions, and many new problems can be formulated to solve with computational solutions. In all cases, it is important to know which data could be used to solve the problem and how to integrate them for the use. For example, when studying protein-protein interactions the information on protein can be obtained from proteomic databases such as Uniprot (for functional and structural information), Interpro (for protein families, domain and functional sites), etc. but other information from genomic databases can be used such as MIPS (for molecular structures and functional networks) or Gene Ontology (for molecular functions, biological process and cellular component). A key question is which data to be used and how to put them in a computational model when they are stored in very different formats?
2. *Understand the power of each data mining method and how the method can be appropriately used to solve a given biomedical problem.* It is known that data mining methods can be generally divided into two groups for classification/prediction and description. Each method can only work well for some tasks and its performance depends on various factors such as method's parameter setting. Usually, the bologists/scientists do not know well which methods to be used for a given task and thus providing them such knowledge is necessary.
3. *Construct the workflow for mining biological data.* The process of knowledge discovery is complicated and should be seen inherently as a process containing several steps. The first step is to understand the application domain, to formulate the problem and to collect data. The second step is to preprocess the data. The third step is that of data mining with the aim of extracting useful knowledge as patterns or models hidden in data. The fourth step is to post-process discovered knowledge, and the fifth step is to put discovered knowledge in practical use. Two data mining primary goals of prediction and description are concerned with different tasks in the step of data mining, such as those for characterization, discrimination, association, classification and clustering. Also, there are different tasks of data cleaning, integration, transformation and reduction in the preprocessing step, and those of interpretation, evaluation, exportation, and visualization of results in the post-processing step. Moreover, each of these tasks can be done with different methods and algorithms. To solve a given discovery problem, the user usually has to go through these steps several times, each time she/he has to run a series of algorithms [12]. A workflow is a depiction of a sequence of operations in steps of the knowledge discovery process. How to formulate an appropriate workflow is not a simple task for most biologists/scientists when using knowledge discovery tools.

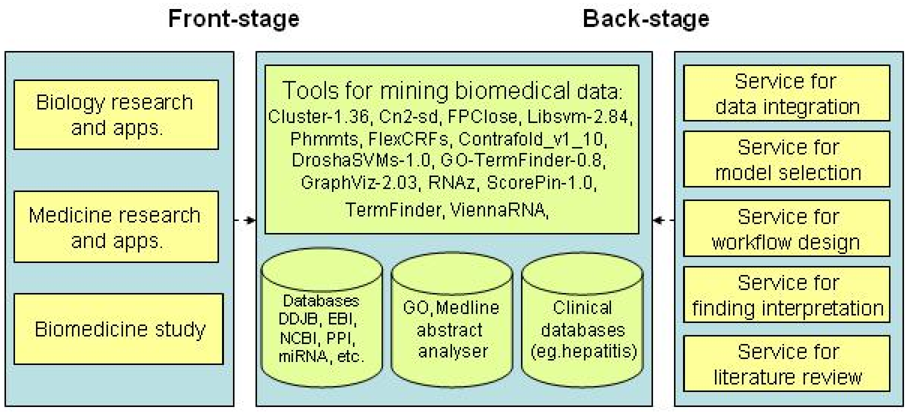

Front-stage
Back-stage
Biology research and apps.
Medicine research and apps.
Biomedicine study
Tools for mining biomedical data:
Cluster-1.36, Cn2-sd, FPClose, Libsvm-2.84, Phmmts, FlexCRFs, Contrafold_v1_10, DroshaSVMs-1.0, GO-TermFinder-0.8, GraphViz-2.03, RNAz, ScorePin-1.0, TermFinder, ViennaRNA,
Databases DDJB, EBI, NCBI, PPI, miRNA, etc.
GO,Medline abstract analyser
Clinical databases (eg.hepatitis)
Service for data integration
Service for model selection
Service for workflow design
Service for finding interpretation
Service for literature review

Fig. 2. Overview of our service-oriented environment for bioinformatics

4. *Interpret and evaluate findings.* The findings obtained from biomedical data via data mining programs are in forms of models (such as decision trees or regression lines) or patterns (such as induction rules). Understanding the meaning of these findings in terms of computation language and interpreting their biological meaning are the final importance. Usually, this is also a difficult task for biologists/scientists as they either are not familiar with computational representation of the findings or those findings can be very new for them.
5. *Find useful information from the literature.* MEDLINE has a special potential for any of biomedical applications. For example, when studying hepatitis we always have to look for information from more than 60,000 abstracts of research papers concerning hepatitis in MEDLINE. How to help to extract information/domain knowledge from MEDLINE abstracts is an expected service for biomedical scientists.

The above critical issues in bioinformatics led us to provide necessary services. Figure 2 shows the overall architecture and currently available resources and tools in our environment being developed.

3.2 Towards a service-oriented environment for bioinformatics

From the analysis in previous section, we are working towards a service-oriented environment for bioinformatics with the following five kinds of services:

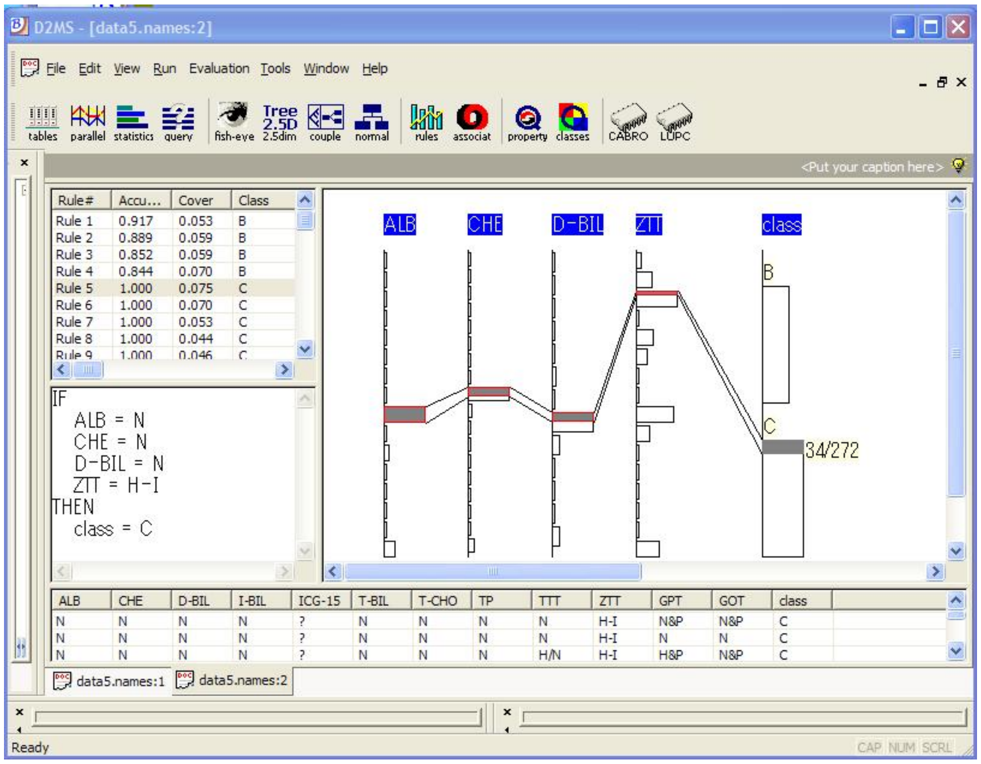
- Services for selecting and integrating data,
- Services for selecting data methods,
- Services for workflow design and creation,
- Services for interpreting the findings,
- Services for supporting the literature review.

Services for selecting and integrating data. The proposed services for preparation of data include

- Getting data from different databases distributed over the network for ready-to-use. Currently, the available biological databases that are daily upgraded include:
  - Genome databases: DDBJ, EBI (Eukaryota, Agambiae, Ataliana, Bvulgaris, Cbiggsa, Celegans, Hsapiens, Mmusculus), NCBI (Blast, Genebank, Genomes, Refseq, B_taurus, D_rerio, H_sapiens, LocusLink, M_musculus, R_norvegicus, X_tropicalis).
  - Proteomic databases: DSSP, TRANSFAC, PIR, PMD, PRF, PRINTS, PRODOM, PROSITE, InterPro, MINT, MIPS, MSMS
  - Protein classification databases: YIPD, PFAM, CluSTr, TIGRFAMs, CATH.
- Providing guidelines for integrating different types of data. In particular, we provide schemes for two ways of integrating data
  - By inductive logic programming (ILP): The technique allows the user to convert any kind of data into ground facts in form of predicates [14, 22].
  - By kernel methods: Kernel methods can deal with any data type (sequence, graph, text, numerical numbers, etc.) as representing the original data by kernel matrices of real numbers. Multiple kernel learning is the problem of integrating various kernels generated from different data types [27]. Two simple but effective services for combining kernels are introduced in [30].
- Providing ready-to-use integrated data for proteomics study within the ILP framework [22, 23]; combining expression data and genomic location data in study of transcriptional regulatory rules [25]; combining epigenetic data and gene expression data in study of acetylation and methylation [26]; or combining miRNA expression data with mRNA and miRNA binding data in study of miRNA regulatory modules in human genome [32].

Services for selecting data methods. The proposed services for selecting data mining methods include

- Meta-knowledge on data mining/machine learning methods and their usage in concrete data mining task in form of rules in an expert systems [15].
- Meta-knowledge embedded in mining systems on the system methods such as done in [12].
- Learning methods that frequently used in biomedical data analysis and their guidance such as ILP [22], CN2-SD [25, 32], etc.
- Machine learning library that are developed for the Web services and thus service-oriented systems such as Weka4WS [29].
- Available ready-to-use tools at the environment: BLAST, ClustalW, Cluster-1.36, Cn2-sd, FPClose, Libsvm-2.84, Phmmts, FlexCRFs, Contrafold_v1_10, DroshaSVMs-1.0, GO-TermFinder-0.8, GraphViz-2.03, RNAz, ScorePin-1.0, TermFinder, ViennaRNA, Wordspy1.5.


D2MS - [data5.names:2]
File Edit View Run Evaluation Tools Window Help
tables parallel statistics query fish-eye 2.5dim couple normal rules associat property classes CABRO LUPC
<Put your caption here>
ALB
CHE
D-BIL
ZTT
class
B
C
34/272
data5.names:1
data5.names:2
Ready
CAP NUM SCRL

Rule#	Accu...	Cover	Class
Rule 1	0.917	0.053	B
Rule 2	0.889	0.059	B
Rule 3	0.852	0.059	B
Rule 4	0.844	0.070	B
Rule 5	1.000	0.075	C
Rule 6	1.000	0.070	C
Rule 7	1.000	0.053	C
Rule 8	1.000	0.044	C
Rule 9	1.000	0.046	C

```
IF
   ALB = N
   CHE = N
   D-BIL = N
   ZTT = H-I
THEN
   class = C
```

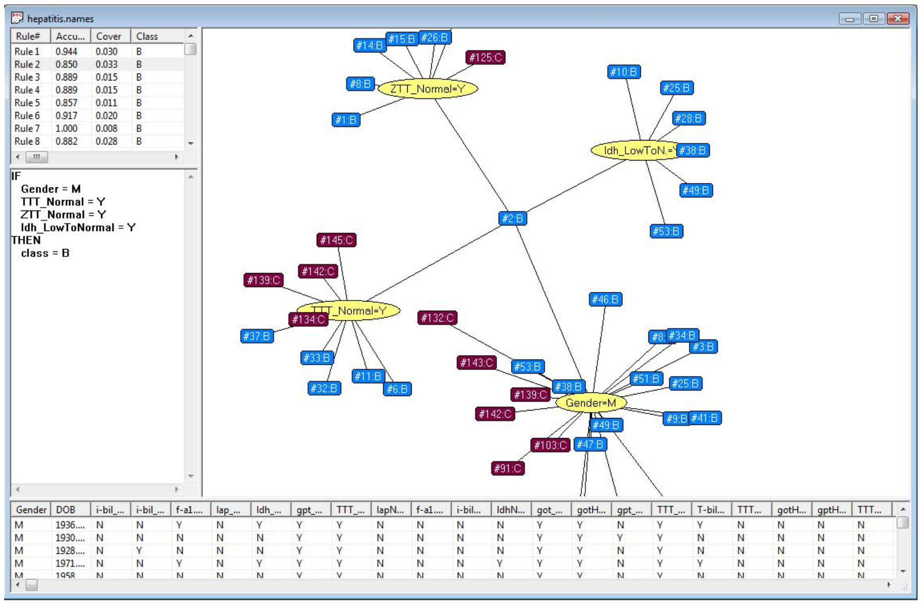
ALB	CHE	D-BIL	I-BIL	ICG-15	T-BIL	T-CHO	TP	TTT	ZTT	GPT	GOT	class
N	N	N	N	?	N	N	N	N	H-I	N&P	N&P	C
N	N	N	N	?	N	N	N	N	H-I	N	N	C
N	N	N	N	?	N	N	N	H/N	H-I	H&P	N&P	C

Fig. 3. Visualization of a discovered rule about hepatitis type C ([12, 13]).

Services for workflow design and creation. The proposed services for design and creation of workflow include

– Recently created Web-services related standards and technology using an XML-standard for workflow specification, such as WSDL (Web services description language), BPEL4WS (business process execution language for Web services) [5], etc.
– Tools for service-oriented modeling and design as a graphical process modeling tools such as jABC [18], Taverna workbench [31], etc.
– The plan management module maintains profiles of algorithms available in preprocessing, data mining, and post-processing of D2MS [12]. An algorithm profile contains information about the algorithm functions, its requirements, types and effect of its parameters. The registration of an algorithm is done via a dialog box when it is added into a plan.

Services for interpreting the findings. The proposed services for interpretation of mining results include



Rule#	Accu...	Cover	Class
Rule 1	0.944	0.030	B
Rule 2	0.850	0.033	B
Rule 3	0.889	0.015	B
Rule 4	0.889	0.015	B
Rule 5	0.857	0.011	B
Rule 6	0.917	0.020	B
Rule 7	1.000	0.008	B
Rule 8	0.882	0.028	B

```
IF
  Gender = M
  TTT_Normal = Y
  ZTT_Normal = Y
  ldh_LowToNormal = Y
THEN
  class = B
```

Gender	DOB	i-bil_...	i-bil_...	f-a1....	lap_...	ldh_...	gpt_...	TTT_...	lapN...	f-a1....	i-bil...	ldhN...	got_...	gotH...	gpt_...	TTT_...	T-bil...	TTT...	gotH...	gptH...	TTT...
M	1936....	N	N	Y	N	Y	Y	Y	N	N	N	N	Y	Y	N	Y	Y	N	N	N	N
M	1930....	N	N	N	N	N	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N	N
M	1928....	N	Y	N	N	N	Y	Y	N	N	N	N	Y	Y	N	Y	N	N	N	N	N
M	1971....	N	N	Y	N	Y	Y	Y	N	N	N	Y	Y	Y	N	Y	Y	N	N	N	N
M	1958	N	N	N	N	N	Y	Y	N	N	N	N	Y	Y	N	Y	N	N	N	N	N

Fig. 4. A discovered rule is visualized in the relation with the others ([12, 24]).

- Visualization of a discovered rule and its relations with the others. Figure 3 illustrates the visualization of a rule discovered by our system D2MS [12, 13]. This rule describes hepatitis type C, where the rule is in the middle-left window and visualized in the upper-right window; the set of rules each with precision and coverage is given in the upper-left window; the bottom window show the description of patient who matched this rule. Figure 4 shows another mode of rule visualization in D2MS where each condition (yellow node) of the rule under consideration (Rule 2) is connected to all other rules having it as a condition [24]. The light blue nodes are rules for hepatitis type B while the brown nodes are rules for hepatitis type C.
- Exploiting findings represented into other forms such as Excel file [12].
- Providing the statistical significance of association rules in learning [11].
- Combining the findings, mining process and context of finding extracted by literature into an integrated dynamic platform. It supports biomedical scientists in interpretation and evaluation of the findings with accessibility to every related sources including original data, intermediate relations, and external references. Biomedical scientists tend to value not only statistical evidence of the findings, but also intuitive accountability on them. This point is indispensable in the discovery activity because most of the findings from individual mining methods are generally fragment sets regarding some certain medical facts, so that biomedical scientists have to reorganize and reallocate them into a biomedical context fitting to their integrated expertise.

Services for supporting for literature review. It is useful if the biologists/scientists have services supporting for literature review. The literature review is likely to very important for biomedical scientists who often do their job based on literature [10]. Services concerning the literature review are proposed to include the following tasks:

– Identifying the topics of interest.
– Searching and appraising public databases of literature to identify the valid and applicable evidence.
– Mining contents of the collections of papers.
– Synthesizing and presenting reviews.
– Provide research trends by emerging trend detection [21].
– Extracting relations among medical tests attempted in [10] to complement the findings from clinical data.
– Support constructing the context information to fulfil the functions for services for interpreting the findings. This can be extracted based on the structure of medical articles because the appearance of biomedical literature like structure, key phrases, statements and so on, reflects what kinds of biomedical objects, techniques, relations motivate their investigations, and how the experts think.

4 Conclusion

This paper addresses the issues of service-oriented environments for bioinformatics research. By analyzing the features of service-oriented architectures, the tendency of service-oriented knowledge discovery, the issues in bioinformatics research, we argued that the service-oriented development is very significant in bioinformatics. We also proposed five services towards service-oriented bioinformatics including data integration, model selection, workflow design, finding interpretation, and literature review.

The construction of our service-oriented environment for bioinformatics is now at the initialized stage but based on various components developed in the past such as data mining methods, knowledge and data visualization, MEDLINE abstract analysis, model selection, etc. We strongly believe that service-oriented bioinforamtics will facilitate biomedical scientists in exploiting computational programs to do their research.

References

1. Arsanjani, A.: Service-oriented architecture and modelling, *IBM Online article*, http://www.ibm.com/developerworks/library/ws-soa-design1/, November 2004.
2. Bruin, J., Kok, J., Lavrac, N., Trajkovski, I.: Towards Service-Oriented Knowledge Discovery: A Case Study, *Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery*, ECML/PKDD'08, 2008.

3. Bruin, J., Kok, J., Lavrac, N., Trajkovski, I.: On The Design of Knowledge Discovery Services: Design Patterns and Their Application In A Use Case Implementation, *Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery*, ECML/PKDD'08, 2008.
4. Buetow, K.H.: Cyberinfrastructure: Empowering a "third way" in biomedical research, *Science*, Vol. 308, No. 5723, 821–824, 2005.
5. Cheung, W.K., Zhang, X.F., Wong, H.F., Liu, J., Luo, Z.W., Tong, F.C.: Service-oriented distributed data mining, *IEEE Internet Computing*, Jul-Aug., 44–54, 2006.
6. Guedes, D., Meira, W., Ferreira, R.: Anteater: An service-oriented architecture for high-performance data mining, *IEEE Internet Computing*, Jul-Aug., 6–43, 2006.
7. Erl, T.: Service-oriented architecture–Concept, technique and design, in *Service-oriented architecture–Concept, technique and design*, Printice Hall, 1–51, 2005.
8. Foster I.: Service-oriented science, *Science*, Vol. 308, 814–817, 2005.
9. Friedrich, C.M., Dach, H., Gattermayer, T., Engelbrecht, G., Benkner, S., Hofmann-Apitius, M.: @neuLink: a service-oriented application for biomedical knowledge discovery, *Studies in Health Technology and Informatics*, Vol. 138, 165–172, 2008.
10. Ho, T.B., Kawasaki, S., Takabayashi, K., Nguyen, C.H.: Integration of learning methods, medical literature and expert inspection in medical data mining, *IEICE Trans. Information and Systems*, Vol. E90-D, No. 10, 1574–1581, 2007.
11. Ho, T.B., Nguyen, T.D., Kawasaki, S.: Failures and Successes in Medical Data Mining, *Chapter 6 in Knowledge-Based Intelligent Systems for Health Care*, T. Ichimura and Yoshida, K. (Eds.), Advanced Knowledge International Publishers, 167–212, 2004.
12. Ho, T.B., Nguyen, T.D., Shimodaira, H., Kimura, M.: A knowledge discovery system with support for model selection and visualization, *Applied Intelligence*, Vol. 19, Issue 1-2, 125–141, 2003.
13. Ho, T.B., Nguyen, T.D., Nguyen, D.D.: Visualization support for a user-centered KDD process, *ACM International Conference on Knowledge Discovery and Data Mining KDD-02*, Edmonton, 23-26 July, 519–524, 2002.
14. Ho, T.B., Nguyen, T.P., Tran, T.N.: Study of protein-protein interactions from multiple data sources, *Advances in Data Warehousing and Mining*, David Taniar (Ed.), IGC Publishers, 280–307, 2007.
15. Ho, T.B., Quinqueton, J., Ralambondrainy, H.: Using expert system techniques for interpretation of data analysis results, *Proceedings of COMPSTAT'86*, 308–311, 1986.
16. Hunter, A., Schibeci, D., Hiew, H.L., Bellgard, M.: Grendel: A bioinformatics Web service-based architecture for accessing HPC resources, *Australasian Workshop on Grid Computing and e-Research*, 29–32, 2005.
17. Kargupta, H., Han, J., Yu, P.S., Motwani, R., Kumar, V.: *Next Generation of Data Mining*, CRP Press, 2008.
18. Margaria, T., Kubczak, C., Steffen B.: Bio-jETI: a service integration, design, and provisioning platform for orchestrated bioinformatics processes, *BMC Bioinformatics*, Vol. 9, Supp. 4, 1–17, 2008.
19. Navas-Delgado, I., Rojano-Munoz, M., Ramirez, S., Perez, A.J., Leon, E.A., Aldana-Montes, J.F., Trelles, O.: Intelligent client for integrating bioinformatics services, *Bioinformatics*, Vol. 22, No. 11, 106–111, 2005.
20. Kelly, N., Jithesh, P.V., Donachy, P., Harmer, T.J., Perrott, J.H., McCurley, M., Townsley, M., McKee, J.J.S.: GeneGrid: A commercial Grid service oriented virtual bioinformatics laboratory, *IEEE International Conference on Services Computing*, Vol. 1, 43–50, 2005.

21. Le, M.H., Ho, T.B., Nakamori, Y.: Detecting Emerging Trends from Scientific Corpora, *International Journal of Knowledge and Systems Science*, Vol. 2, No. 2, 53–59, 2005.
22. Nguyen, T.P., Ho, T.B.: An integrative domain-based approach to predicting protein-protein interactions, *Journal of Bioinformatics and Computational Biology*, Vol. 6, Issue 6, 1115–1132, 2008.
23. Nguyen, T.P., Ho, T.B.: Discovering signal transduction networks using signaling domain-domain interactions, *Genome Informatics*, Vol. 17, No. 2, 35–45, 2006.
24. Nguyen, D.D., Ho, T.B., Kawasaki, S.: Knowledge Visualization in Hepatitis Study, *Asia Pacific Symposium on Information Visualization APVIS 2006*, Austra-lian Computer Society, 59–62, 2006.
25. Pham, T.H., Clemente, J., Satou, K., Ho, T.B.: Computational discovery of transcriptional regulatory rules, *Bioinformatics*, Vol. 21, Supp. 2, 101–107, 2005.
26. Pham, T.H., Tran, D.H., Ho, T.B., Satou, K., Valiente, G.: Qualitatively predicting acetylation and methylation areas in DNA sequences, *Genome Informatics*, Vol. 15, No. 2, 3–11, 2005.
27. Scholkopf, B., Tsuda, K., Vert, J.P.: *Kernel Methods in Computational Biology*, The MIT Press, 2004.
28. Sulakhe, D., Rodriguez, A., D'Souza, M., Wilde, M., Nefedova, V., Foster, I., Maltsev, N.: GNARE: An environment for Grid-based high-throughput genome analysis, *IEEE International Symposium on Cluster Computing and the Grid*, Vol. 1, 455–462 2005.
29. Talia, D., Trunfio, P., Verta, O.: The Weka4WS framework for distributed data mining in service-oriented Grids, *Concurrency and Computation: Practice & Experience*, Vol. 20, Issue 6, 1933-1951, 2008.
30. Tanabe, H., Ho, T.B., Nguyen, C.H., Kawasaki, S.: Simple but effective methods of combining kernels for the prediction problem in biology, *IEEE International Conference on Research, Innovation, and Vision for the Future in Computing and Communication Technologies RIVF08*, Ho Chi Minh city, 71–78, 2008.
31. Taverna, [http://taverna.sourceforge.net]
32. Tran, D.H., Satou, K., Ho, T.B.: Finding microRNA regulatory modules in human genome using rule induction, *BMC Bioinformatics*, Vol. 9, No. Supp 11, 1–10, 2008.
33. Wang. J.T.L., Zaki, M.J., Toivonen, H.T.T., Shasha, D. (Eds.): *Data Mining in Bioinformatics*, Springer, 2005.
34. Xiang, X., Madey, G.: A service-oriented data integration and analysis environment for in silico experiments and bioinformatics research, *40th Hawai International Conference on System Science*, 171–180, 2007.
35. Yang, Q., Wu, X.: 10 challenging problems in data mining research, *International Journal of Information Technology and Decision Making*, Vol. 5, No. 4, 597–604, 2006

OntoDM: Towards an Ontology of Data Mining Investigations (Extended Abstract) *

Panče Panov[1], Larisa N. Soldatova[2], Sašo Džeroski[1]

[1] Jožef Stefan Institute, Jamova cesta 39, SI-1000 Ljubljana, Slovenia
{Pance.Panov,Saso.Dzeroski}@ijs.si
[2] Aberystwyth University, Penglais, Aberystwyth, SY23 3DB, Wales, UK
lss@aber.ac.uk

1 Introduction

In recent years use of term ontology has become prominent in the area of computer science research and the application of computer science methods in management of scientific and other kinds of information. In this sense the term ontology has the meaning of a standardized terminological framework in terms of which the information is organized. When one sets out to construct an ontology then, what one is doing is designing a representational artifact that is intended to represent the universals and relations amongst universals that exist, either in a given domain of reality (e.g data mining domain) or across such domains.

In the domain of data mining and knowledge discovery, researchers have tried to construct semantic representations describing data mining entities that were targeted to solve specific problems. Most of the developments are with the aim of automatic planning of data mining workflows [1, 17, 7, 5] and description of data mining services on the GRID [4, 3]. Some of the focus has been put on description of experiments in machine learning by Lot of work has been done in the formalization of scientific investigations and experimentation mostly in the area of biomedicine and automation of science [15, 8, 14].

Most of the semantic representations for data mining proposed so far are based on so called light-weight ontologies [9].The current proposals for an ontology of data mining are not based on upper level ontology categories nor have used a predefined set of relations based on an upper level ontology. In contrast to many other domains, data mining requires elaborate inference over its entities, and hence requires rigid heavy-weight ontologies with the aim of improving the KDD (Knowledge Discovery in Databases) process and providing support for development of new data mining approaches and techniques.

The motivation for developing a heavy-weight ontology of data mining is multi-fold. Firstly, formalization of data mining entities and identification of the relationships between the entities in the form of an ontology is the first step towards developing a general framework for data mining. Secondly, the domain of data mining needs to be described as broad as possible and lot of attention has

* Full version of this paper is presented in Procedings of Twelfth International Conference on Discovery Science (DS09), Porto, Portugal, 3-5 October 2009

to be focused on the rigorous meaning of each entity by introducing semantically rigorous relations between the entities and compliance to an upper level ontology categories. Finally, an ontology of data mining should define what is the minimum information required for the description of a data mining investigation. Biology is leading the way in developing standards for recording and representation of scientific data and biological investigations (e.g., already more than 50 journals require compliance of papers reporting microarray experiments to the Minimum Information About a Microarray Experiment - MIAME standard [2]). The researchers in the domain of data mining should follow this good practice and the ontology of data mining would support development of standards for performing and recording of data mining investigations.

In this work we propose an extended and updated version of the ontology of data mining named OntoDM. Our ontology design takes into consideration the best practices in ontology engineering. We use an upper level ontology BFO (Basic Formal Ontology)[3] to define the upper level classes, the OBO Relational Ontology (RO)[4] to define the semantics of the relationships between the data mining entities, and provide is-a completeness and single is-a inheritance for all DM entities. We also developed our ontology in the most general fashion in order to be able to represent the complex entities in data mining that are becoming more and more popular research areas such as mining structured data and constraint-based mining.

2 OntoDM Design and Description

Our ontology of data mining (OntoDM) aims to provide a structured vocabulary of entities sufficient for the description of data mining scenarios and workflows. OntoDM aims to follow the OBO Foundry principles[5] in ontology engineering that are widely accepted in the biomedical domains. In this way, OntoDM is built on a sound theoretical foundation, will be compliant with other (e.g., biological) domains and can be widely re-usable. OntoDM is expressed in OWL-DL and is being developed using the Protege ontology editor[6]. OntoDM is available at: `http://kt.ijs.si/panovp/OntoDM/`.

In our preliminary work [10] we presented an initial version of OntoDM sufficient for the representation of data mining tasks and complex data types. The initial version of OntoDM was using the philosophy of Ontology of Scientific Experiments (EXPO) [15] and ontology of biomedical investigations (OBI)[7] for identification and organization of entities in a *is-a* class hierarchy. The ontology is based on the proposal for a general framework for data mining presented in [6]. From the framework proposal we identified a set of basic entities of data mining. The entities listed above are used to describe different dimensions of

[3] BFO: `http://www.ifomis.org/bfo`
[4] RO: `http://www.obofoundry.org/ro/`
[5] OBO Foundry: `http://ontoworld.org/wiki/OBO_foundry`
[6] Protege: `http://protege.stanford.edu`
[7] OBI: `http://obi-ontology.org/`

data mining. These are all orthogonal dimensions and different combinations among these should be facilitated. Through combination of these basic entities, one should be able to describe most of the diversity present in data mining approaches today.

The version of OntoDM described in the current paper has been sufficiently updated in several ways. First, the structure of the ontology was fully aligned with the top level structure of the OBI ontology. This procedure requested revising the representation of several data mining entities and also introduced new entities in the ontology in order to represent different aspects of data mining entities: specification, implementation and process (e.g., the entity data mining algorithm was split into three entities each capturing different dimension of a description; algorithm specification, algorithm implementation and algorithm description). It is necessary to have all three aspects represented separately in the ontology as they have distinctly different nature and this will facilitate different usage of the ontology. The process aspect can be used for constructing data mining workflows and definition of participants of workflows and its parts; the specification aspect can be used to reason about components of data mining algorithms; the implementation aspect can be used for search over implementations of data mining algorithms and to compare various implementations. Second, we extended the set of relations used in the initial version with relations defined in the OBI ontology in order to express the relations between informational entities, entities that are realized in a process and processes. Finally, we extended the OBI classes with data mining specific classes for describing complex entities (e.g., data mining scenarios, queries).

In this version of the ontology we mapped the entities more closely to the structure of the OBI ontology. We use BFO upper level classes to represent entities which exist in the real world (i.e., processes, informational entities created in human brain), and in addition we use extensions of EXPO <*abstract entity*> to represent mathematical entities. Recently, due to the limitations of BFO in dealing with information, an Information Artifact Ontology (IAO) has been proposed as a spin-off of the OBI project[8]. Currently IAO is available only in a draft version, but we have included the most stable and relevant classes into OntoDM.

The OntoDM ontology includes and different types of formally defined ontological relations in order to achieve the desired level of expressiveness. The initial version of the ontology [10] included: fundamental relations (*is-a*, *part-of*), relations from RO [13] *has-participant*, *has-agent*, relations from EXPO/LABORS [15] (*has-representation*), relations from EXACT[14](*has-information*) and relations from OBI (*has-role*, *has-quality*, *has-specified-input*,*has-specified-output*). In this version of the ontology we include relations for expressing relationships between: a process and realizable entity (*realizes*), a planned process and objective specification (*achieves-planned-objective*) and informational entity about a realizable and a realizable entity (*is-concretized-as*). These relations are defined in the OBI ontology.

[8] IAO:`http://code.google.com/p/information-artifact-ontology/`

3 Conclusion

The ontology OntoDM as presented here is in its early stages of development and hence much work remains to be done. We first need to populate the proposed classes of data mining entities with individuals, identify shortcomings of our ontology in the process and refine the structure of OntoDM as needed. While the current version of OntoDM is expressed in OWL-DL, the next level of development would require it to be translated into first-order logic and extended with axioms.

Formalizing the knowledge about the domain of data mining and building of a heavy weight ontology of data mining is a time and resource consuming task and should be a community effort. That is why one of the aims of our work is also to invite researchers from the area of data mining to contribute to the ontology by suggesting improvements in the definitions of the entities and by using the knowledge in the ontology in their applications. Our goal is to have a mature ontology of data mining that is sufficient and expressive enough to describe the current trends in data mining. This would be also be a helpful step in developing standards for data mining and would lead towards an ontology of data mining investigations.

References

1. Abraham Bernstein, Foster Provost, and Shawndra Hill. Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Trans. on Knowl. and Data Eng.*, 17(4):503–518, 2005.
2. Alvis Brazma et al. Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nature Genetics*, 29:365–371, December 2001.
3. Peter Brezany, Ivan Janciak, and A Min Tjoa. *Data Mining with Ontologies: Implementations, Findings and Frameworks*, chapter Ontology-Based Construction of Grid Data Mining Workflows. IGI Global, 2007.
4. Mario Cannataro and Carmela Comito. A data mining ontology for grid programming. In *Proceedings of the 1st Int. Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGrid2003)*, pages 113–134, 2003.
5. Claudia Diamantini and Domenico Potena. Semantic annotation and services for kdd tools sharing and reuse. In *ICDMW '08: Proceedings of the 2008 IEEE International Conference on Data Mining Workshops*, pages 761–770, Washington, DC, USA, 2008. IEEE Computer Society.
6. Sašo Džeroski. Towards a general framework for data mining. In Saso Dzeroski and Jan Struyf, editors, *KDID*, volume 4747 of *Lecture Notes in Computer Science*, pages 259–300. Springer, 2006.
7. Alexandros Kalousis, Abraham Bernstein, and Melanie Hilario. Meta-learning with kernels and similarity functions for planning of data mining workflows. In Pavel Brazdil, Abraham Bernstein, and Larry Hunter, editors, *Proceedings of the Second Planning to Learn Workshop (PlanLearn) at the ICML/COLT/UAI 2008*, pages 23–28, 2008.

8. Ross D. King, Jem Rowland, Stephen G. Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N. Soldatova, Andrew Sparkes, Kenneth E. Whelan, and Amanda Clare. The Automation of Science. *Science*, 324(5923):85–89, 2009.
9. Riichiro Mizoguchi. Tutorial on ontological engineering - part 3: Advanced course of ontological engineering. *New Generation Comput.*, 22(2), 2004.
10. Panče Panov, Sašo Džeroski, and Larisa Soldatova. OntoDM: An ontology of data mining. In *ICDMW '08: Proceedings of the 2008 IEEE International Conference on Data Mining Workshops*, pages 752–760, Washington, DC, USA, 2008. IEEE Computer Society.
11. Ross J. Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., 1993.
12. D. Schober, W. Kusnierczyk, S. E Lewis, and J. Lomax. Towards naming conventions for use in controlled vocabulary and ontology engineering. In *Proceedings of BioOntologies SIG, ISMB 2007*, pages 29–32, 2007.
13. Barry Smith, Werner Ceusters, Bert Klagges, Jacob Kohler, Anand Kumar, Jane Lomax, Chris Mungall, Fabian Neuhaus, Alan L Rector, and Cornelius Rosse. Relations in biomedical ontologies. *Genome Biology*, 6(5), 2005.
14. Larisa N. Soldatova, Wayne Aubrey, Ross D. King, and Amanda Clare. The exact description of biomedical protocols. *Bioinformatics*, 24(13), 2008.
15. Larisa N. Soldatova and Ross D. King. An ontology of scientific experiments. *Journal of the Royal Society Interface*, 3(11):795–803, 2006.
16. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.
17. Monika Zakova, Petr Kremen, Filip Zelezny, and Nada Lavrac. Planning to learn with a knowledge discovery ontology. In Pavel Brazdil, Abraham Bernstein, and Larry Hunter, editors, *Proceedings of the Second Planning to Learn Workshop (PlanLearn) at the ICML/COLT/UAI 2008*, pages 29–34, 2008.