© The Author 2011. Published by Oxford University Press on behalf of The British Computer Society. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
Advance Access publication on 7 August 2011
doi:10.1093/comjnl/bxr077

Orange4WS Environment for Service-Oriented Data Mining

VID PODPEČAN[1], MONIKA ZEMENOVA[2] AND NADA LAVRAČ[1]

[1]*Jožef Stefan Institute, Ljubljana, Slovenia*
[2]*IZIP Inc., Prague, Czech Republic*
[*]*Corresponding author: vid.podpecan@ijs.si*

Novel data-mining tasks in e-science involve mining of distributed, highly heterogeneous data and knowledge sources. However, standard data mining platforms, such as Weka and Orange, involve only their own data mining algorithms in the process of knowledge discovery from local data sources. In contrast, next generation data mining technologies should enable processing of distributed data sources, the use of data mining algorithms implemented as web services, as well as the use of formal descriptions of data sources and knowledge discovery tools in the form of ontologies, enabling automated composition of complex knowledge discovery workflows for a given data mining task. This paper proposes a novel Service-oriented Knowledge Discovery framework and its implementation in a service-oriented data mining environment Orange4WS (Orange for Web Services), based on the existing Orange data mining toolbox and its visual programming environment, which enables manual composition of data mining workflows. The new service-oriented data mining environment Orange4WS includes the following new features: simple use of web services as remote components that can be included into a data mining workflow; simple incorporation of relational data mining algorithms; a knowledge discovery ontology to describe workflow components (data, knowledge and data mining services) in an abstract and machine-interpretable way, and its use by a planner that enables automated composition of data mining workflows. These new features are showcased in three real-world scenarios.

Keywords: data mining; knowledge discovery; knowledge discovery ontology; e-science workflows; automated planning of data mining workflows

Received 20 December 2010; revised 30 May 2011
Handling editor: Yannis Manolopoulos

1. INTRODUCTION

Fast-growing volumes of complex and geographically dispersed information and knowledge sources publicly available on the web present new opportunities and challenges for knowledge discovery systems. Principled fusion and mining of distributed, highly heterogeneous data and knowledge sources requires the interplay of diverse data processing and mining algorithms, resulting in elaborate data mining workflows. If such data mining workflows were built on top of a service-oriented architecture, the processing of workflow components (e.g. data mining algorithms) can be distributed between the user's computer and remote computer systems. Therefore, as the use of data mining algorithms (implemented as services) is no longer limited to any particular data mining environment, platform or scenario, this can greatly expand the domains where data mining and knowledge discovery algorithms can be employed. As an example, state-of-the-art data mining and knowledge discovery methods can become widely available in bioinformatics, business informatics, medical informatics and other research areas. Moreover, existing domain-specific services can become seamlessly integrated into service-oriented data mining environments.

There is another important aspect that makes data mining difficult for non-expert users. While the mutual relations of specialized algorithms used in the workflows and principles of their applicability are easily mastered by computer scientists, this cannot be expected from all end-users, e.g. life scientists. A formal capture of the knowledge of data mining tasks, and input–output characteristics of data mining algorithms is thus needed, which can be captured in the form of ontologies of relevant services and knowledge/data types, to serve as a basis for intelligent computational support in

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

knowledge discovery workflow composition. A formal capture of knowledge discovery tasks can then be used to improve repeatability of experiments and to enable reasoning on the results to facilitate their reuse.

This paper proposes a novel Service-oriented Knowledge Discovery (SoKD) framework, and its implementation that address the challenges discussed earlier. Building such a framework has been recognized as an important aspect of third-generation data mining [1]. A practical implementation of the proposed third-generation knowledge discovery platform, named Orange4WS (Orange for Web Services), has been conceived as an extension of the existing data mining platform Orange [2].

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 (best known examples being Weka [3], KNIME [4], RapidMiner [5] and Orange [2]), the Orange4WS platform provides the following new functionalities: (a) user-friendly composition of data mining workflows from local and distributed data processing/mining algorithms applied to a combination of local and distributed data/knowledge sources, (b) simplified creation of new web services from existing data processing/mining algorithms, (c) a knowledge discovery ontology of knowledge types, data mining algorithms and tasks and (d) automated construction of data mining workflows based on the specification of data mining tasks, using the data mining ontology through an algorithm that combines planning and ontological reasoning. This functionality is based on—and extends—a rich collection of data processing and mining components as well as data and information sources provided by local processing components as well as remote web services.

While each individual extension of the existing data mining technologies is not scientifically ground-breaking, the developed Orange4WS environment as a whole is a radically new data mining environment from many perspectives. From the machine learning and data mining perspective, the uniqueness of this platform is in the incorporation of propositional data mining as well as relational data mining algorithms (implemented in Prolog) in a unique data mining framework. On the other hand, from the Artificial Intelligence perspective, a unique feature of the proposed SoKD framework is the use of the developed knowledge discovery ontology of data types and data mining algorithms for automated data mining workflow construction using a fast-forward planning algorithm. From the e-Science perspective, Orange4WS substantially improves the existing environments that support manual construction of scientific workflows (such as Taverna [6] and Triana [7]) by incorporating advanced propositional and relational data mining algorithms as well as by supporting automated workflow construction. Finally, from the web services perspective, simplified creation of new web services from existing data processing/mining algorithms is a valuable extension of existing web-service-based environments. In the presented work, some of these unique features of Orange4WS are show-cased in three complex data mining scenarios, presented in Section 6.

The paper is structured as follows. Section 2 presents a motivating use case for developing and using a service-oriented knowledge discovery platform. Section 3 presents our approach to developing a novel SoKD framework and its implementation that upgrades the existing data mining system Orange into a new SoKD platform Orange4WS.[1] Sections 4 and 5 upgrade the implemented solution by introducing a knowledge discovery ontology of annotated types of data and knowledge resources, data mining algorithms and data mining tasks, and a facility for automated data mining workflow planning based on these annotations. Section 6 presents three use cases illustrating the advantages of the new platform. The Weka use case in Section 6.1 demonstrates that Weka algorithms can easily be integrated as services into the Orange4WS platform. The relational data mining use case in Section 6.2 shows how to combine propositional and relational data preprocessing and mining algorithms in a single environment. Section 6.3 illustrates a complex systems biology use case, which combines (a) a complex relational subgroup discovery system SEGS that uses biological ontologies and background knowledge for learning, and (b) a complex reasoning and visualization environment Biomine that includes data from numerous biological databases. Section 7 presents the related work. Section 8 concludes with a summary and plans for further work.

2. A SAMPLE TEXT MINING USE CASE

This section presents a motivating use case for developing and using a service-oriented knowledge discovery platform, including a user-friendly workflow editor. The use case is built upon text mining web services, available from LATINO[2] text mining library, which provides a range of data mining and machine learning algorithms, with the emphasis on text mining, link analysis and data visualization.

The goal of this use case is to produce a compact and understandable graph of terms, which could potentially give insights into relations between biological, medical and chemical terms, relevant to the subject of a user-defined query. A manually constructed Orange4WS workflow of processing components is shown in Fig. 1.

The use case demonstrates the need for a service-oriented platform able to combine publicly available data repositories (PubMed) with third-party data analysis tools (LATINO), specialized algorithms (Pathfinder) and powerful local visualization components (Orange graph visualizer).

[1] The Orange4WS software environment is available under the GPL licence at http://orange4ws.ijs.si.

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

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

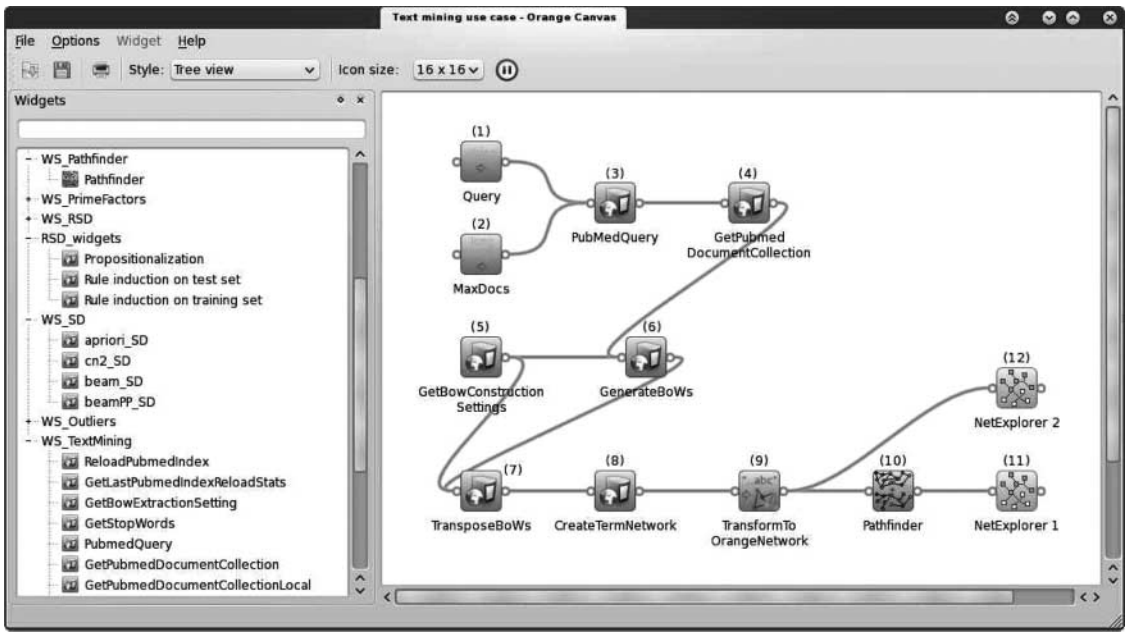

FIGURE 1. An Orange4WS workflow of text mining services in the Orange workflow execution environment. Components numbered 3, 4, 5, 6, 7, 8 and 10 are web services; components 1, 2 and 9 are Orange4WS supporting widgets; components 11 and 12 are instances of the native Orange graph visualizer.

PubMed search web services is queried with a user-defined query string and a parameter defining the maximal number of documents returned (components 1, 2 and 3). It returns a collection of IDs of relevant documents. Then, the obtained IDs are used to collect titles, abstracts and keyword of these documents (component 4). Next, bag-of-words (BoW) sparse vectors are created from the collection of words (component 6). To simplify the setting of parameters for unexperienced users, there is a service providing a suitable set of default values that can be used as an input to the web service that constructs BoW vectors (component 5). BoW vectors are then transposed (component 7) and a network of words/terms is created (component 8) in the .net format of the well-known Pajek social network analysis tool.[3] The resulting graph of terms in the .net format is then transformed into Orange's native data structure for representing graphs (component 9), and simplified using a sparse variant of the Pathfinder algorithm that is implemented as a web service (component 10). Finally, the original and pruned graph are visualized using the Orange's native `Network explorer` (components 11 and 12).

This Orange4WS workflow, implementing a complex text mining scenario, was designed and constructed manually in the Orange's user-friendly workflow editor. In Section 5, we will demonstrate how this workflow can be constructed automatically using a workflow planner and an ontology, which provides information about workflow operators and their input and output knowledge types.

3. THE ORANGE4WS PLATFORM

This section briefly describes the structure and design of the proposed software platform. We explain and comment our decisions concerning the selection of technologies and software tools used. The main part of this section describes the design of the Orange4WS platform and the accompanying toolkit for producing new web services.

3.1. Technological background

Our goal was to develop a simple, user-friendly software platform that is able to seamlessly integrate web services and local components in terms of workflow composition, originating from different communities (propositional data mining, relational data mining, text mining, systems biology, etc.), including also a knowledge discovery ontology to support the automatization of workflow construction.

The proposed software platform, named Orange4WS, is built on top of two open-source scientific-community-driven projects: (a) the Orange data mining framework [2] that provides a range of preprocessing, modeling and data exploration techniques and a user-friendly workflow execution environment, and (b) the Python Web Services project[4] (more

[3] User manual of the Pajek software tool for the analysis and visualization of large social networks is available at http://pajek.imfm.si/doku.php.

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

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

specifically, the Zolera SOAP infrastructure) that provides libraries for the employment and development of web services using the Python programming language by implementing various protocols, including SOAP, WSDL, etc.

In contrast with other freely available workflow environments such as Weka, Taverna, Triana, KNIME, RapidMiner, etc., the Orange4WS framework offers a rather unique combination of features: (a) a large collection of data mining and machine learning algorithms, efficiently implemented in C++ (Orange core); (b) a three-layer architecture: C++, Python, as well as Orange and Orange4WS Widgets; (c) a collection of very powerful yet easy to use data visualization widgets; (d) incorporation of propositional as well as selected relational data mining algorithms, (e) simplicity of workflow composition in the Orange canvas and (f) automated workflow construction using a knowledge discovery ontology and a planner. Moreover, by using an interpreted high-level programming language (Python), it is possible to avoid the compile-test-recompile development cycle. Also, high-level interpreted languages are a perfect choice for rapid software development using emerging web technologies such as RESTful web services[5] or WEB APIs.[6]

3.2. Platform design

Apart from the Orange core in C++ and its interface to the Python programming language, the Orange framework enables *visual programming* achieved by graphically composing processing components into *workflows*. Workflows are—essentially—executable visual representations of complex procedures. They enable repeatability of experiments as they can be saved and reused. Moreover, workflows make the framework suitable also for non-experts due to the representation of complex procedures as sequences of simple steps.

Workflow construction in Orange is supported by the *Orange Canvas*, an interactive graphical user interface component. It enables graphical construction of workflows by allowing workflow elements called *Orange Widgets* to be positioned in a desired order, connected with lines representing flow of data, adjusted by setting their parameters and finally executed.

An Orange Widget is defined by its inputs, outputs and the graphical user interface. Inputs and outputs are defined by the so-called typed channels, which specify the name of the channel, multiplicity (inputs only), data type, and a handler function (inputs only), which is invoked when the input data are available. For example, one of the most common inputs (outputs) is the Orange `ExampleTable`, a data structure used to store tabular and/or sparse data.

[5]A RESTful web service is a simple web service implemented using HTTP and the principles of REST [8].

[6]A Web API is a defined set of HTTP request messages along with a definition of the structure of response messages, most commonly expressed in JSON or XML.

Orange4WS extends and upgrades Orange on three levels. First, it provides tools that ease the employment of web services from the Python interpreter. Second, it upgrades the Orange Canvas with the ability to use web services as workflow components. Note that this level also provides a number of local Orange4WS widgets that are required for web service integration such as data transformation, data serialization and deserialization etc. Third, it enables automatic workflow construction by integrating a knowledge discovery ontology and a planner.

The functionality of Orange4WS is provided by several components (modules). The most important modules are: web service widget code generator, web service types extractor, web services stubs importer and the subsystem for automated workflow construction. The latter offers a number of supporting modules and functions as well as a general knowledge discovery ontology (KD ontology) that enables automated workflow planning. A high-level overview of the design of Orange4WS showing the main components and their interaction is shown in Fig. 2. The structure of the subsystem for automated workflow planning is discussed in more details in Section 5.

The `Web service stubs importer` module provides the base functionality that is required by the majority of other components. It dynamically loads web service consumer classes (web service client) generated by the Zolera SOAP infrastructure library using the provided link to the WSDL description of the service. These classes provide a high-level access to all methods provided by a given SOAP web service.

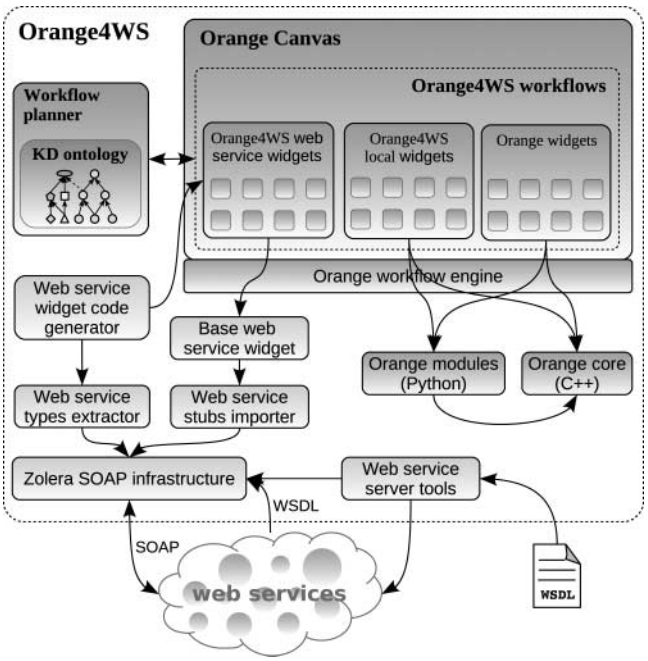

FIGURE 2. The structural design of the Orange4WS platform. A more detailed structure of the workflow planner component is shown in Fig. 8.

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

The role of the `Web service types extractor` module is to extract all type information for a given web service client instance, which was imported by the `Web service stubs importer` module. All web service functions and their input and output parameters are analyzed in a recursive manner, and full type as well as multiplicity[7] information is extracted. Simple data types are mapped to equivalents from the Python language, while complex types are mapped to objects, respectively.

The `Web service widget code generator` implements one of the main functionalities of Orange4WS: fully automated creation of widgets from web services. It relies on the modules, described earlier, to import generated web service consumer code and to define web service widget's inputs and outputs according to the extracted types. For a given web service, a widget is generated for each of its methods, and each input (output) parameter of a given method is mapped to one input (output) typed channel. Every web service widget is a subclass of the `BaseWebServiceWidget` class that takes care of the execution of the corresponding method, error detection and reporting, user notification, etc.

Since the main design goals of Orange4WS are simplicity and automatization, all technical details of creating new Orange4WS widgets from web services are summarized as a single user-interface command `import web service`. It invokes the web service widget code generator, which implements all required steps to enable access to the web services through a collection of Orange4WS widgets representing its methods. The details of actual invocation of a given web service method are thus 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. Essentially, the Orange Canvas is not aware of a non-local nature of web service widgets. Such simplicity is essential as the platform is intended to be used by scientists from very different domains, including bioinformatics, natural language processing, etc.

3.3. Composition and execution of workflows

One of the most important features of Orange, also inherited by Orange4WS, is an easy-to-use interactive workflow construction in Orange Canvas. Workflows components (widgets) represented with icons can be dragged to the appropriate position on the Canvas, while their inputs and outputs can be connected visually by drawing lines. The `Signal manager`, Orange's main workflow management component, enables or disables the connectivity of inputs and outputs according to their types. It also prevents the user from creating loops while connecting widgets by detecting cycles in the corresponding directed graph. If a widget supports the adjustment of its parameters, this can be done from widget's user interface, which can also enable data and results visualization as well as other interactive features. Finally, a constructed workflow can be saved into an XML format that corresponds to a predefined XML schema. This ensures repeatability of scientific experiments as well as user collaboration.

Orange4WS extends the manual composition of workflows in Orange with the ability to construct workflows automatically. Automated workflow construction is treated as a planning task where available workflow components represent operators while their input and output knowledge types represent preconditions and effects. The `Workflow planner` that is used to discover workflows satisfying the specified knowledge discovery task queries the developed knowledge discovery ontology where the available operators are annotated. The discovered workflows are available in the Orange's XML format, and can be loaded, instantiated and executed in Orange4WS. Section 5 discusses this feature of Orange4WS in details.

Orange's approach to workflow execution differs from the conventional workflow execution engines [9]. Because Orange workflows tend to be simple and as interactive as possible, the execution is provided on per-widget basis. As such, workflow components are treated as standalone steps in interactive analysis of data. Essentially, Orange does not provide a central workflow execution engine. Instead, the decision on how and when a widget is to be executed is left to the widget itself. Widgets are basically GUI wrappers around data analysis and visualization algorithms [2] implemented in Orange (note that Orange4WS extends Orange with web service widgets). In comparison with the Taverna workflow management system [10], this allows for rich and complex workflow components enabling user interaction and visualizations but also decreases the overall complexity of workflows (note that this is a well-known tradeoff between the complexity of workflows and the complexity of their components).

Essentially, there are two types of widgets: *flow-through* widgets and *on-demand* widgets. Flow-through widgets are executed as soon as all required input data are available. On the other hand, on-demand widgets are executed only when the user request their execution (all required input data must also be present). This type of execution is usual in the case of rich and complex widgets that require user interaction prior to the actual execution.

Orange4WS workflows are executed in the same manner as Orange workflows only with the following differences. First, Orange4WS provides components that simulate unconditional looping. The `Emitor` and `Collector` widgets enable processing of data sequences by emitting unprocessed data and collecting the results, respectively. Second, unlike Orange where the majority of widgets are of the on-demand type, all auto-generated Orange4WS web service widgets are flow-through. This corresponds to the base principle of

[7] Parameter multiplicity can be one of the following: required (1..1), optional (0..1), zero or more (0..*), one or more (1..*).

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

service-oriented design according to which a web service should encapsulate only one well-defined functionality that should not require complex user interaction. However, using the supporting modules and tools Orange4WS provides, any kind of web service widget can be developed. For example, an on-demand-type web service widget with progress polling was developed to interact with the computationally complex web service implementing the SEGS algorithm [11] (Section 6.3 discusses this service in more detail). Finally, the actual flow of data in Orange4WS workflows depends on the types of web services. In the case of location unaware web services, the results of the execution are always sent back to the caller (Orange4WS), while in the case of location aware web services,[8] Orange4WS only coordinates the execution while the actual data are not transmitted.

3.4. Creation of new web services

A separate part of our service-oriented knowledge discovery platform, also shown in Fig. 2 as the *Web service server tools* component, is a package of tools that ease the creation of new web services. These tools closely follow the general *WSDL first* design principle [12]. This principle promotes clearly designed, interoperable and reusable services by separating the design of interfaces from the actual logic. Essentially, our tools extend the Python language framework by using the Python Web Services package, enhanced with multiprocessing capabilities, security, logging and other related functionalities. By using these tools, any code can easily be transformed into a SOAP web service and used as an ingredient for Orange4WS workflow composition (or in any other workflow environment capable of using web services). Moreover, the provided tools support the creation of simple request/response stateless services as well as more complex batch (job) processing services, which can be used for time-consuming server-side processing. Such batch processing services also store results which can be retrieved later.

We have successfully created web services for the Relational subgroup discovery algorithm [13] implemented in Prolog. As a result, this relational data mining algorithm is available as a processing component in a propositional workflow-enabled environment. Also, the SEGS algorithm [11], a highly computationally complex rule discovery algorithm that uses biological ontologies as background knowledge, was transformed into a web service that greatly improved its processing capability, availability and also its ontology update mechanisms, which are now automated. Section 6 provides more details on these web services.

4. KNOWLEDGE DISCOVERY ONTOLOGY

To enrich the proposed knowledge discovery platform with semantics, we have developed the *Knowledge Discovery ontology* (the KD ontology, for short). The ontology defines relationships among the components of knowledge discovery scenarios, both declarative (various knowledge representations) and algorithmic. The primary purpose of the KD 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 knowledge discovery tasks. In addition, the ontology can also be used for automated annotation of manually created workflows facilitating their reuse.

An illustrative part of the top-level structure of the ontology is shown in Fig. 3. The three core concepts are: `Knowledge`, capturing the declarative elements in knowledge discovery; `Algorithm`, which serves to transform knowledge into (another form of) knowledge; `Knowledge discovery task`, which describes a task that the user wants to perform mainly by specifying the available data and knowledge sources and the desired outputs. The ontology is implemented in semantic web language OWL-DL.[9] The primary reasons for this choice were OWL's sufficient expressivity, modularity, availability of ontology authoring tools and optimized reasoners. The core part of the KD ontology currently contains around 150 concepts and 500 instances and is available online.[10] The structure of workflows is described using OWL-S.[11]

In the following sections, we describe `Knowledge` and `Algorithm` concepts in more detail and provide information on the annotation of algorithms available locally in the Orange4WS toolkit and in the LATINO library.

4.1. Knowledge

All the declarative components of the knowledge discovery process such as datasets, constraints, background knowledge, rules, etc. are instances of the `Knowledge` class. In data mining, many knowledge types can be regarded as sets of more elementary pieces of knowledge [14], e.g. first-order logic theories consist of formulas. This structure is accounted for through the property `contains`, so e.g. a first-order theory `contains` a set of first-order formulas.

Moreover, some knowledge types may be categorized according to the expressivity of the language in which they are encoded. For this purpose, we have designed a hierarchy of language expressivity (see Fig. 3, `Expressivity`). We further distinguish knowledge types that play special roles in knowledge discovery, e.g. the `Dataset` class, defined as `Knowledge`, that contains `Examples`. `Expressivity` can also be defined for datasets to distinguish between propositional datasets and relational datasets.

All the other types of knowledge such as pattern sets, models and constraints are clustered under the class `NonLogical-`

[8] Location aware web services only exchange references to the actual data that are usually stored on shared data storage resources.

[9] http://www.w3.org/TR/owl-semantics/.

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

[11] http://www.w3.org/Submission/OWL-S/.

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

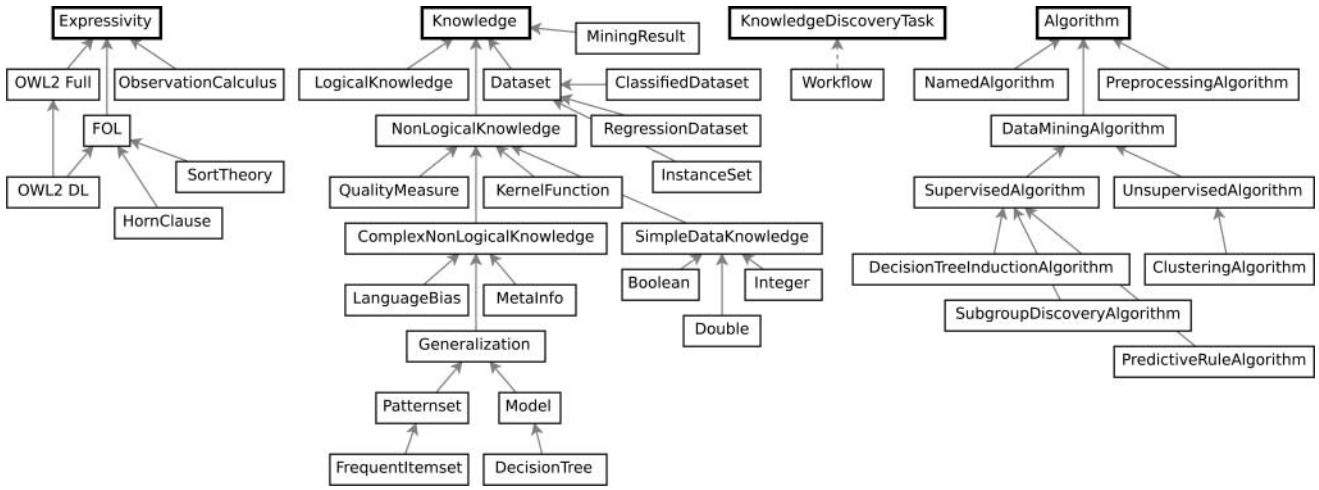

FIGURE 3. Part of the top level structure of the KD ontology (the whole ontology contains more than 150 concepts and 500 instances). Subclass relations are shown through solid arrows. The relation between `KnowledgeDiscoveryTask` and `Workflow`, shown through a dashed arrow, is defined as `forTask` relation. The `Workflow` class is a specialization of the OWL-S class `CompositeProcess`. The `Algorithm` class is a specialization of the OWL-S class `Process`, while the `NamedAlgorithm` class is a specialization of the OWL-S class `AtomicProcess`. The top-level classes shown in bold are subclasses of the predefined OWL class `Thing`.

`Knowledge`. It contains concept `Generalization`, which describes knowledge produced by data mining algorithms. The `Generalization` class currently contains two subclasses, `Patternset` and `Model` which can be distinguished by the property of decomposability and also by the type of algorithms used to produce them.

4.2. Algorithms

The notion of an algorithm involves all executable routines used in a knowledge discovery process, ranging from inductive algorithms to knowledge format transformations. Any algorithm turns a knowledge instance into another knowledge instance, e.g. inductive algorithms will typically produce a `Generalization` instance out of a `Dataset` instance. The `Algorithm` class is a base class for all algorithms, such as the *APriori* algorithm for association rule induction implemented in Orange or the *GenerateBows* algorithm implemented in the LATINO text mining library for constructing the bag of words representation of a collection of documents. For this work, we have refined the hierarchy of fully defined classes, such as `DecisionTreeAlgorithm` or `DataPreprocessingAlgorithm` for fine-grained categorization of data mining algorithms according to their functionality. This fine-grained hierarchy allows for the formulation of additional user constraints on the workflows. For example, constraints can refer to some particular category of data mining algorithms, e.g. `DiscretizationAlgorithm`, `FormatChangingAlgorithm`, `ClusteringAlgorithm`, etc.

Each algorithm configuration is defined by its input and output knowledge specifications and by its parameters. The `Algorithm` class is a specialization of the OWL-S class `Process` and an algorithm configuration is an instance of its subclass `NamedAlgorithm`.[12] Both the input knowledge and the parameters are instances of `AlgorithmParameter` and are 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.

4.3. Annotating 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. We have also annotated the components of the LATINO text mining library according to their WSDL descriptions, using the approach described by Kalyanpur *et al.* [15]. As an example, we present a definition of the `GenerateBows` algorithm. `GenerateBows` is defined as an algorithm that can be applied to a collection of documents and produces a bag of words representation of these documents. The settings are quite complex; therefore, they are provided as a single input object. The definition in the description logic notation using the extended ABox syntax [16] is shown in Fig. 4.

The locally available Orange4WS algorithms were annotated manually, since no systematic description of these algorithms,

[12]The `DataMiningAlgorithm` class represents categories of data mining algorithms, e.g. subgroup discovery algorithm or decision tree induction algorithms, while the `NamedAlgorithmClass` represents concrete algorithms, such as *CN2* for subgroup discovery or *C4.5* for decision tree algorithms.

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

$$
\begin{aligned}
\{\texttt{GenerateBows}\} &\sqsubseteq \texttt{NamedAlgorithm}\\
&\sqcap \exists \texttt{output} \cdot \{\texttt{GenerateBows-O-Bows}\}\\
&\sqcap \exists \texttt{input} \cdot \{\texttt{GenerateBows-I-Docs}\}\\
&\sqcap \exists \texttt{input} \cdot \{\texttt{GenerateBows-I-Settings}\}\\
\{\texttt{GenerateBows-I-Docs-Range}\} &\equiv \texttt{isRangeOf} \cdot \{\texttt{GenerateBows-I-Docs}\}\\
&\equiv \texttt{DocumentCollection}\\
\{\texttt{GenerateBows-O-Bows-Range}\} &\equiv \texttt{isRangeOf} \cdot \{\texttt{GenerateBows-O-Bows}\}\\
&\equiv \texttt{BowSpace}
\end{aligned}
$$

FIGURE 4. A definition of the `GenerateBows` method in the description logic notation using the extended ABox syntax.

e.g. in PMML[13] or WSDL[14] was available. The algorithms available in LATINO were also annotated manually based on their WSDL descriptions. The annotated algorithms also served as case studies to validate and extend the KD ontology, while the development of a procedure for semi-automatic annotation is a subject of future work.

5. AUTOMATED WORKFLOW CONSTRUCTION

The focus of this section is on automatic construction of abstract workflows of data mining algorithms. The mapping to concrete computational resources, particular data sets and algorithm parameters are not taken into account during abstract 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 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 the KD ontology, we implemented a planning algorithm capable of directly querying the KD ontology using the Pellet[15] reasoner. The main motivation for using Pellet was its ability to deal with literals, its availability in Protégé,[16] which we used for ontology development, and processing of SPARQL-DL [17] queries.

Our work was originally motivated mainly by complex relational data mining tasks, where the number of alternative workflows, which can be produced, is quite small, due to use of complex knowledge types and specialized algorithms [18]. This is also the case for the motivating text mining scenario from Section 2. The LATINO web services, which were annotated as specified in Section 4.3, can now be used in the process of automated workflow construction. Our planner was able to automatically (re)construct the workflow, presented in Section 2, according to the given instance of `KnowledgeDiscoveryTask` that specified the input data and the desired output. Note, however, that the *Pathfinder* algorithm is not present in the automatically generated workflow, as the corresponding web service is not yet annotated in the KD ontology. Figure 5 shows the automatically generated abstract workflow for the text mining scenario as well as an executable instantiation of the same workflow in the Orange Canvas inside Orange4WS.

As we have extended the KD ontology with annotations of algorithms available in the Orange and Orange4WS toolkits, 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.

5.1. Exploiting algorithm hierarchy

The planning algorithm used to generate abstract workflows automatically is based on the Fast-Forward (FF) planning system [19]. 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 during the planning time by posing an appropriate SPARQL-DL query to the KD ontology. In the original version of the planner [18], there are no mechanisms for exploiting the algorithms hierarchy. In this work, we have enhanced the algorithm in two ways: a hierarchy of algorithms based on defined classes and input/output specifications is 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`. It holds that $A_j \sqsubseteq A_i$ if for every input I_{ik} of 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 (defined by a class or set of classes in the KD ontology)

[13] http://www.dmg.org/pmml-v4-0.html.
[14] www.w3.org/TR/wsdl.
[15] http://clarkparsia.com/pellet/.
[16] http://protege.stanford.edu/.

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

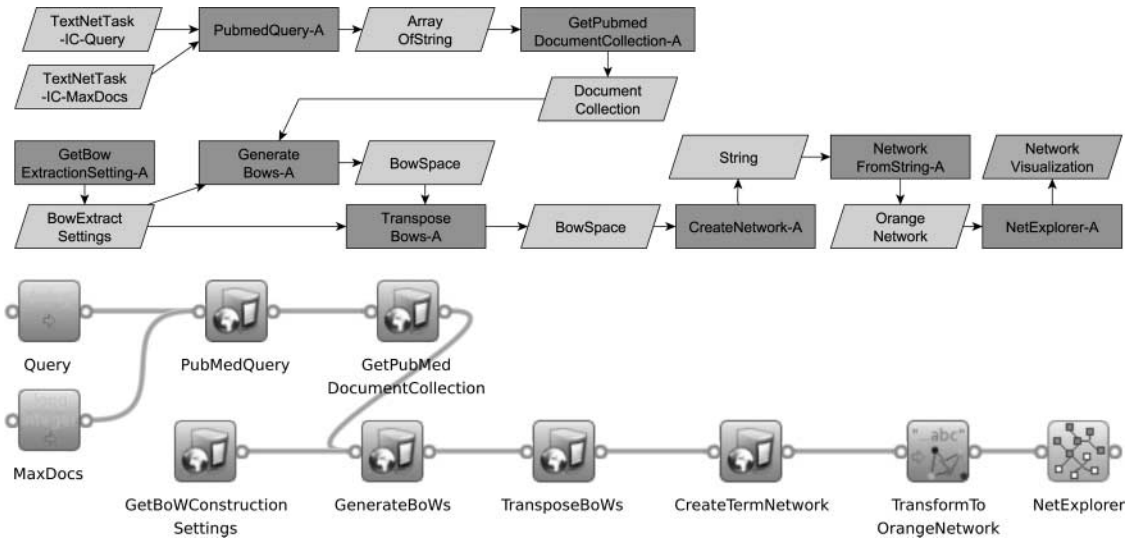

FIGURE 5. A schema of automatically generated abstract workflow and its executable instantiation in the Orange4WS environment. The underlying knowledge discovery task is a text-mining scenario of Section 2 for the analysis of a graphs of terms, obtained by querying the PubMed database using a publicly accessible web service.

```
task - instance of KnowledgeDiscoveryTask
maxSteps - max length of the workflow
constr - additional constraints
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);
```

FIGURE 6. A skeleton of the procedure for automatic workflow composition using the KD ontology.

in a linear part of the workflow are built into the planner. Additional constraints on workflows are used only for filtering the generated workflows during post-processing (procedure `filterWorkflows`). Workflows for all the members of an equivalence set are generated using the `expandWorfklows` procedure. The information about algorithms subsumption is also used when presenting the workflows to the user. The whole procedure for workflow generation is outlined in Fig. 6.

The generated workflows are presented to the user through interactive visualization, which enables the user to browse the workflows from the most abstract level to any specific combination of algorithm instances. Workflows consisting of the smallest number of steps are presented first. An example of a set of workflows generated for discovering association rules in Orange4WS is shown in Fig. 7.

The set of generated workflows shown in Fig. 7 illustrates the use of the algorithm hierarchy for workflow presentation. Since there are four discretization, four sampling, five ranking and six continuization algorithms, it would be infeasible to present all the generated workflows without using the algorithm hierarchy. Automatic selection of a relevant subset of workflows is non-trivial and is the subject of future work.

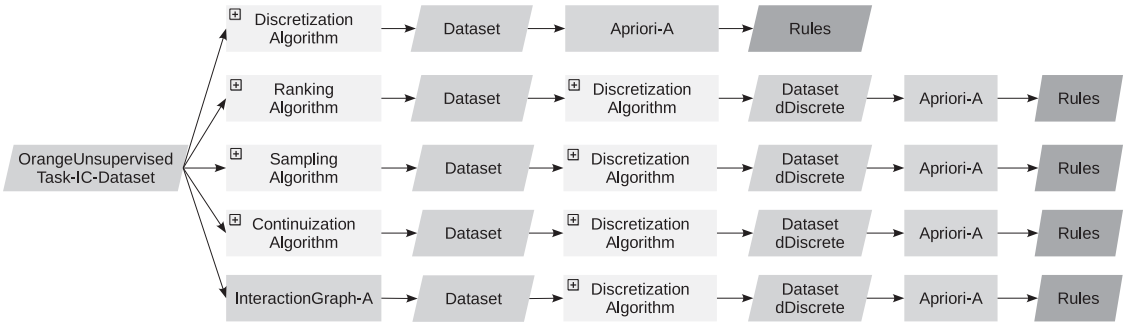

FIGURE 7. A set of automatically generated abstract workflows for discovering association rules in Orange4WS.

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

5.2. Integrating annotations and planning into Orange4WS

We have developed a framework for integrating our methodology into the Orange4WS platform, so that the workflows, which were constructed manually using the Orange4WS GUI and which contain only algorithms represented in the KD ontology, 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 Orange4WS representation is preserved in the annotation. Therefore, Orange4WS workflows can be recreated from the annotations and executed again in the Orange4WS toolkit. On the other hand, workflows generated by the planner using KD annotations of Orange4WS algorithms can be converted to the Orange4WS representation and executed in Orange4WS.

An overview of the framework is shown in Fig. 8. The Orange2Onto module, which acts as an interface between Orange4WS and the ontology representation, does not work directly with the internal representation of Orange4WS, but works with the OWS format used in the standard Orange distribution to store workflows in the XML format.

In order to formally capture the mapping between the internal Orange4WS 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 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.

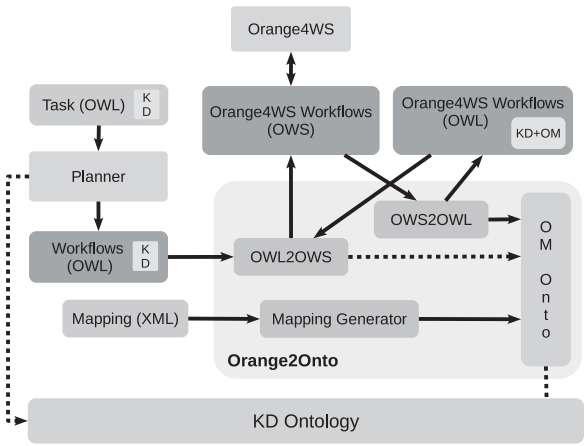

FIGURE 8. An overview of the framework for integration of annotations and planning into Orange4WS.

Annotation of a new algorithm available in Orange4WS 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) generate a mapping using the OM ontology by means of the provided tools.

Annotations of Orange4WS 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 a form of `Knowledge` expressed as a string. The annotations of such workflows can therefore be used for querying and for repeating the experiments; however, the generated annotation of the unknown algorithm is not suitable for planning.

The procedures for converting the Orange4WS representation to OWL and vice versa were implemented in Python using JPype[17] cross-language bridge to enable access to the Jena[18] ontology API implemented in Java.

6. USE CASES ILLUSTRATING THE UTILITY OF ORANGE4WS

This section presents three use cases from different domains, which illustrate some of the capabilities of the Orange4WS implementation. The presented workflows were not constructed automatically since not all workflow components and services were annotated in the KD ontology. Although the use cases presented here are simple, they give an overview of what our implementation is capable of, and illustrates the potential of web services technology for knowledge discovery.

6.1. Use case illustrating the availability of WEKA algorithms

A data mining practitioner would ideally like to have all the standard data mining algorithms at his disposal. While some of these are already provided in the Orange data mining toolkit[19] [2], data mining practitioners might also like to have

[17] http://jpype.sourceforge.net/.
[18] http://jena.sourceforge.net/.
[19] Implementations of classic data mining algorithms in Orange typically include several improvements, but some additions are not well documented, which is undesirable.

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

the classical Weka algorithms [3] available as well. Workflow tools, which are based on the Java technology (e.g. KNIME, RapidMiner, Taverna), typically include the Weka core (i.e. algorithm implementations), and manually written wrappers. In Orange4WS, this is simply achievable through Weka web services already available on the internet, or created with our tools described in Section 3.4. The advantage of a web-service-based approach is twofold. First, through web services, the computation is distributed among servers hosting the services. Second, the latest versions of underlying software libraries are provided automatically to all clients given that the services are updated regularly.

A collection of Weka web services has been made available by A. Bosin.[20] There are eight services available: `atributeRank`, `attributeSelect`, `datasetFilter`, `datasetDiscretize`, `modelTest`, `modelApply`, `classifierBuild` and `clustererBuild`. Although these services currently have poor semantics (they operate using string representations of native WEKA data types), major functionality of Weka is available (attribute evaluation, data filtering, model building and testing) and can be used in the construction of data mining workflows.

This simple but illustrative 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 testing set, (4) building a classifier and evaluating it on the test set and (5) reporting the results to the user. This is accomplished by connecting 16 processing entities, 6 of which are web services, 3 are native Orange widgets while the rest are the supporting widgets provided by Orange4WS (data transformation and creation of integral data types). Note, however, that annotating the semantics of these services would enable reasoning and automatic planning of such workflows, and incorporation into larger and more complex scenarios. The workflow, created and executed within the Orange4WS platform, is shown in Fig. 9.

For illustrational purposes, we tested the created workflow with the *voting* dataset. Seven most important attributes were chosen and stratified random sampling was used to partition the data into training (75% of all instances) and test (25% of all instances) data. Weka's J48 decision tree induction algorithm was used to build a decision tree model, which was then applied to the test data. The `modelTest` web service provided Weka's model evaluation output, which was finally visualized locally with a data viewer component.

6.2. Relational data mining use case

This use case is built upon the propositionalization-based approach to relational subgroup discovery. The implementation of the relational subgroup discovery algorithm RSD, developed by Železný and Lavrač [13], is used to illustrate the use

[20] http://www.dsf.unica.it/~andrea/webservices.html.

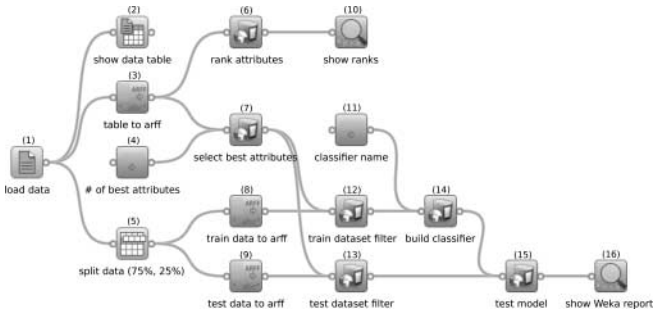

FIGURE 9. A workflow of Weka data mining services and local processing elements constructed within the Orange4WS platform. Components number 6, 7, 12, 13, 14, 15 are Weka web services; components 1, 2, and 5 are native Orange widgets. Other components are the supporting widgets provided by Orange4WS.

of our platform in a relational data mining scenario. The input to the RSD algorithm consists of a relational database containing (a) one main relation defined by a set of ground facts (training examples), each corresponding to a unique individual and having one argument specifying the class, and (b) background knowledge in the form of a Prolog program including functions, recursive predicate definitions, syntactic and semantic constraints, defined for the purpose of first-order feature construction.

Relational data mining and inductive logic programming are relatively separate research areas from standard propositional data mining. The main reason is the background of this research in logic programming, typically requiring a Prolog execution environment. Also, the data representation formalism is different (Prolog clauses), and taking into account relational background knowledge into the learning process requires a conceptually different approach from propositional learning, which only accepts tabular data as the input to a data mining algorithm. Consequently, standard data mining environments do not deal with relational data mining, and only once a service-oriented approach is considered, the two data mining frameworks can be handled within the same data mining environment.

The implementation of RSD, although efficient and stable, requires a YAP Prolog interpreter and specific implementation-related knowledge. Therefore, in order to be used in the Orange4WS environment, web services were created, which expose its abilities to the outside world. More specifically, using our tools for service development described in Section 3.4, we created a service for propositionalization and rule induction, respectively. In this use case, however, only the propositionalization service was used as we combined it with other, classic propositional data mining algorithms, also available as services. We employed the CN2-SD subgroup discovery algorithm [20], the SD algorithm [21], which implements beam search, and the APRIORI-SD algorithm [22]. It is worth noting that all three implementations are able to

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

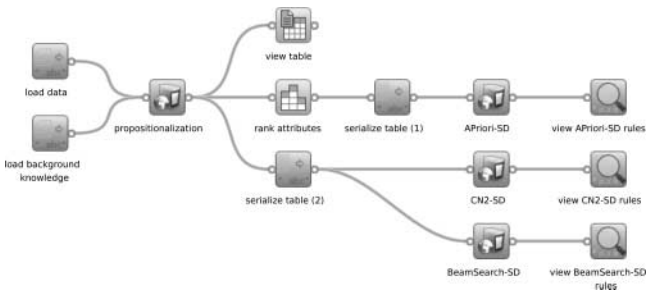

FIGURE 10. A workflow combining propositionalization of relational data, feature ranking, and subgroup discovery. Workflow components for `propositionalization`, `APRIORI-SD`, `CN2-SD` and `BeamSearch-SD` are web services, respectively.

produce results in the PMML[21] format, which makes them compatible with processing components outside Orange4WS.

The workflow of this use case, shown in Fig. 10, is illustrated on the Trains dataset [23], which is well-known in the area of relational data mining and Inductive Logic Programming. Altogether, 125 features were generated from the given relational data. As this was too much for the APRIORI-SD algorithm, feature selection was performed to obtain 10 best features (the other two algorithms were able to handle the complete feature set). For example, the highest ranked feature *f8* is as follows:

```
f8(Train) :- hasCar(Train,Car),
             carShape(Car,rectangle),
             carLength(Car,short),
             hasSides(Car,not_double).
```

Two example subgroups (one for each class), that are generated by the CN2-SD algorithm are shown as follows.

```
class = eastboundTrain IF f8  = true  AND
                          f82 = false AND
                          f25 = false AND
                          f40 = false

class = westboundTrain IF f121 = false AND
                          f5   = true  AND
                          f62  = false AND
                          f65  = false
```

6.3. Complex real-life systems biology use case

This use case is built upon two tools used in systems biology: the SEGS algorithm [11] and the Biomine system [24]. The combination of these systems, both of which make use of publicly available databases such as GO, Entrez, KEGG, PubMed, UniGene, OMIM and KEGG, enables novel scenarios for knowledge discovery from biological data.

In data mining terms, the SEGS (Search for Enriched Gene Sets) algorithm [11] is a specialized semantic subgroup discovery algorithm capable of inducing descriptions of groups of differentially expressed genes in terms of conjunctions of first-order features constructed from ontological relations available in public biological ontologies. The novelty of SEGS is that the method does not only test existing gene sets for differential expression but it also generates new gene sets that represent novel biological hypotheses. In short, in addition to testing the enrichment of individual GO and KEGG terms, this method tests the enrichment of newly defined gene sets constructed by the intersection and conjunctions of GO ontology terms and KEGG pathways.

The two new operators, *interact()* and *intersect()*, can yield to the discovery of gene sets that cannot be found by any other currently available gene set enrichment analysis software. They can be formalized as follows. If S is a gene set and ENTREZ is a database of gene–gene interactions, then the new interacting geneset INT(S) is defined as

$$\mathrm{INT}(S) = \{g : \exists g\prime \in S : \exists \mathrm{ENTREZ}(g, g\prime)\}. \quad (1)$$

Additionally, if S_1 is a term from the *molecular function* domain of the GO ontology, and S_2 belongs to the *cellular component* domain, and S_3 belongs to the *biological process* domain, and K is a KEGG pathway, then the gene set S defined by the *intersect()* operator is constructed as follows:

$$S_{S_1,S_2,S_3,K} = \{g : g \in \{S_1 \cap S_2 \cap S_3 \cap K\}\}. \quad (2)$$

As a result, the SEGS algorithm is able to discover complex rules that cannot be found by any other gene set enrichment analysis method or tool.

In the scope of the Biomine project, data from several publicly available databases were merged into a large graph (currently, ∼2 million nodes and 7 million edges) and a method for link discovery between entities in queries was developed. In the Biomine framework, nodes correspond to entities and concepts (e.g. genes, proteins, GO terms), and edges represent known, probabilistic relationships between nodes. A link (a relation between two entities) is manifested as a path or a subgraph connecting the corresponding nodes. The Biomine graph data model consists of various biological entities and annotated relations between them. Large, annotated biological data sets can be readily acquired from several public databases and imported into the graph model in a relatively straightforward manner. Currently used databases are: EntrezGene, GO, HomoloGene, InterPro, MIM, STRING, SwissProt, Trembl and UniProt.

The Biomine project provides the Biomine search web service (more specifically, a web API based on the HTTP

[21]The Predictive Model Markup Language (PMML) is an XML-based markup language that enables applications to define models related to predictive analytics and data mining and to share those models between PMML-compliant applications.

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

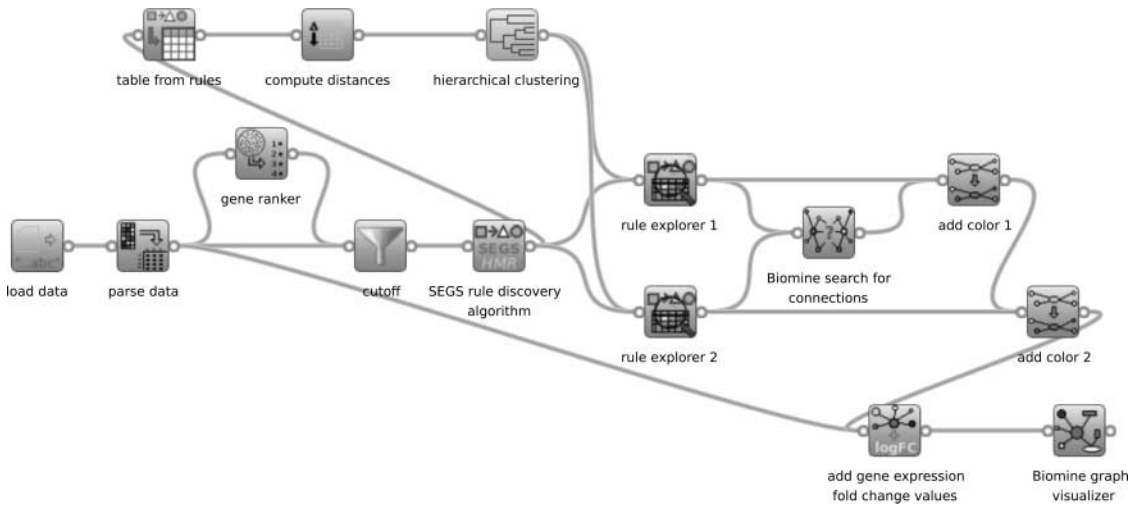

FIGURE 11. A workflow implementing the knowledge discovery scenario using the SEGS algorithm and the Biomine system. The component for computing rule distance and the interactive widget for hierarchical clustering are provided by Orange, other components are part of Orange4WS. The SEGS rule discovery algorithm is available as a SOAP web service while the Biomine search service is based on JSON.

protocol and JSON[22]), interactive web application and a powerful platform independent graph visualizer, implemented as a Java applet. The presented use case employs the Biomine search web service as well as the graph visualizer, which runs locally as an Orange4WS widget.

The original implementation of the SEGS algorithm was transformed into a SOAP 1.1 compatible web service using our tools described in Section 3.4. This greatly improved its flexibility and portability since the actual processing is now performed on more powerful server-side hardware employing massive parallel processing, and can be accessed from any workflow tool capable of calling web services. Moreover, publicly available databases, used by SEGS, can now be regularly updated by an automated update mechanisms. For space limitations, we do not provide a complete description of the SEGS service because it has a lot of input parameters but rather a short description of the provided functions and sample results.

As the SEGS algorithm has a large time complexity, the corresponding web service is designed as a partially stateful service. The SEGS service is actually a batch (job) processing service that stores the results of rule discovery; so they can be retrieved later using an unique user identifier. Apart from this, no consumer-specific context is stored or shared, and the invocations have no correlation to prior interactions. The service is able to report progress, and stores computed results indefinitely. It offers three functions: `runSEGS`, `getProgress` and `getResult`. The `getResults` function returns constructed rules, evaluated with the SEGS's built-in gene set enrichment tests (currently, Fisher's exact test, GSEA and PAGE).

A typical general scenario of knowledge discovery from gene expression data by using the SEGS algorithm and the Biomine system consists of the following steps:

(1) raw data preprocessing (normalization, missing values removal, merging, etc.);
(2) gene ranking (most typically, the Relief ranker or *t*-test is used);
(3) rule discovery using the SEGS algorithm;
(4) postprocessing of obtained SEGS rules (e.g. clustering);
(5) employing the Biomine system to discover interesting links, thus providing insights into the underlying biological processes.

The presented scenario, implemented as a workflow in the Orange4WS toolkit, is shown in Fig. 11. It is composed of local Orange4WS widgets, Orange components (clustering, example distances computation) and web services (the SEGS algorithm, Biomine search). First, the data are loaded and parsed, and the present genes are ranked. Then, the cutoff is applied to remove genes that seem not be involved in the observed biological processes. The resulting list of genes is fed to the SEGS algorithm to discover and evaluate rules composed of GO ontology terms, KEGG pathways as well as term interactions. The induced rules (if any) are sent to interactive hierarchical clustering component. The rules as well as clusters can be displayed in a user-friendly HTML browser where the user can select an interesting cluster or individual rule to be sent to the Biomine system.

The Biomine search web service returns the most reliable subgraph, which can be visualized using the provided interactive graph visualizer component. Such graphs offer non-trivial

[22]JSON is an acronym for JavaScript Object Notation, a lightweight text-based open standard designed for human-readable data interchange.

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

```
class of differentially expressed genes :-
  leukocyte differentiation AND
  interact(primary immunodeficiency)

  leukocyte differentiation AND
  membrane AND
  interact(natural killer cell mediated cytotoxicity)

  lymphocyte differentiation AND
  interact(fc epsilon RI signaling pathway) AND
  interact(fc gamma R-mediated phagocytosis)
```

FIGURE 12. The top three rules describing the class of differentially expressed genes from a classical acute lymphoblastic leukemia (ALL) dataset. The rules are composed of terms from the GO ontology and KEGG pathways.

insights into biological relations that are of interest to domain experts, and can potentially reveal previously unknown links (literature search is also included in Biomine).

For illustrative purposes, the presented knowledge discovery scenario was tested on a sample microarray dataset, a classical acute lymphoblastic leukemia (ALL) dataset [25]. The top three rules (according to the *P*-value obtained by permutation testing) that describe the class of differentially expressed genes are shown in Fig. 12. The rules are composed of terms from the GO ontology and KEGG pathways, respectively.

7. RELATED WORK

This section presents the work related to the key components of our framework: knowledge discovery domain formalization for workflow construction and reuse, workflow editing and execution environment and service-oriented architecture for knowledge discovery.

Construction of analytic workflows has been the topic of substantial research and development in the recent years. The best known systems include the Triana [7] workflow environment for P2P and Grid containing a system for integrating various types of middleware toolkits, and the Taverna [6] environment for workflow development and execution (primarily used in bioinformatics). However, these two system currently do not provide means for automatic workflow construction. Although Triana and Taverna are not specialized to support data mining tasks, there are projects aimed to incorporate general data mining components into these two sotfware systems. In the context of the DataMiningGrid project [26], which used Triana as a front end, generic and sector-independent data mining tools and services for the grid were developed. Similarly, a number of systems biology related data mining web services have become available in the myExperiment Virtual Research Environment[23] which can be used in Taverna (or any other tool capable of using web services).

On the other hand, the specialized data mining platforms Weka [3], KNIME [4], RapidMiner [5] and Orange [2] have mostly failed to recognize and adopt the web services computing paradigm, and the need for unification and formalization of the field of data mining. Currently, only RapidMiner offers some support for service-oriented computing through the Web Extension component, while none integrates an ontology of data, algorithms and tasks.

There has been some work on workflows for distributed data mining using a service-oriented architecture, e.g. Guedes *et al.* [27] and Ali *et al.* [28]. These systems focus on demonstrating the feasibility of a service-oriented approach for distributed data mining with regard to parallelization and distributed data sources, while none of these approaches enable automated data mining workflow construction.

Also relevant for our work is Weka4WS [29], a framework that extends the Weka toolkit to support distributed data mining on the Grid. The Weka4WS user interface supports the execution of both local and remote data mining tasks but only native Weka components and extensions are available, and the framework does not support arbitrary web services that can be found on the internet.

There exist several systems using a formal representation of data mining (DM) operators for automated workflow composition and ranking, including IDEA [30], NExT [31] and KDDVM [32], which focus solely on propositional data mining, and do not offer a general scientific workflow environment for data mining, whereas our approach allows also for the inclusion of complex relational data mining and text mining algorithms in a general workflow-based data mining environment.

Other efforts to provide a systematic formalization of the data mining tasks include projects MiningMart [33], DataMiningGrid [26], and a system described by Li *et al.* [34]. The first two focus on mining propositional patterns from data stored in a relational database. None of the systems provide means for automated workflow construction.

Another, very practically oriented approach to the generalization data mining algorithm implementations was introduced by Zaki *et al*. [35]. The proposed Data Mining Template Library is built using the principle of generic programming.[24] The library is generic with respect to the algorithm, data source and format, data structure, storage management and pattern to be mined. Nevertheless, this approach focuses solely on frequent pattern mining, and only provides generic templates in implementation-specific programming language instead of a general and independent ontology.

Parallel to our work, the OntoDM [36] ontology is currently being developed, adopting a principled top-down approach aimed at achieving maximal generality of the developed ontology. Given the complexity of the domain subject to be

[23] http://www.myexperiment.org/

[24] The Generic Programming paradigm focuses on finding suitable abstractions so that a single, generic algorithm can cover many concrete implementations.

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

modeled, the ontology is currently not sufficiently refined for the purpose of automated workflow construction. Also, unlike our ontology, OntoDM is not compatible with OWL-S. Recent work aimed at the development of a data mining ontology includes also [37, 38], where the work by Hilario *et al.* [37] has been influenced also by the knowledge discovery ontology described in this paper.

Solutions to the problem of web service composition in the context of planning are also relevant for our work. The work of Lecue *et al.* [39] relies on computing a *causal link matrix* for all the available services. In contrast, we work with a more general, non-linear notion of a plan. Work by Sirin *et al.* [40], Klusch *et al.* [41] and Liu et al. [42] translate an OWL description to a planning formalism based on PDDL. While the work presented in [41] and [42] use classical STRIPS planning, Sirin *et al.* [40] employ Hierarchical Task Network (HTN) planning. HTN is not applicable in our framework as it is not constrained to tree-based task decomposition. The approach presented by Liu *et al.* [42] and Klusch *et al.* [41] uses a reasoner in the pre-processing phase; we take a step further by integrating the reasoning engine directly with the planner. Planning directly in description logics is addressed by Hoffmann [43]. Currently, the algorithm can only deal with DL-Lite descriptions with reasonable efficiency.

8. CONCLUSIONS

This paper proposes a third-generation knowledge discovery framework and its implementation in a service-oriented data mining platform named Orange4WS. Based on the Orange data mining toolkit, which supports the 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 that is becoming more and more popular, we believe their use in data mining and knowledge discovery will increase rapidly. We have added semantic capabilities to the framework by proposing a methodology for integrating semantic annotation and planning into our data mining platform by means of the developed KD ontology. We have developed a planner, which exploits the hierarchy of algorithms annotated using the KD ontology.

In summary, the described service-oriented knowledge discovery paradigm shift, implemented in the Orange4WS platform, was achieved through the integration of latest achievements in the field of service-oriented approaches to knowledge discovery, knowledge discovery ontologies and automated composition of scientific workflows. This paradigm shift can potentially lead to the development of a novel intelligent knowledge discovery process model for data mining, extending the current CRISP-DM data mining methodology.[25] This paradigm shift will enable the orchestration of web-based data mining services and fusion of information of various formats, as well as design of repeatable data mining and information fusion workflows used in novel life science, bioinformatics and e-science applications.

Similarly to all other service-based solutions, a potential drawback of the presented platform is that the execution of workflows depends on the availability and reliability of remote services. As a result, the enactment of a selected workflow is not entirely under the control of the user, and there is no guarantee of successful completion of experiments. Also, the presented platform is still conventional in the sense that it does not support Web 2.0 collaborative work functionalities. Finally, our platform is platform-independent but system independence is not addressed. Note that this would require a complete reimplementation of the user interface and local processing components using web technologies only. Such reimplementation would allow for employing Orange4WS on any system equipped with a modern web browser, including mobile devices.

In future work, we will explore adding means for semantic web service discovery and their semi-automatic annotation. The planner will also be a subject of future improvements as we aim to incorporate the ability of satisfying user-defined constraints and preferences. We will add support for web service libraries other than ZSI such as the WSO2 web service framework (based on Apache Axis2/C), lightweight SOAP client SUDS and the *pysimplesoap* library, which will greatly expand the range of supported web services.

Finally, the proposed SoKD framework and its implementation in the Orange4WS platform will enable also for meta-mining of data mining workflows, which is a challenging topic of future research.

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

REFERENCES

[1] Finin, T. *et al.* (2007). National Science Foundation Symposium on Next Generation of Data Mining and Cyber-Enabled Discovery for Innovation (NGDM'07). Final Report.

[2] Demšar, J., Zupan, B., Leban, G. and Curk, T. (2004) Orange: From Experimental Machine Learning to Interactive Data Mining. In Boulicaut, J.-F., Esposito, F., Giannotti, F. and Pedreschi, D. (eds), *PKDD*, Lecture Notes in Computer Science 3202, pp. 537–539. Springer.

[3] Witten, I.H., Frank, E. and Hall, M.A. (2011) *Data Mining: Practical Machine Learning Tools and Techniques* (3rd edn). Morgan Kaufmann, Amsterdam.

[4] Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K. and Wiswedel, B. (2007) KNIME: The Konstanz Information Miner. In Preisach, C., Burkhardt, H., Schmidt-Thieme, L. and Decker, R. (eds), *GfKl*, Studies in Classification, Data Analysis, and Knowledge Organization, pp. 319–326. Springer.

[5] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M. and Euler, T. (2006) YALE: Rapid Prototyping for Complex Data Mining

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

Tasks. In Eliassi-Rad, T., Ungar, L.H., Craven, M. and Gunopulos, D. (eds) *KDD*, pp. 935–940. ACM.

[6] Roure, D.D., Goble, C.A. and Stevens, R. (2009) The design and realisation of the myExperiment virtual research environment for social sharing of workflows. *Future Gener. Comput. Syst.*, **25**, 561–567.

[7] Taylor, I., Shields, M., Wang, I. and Harrison, A. (2007) The Triana workflow environment: architecture and applications. *Workflows e-Sci.*, **1**, 320–339.

[8] Fielding, R.T. (2000) Architectural styles and the design of network-based software architectures. PhD Thesis, University of California, Irvine CA 92697, USA.

[9] Zupan, B., Leban, G., Demšar, J. and Curk, T. (2003) Widgets and Visual Programming. Technical Report. Bioinformatics Laboratory, Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia.

[10] Hull, D., Wolstencroft, K., Stevens, R., Goble, C.A., Pocock, M.R., Li, P. and Oinn, T. (2006) Taverna: a tool for building and running workflows of services. *Nucleic Acids Res.*, **34**, 729–732.

[11] Trajkovski, I., Lavrač, N. and Tolar, J. (2008) SEGS: search for enriched gene sets in microarray data. *J. Biomed. Inf.*, **41**, 588–601.

[12] Erl, T. (2005) *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[13] Zelezný, F. and Lavrač, N. (2006) Propositionalization-based relational subgroup discovery with RSD. *Mach. Learn.*, **62**, 33–63.

[14] Dzeroski, S. (2006) Towards a General Framework for Data Mining. In Dzeroski, S. and Struyf, J. (eds), *KDID*, Lecture Notes in Computer Science 4747, pp. 259–300. Springer.

[15] Kalyanpur, A., Pastor, D.J., Battle, S. and Padget, J.A. (2004) Automatic Mapping of OWL Ontologies into Java. In Maurer, F. and Ruhe, G. (eds) *SEKE*, pp. 98–103.

[16] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D. and Patel-Schneider, P.F. (eds) (2003) *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

[17] Sirin, E. and Parsia, B. (2007) SPARQL-DL: SPARQL Query for OWL-DL. In Golbreich, C., Kalyanpur, A. and Parsia, B. (eds), *OWLED*, CEUR Workshop Proceedings, Vol. 258. CEUR-WS.org.

[18] Žáková, M., Křemen, P., Železný, F. and Lavrač, N. (2008) Planning to Learn with a Knowledge Discovery Ontology. *Planning to Learn Workshop (PlanLearn 2008) at ICML 2008*. Helsinki, Finland.

[19] Hoffmann, J. and Nebel, B. (2001) The FF planning system: fast plan generation through heuristic search. *J. Artif. Intell. Res. (JAIR)*, **14**, 253–302.

[20] Lavrač, N., Kavšek, B., Flach, P.A. and Todorovski, L. (2004) Subgroup discovery with CN2-SD. *J Mach. Learn. Res.*, **5**, 153–188.

[21] Gamberger, D. and Lavrač, N. (2002) Expert-guided subgroup discovery: methodology and application. *J. Artif. Intell. Res. (JAIR)*, **17**, 501–527.

[22] Kavšek, B. and Lavrač, N. (2006) Apriori–SD: adapting association rule learning to subgroup discovery. *Appl. Artif. Intell.*, **20**, 543–583.

[23] Michie, D., Muggleton, S., Page, D. and Srinivasan, A. (1994) To the International Computing Community: A New East–West Challenge. Technical Report. Oxford University Computing laboratory, Oxford, UK.

[24] Sevon, P., Eronen, L., Hintsanen, P., Kulovesi, K. and Toivonen, H. (2006) Link Discovery in Graphs Derived from Biological Databases. In Leser, U., Naumann, F. and Eckman, B.A. (eds), *DILS*, Lecture Notes in Computer Science 4075, pp. 35–49. Springer.

[25] Chiaretti, S., Li, X., Gentleman, R., Vitale, A., Vignetti, M., Mandelli, F., Ritz, J. and Foa, R. (2004) Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival. *Blood*, **103**, 2771–2778.

[26] Stankovski, V., Swain, M.T., Kravtsov, V., Niessen, T., Wegener, D., Kindermann, J. and Dubitzky, W. (2008) Grid-enabling data mining applications with DataMiningGrid: An architectural perspective. *Future Gener. Comput. Syst.*, **24**, 259–279.

[27] Guedes, D., Meira, W. and Ferreira, R. (2006) Anteater: a service-oriented architecture for high-performance data mining. *IEEE Internet Comput.*, **10**, 36–43.

[28] Ali, A.S., Rana, O.F. and Taylor, I.J. (2005) Web Services Composition for Distributed Data Mining. *ICPP Workshops*, pp. 11–18. IEEE Computer Society. Oslo, Norway.

[29] Talia, D., Trunfio, P. and Verta, O. (2005) Weka4WS: A WSRF-Enabled Weka Toolkit for Distributed Data Mining on Grids. In Jorge, A., Torgo, L., Brazdil, P., Camacho, R. and Gama, J. (eds), *PKDD*, Lecture Notes in Computer Science 3721, pp. 309–320. Springer.

[30] Bernstein, A., Provost, F.J. and Hill, S. (2005) Toward intelligent assistance for a data mining process: an ontology-based approach for cost-sensitive classification. *IEEE Trans. Knowl. Data Eng.*, **17**, 503–518.

[31] Bernstein, A. and Dänzer, M. (2007) The NExT System: Towards True Dynamic Adaptations of Semantic Web Service Compositions. In Franconi, E., Kifer, M. and May, W. (eds), *The Semantic Web: Research and Applications*, Chapter 52, Lecture Notes in Computer Science 4519, pp. 739–748. Springer, Berlin, Heidelberg.

[32] Diamantini, C., Potena, D. and Storti, E. (2009) Ontology-Driven KDD Process Composition. In Adams, N.M., Robardet, C., Siebes, A. and Boulicaut, J.-F. (eds), *IDA*, Berlin, Lecture Notes in Computer Science 5772, pp. 285–296. Springer.

[33] Morik, K. and Scholz, M. (2003) The MiningMart Approach to Knowledge Discovery in Databases. In Zhong, N. and Liu, J. (eds), *Intelligent Technologies for Information Analysis*, pp. 47–65. Springer.

[34] Li, Y. and Lu, Z. (2004) Ontology-based universal knowledge grid: enabling knowledge discovery and integration on the grid. *IEEE SCC*, pp. 557–560. IEEE Computer Society. Shanghai, China.

[35] Hasan, M.A., Chaoji, V., Salem, S., Parimi, N. and Zaki, M.J. (2005) DMTL: A Generic Data Mining Template Library. *Proc. Workshop on Library-Centric Software Design, Object-Oriented*

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013

Programming, Systems, Languages and Applications Conf. (OOPSLA '05), San Diego, CA, USA, pp. 53–63. Rensselaer Polytechnic Institute.

[36] Panov, P., Džeroski, S. and Soldatova, L.N. (2008) OntoDM: An Ontology of Data Mining. *ICDM Workshops*, pp. 752–760. IEEE Computer Society. Pisa, Italy.

[37] Hilario, M., Kalousis, A., Nguyen, P. and Woznica, A. (2009) A Data Mining Ontology for Algorithm Selection and Meta-Mining. *Proc. 2nd Workshop on Service-Oriented Knowledge Discovery (SoKD '09): Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery, ECML PKDD Conf.*, Bled, Slovenia, September 7–11, pp. 76–87.

[38] Diamantini, C., Potena, D. and Storti, E. (2009) KDDONTO: An Ontology for Discovery and Composition of KDD Algorithms. *Proc. 2nd Workshop on Service-Oriented Knowledge Discovery (SoKD '09): Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery, ECML PKDD Conf.*, Bled, Slovenia, September 7–11, pp. 13–24.

[39] Lécué, F., Delteil, A. and Léger, A. (2007) Applying Abduction in Semantic Web Service Composition. *ICWS*, pp. 94–101. IEEE Computer Society. Salt Lake City, Utah, USA.

[40] Sirin, E., Parsia, B., Wu, D., Hendler, J.A. and Nau, D.S. (2004) HTN planning for web service composition using SHOP2. *J. Web Sem.*, **1**, 377–396.

[41] Klusch, M. and Gerber, A. (2005) Semantic Web Service Composition Planning with OWLS-XPlan. *Proc. 1st Int. AAAI Fall Symp. Agents and the Semantic Web*, pp. 55–62. Arlington, Virginia, USA.

[42] Liu, Z., Ranganathan, A. and Riabov, A. (2007) A planning approach for message-oriented semantic web service composition. *AAAI*, pp. 1389–1394. Proc. of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22–26, 2007, Vancouver, British Columbia, Canada.

[43] Hoffmann, J. (2008) Towards Efficient Belief Update for Planning-Based Web Service Composition. In Ghallab, M., Spyropoulos, C.D., Fakotakis, N. and Avouris, N.M. (eds), *ECAI*, Frontiers in Artificial Intelligence and Applications 178, pp. 558–562. IOS Press.

Downloaded from http://comjnl.oxfordjournals.org/ at Jozef Stefan Institute on March 25, 2013