SINDBAD SAILS: A Service Architecture for Inductive Learning Schemes

Jörg Wicker, Christoph Brosdau, Lothar Richter, and Stefan Kramer

Technische Universität München, Institut für Informatik I12, Boltzmannstr. 3, D-85748 Garching b. München, Germany
{joerg.wicker, lothar.richter, stefan.kramer}@in.tum.de,
christoph.brosdau@campus.lmu.de

Abstract. The paper presents SINDBAD SAILS (Service Architecture for Inductive Learning Schemes), a Web Service interface to the inductive database SINDBAD. To the best of our knowledge, it is the first time a Web Service interface is provided for an inductive database. The combination of service-oriented architectures and inductive databases is particularly useful, as it enables distributed data mining without the need to install specialized data mining or machine learning software. Moreover, inductive queries can easily be used in almost any kind of programming language. The paper discusses the underlying concepts and explains a sample program making use of SINDBAD SAILS.

1 Introduction

Inductive databases are databases which handle data, patterns and models, and which support the complete knowledge discovery process on the basis of inductive query languages. Many of the proposals for inductive databases and constraint-based data mining are restricted to single pattern domains (such as itemsets or molecular fragments) or single tasks, such as pattern discovery or decision tree induction. Although the closure property is fulfilled by many of those approaches, the possibilities of combining various techniques in multi-step and compositional data mining are rather limited. In previous work [1, 2], we presented a prototype system, SINDBAD (structured inductive database development), supporting the most basic preprocessing and data mining operations such that they can be combined more or less arbitrarily. One explicit goal of the project is to support the complete knowledge discovery process, from pre-processing to post-processing, on the basis of database queries. The research extends ideas discussed at the Dagstuhl perspectives workshop "Data Mining: The Next Generation" [3], where a system of types and signatures of data manipulation and mining operators was proposed to support compositionality in the knowledge discovery process. One of the main ideas was to use the simplest possible signature (mapping tables onto tables) as a starting point for the exploration of more complex scenarios. The relational model was chosen, as it possesses several useful properties, from closure to convenient handling of collections of tuples. Moreover, it is possible to take advantage of mature and optimized database technology. Finally, systems supporting (variants of) SQL are well-known and established, making it easier to get users acquainted with new querying facilities.

In this work, we present a Web Service interface to SINDBAD. Using this interface, all features of SINDBAD can be made available on a dedicated server. In this way, SINDBAD data mining services can be started from arbitrary clients. It is possible to distribute tasks over multiple servers to decrease the load on each machine. Another effect is the availability of features of SINDBAD in many different programming languages and on many different platforms. Thus, machine learning and data mining methods can be combined easily with native programming language constructs, such as conditional statements or loops, without having to install specialized libraries or software packages.

This paper is organized as follows. First we give a brief introduction to SINDBAD, its concepts and its implementation. For a more detailed introduction we have to refer to previous publications [1, 2]. Subsequently, we give an overview of Web Services in general. The following sections describe the Web Service interface of SINDBAD. Finally, we summarize related work on Web Services for machine learning and data mining.

2 SINDBAD

2.1 Concepts and SiQL

SiQL (structured inductive database query language), the query language of the SINDBAD system, is a straightforward extension of SQL. Instead of just adding complicated data mining operators to SQL, we focused on incorporating small, but extensible and adjustable operators that can be combined to build more complex functions. The query language supports the knowledge discovery process by a successive transformation of data. As each pre-processing and data mining operator returns a table, the queries can be nested arbitrarily, and the kind of compositionality needed in multi-step data mining can be achieved easily. The mining operators were designed in analogy to relational algebra and SQL: For instance, we made heavy use of the extend-add-as operator which adds the results of a data mining operation as new columns to a relation (see below). Moreover, we devised a feature-select clause in analogy to the select clause. It selects features that fulfill a given condition, e.g., an information gain above a certain threshold or the highest correlation coefficient with respect to a given column.

From each category of preprocessing/mining algorithms, we implemented most fundamental representatives. For discretization, we included equal frequency or equal width, for feature selection a filter approach based on information gain or variance, for pattern mining, the computation of frequent itemsets using APriori, for clustering k-Medoids, and for classification k-nearest neighbor and rule induction (pFOIL, a propositional variant of FOIL [4]). External tools can be integrated via wrappers.

We adopted the `extend` [5] operator to add the results of various data mining operations as new attributes to a given relation. It computes a function for each tuple and adds the result as the value of a new attribute. In SINDBAD, the `extend` operator adds the result of clustering, instance- or rule-based predictions, and sampling to a table. For clustering/classification, the cluster/class membership is indicated by an additional attribute. In sampling, the sample membership determined by a random number generator is given in the new attribute. In this way, we can split datasets, for instance, into a training set and a test set. For clustering and instance-based learning (k-nearest neighbor), other methods for handling tuples and distances are provided as well.

One of the central concepts of SINDBAD is that of distances between objects. This is not restricted to tuples of a single relation. Using relational distance measures, it is possible to apply clustering and instance-based learning to multi-relational data [6]. Most relational distance measures are based on recursive descent and set distances, i.e., distances between sets of points. In the simplest case, the computation of a distance between two sets of tuples A and B boils down to computing the minimum distance between two elements of each set (single linkage), $d_{SL}(A, B) = \min_{a \in A, b \in B} d(a, b)$.

One of the most recent additions is the integration of full-fledged predictive models in the form of rule sets. For simplicity, we chose pFOIL, a propositional variant of the traditional FOIL algorithm [4]. The addition of models required significant extensions of the data model of the system. Models can be composed of component models. The evaluations of *component models* (e.g., class predictions) can be aggregated by *combining functions*. Combining functions can be defined in terms of logical or arithmetic operators. In this way, rule sets, weighted rule sets, trees, linear classifiers, and ensembles can be handled conveniently. For details of the query language and the implementation, we have to refer to a more comprehensive publication [1]

2.2 Implementation

The SINDBAD prototype is implemented in Java on top of a PostgreSQL database. For parsing the queries, we used the lexical analyzer generator JFlex (see `http://jflex.de/`) and the parser generator Cup (see `http://www2.cs.tum.edu/projects/cup/`). The implementation supports arbitrarily nested queries. In the future, we are planning to integrate a full-fledged analysis of parse trees, opening possibilities for query optimization. The system is built on top of PostgreSQL (see `http://www.postgresql.org/`), an open source relational database management system. Most of the inductive queries are broken down and translated into a larger number of less complex non-inductive queries. The implementation of data mining features as PostgreSQL functions seems to be critical for performance.

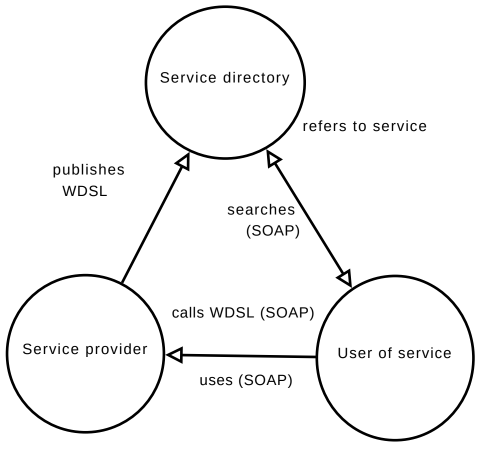
Table 1. Excerpt of a SINDBAD session. SiQL is used to learn rules for the activity of chemical structures against HIV. In the first few queries, the partitioning into a training and a test set is configured (20)-(22). In the next step, a test (23) and a training (24) set is generated using the given parameters (25% of examples set into test set, the remaining into training set, the distribution in column `activity` is preserved in both sets). Subsequently, the FOIL algorithm is configured to use MDL pruning during the learning process (26) and rules are learned on the training data using the FOIL algorithm (27). The resulting rules are stored in the relation `hiv_rules`. Finally, the learned rules are displayed (28) and applied to the test set (29). In this step, an additional column per learned rule is appended to the test set showing the prediction of the rule.

```
(20)> configure sampling_method = 'holdout';
(21)> configure sampling_percentage = '0.25';
(22)> configure sampling_keep_ratio_column =
     > 'activity';
(23)> create table hiv_train_test as
     > extend hiv_formatted
     > add sample membership as test_flag;
(24)> create table testset as
     > select * from hiv_formatted
     > where  test_flag = false;
(25)> create table trainset as
     > select * from hiv_formatted
     > where  test_flag = true;
(26)> configure foil_mdl = 'true';
(27)> create table hiv_rules as learn rules
     > for activity in  trainset;
(28)> show table hiv_rules;
(activity = true  <-  f683 = true AND
 f262 = true AND f219 =  true AND
 f165 = true)
...
(29)> extend testset add
      > model prediction of
      >  hiv_rules
      > as learned_activity;
```

3 Web Services

A Service-Oriented Architecture (SOA) is a design paradigm for distributed computational resources, described by their capabilities and typically made accessible on the internet [7]. Encapsulating functionality in an SOA, parts of a software system can be reused regardless of specific requirements on the underlying system, programming language or location of the provided service.

Fig. 1. Components of a Web Service environment



One possible implementation of an SOA is a Web Service (see Figure 1). Web Services are offered on the internet and can be accessed by the Simple Object Access Protocol (SOAP). The specification of a Web Service is split into three parts:

1. SOAP (Simple Object Access Protocol), an XML-based message format for the communication and embedding into transport protocols,
2. WSDL (Web Service Description Language), an XML-based description language to describe the Web Service, its interfaces and parameters, and
3. UDDI (Universal Description, Discovery and Integration Protocol) (optional), the directory service for Web Services, specifying the standardized directory structure for administration and search for Web Service meta-data.

4 SINDBAD SAILS

SINDBAD SAILS (SINDBAD Service Architecture for Inductive Learning Schemes) introduces an implementation of a Web Service interface for SINDBAD. It can be used to access SINDBAD on one or more dedicated servers.

Thus, it is possible to access it from multiple clients and distribute tasks over multiple servers.

4.1 Motivation

Building a Web Service on top of an inductive database offers many advantages: First of all, it is possible to run the (in most cases) computationally intensive operations on separate systems and distribute work that can be done simultaneously. As the computations are carried out on the server, the hardware requirements on the client are not very high. While this feature can be achieved by other implementations than a Web Service interface, common implementations in most cases require certain packages or software on the client machines. When using Web Services, this does not necessarily apply. In some cases it is beneficial to install packages to handle the access to the service. This depends on the used programming language on the clients. However, as the implementation of the service and the client are completely independent, it is up to the user which language, package, or software is used.

The distinction between the implementation of the data mining and preprocessing algorithms on the server running SINDBAD and the implementation of the user code on the client side makes it easier to use the data mining algorithms. The users do not need to know the details of the algorithms: They just need to submit the data in the right format and send it to the server.

The advantage of inductive databases compared to other possible implementations is due to the status of patterns and models in such systems. Just as regular data items, patterns and models are viewed as *first-class objects* in inductive databases. Taking advantage of a service-oriented architecture, it is possible to transfer data, patterns and models from one inductive database to another. In this way, distributing data mining tasks and integrating methods and results from multiple servers becomes an easy task.

Finally, the use of data mining and machine learning features in programming languages has not received much attention so far. Whereas machine learning is considered important in the context of reasoning, or more generally, artificial intelligence systems [8], the use of inductive queries in regular computer programs has not yet been discussed in the literature. The approach differs from R and MATLAB implementations and interfaces, and older libraries like MLC++ [9], in its additional layer of abstraction offered by the query language SiQL. Using SINDBAD SAILS, it is easily possible to use basic machine learning and data mining in (almost) arbitrary programming languages, without the need to install specialized software.

4.2 Features and Implementation

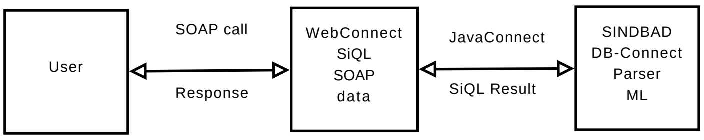
SINDBAD SAILS is a combination of a SINDBAD database and a Web Service implemented in PHP. The library PEAR::SOAP (see `http://pear.php.net/package/SOAP/`) is used to generate the WSDL file from the main class that handles the input and output and organizes further processing. The PHP classes

Table 2. Data Mining methods and their parameters

Method	Details
`cluster( numberCluster, data)`	Method to start execution of KMedoids algorithm in SINDBAD. Parameters set the number of clusters and the input data.
`classify( numberNeighbours, data, classified, column)`	Starts classification with KNearest Neighbor algorithm. Parameters set the number of neighbours to be considered, the data to classify, the training data and the column containing the class informations.
`ruleLearning( target, data)`	Learns rules for a given target concept in the data. The result is returned in multiple relations.
`frequentItems( minsupport, data)`	Searches frequent itemsets in the input relation. A parameter sets the minimum support of an itemset to be considered as frequent.

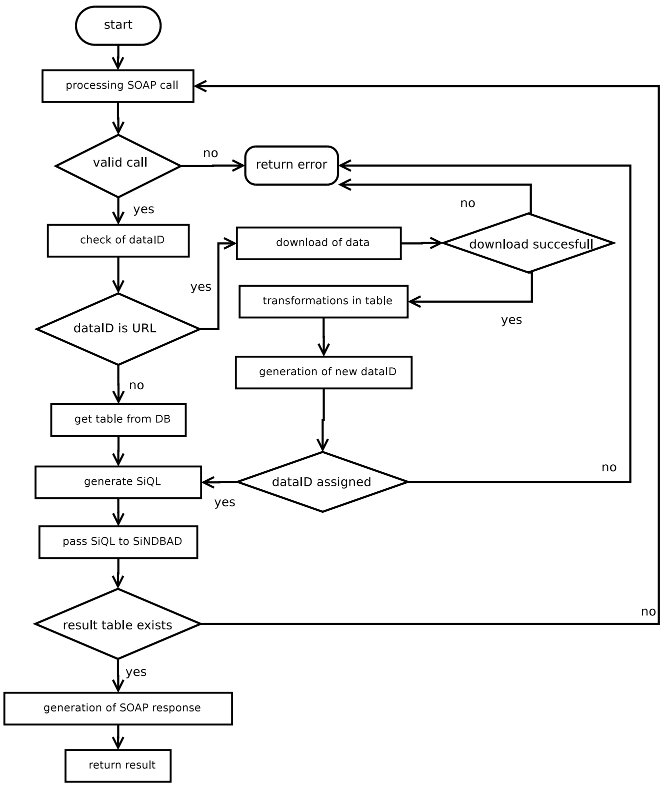
provide several methods which can be called over the http interface (see Table 2). All the functionality needed to archieve WSDL/PHP handling is provided by the PEAR:SOAP library. An integration of a SINDBAD call over a SINDBAD SAILS interface into an external programm is executed in three steps (see Figure 2). The user – in this case the program – issues a SOAP call using the SOAP http interface. The interface translates the method into one or more SiQL queries and passes them by a command line call to SINDBAD. In SINDBAD, the SiQL calls are processed, and results are returned to the SOAP interface. The SOAP interface returns the results, depending on a user-defined parameter, on a certain level of detail (see below).

Fig. 2. Workflow of a SINDBAD SAILS call



A detailed overview over a SINDBAD SAILS call is given in Figure 3: Data Mining methods are translated into SiQL queries and executed by the underlying SINDBAD database. Methods for uploading the data on and downloading results from the server are provided. The input for the algorithms is either uploaded from a given URL or a result of a former query on the database is used. We decided not to upload the data via transformation into XML and back, as this would require the client to implement this functionality, and we intended to keep the requirements on the client side as low as possible. The data is processed by the preprocessing and data mining algorithms and the results are stored on the server. They can be downloaded by using a PHP method or used in further computations. To use data from earlier sessions, an identification number is associated with each result. This id can be passed to the PHP methods instead of the URL to the input data. Additionally, each intermediate result can be obtained from the server. Each Web Service method call returns two integers, the identifier of the input table and the identifier of the result table. The generated queries are passed to SINDBAD via command line, the output of SINDBAD is parsed for thrown exceptions and problems. If any exception is thrown, it is passed to the client system in a special type which is described in the SOAP specifications.

Fig. 3. UML Activity diagram of a SINDBAD SAILS call



The results of the session are returned to the user depending on a parameter which sets the level of detail of the results. In the default case, the complete result tables are returned, but it is possible to reduce the returned output. For instance, for classification tasks, it is possible to return the predicted and actual classes of each instance instead of the complete instance including that information. If even less information is sufficient, the system can return just the confusion matrix or the error rate/AUC. In itemset mining, the full set of solutions, the borders, or just aggregate statistics of the results can be returned.

A sample call in Java is shown in Table 3. To connect to the Web Service, the Axis package of Apache is used (see `http://ws.apache.org/axis/`). First, a connection to the Web Service is established. Then the task is set to frequent itemset mining, which is done by the APriori algorithm. Finally, an URL of the data is sent to the service for download. With this URL, the minimum support and the level of detail for the results is passed to SINDBAD. If any errors occur during the execution of the APriori algorithm, an exception is thrown, containing a detailed error message from SINDBAD. The example is given in Java, but similar programs can be written in almost any programming language.

Table 3. Simple Java example program using the Web Service interface of SINDBAD. In the example, the Axis package of Apache is used to establish a connection to the Web Service.

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import javax.xml.namespace.QName;
public class SoapClient
{
    public static void main(String[] args) throws Exception
    {
        String endpoint = "http://SINDBAD.in.tum.de/soap";
        Service service = new Service();
        Call call = (Call) service.createCall();
        call.setTargetEndpointAddress( new java.net.URL(endpoint) );
        call.setOperationName("frequentItemsets");
        Object[] returnset =
            (Object[]) call.invoke(
                new Object[]{
                    'http://wwwkramer.in.tum.de/soybean.arff',
                    '0.5',
                    'low'} );
    }
}
```

5 Related Work

We presented a Web Service interface to the inductive database SINDBAD. The main difference from similar approaches combining Web Services and data mining is the use of an inductive database as a basis for data mining. Similar approaches exist on the basis of the WEKA workbench [10]. Weka4WS [11] (see `http://weka4ws.wordpress.com/`) is a framework with the aim of providing distributed data mining in grid environments. To implement the Web Service environment, it uses the Web Service Resource Framework (WSRF). Additionally, it provides a modified Weka Explorer to enable computation on remote or local hosts. While the Weka4WS offers access to a great number of data mining algorithms, it does not seem to aim for an easy integration into external programs. The main focus of the projects seems to be the support of distributed data mining.

A similar approach introduces a Dynamic Data Mining Process system based on SOA [12]. Although the WEKA workbench is not used as a backend, the Web Services are used to manage the internal communication between different data mining processes. Each data mining process is provided by a single Web Service, a complete data mining session is a combination of several Web Services. As the Web Services work independently from each other, it is not obvious how to combine several different data mining operations into one procedure.

Ghanem *et al.* [13] presented a tool for biological text mining on the basis of Web Services. They focus on biological applications and provide a visualization for non-programmers.

The approach most similar to SINDBAD SAILS is the FAEHIM project [14, 15], which provides a similar functionality, but again uses WEKA as a basis for most of its data mining algorithms. At the moment, about 75 data mining algorithms are supported.

A more specialized approach is taken by the Taverna workflow management system (see `http://taverna.sourceforge.net/`), which offers support for a broad range of bioinformatics applications. While Taverna focuses on tasks arising in biological applications, SINDBAD SAILS is intended for general-purpose data mining tasks.

6 Conclusion

We presented a data mining Web Service using the SINDBAD inductive database. To the best of our knowledge, it is the first time a Web Service interface is offered for an inductive database. Although the combination with SINDBAD limits the number of currently available data mining algorithms so far, it enables easy access to some of the most fundamental algorithms. The main advantage of the inductive database approach is the possibility to use and combine the mining results almost arbitrarily.

SINDBAD SAILS will support all features of SINDBAD. In addition to providing a broad range of data mining algorithms publicly on the internet, it is possible to set up a network of inductive databases and distribute the work over several servers exchanging result objects between them. Another attractive feature of the approach is the straightforward use of machine learning and data mining in programming languages without the need for specialized libraries. As such, it also points into an interesting direction: the transition from inductive query languages to inductive programming languages.

References

1. Kramer, S., Aufschild, V., Hapfelmeier, A., Jarasch, A., Kessler, K., Reckow, S., Wicker, J., Richter, L.: Inductive databases in the relational model: The data as the bridge. In Bonchi, F., Boulicaut, J.F., eds.: KDID. Volume 3933 of Lecture Notes in Computer Science., Springer (2005) 124–138
2. Richter, L., Wicker, J., Kessler, K., Kramer, S.: An inductive database and query language in the relational model. In: Proceedings of the 10th International Conference on Extending Database Technology (EDBT 2008), ACM Press (2008) 740–744
3. Agrawal, R., Bollinger, T., Clifton, C.W., Dzeroski, S., Freytag, J.C., Gehrke, J., Hipp, J., Keim, D., Kramer, S., Kriegel, H.P., Liu, B., Mannila, H., Meo, R., Morishita, S., Ng, R., Pei, J., Raghavan, P., Ramakrishnan, R., Spiliopoulou, M., Srivastava, J., Torra, V., Tuzhilin, A.: Data mining: The next generation. Report based on a Dagstuhl perspectives workshop organized by R. Agrawal, J-C. Freytag, and R. Ramakrishnan (2005)
4. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning **5** (1990) 239
5. Date, C.J.: An Introduction to Database Systems. 4th edn. Addison Wesley (1986)
6. Ramon, J., Bruynooghe, M.: A polynomial time computable metric between point sets. Acta Informatica **37** (2001)
7. Ferris, C., Booth, D., Champion, M., Haas, H., Orchard, D., Newcomer, E., McCabe, F.: Web services architecture. W3c note, W3C (2004) http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.
8. Domingos, P.: Structured machine learning: Ten problems for the next ten years. In: Proceedings of Seventeenth International Conference on Inductive Logic Programming, Corvallis, Oregon, Springer (2007)
9. Kohavi, R., Sommerfield, D., Dougherty, J.: Data mining using MLC++: A machine learning library in C++. In: Tools with Artificial Intelligence, IEEE Computer Society Press (1996)
10. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. 2nd edition edn. Morgan Kaufmann, San Francisco (2005)
11. Talia, D., Trunfio, P., Verta, O.: Weka4WS: A WSRF-enabled Weka toolkit for distributed data mining on grids. In: Proc. of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2005). Volume 3721 of LNAI., Porto, Portugal, Springer-Verlag (2005) 309–320 ISBN 3-540-29244-6.
12. Tsai, C.Y., Tsai, M.H.: A dynamic web service based data mining process system. In: CIT '05: Proceedings of the The Fifth International Conference on Computer and Information Technology, Washington, DC, USA, IEEE Computer Society (2005) 1033–1039
13. Ghanemn, M., Chortaras, A., Guo, Y.: Web service programming for biological text mining. In: Proceedings of the ACM SIGIR'04 Workshop on Search and Discovery in Bioinformatics. (2004)

14. Ali, A.S., Rana, O., Taylor, I.: Web services composition for distributed data mining. In: ICPPW '05: Proceedings of the 2005 International Conference on Parallel Processing Workshops, Washington, DC, USA, IEEE Computer Society (2005) 11–18
15. Ali, A., Ludwig, S., Rana, O.: A cognitive trust-based approach for web service discovery and selection. In: Third IEEE European Conference on Web Services (ECOWS 2005). (2005) 12 pp.