Proceedings of the First Annual Workshop on Data Mining Standards, Services, and Platforms

KDD 2003
August 27, 2003
Washington DC

Edited by Robert Grossman
University of Illinois at Chicago
& Open Data Partners

Table of Contents

Affiliations

Robert Allan, London Metropolitan University

Joseph Bugajski, Visa International

Robert Chu, SAS

John Gordon, London Metropolitan University

Robert Grossman, Laboratory for Advanced Computing and National Center for Data Mining, University of Illinois at Chicago and Open Data Partners LLC

Na Helian, London Metropolitan University

Mark Hornick, Oracle

Mehmed Kantardzic, University of Louisville

Anup Kumar, University of Louisville

Christoph Lingenfelder, IBM

Gregor Meyer, IBM

Frank Wang, London Metropolitan University

Part 1.

Data Mining Services and Platforms

Building Standards Based Analytics for Financial Services

Joseph M. Bugajski
Visa International

Risk applications provide critical infrastructure for financial institutions. The core of a risk application is an analytical model. Creating and deploying effective analytical models depends on having consistent, accurate and timely data.

In this paper, we describe a service-oriented approach to working with data, metadata, and derived data. We also describe a service-oriented approach for creating PMML-based analytic models using this data and derived data.

Finally, we describe a framework for measuring the accuracy and consistency of the data, derived data, and analytic models in the framework.

Introduction to Risk Management

Every financial institution ostensibly "bets" that the amount of money returned to them from investments will exceed the principal invested plus the amount of risk associated with the investments. Risk measures the potential for losses sustained following untoward events. Risk models provide a stochastic description of untoward events for calculating expected loss due to untoward events in an investing activity. Risk, *R* (Eq1) is

$$R \equiv A\,\overline{P}(X)$$

Equation 1: Risk

where *A* is principal invested, X is a matrix of random variables representing "untoward events", and P is the associated "risk" model of those untoward events.

The risk value proposition is equally simple (Eq2). Let *V* be total value from "*N*" investments of the i^{th} principal *A*. Let R_i be the associated risk. We require that

$$V \geq \sum_i \left(A_i + R_i\right) = \sum_i A_i \left(1 + \overline{\overline{P}}(X)_i\right)$$

Equation 2: Investment Value

Clearly, investors seek returns that exceed the sum of principal invested and risk.

Financial institutions with global operations require no less an effective risk model than a local bank. Indeed, global financial institutions demand closer scrutiny of the components of their risk and analytical environments. Why? First, large financial institutions invest worldwide. Second, their analytical environments tend to be distributed globally. Because analytical models are core to risk management, and because peer group histories of untoward events drive analytical models, we seek standardization of analytics and data so that we may assure the global financial institution of a coherent risk management

program. A risk management program, like a security program, is only as good as its weakest component. Standards for data and analytical methods strengthen risk models.

Components of a Risk Management Program

Every part of the risk management program deserves consideration of the relevant standards protocols for data and for analytical methods. When a financial firm undertakes an investing activity, they follow a routine process. They

1) Build a business model of the investing activity
2) Enumerate untoward events *X* relative to the investing activity
3) Collect and manage historical data about the investing activity
4) Determine risk factors correlative to such *X*
5) Provision a risk model for the investing activity
6) Measure the efficacy of the risk model against experience
7) Improve the accuracy and consistency of the model

We will briefly discuss each of the processes.

1) Components of a business model for an investing activity

A financial product manager defines a new financial product or service. A business architect, or enterprise architect, translates the description of the product into UML models. The models define operations for which the risk analyst must characterize the risk environment. We will follow a simple model of a credit card product.

Credit card programs, like those offered by Visa International, consist of five operations (fig 1). (i) Operating rules and regulations govern the use of the card product. (ii) Banks may offer special services, like car rental insurance. (iii) The card company (Visa, MasterCard, and American Express) provides a delivery environment to move payment messages. (iv) The payment messages must follow standard formats to communicate bank instructions. (v) Card companies deliver services like currency conversion, stand-in processing, and risk management. Risk control is a key consideration for each of the five elements of the card product program, including controls for manufacturing cards, rules governing liquidity of issuing banks, and merchant monitoring for fraud activity. The delivery environment itself requires risk controls given that a large amount of cash cycles daily through it.

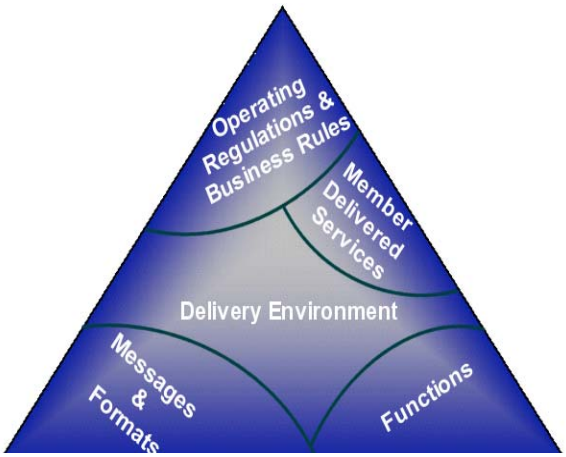


Figure 1: Credit Card Product Program

2) Enumerate untoward events *X* relative to the investing activity

There are a number of ingenious ways for criminals to defraud card issuing banks and card accepting merchants. Each of these "untoward events" becomes part of a risk model:

- Fraudulent use of a card account
- Illiquid banks
- Overuse of charge-backs by consumers or merchants
- Production of counterfeit magnetic stripes
- Misappropriation and use of personal information (identity theft)
- Merchants making a business of stealing from card services
- Abrupt changes in financial condition of cardholders

A risk analyst statistically models the untoward events *X*, study reports of similar events, and calculates the loss expected with each event. The resultant model *P(X)* describes the conditions that best match known events.

An example of fraudulent use of a stolen credit card illustrates the modeling process. A risk analyst will research true cases of fraudulent uses of credit cards. When the fraud perpetrator has a stolen credit card, he or she will often use it for a low value purchase, gasoline. If it works, he or she will use that card to buy a fungible item like jewelry or electronics. The thief will continue to use the card until all available cash is used.

A model of the example requires two sources of data: fraud reports and records of legitimate cardholder activity. In addition, we desire the models to neither falsely indicate misuse nor falsely indicate proper use.

3) Collect and manage historical data about the investing activity

The sources of data about fraud and records of card use must be available to the analyst. Card associations all require reports of fraudulent use of payment cards. This data must be available in a standard form and schema to assure it contains sufficient information to characterize fraud, and relate the case to original data sources from which the analyst will obtain training data. The data will include a process description to formulate controls for activity sequences. Other data about events occurring near the time of the fraud permit statisticians to model the likelihood of the event recurring.

Data storage and retrieval is a critical part of the risk infrastructure. Reports of fraud are of interest to police, banks, merchants, card companies, and acceptance device manufacturers. Service oriented systems, and XML formatting for reports, make these common processes less burdensome.

4) Determine risk factors correlative to such *X*

The risk analyst uses reported fraud and records of valid payment activity to characterize the loss pattern in an analytical model. Although we cannot discuss particular risk factors, the reader may understand the general treatment of the problem by the above example.

Excessive purchases of small amounts of gasoline followed by jewelry may indicate fraud. Hence, an analyst will develop a model of card use by category of merchant. Card

activity in rapid succession also indicates a fraud pattern. Use of the card in strange locale or unusual times is another indicator of fraud that the risk model may incorporate.

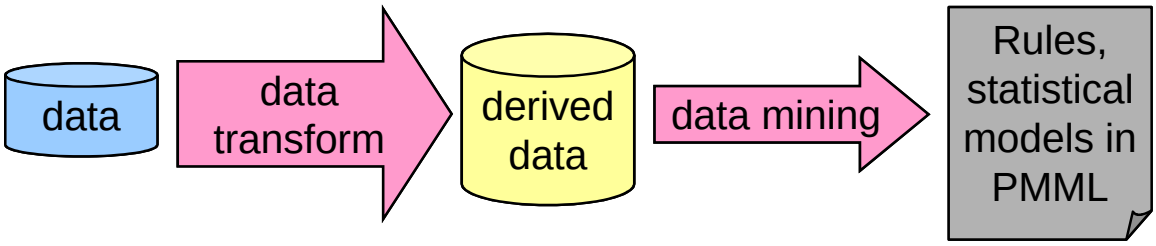


Figure 2: Data and Derived Data in Analytical Models

Each element of model building demands standards for consistency. XML data permits variation in content independently of application structure. Similarly, data transformation services are available for novel and existing tasks without adding new code. Derived data is consistently determined when the transformation service and input data follow standard methods. The system produces resultants of known quality for numerous modeling activities. A service oriented design makes enhancement simple and reliable.

5) Provision a risk model for the investing activity

The correlative factors are the basis of a risk model. PMML provides a standard form for invoking models used to score transactions for fraud. A card company or bank deploys a risk model for reading payment records within moments of use. The model scores each record. Scores exceeding a predetermined threshold trigger an alert message. The alert is broadcast to customer service for verification of card use, to a risk control team for action, and to the payment authorization system to decline approval.

6) Measure the efficacy of the risk model against experience
7) Improve the accuracy and consistency of the model

Model update and maintenance are not always considered during system design. An analytical model ideally anticipates untoward events not explicitly understood from the training data. Artificial neural networks and genetic algorithms bear this property, but regression trees and Bayesian inference engines do not. In addition, standards improve over time as adherents put them to work necessitating system updates.

The service oriented risk management system is a loosely coupled architecture. This means that maintenance engineers may change parts of the system without disturbing other parts. This feature is critical in high volume production modeling environments.

Finally, the risk model is only as good as the data from which it derives. Fraud reports must be available as soon after the event as possible. The data used to construct a measure must be readily accessible for analysis. Taken together with the above factors, provisioning a service oriented, standards based analytical model suite, will deliver consistent, reliable, and coherent performance across the enterprise.

Meta-Model of a Risk Management System

An analytical program for risk modeling that is standards based may be erected in a service-oriented framework. Such a framework admits a distributed solution to the problem of managing risk for complex investment activities. Furthermore, a service-oriented framework with standards based analytics provides a high level of coherence independently of location by permitting access to consistent meta-models, metadata, data, derived data, and analytical methods. Clearly, such a framework demands open standards based communication protocols. We assume XML Metadata Interchange Format (XMI) for metadata communication (OMG).

We present here a high level view of a service-oriented architecture (fig.3) for risk management. An investment *meta-model* derives from a business model and enumerated untoward events. Data collection prescribes *metadata* for correlative measures. The determination of correlative factors, also called "features", requires both *data* and *derived data*. The risk model employs a *service* model, which shows how predictive model mark-up language (PMML) contributes to the overall risk management system. The quality and consistency of the ensemble follows from the framework; i.e., measures of the quality of service are integral to the entire framework.

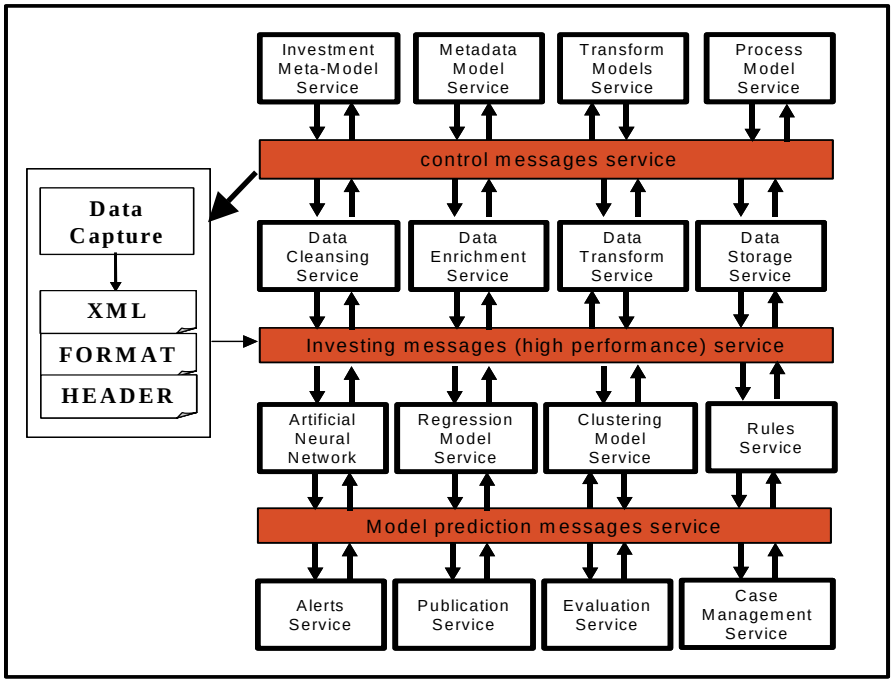


Figure 3: Service Oriented Architecture for Risk Models

The meta-model (fig3) is a service-oriented architecture for risk management. Risk analysts build the set of services shown as the top layer periodically, as a model is built, and later, when it is updated. The information from these services pass to other services subscribed to such messages. Hence, metadata and service meta-models move information seamlessly to services for data cleansing and transformation (the second layer of services from the top). The third layer of services perform prediction scoring,

rules checking, and classification of source data. It also provides a data store for model development. Data formats are standard across the payment system defined by way of the data preparation services. The bottom layer of services issues alerts, communicates with risk analysts, provide case management services and permit the risk analysts to monitor the effectiveness of the models.

A service oriented architecture has discoverable analytical services (PMML), a message platform (HTTP/S, TCP/IP), message oriented middleware (JMS, CORBA, RPC, DCOM, SOAP), service meta-models (OMG's UML specification and model driven architecture), service discovery and description (UDDI, WSDL), metadata exchange (XMI), and a technology platform (J2EE, .Net) on which the project will be delivered.

Final Observations and Summary

Every financial institution uses risk application software to measure, classify, manage, monitor, and avoid risks of losses in their investment portfolios. Banks, insurance companies, mutual funds, equity and bond traders, and departments of companies and governments run risk software. Some of these applications are custom, some commercially available. They run on every type of computer from the lowliest calculator to supercomputers. Existing, mostly proprietary, applications fail to interoperate, do not deliver a coherent risk environment, and cannot be distributed on a global basis. A service-oriented architecture for risk management potentially can solve this problem.

The determination of risk factors and the ability to present an analytical description of risks is key to the development of every financial product. The characterization of risk probabilistically for purposes of making predictions about the outcome of an investing activity sets a minimum for investment returns.

We described a service-oriented approach for creating PMML-based analytic models using payment financial data and derived data. We provided for standards based data and services definitions. The ensemble of such services allowed us to construct a coherent model of a platform for distributed risk management. The system would also contribute to more timely updates of models and fewer losses.

Standards, Services and Platforms for Data Mining: A Quick Overview

Robert Grossman

1. Introduction

Today, most data mining takes places in one of two ways. In the first way, a client server or 3-tier based data mining application accesses and analyzes local data. In the second way, data mining is embedded in another application, either explicitly or implicitly. For example, today data mining is embedded into the databases marketed and sold by IBM, Microsoft and Oracle. Data mining is also commonly embedded into a variety of applications, for example in CRM applications and financial risk applications.

During the past several years, web services have matured to the point that it is now becoming practical to create distributed data mining infrastructures and platforms based upon web services. In this paper, we briefly survey this area.

There is the potential for web services to change in a fundamental way the infrastructure used to analyze data. Consider the following: today, many people find it quicker to locate a preprint by using Google then to search for it on their own local disk. On the other hand, almost all data analysis is done using local data. As bandwidth becomes a commodity, accessing remote data and remote services will become easier, and one day it may be as easy to work with remote data as it is to work with local data.

This paper is a preliminary version of a paper by the same name. Sections 5, 6 and 7 are based in part on [Grossman:03].

2. Background

It is convenient to think of data mining systems that were developed during the past decade as comprising three generations: 1) client-server systems; 2) component and agent based systems; and 3) systems based upon web services.

Today, most data mining takes place using first generation data mining systems. The data is local data and the architecture is either a client server architecture or a 3-tier based architecture. With these systems, a client front end is used to access a server (possibly on the same machine) hosting the data mining application. With a client server model, the server also manages the data; with a 3-tier model, the data is accessed from another host using ODBC, JDBC, or a related protocol. The most common commercial systems of this type include SAS(tm), SPSS(tm), and SPlus(tm). There is also an open source version of SPlus called R.

The next generation of data mining systems that were developed were component based. The components could be local, relying on Microsoft's COM or DCOM platforms for

example, or global, relying on Suns J2EE platform for example. Angoss [Angoss] is an example of the former and Kennsington [Inforsense] is an example of the latter.

More or less at the same time, various experimental agent based data mining systems were developed. The basic assumption in these systems is that the data is distributed and agents are used to either move the data, move the models produced by a local data mining system, or move the results of a local data mining computation. Today, very few agent-based systems are used in practice. Theis is probably because no agent-based infrastructure, over which agent-based data mining systems were built, was ever widely adopted. Examples of agent based distributed data mining systems include JAM [Stoflo:97], Papyrus [Grossman:99], and BODHI [Kargupta:97].

Somewhat later, the next generation of service-based data mining systems began to emerge. These generally are built using W3C's standardization of web services. Examples include DataSpace [Grossman:02a] and data mining systems developed by IBM, Microsoft and SAS that employ the XML for Analysis standard [XMLA].

More general service based infrastructures, such as grids or data grids [Foster:99], are also used for data mining, especially when large computational resources are required. A data grid uses Globus or an equivalent infrastructure to provide a security infrastructure and resource management infrastructure so that distributed computing resources can be used. In addition, Globus provides a high performance data transport mechanism called GridFTP. Recently, the Grid community has begun an effort called the Open Grid Service Architecture or OGSA, which provides a web service-based access to some grid services [OGSA]. The term *knowledge grid* is sometimes used for data mining services deployed using grids or data grid services.

Although grids have been used for some data mining applications, their use has been limited since the critical path for many data mining applications is not the lack of computational resources but rather the time and effort required to deploy data mining into operational systems and the time and effort required to prepare data for data mining. We address these issues in the next section.

3. From Data Mining Systems to Data Mining Middleware

Producing Models. At the core, a data mining system produces one or more statistical or data mining models. Today, these models are generally described using the XML markup language called the Predictive Model Markup Language or PMML [DMG]. These models may be descriptive (e.g., a cluster model or association rules) or predictive (e.g., a regression model or neural network). From this point of view a data mining system takes a data set as input (the learning set) and produces a PMML model as the output. Sometimes these types of applications are called PMML producers.

Producing Scores. Initially, the majority of data mining systems were designed as stand alone applications and were generally difficult to integrate with operational systems. In an operational system, the role of data mining is often simply to take a data record and

produce a score. For this reason, specialized scoring engines began to be developed for this purpose. Today there are several data mining scoring engines that take a PMML model as input and then score one or more records. For example, a scoring engine would take a data record as an input and produce as output the result of applying the model to the data record. Scoring engines have taken a while to mature. The first work in this area began in 1997. Today scoring engines are produced by a variety of vendors including IBM, SPSS, SAS, and Magnify. Sometimes these types of applications are called PMML consumers.

Preparing Data. More recently, there has been an effort, most notably among members of the Data Mining Group, to standardize the data transformations, aggregations, normalizations and other functions required to prepare data for data mining. Although this work is still immature, PMML version 2.0 already includes many common data mining transformations, but perhaps not yet in a format that makes them easy to integrate into data mining systems [DMG]. The typical input for a system for preparing data consists of one or more tables of data records. The output consists of a table containing the data records produced by the transformations. The transformations may be described by a PMML file, for example. Today, data preparation is generally done using a database or a data mining system, such as SAS or R.

Accessing Data. It is probably still the case that most data that is mined and analyzed is made available in ASCII files whose fields are delimited by commas, tabs, vertical bars or some other special character and whose records are delimited by carriage returns. On the other hand, more and more business data is being stored in relational databases, where data mining applications can access the data through ODBC or JDBC. More recently, web service based protocols such as the DataSpace Transfer Protocol or DSTP have provided a simple mechanism for accessing remote and distributed data and direct support for some of the more common operations for accessing data, such as the ability to retrieve metadata, to select rows and columns, and to sample data [Grossman:02a].

We use the term *data mining middleware* to refer to systems or services that are used to produce models, produce scores, prepare data, or access data. Standards have at least two important roles in data mining middleware [Grossman:02b]:

1. Standards are used to specify the inputs, outputs and interfaces to the various data mining middleware services described above. For example, PMML is used to specify the output of service producing models. As another example, the Web Service Description Language (WSDL) [W3C:WS] can be used to describe a data mining web service.
2. Standards are used to specify the APIs to other languages and systems. There are standard data mining APIs for Java and SQL for example. Using the appropriate API, an application can build a classification tree using data in a SQL database.

5. Using XML to Define Inputs, Outputs and Interfaces

The Predictive Model Markup Language (PMML) is being developed by the Data Mining Group, a vendor led consortium that currently includes over a dozen vendors including statistical and data mining software [DMG]. PMML can be used to specify the inputs and outputs of data mining consumers and producers. PMML can also be used to specify the transformations used to prepare data for data mining.

PMML consists of the following components:

1. **Data Dictionary.** The data dictionary defines the fields that are the inputs to models and specifies the type and value range for each field.
2. **Mining Schema.** Each model contains one mining schema that lists the fields used in the model. These fields are a subset of the fields in the Data Dictionary. The mining schema contains information that is specific to a certain model, while the data dictionary contains data definitions that do not vary with the model. For example, the Mining Schema specifies the usage type of an attribute, which may be active (an input of the model), predicted (an output of the model), or supplementary (holding descriptive information and ignored by the model).
3. **Transformation Dictionary.** The Transformation Dictionary defines derived fields. Derived fields may be defined by normalization, which maps continuous or discrete values to numbers; by discretization, which maps continuous values to discrete values; by value mapping, which maps discrete values to discrete values; or by aggregation, which summarizes or collects groups of values, for example by computing averages.
4. **Model Statistics.** The Model Statistics component contains basic univariate statistics about the model, such as the minimum, maximum, mean, standard deviation, median, etc. of numerical attributes.
5. **Model Parameters.** PMML also specifies the actual parameters defining the statistical and data mining models per se. Models in PMML Version 2.1 include regression models, clusters models, trees, neural networks, Bayesian models, association rules, and sequence models.

6. Standards for Data Mining APIs

Developing standards based data mining applications and services is facilitated by defining standard API's to common languages such as Java, SQL, and Microsoft's OLD DB.

The data mining extensions in SQL are part of the SQL Multimedia and Applications Packages Standard or SQL/MM. The particular specification, called SQL/MM Part 6: Data Mining, specifies a SQL interface to data mining packages.

The Java Specification Request 73 (JSR-73), known as *Java Data Mining* (JDM), defines a pure Java (tm) API to support data mining operations. These operations include model building, scoring data using models, as well as the creation, storage, access and

maintenance of data and metadata supporting data mining results [JSR-73]. It also includes selected data transformations and provides a framework so that new mining algorithms can be introduced.

Microsoft's OLE DB for Data Mining (OLE DB for DM) defines a data mining API to Microsoft's OLE DB environment [Microsoft:OLEDB]. OLE DB for DM doesn't introduce any new OLE DB interfaces, but rather uses a SQL-like query language and a specialized data structure called a rowset so that data mining consumers can communicate with data mining producers using OLE DB. In 2002, OLE DB for DM was subsumed by Microsoft's Analysis Services for SQL Server 2000 [Microsoft:XMA]. Microsoft's Analysis Services provide APIs to Microsoft's SQL Server 2000 services which support data transformations, data mining and OLAP operations.

7. Data Mining as a Web Services

The W3C has led the development of standards for web services. Although there are several variants, a web service can be defined using the Web Service Description Language (WSDL), and XML data can be transported using the Simple Object Access Protocol or SOAP. Finally, web services can be discovered using the Universal Description, Discovery and Integration or UDDI service. For details see [W3C:WS].

Work in this area includes:

- A vendor led group that includes Microsoft, SAS, and Hyperion, has developed a web services for data mining standard called XML for Analysis or XMLA [XMLA].
- Several "knowledge grid" applications have been built over grid and data grid services [Cannataro:2002].
- Version 3.0 of the DataSpace Transfer Protocol or DSTP interoperates with both web services and OGSA compliant services [Grossman:02a and 03].

8. Conclusions and Future Directions

Given the growing amount of web accessible data, the declining cost of bandwidth, and the maturing of web services, it will become more and more common to analyze and to mine data with data mining services.

One of the obstacles to the wider adoption of data mining has been the difficulty in preparing data for data mining. The incorporation of data mining as an embedded database application and the emergence of web services to prepare and transform data may facilitate the broader use of data mining.

References

[Angoss] Angoss, KnowledgeStudio, retrieved from www.angoss.com on August 5, 2003.

[Cannataro:2002] Mario Cannataro, Domenico Talia, and Paolo Trunfio, The Knowledge Grid: Towards an Architecture for Knowledge Discovery on the Grid, to appear.

[DMG] Predictive Model Markup Language (PMML), Data Mining Group, retrieved from http://www.dmg.org on August 5, 2003.

[JSR-73] Java Specification Request 73. Retrieved from http://jcp.org/jsr/detail/073.jsp on March 8, 2002.

[Foster:99] I. Foster and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, San Francisco, California, 1999

[Grossman:99] Robert L. Grossman, Stuart Bailey, A. Ramu, Balinder Malhi, Harinath Sivakumar, Andrei Turinsky, Papyrus: A System for Data Mining over Local and Wide Area Clusters and Super-Clusters, Proceedings of SC 1999, 1999.

[Grossman:02a] Robert Grossman, and Marco Mazzucco, DataSpace - A Web Infrastructure for the Exploratory Analysis and Mining of Data, IEEE Computing in Science and Engineering, July/August, 2002, pages 44-51.

[Grossman:02b] Robert Grossman, Mark Hornick, and Gregor Meyer, Data Mining Standards Initiatives, Communications of the ACM, Volume 45-8, 2002, pages 59-61.

[Grossman:03] Robert Grossman, Mark Hornick, and Gregor Meyer, Emerging Standards and Interfaces in Data Mining, Handbook of Data Mining, Nong Ye, editor, Kluwer Academic Publishers.

[Kargupta:97] H. Kargupta, I. Hamzaoglu and B. Stafford, Scalable, Distributed Data Mining Using an Agent Based Architecture, KDD97, pages 211-214.

[Microsoft:OLEDB]. Microsoft OLE DB for Data Mining Specification 1.0 Retrieved from www.microsoft.com/data/oledb/default.htm on March 8, 2002.

[Microsoft:SQL]. Microsoft SQL Server 2000 Analysis Services. Retrieved from www.microsoft.com/SQL/techinfo/bi/analysis.asp on March 8, 2002.

[OGSA] The Globus Project, Towards Globus Toolkit 3.0: Open Grid Services Architecture, retrieved from www.globus.org/ogsa/, on January 10, 2003.

[Stolfo:97] S. Stolfo, A. L. Prodromidis and P. K. Chan, JAM: Java Agents for Meta-Learning over Distributed Databases, KDD97.

[W3C:SW] World Wide Web Consortium (W3C), Semantic Web, retrieved from www.w3c.org/2001/sw on March 8, 2002.

[W3C:WS] World Wide Web Consortium (W3C), Web Services, retrieved from http://www.w3.org/2002/ws/ on August 5, 2003.

[XMLA] XML for Analysis Consortium, XML for Analysis, retrieved from http://www.xmla.org

Grid Application Protocols and Services for Distributed Data Mining

Anup Kumar and Mehmed Kantardzic

1. Introduction

Information technology industry in general, and data mining technology in particular, has lingered too long in an era of over-specialization in which integration is just another specialty. We have made tremendous progress in almost every aspect of computing through these specializations in computer science and engineering. Some components are smaller, others are faster, cheaper, easily connectable, more precise, and have more capacity. But, how do we deal with the complexity of the entire environment, as a system, generated by all that "smaller/faster/cheaper" focus, where complexity is expressed by heterogeneity and large interconnectivity of powerful software, hardware, and data components? Though this question is applicable to different IT disciplines, our concentration is on solutions for a data-mining domain. Data mining technology is a typical example where research has made significant progress in specific algorithms, but there are not enough results on their integration and simplified use in complex, distributed Internet based environments.

There are a few systems developed for distributed data mining. The JAM system developed by Professor Stolfo et. al. [26] uses local learning, and outputs from local learning can be combined to build meta-learning. JAM provides a set of learning programs that execute models over data stored locally and also provides a set of agents for combining the results from multiple sites. The Kensington [9,18] data-mining infrastructure developed by Professor Guo allows access to data from anywhere on the Internet. This remote access to data and mining framework are based on CORBA. The BODHI [23, 24] developed by Professor Kargupta is an agent based distributed knowledge discovery system. It uses local learning schemes that can be combined at a central location to build meta-knowledge. The Papyrus system [17] developed by Professor Grossman is based on a layered infrastructure for high performance and wide area data mining. It can support clustesr of workstations connected by high performance networks. These existing approaches in modeling data mining infrastructure suffer from one or more of the following limitations:

- The existing frameworks only support on demand data mining. But in the Internet domain, on-line and incremental data mining is needed for many applications.

- The data location and format of data has to be known before hand. In the Internet world data sources are added every day. Data mining frameworks must have the flexibility to discover new data sources for a particular domain dynamically.

- The primary focus of the existing frameworks is on just building data mining models such as described in PMML [16]. But building data mining models is only one phase in the iterative data mining process.

- The data mining algorithms are applied independently on different sites and their results are integrated to provide global learning models. Lack of coordination between different sites during the process of local model building affects the quality of results in global learning models [22].

- Most of the existing implementations of data mining infrastructures are tightly coupled and require both ends of communication to use the same distributed object model. This may not work across firewall or proxy servers.

- Most of the data mining models do not discuss the security/privacy structure for data access and execution of distributed agents. Moreover, there is no discussion of cost / charge for data access and distributed agent execution on the data site.

- Some of the existing frameworks provide an integrated web of data such as DataSpace [16]. These frameworks allow queries to be executed on distributed data sets, but the data types in these frameworks are very limited. The goal of distributed data mining must include a web of distributed algorithms and other resources in addition to a web of data.

- In all the existing frameworks, a fixed number of distributed data mining algorithms is implemented. This rigid scheme of implementing proprietary data mining algorithms in a framework limits the integration of new and better algorithms developed by a third party that are available on the Internet. The data-mining framework must be flexible to allow incremental addition to the algorithm knowledge base.

In this paper we are concentrating on the complexity problem of the data-mining infrastructure, and we are proposing a layered approach to overcome this complexity using Grid technology. By hiding the complexity of both, data and software components in the Grid based architecture; we will obtain more user oriented data mining support system [16]. The user will have a powerful, intelligent infrastructure to specify data mining problems at a more abstract level, without knowing all the specific characteristics of the required components for data mining. The proposed architecture will allow users to concentrate on what they want to accomplish rather than figuring out how to solve all the technical details in tuning computing systems for executing data mining models. The proposed approach provides a reliable infrastructure that can accommodate the rapid growth of computing and data resources on the Internet, and hide system complexity from the users. In grid-based infrastructure exchange and integration of data and tasks (tools, libraries, device drivers, middleware, etc.) can be implemented through open standards defining the identification of components, their communication protocol and negotiation protocol among them. The Grid based distributed data mining approach is an attempt to build an infrastructure that will have consistent user-friendly interfaces and

control while allowing integration of a distributed, heterogeneous environment and complex interconnectivities.

This paper explains basic components and characteristics of a distributed data mining process in Section 2, and fundamental principles of Grid architecture and its relation to proposed model are discussed in Section 3. Application layer of protocols and services for distributed data mining based on the Grid architecture is specified in Section 4.

2. Hierarchical Structure for Distributed Data Mining Process

The benefit of understanding large, complex, and information-rich data sets is common to all fields of businesses, science, and engineering. Globally, data mining is defined as a process of discovering various models, summaries, derived nontrivial information, and patterns from a given collection of data [20]. The popularity of the Internet and the web makes it imperative that the data-mining framework is extended to include distributed and time dependent information and tools. Moreover, the assumptions and approaches used for static and centralized data mining process are not any more valid for distributed environments. Distributed data mining refers to the mining of distributed data sets using distributed computational resources. In many situations, local data sets cannot be transferred and centralized into a data warehouse because of [22]:

- Security of local data,
- Autonomy of local data and their different frameworks, or
- Size of local data sets.

Standardization on one hand, and the use of new technologies for an infrastructure improvement on the other side, will make large, complex data mining projects less costly, more reliable, more repeatable, more manageable, more efficient, and very importantly will also provide higher quality in data mining results. Our approach in standardization of distributed data mining methodology is based on a model of a data mining process having four levels of abstraction [20, 21]. The model is organized in a hierarchical structure: a) phases of data mining, b) generic tasks, c) specialized tasks, and d) process instances. For example, in the *exploration phase* one of the generic tasks is to “build the model”, while its specialized task is “build classification model”, and corresponding process instances are: “neural network”, “decision rules”, or ”logistic regression”. The other example of task hierarchy is *data preparation phase*, where “data cleaning” is a generic task, a possible specialized task is “elimination of missing values”, and corresponding process instances are: “mean value algorithm” or “clustering”. These examples of hierarchy are given in Figure 1.

While higher level of abstract concepts in the hierarchy explain *what to do*, lower level components provide details of data processing and answer the question *how to do*. This paper proposes bottom-up approach in building Grid services infrastructure for data mining. The implementation will start with process instances as components of core application services, and then integrate them into more abstract, and at the same time more complex and intelligent data mining specialization tasks.

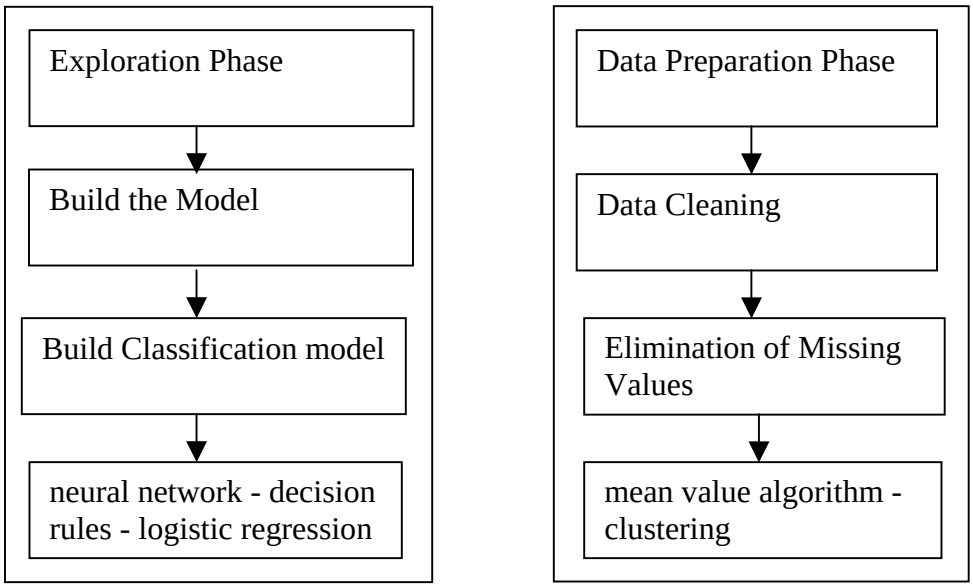


Figure 1. Examples of paths in hierarchical structure of a data mining process

The main goal of the proposed approach is to develop a core infrastructure for better and easier implementation of standard data mining tasks in a distributed environment. Every data mining task specifies dynamically cross-organizational sharing of resources (data, software tools, and computations) defining Virtual Organization (VO) in a distributed Internet-based environment. Sharing of resources is performed in a controlled fashion, through collaboration of participants, taking into account all limitations on requirements of how a resource may be used [19]. Data mining task based virtual organizations (DMVO) differ in many respects: the number and type of data sources, the types of mining activities, the duration and scale of the interaction, and the amount and way the resources are being shared. Traditional approaches to data mining and traditional software tools usually do not support this dynamic, scalable environment, while layered grid architecture represents the right framework for defining data mining based virtual organizations.

3. Grid Computing and Distributed Data Mining Requirements

3.1 Grid Technology: Characteristics, Architectures, and Implementation

Grid architecture provides co-coordinated resource sharing and problem solving in distributed heterogeneous environments. It supports virtual organization users and resources to negotiate, establish and manage relationships through the use of standard protocols and services [13]. The Grid is not an alternative to the Internet in a case of distributed data mining applications; it is rather a set of additional protocols and services that are built on the Internet to support the creation and use of resources in the heterogeneous environment. It is a layered architecture as shown in Figure 2. Connectivity and fabric layers provide low-level services to domain specific application and collective services.

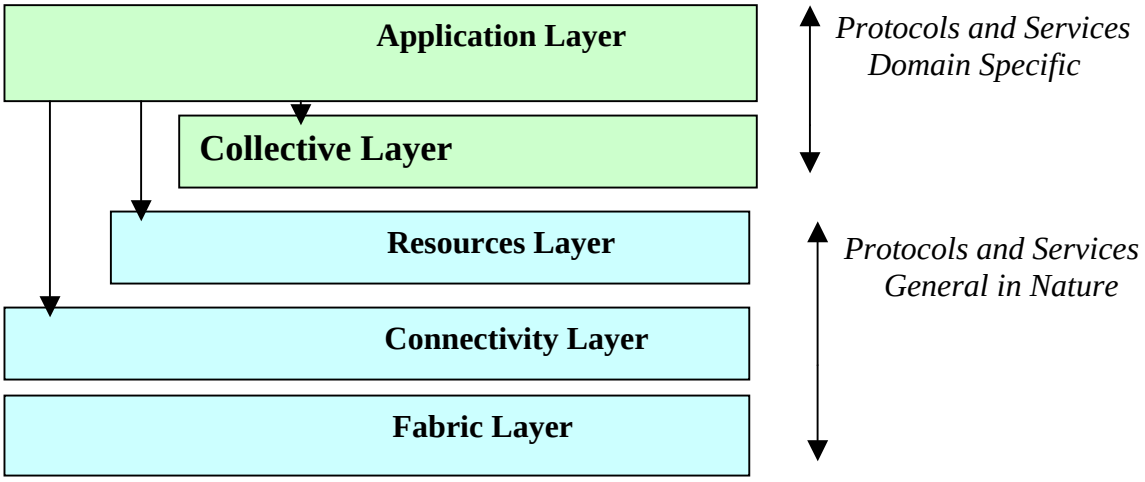


Figure 2. The Layered Grid Architecture

There has been a lot of work done in developing grid standards and Globus toolkit [4, 5, 6, 7, 8, 11, 12, 15, 19, 28, 30]. Due to recent advances in web services , the traditional grid framework has adopted the web service description language (WSDL) for describing grid services in Open Grid Service Architecture (OGSA) [3]. The alpha release of Globus 3.0 toolkit [10, 28] provides open source implementation of open grid service infrastructure (OGSI) [4]. The OGSI provides grid service features such as invocation, lifecycle management, a service data interface and a security interface that guarantees a fundamental level of interoperability between grid services [4].

The current grid protocols and services do not support high-level services needed for developing a distributed data mining framework. Distributed data mining is characterized by the following resources:

a) General purpose resources:
   - Computational resources,
   - Storage resources,
   - Network resources,

b) Problem specific resources:
   - Code repositories,
   - Data repositories,
   - Catalogs.

The key issues, to be addressed in a grid services based distributed data mining framework, are:

- How to register data in data repositories with description of data?
- How to register algorithms in code repositories with their input and out specification along with their security and execution time constraints?

- How to discover data sets and algorithms based on user requirements efficiently?
- How to integrate data available in different formats?
- How to represent the relationship between different algorithms that can cooperate to perform the data-mining task?
- How, When, and Where to schedule an execution of an algorithm for data mining based on the above relationships?
- How to enforce various constraints such as security, disk quota, and execution time on data access and algorithm execution?
- How to monitor the progress of data mining operation?
- How to implement application layer services using Grid based protocols?

Grid architecture for a data mining based virtual organization should support management for all six classes of resources enabling secure: negotiation, monitoring, control & coordination, and accounting & payment. This management is performed primarily through the resources layer. While fabric, connectivity and resources layers must be general in nature and widely deployed, domain specific requirements are specified on collective and application layers. Collective layer spans the spectrum from general purpose to domain specific. The main functions of a collective layer can be implemented as:

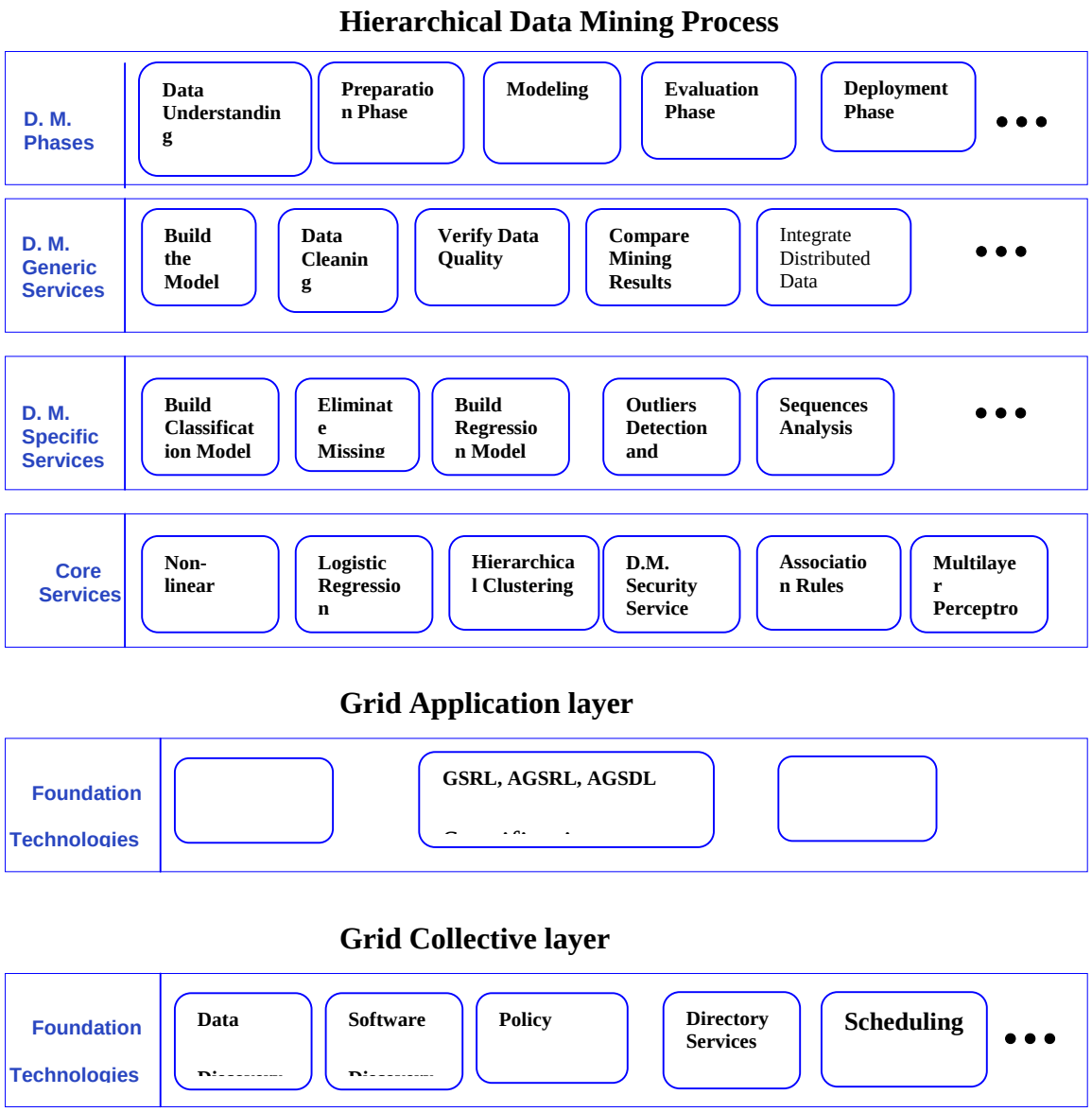
- Persistent services with associated protocols, or
- Software development kits (SDK) with associated application program interfaces (API), designed to be linked with other applications.

In order to build a grid-based data mining framework, we propose the implementation of the following collective level services:

- *Directory services* – they allow VO users to register new data mining resources such as datasets and algorithms.
- *Data discovery services* – they discover and select distributed data sets based on high level specification of data mining problem.
- *Software discovery services* – they discover and select the best software implementation and execution platform based on characteristics of a data mining problem.
- *Policy enforcement services* – these services enforce community policies governing resources access and use, and they include security and privacy aspects.
- *Scheduling services* – they allow users in VO to establish relationships between different types of resources for a specific data-mining task.
- *Monitoring services* – they support monitoring and diagnostics of resources for failure, overload, etc.

3.2 Grid Application Services for Distributed Data Mining

This section provides an integration framework between data mining process layers and the grid application layer. Various components of the proposed data mining application layer are shown in Figure 3. The six collective services will be developed for the data mining framework. There will be two application level activities taking place in this model. The Grid Services User Interface (GSUI) provides a link between various data mining processes shown in the top four layers. User can drill down to any level and can request for data sets and possible algorithms for data mining available over the Internet. Based on data types selected and the algorithms specified for performing a data mining task, a grid service relationship specification will be generated for execution. This relationship will specify the sequence of data mining algorithm execution, resource restrictions, allowed privileges for execution and data access. Scheduling, policy enforcement and monitoring services will use these relationships.



Other Grid Layers: ***Resources Layer, Connectivity Layer, and Fabric Layer***

Figure 3: Components of Grid Services

4. Rationale for Enhanced Grid Services for Data Mining

The proposed approach requires the following protocols and services for a data mining framework:

4.1 Advanced Grid Service Registration Language (AGSRL)

In data mining applications the data providers and algorithm providers will register their data and algorithm services with UDDI registries. In the current Web Service Description Language (WSDL) standard adopted by OGSA [3], the service provider can specify the methods implemented in the grid service with its input and output parameters. In data mining, many times it is desirable to move algorithms from their current location to data sites for execution due to large data set transfer time. Moreover if large data sets are to be processed at different locations it will be more efficient to transfer algorithms to data sites. The current WSDL is not suitable for describing many characteristics that are needed for data mining purposes. These include:

- How to specify the requirements for a given grid service if it can be downloaded to other site for computation?
- What are the computation requirements if the program / grid service can be downloaded to other site?
- How data provider will specify the structure (format and semantics) of data set available for data mining?
- How data providers can specify detailed restrictions of execution environment if algorithm gets downloaded on their site?

Moreover, the existing WSDL does not allow the specification of various security accesses, execution constraints on that node. For data set registration there should be a way where a data set provider can describe their data sets using standard XML vocabularies. There are many domain specific XML vocabularies available in oasis-open forum. The schema of these vocabularies can be used to describe a data set provided by a data provider.

The current WSDL should be extended to develop AGSRL that will be integrated with the OGSI framework so that the above features can be included in the registration process for data service providers. In addition, the data set provider and algorithm providers' can also specify if code can be executed at the (data) site or data can be transferred at (code) site. The AGSRL will be used by directory services for registration of algorithms and data sets.

4.2 Advanced Grid Service Discovery Language (AGSDL)

In web services application development, a service discovery phase is most time consuming. It was observed [28] that performance of an application could degrade by an order of magnitude due to dynamic discovery of a service. Current discovery protocols are suited for service identification only. There is no way to perform search-using UDDI for data sets. In addition, data required for mining application may be available at

multiple sites; in this case the data discovery phase must be as efficient as possible. With the existing UDDI discovery model, the search may be performed sequentially to identify various data items that may not be acceptable for data mining applications. In this research we expect two important contributions will be made to the existing UDDI discovery structure:

- UDDI API will be extended to discover data sets based on the standard vocabularies in a particular data-mining domain.
- Efficient XML based AGSDL for specifying parallel search operations and integration will be developed. Moreover, this discovery language will be integrated in OGSI framework.

The AGSDL will be used by software and data discovery services.

4.3 Grid Service Relationship Language (GSRL)

The detailed grid service information will be described in AGSRL (extended-WSDL) document. The GSUI will perform the discovery using AGSDL and based on the user requirements and the description in AGSRL, it will generate a complete sequence of execution of algorithms and data sets. This will include which algorithms need to be executed, where, and in what order and what are the connections. The existing grid standards used in OGSA do not have any provision for specifying this relationship.

A GSRL will be developed to specify relationships between resources used for a particular data-mining task. These relationships could be generated based on user specified data sets and algorithms, and they will be used by scheduling, policy enforcement and monitoring services.

4.4 Adaptive Grid Service Invocation Mechanism (AGSIM)

In the existing method invocation framework, a method with certain signature can be invoked with that type of dataset only. If one of the parameters is not of the same type, an exception will be generated. In data mining applications, there may be cases that data is semantically similar but syntactically dissimilar. If the data required by a method is integer but that data set has a real value, there should be automated adaptation mechanism that will convert the data to the correct type and invoke the method. In this research a mechanism to implement this automatic format conversion will be developed. Primarily scheduling services will use this.

5. Conclusions

This paper presents an approach which provides an application-oriented Grid based distributed data mining framework for development of distributed applications that are transparent, high performance, and incrementally expandable. The proposed framework:

- Hides the complexity of data mining process
- User-Oriented specification of data mining process
- Allows code execution at data-set location
- Support integration of heterogeneous technologies, data and algorithms.
- Improves the flexibility and performance of a data mining system.
- Improves resource utilization and adapts to demand fluctuations.
- Grid provides reliable platform for performing distributed data mining.

The Grid framework allows users the ability to discover new algorithms and data sets for their data-mining task dynamically defining new virtual organization. Grid application services for data mining allow building of a scalable data mining framework. In addition the framework allows access to applications through firewalls using a standard protocol.

One of the major benefits of the project comes from the fact that it is developed based on open standards that allows multiple universities, companies and other organizations to participate in extending all layers of the Grid architecture.

References:

[1] ACM SIGKDD, *Workshop on Distributed Data Mining*, http://www.eecs.wsu.edu/~hillol/DKD/dpks2000.html, 2000.

[2] ADELFI: A model for Deployment of High Performance Solutions on Internet/Intranets, Esprit Framework-4 Project, http://ruby.doc.ic.ac.uk/adelfi/, 2000.

[3] Foster I., Kesselman C., Nick J., Tueck S. The Physiology of The Grid, *http://www.globus.org/research/papers/osga.pdf, 2002.*.

[4] Tuecke S, Czajkowski K, Foster I, etal.;Grid Service Specification. Open Grid Service Infrastructure WG, Global Grid Forum, Draft 2, 7/17/2002..

[5] Foster I, The Grid: A New Infrastructure for 21st Century Science. *Physics Today*, 55(2):42-47, 2002.

[6] Chervenak A. , Deelman E. , Foster I, Guy L, Hoschek W, etal. Giggle: A Framework for Constructing Sclable Replica Location Services. *Proceedings of Supercomputing 2002 (SC2002)*,November 2002.

[7] Allcock B, Bester J, Bresnahan J, Chervenak A. L. , Foster I, Kesselman C, etal Data Management and Transfer in High Performance Computational Grid Environments**,** *Parallel Computing Journal*, Vol. 28 (5), May 2002, pp. 749-771.

[8] Ranganathan K and Foster I, Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications**.** *Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, Edinburgh, Scotland, July 2002.

[9] Chattratichat J., Darlington J., Guo Y., Hedvall S., Köhler M., Syed J., An Architecture for Distributed Enterprise Data Mining, *Proceedings of the HPCN Conference*, Amsterdam, 1999.

[10] Sandholm T., Seed R., Gawor J., Globus Toolkit 3 Core, http://www.globus.org/toolkit/ .

[11] Schopf S. H. , Vazhkudai S, Predicting Sporadic Grid Data Transfers. *11th IEEE International Symposium on High-Performance Distributed Computing (HPDC-11)*, IEEE Press, Edinburgh, Scotland, July 2002.

[12] Frey J, Tannenbaum T, Foster I, Livny M, Tuecke S. Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Cluster Computing*, 5(3):237-246, 2002.

[13] Foster I., Kesselman C., Tuecke S., The Anatomy of Grid: Enabling Scalable Virtual Organization, int. pub., Mathematics and Computer Science Division, Argonne National Lab, Argonne, IL, 2001.

[14] Forman G., Zhang B., Distributed Data Clustering can be Efficient and Exact, *SIGKDD Explorations*, Vol. 2, Issue 2, 2000, pp. 34-38.

[15] Angulo D, Foster I, Liu C, and Yang L. Design and Evaluation of a Resource Selection Framework for Grid Applications. *Proceedings of IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, Edinburgh, Scotland, July 2002.

[16] Grossman R., Data Space Project, University of Illinois at Chicago, http://www.dataspaceweb.net/dataspace, January 2000.

[17] Grossman R., S. Bailey, A. Ramu, B. Malhi and A. Turinsky, The Preliminary Design of Papyrus: A System for High Performance, Distributed Data Mining over Clusters, in "*Advances in Distributed and Parallel Knowledge Discovery*", H. Kargupta and P. Chan, editors, AAAI Press/The MIT Press, Menlo Park, California, 2000, pages 259-275.

[18] Guo Y., Sutiwaraphun J., Integrating Knowledge in Distributed Data Mining, *PCW Conference*, 1998.

[19] Berman F, Chien A, Cooper K, Dongarra J, etal. The GrADS Project: Software Support for High-Level Grid Application Development. *International Journal of High-Performance Computing Applications*, 15(4), 2002.

[20] Kantardzic, M., Data Mining: Methods, Tools and Techniques, IEEE Press and John Wiley, 2002, Pages 380.

[21] Kantardzic M., Kumar A., Toward Autonomic Distributed Data Mining With Intelligent Web Services, accepted for The 2003 International Conference on Information and Knowledge Engineering, IKE'03, June 23-26, 2003, Las Vegas.

[22] Kantardzic M., Soliman M., An Approach for Mining Frequent Sequences in Distributed Environment, *Proceedings of the 2002 International Conference on Information and Knowledge Engineering*, Las Vegas, Nevada, June 2002.

[23] Kargupta H., "Distributed Knowledge Discovery from Heterogeneous Sites", DIADIC Laboratory, University of Maryland at Baltimore County, http://www.cs.umbc.edu/~hillol.

[24] Kargupta H., Park B., Johnson E., Hershberger D., Huang W., Ayyagari R., Ghosh S., Distributed Knowledge Discovery from Heterogeneous Sites, *Project Proposal*, http://www.cs.umbc.edu/~hillol/DKD/ddm_research.html.

[25] McKarthy J., Phenomenal Data Mining: From Data to Phenomena, SIGKDD *Explorations*, Vol. 1, No. 2, 2000, pp. 24-29.

[26] Prodromidis A. L., Chan P. K., Stolfo S. J., Meta-Learning in Distributed Data Mining Systems: Issues and Approaches, In *"Advances in Distributed and Parallel Knowledge Discovery "*, Kargupta and Chan (eds.), MIT/AAAI Press, 1999.

[27] Steve J. Gardner, A Wireless Printing Application Using Web Service Technology, *M.S. Thesis*, May 2002, (Kumar).

[28] Status and Plan for Globus Toolkit 3.0, http://www.globus.org/toolkit/gt3-factsheet.html.

[29] Wirth R., Borth M., Hipp J., When Distribution is Part of Semantics: A New Problem Class for Distributed Knowledge Discovery, http://www.cs.umbc.edu/~hillol/pkdd20, *PKDD*, 2001.

[30] Foster I, Kesselman C, Nick J, Tuecke S. Grid Services for Distributed System Integration. *Computer*, 35(6), 2002.

Implementing Relational Grid Monitoring Architecture (R-GMA) provided by European Data Grid (EDG) to Prototype a Knowledge Discovery Infrastructure

Frank Wang, John Gordon, Na Helian and Robert Allan

Abstract. The Grid has developed standards based infrastructures and services and platforms relevant to data mining. For example, the European Data Grid (EDG) provides a Relational Grid Monitoring Architecture (R-GMA) for distributed resources that expose a relational model with SQL support to provide static as well as dynamic information about Grid resources and for use within application monitoring. The relational model R-GMA is very flexible and allows complex queries that make use of information in multiple objects. R-GMA makes information from Producers available to Consumers as relations (tables). This paper describes the implementation of a ScanOnce algorithm in SQL for quick association rule mining and the development of a data mining infrastructure JetGrid. The architecture of JetGrid is designed to be compatible with lower-level grid mechanisms since it is to operate on top of R-GMA. JetGrid for quick knowledge discovery was preliminarily prototyped and it is extensible, using an object-oriented design that was coded in C++. Mining agents will be staged to one or more computing elements on the EDG. The agent also performs acquisition of data. Intensive tests are being carried out using GridFTP and other Globus transport mechanism for shipping raw data D across the grid to one node for processing (MD: Move Data Model) and processing the data locally until a result R is obtained and ship the result to one node for further processing (MR: Move Results Model), which provides some indication of the performance of the JetGrid data mining infrastructure on top of EDG.

Keywords: Grid Computing, Data Mining, SQL, European Data Grid (EDG), Relational Grid Monitoring Architecture (R-GMA), e-Science

1. Introduction

The idea of computational and data grids dates back to the first half of the 90's. The vision behind them is often explained using the electric power grid metaphor. The electric power grid delivers electric power in a pervasive and standardized way. One can use any device that requires standard voltage and has a standard plug if he/she is able to connect it to the electric power grid through a standard socket[Foster, 1997] [Foster, 1999]. Currently there are millions of computing and storage systems all over the planet connected through the Internet. What we need is an infrastructure and standard interfaces capable of providing transparent access to all this computing power and storage space in a uniform way. The user doesn't have to know which resources they are using or where

they are. They just get computing power and storage space from the Grid through a standard interface.

The European Data Grid (EDG) is a project funded by the European Union that aims to enable access to geographically distributed computing power and storage facilities belonging to different institutions[http://eu-datagrid.web.cern.ch/eu-datagrid/]. This will provide the necessary resources to process huge amounts of data coming from scientific experiments in three different disciplines: High Energy Physics, Biology and Earth Observation. Researchers need to access all of the resources in a uniform, transparent and easy way and many challenges have to be solved to achieve this goal. Different institutions may use different computing and storage systems and will also have local security rules. Currently, the coordinated use of, say, 50 different computing systems (based on different standards) implies logging into each single system and knowing how to use it. Few people therefore have the necessary expertise to exploit this huge computing power. Additionally, to use such resources effectively, the user needs effective and dependable information systems that allow automatic resource discovery and allocation. It is EDG's goal to provide these solutions.

The Grid is a natural platform for deploying a high performance data mining service[Hinke, 2000][Chervenak, 2001][Orlando, 2002][Cannataro and Talia, 2003]. The Grid has also developed standards based infrastructures and services relevant to data mining. These new standards and standards based services and platforms have the potential for changing the way the data mining is used. In this paper, an integration of a new ScanOnce algorithm for quick association rule mining in SQL and the EDG has been attempted, which enriches the National e-Science Programme launched in 2001[http://www.escience.clrc.ac.uk/web]. A grid infrastructure JetGrid for quick knowledge discovery was prototyped. The architecture of JetGrid is designed to be compatible with lower-level grid mechanisms since it is to operate on top of Relational Grid Monitoring Architecture (R-GMA) provided by European Data Grid (EDG). The JetGrid data mining infrastructure is extensible, using an object-oriented design that was coded in C++.

2. ScanOnce Algorithm

The implementation of a ScanOnce algorithm for quick association rule mining on top of the EDG will be done in this work. In Figure 1, N denotes the number of transactions in the database and T the transaction being currently scanned. The data structure A is a set of entries of the form (IS, f), where IS is an itemset enumerated from the current transaction and f is an integer representing its frequency. This is a purely sequential (rather than recursive) count procedure. To count a certain transaction (represented by Item-IDs), we merely start at the first Item-ID and then sequentially traverse the remaining transaction itemset item by item. Initially, A is empty. The contribution from each transaction is comprehensively taken into account by growing a prefix tree for each transaction and enumerating all subsets of the transaction itemset. Figure 2 shows the enumerating procedure of all subsets of an exemplified transaction “abde” by growing a prefix tree. In order to find the frequent itemsets, we have to count the transactions they are contained in. Our implementation is based on the idea to organize the counters for the

itemsets in a special kind of prefix tree, which not only allows us to store them efficiently (using little memory), but also supports processing the transactions as well as generating the rules. A node in the tree represents an itemset consisting of Item-IDs in that node and all its ancestors, as underlined in the figure. The itemsets enclosed in a dashed rectangle share the same ancestor. This full prefix tree is created level by level. Whenever a new itemset IS arrives, we first lookup A, to see whether an entry for IS already exists or not. If the lookup succeeds, we update the entry by incrementing its frequency f by one. Otherwise, we create a new entry of the form (IS, f). For an entry (IS, f), f represents the exact frequency count of IS ever since this entry was inserted into A. In summary, we process the input data stream transaction by transaction. There is no need at all to store and re-scan the previously-scanned transactions, which will be discarded after a single pass. The amount of main memory available can be devoted to itemset counter arrays. The preliminary results indicate that this ScanOnce algorithm is about an order of magnitude faster than the well-used Apriori algorithm in large centralized databases and this gap grows wider when the volume of transactions further grows up[Wang, 2002][[Wang, 2003]. In summary, we process the input data stream transaction by transaction in each local site. This is 100% sequential counting procedure and therefore there is no need at all to store and re-scan the previously-scanned transactions, which will be discarded after a single pass. In [Manku et al, 2002], they try to fill available main memory with as many transactions as possible, and then process recursively such a batch of transactions together. This is where our algorithm differs from that one. The amount of main memory available can be devoted to itemset counters.

```
1. A ← Ø  //A: The set of all counters
2. T ← next transaction //T: Transaction (item-IDs)
3. Grow subset tree for T and enumerate
all subsets of the current transaction T (Fig.2)
4. IS ← each subset
5. if (IS, f) exists in A do  //f: frequency
6.      f ← f+1
7. else do
8.      insert (IS, 1) to A
9. endif
10. Goto 2
11.
12. scan A and prune infrequent itemsets
13. if f ≥ min_sup x N //min_sup: minimum support, N: Number of
transactions
14.  output (IS, f)
15. endif
16.
17. Generate rules from frequent itemsets IS satisfying minimum
confidence c specified by the user
```

Fig. 1 Pseudo-code for the ScanOnce algorithm.

In principle, the ScanOnce algorithm should work well for distributed, mobile and streaming data mining including the Grid platform in terms of scanning the data sets sequentially rather than repeated like in traditional manners. In each site, attributing to the integrity of the data structure designed in this algorithm, the locally enumerated itemset counters can be stored in a pair of local relational tables. The power of generating ad hoc queries in SQL ensures fast access to any desired counter. The local absolute support count for each enumerated itemset is first found. The global absolute support count for each itemset can then be determined by summing up through either GridFTP or Globus transport mechanism, for each enumerated itemset, the local support of that itemset in all the distributed sites.

3. EDG Data Mining Standards, Services and Platforms

For the scientist in an e-Science scenario, the vision that is now becoming reality is as follows:

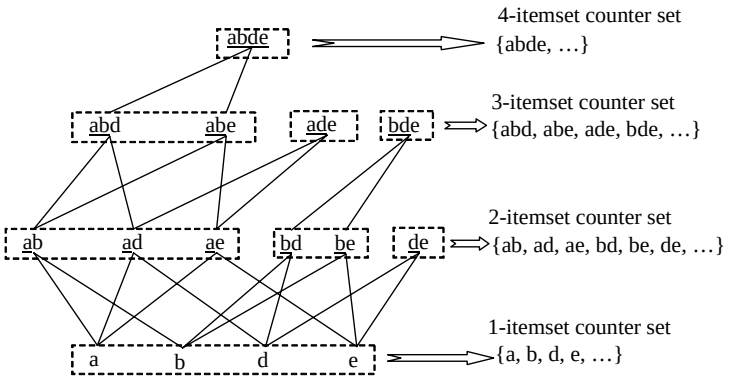


Fig.2 Enumeration of all subsets of an exemplified transaction “abde” by growing a prefix tree.

1. The user submits his request through a Graphic User Interface (GUI) just specifying high level requirements (the kind of application he wants to use, the operating system,...) and eventually providing input data;
2. The Grid finds and allocates suitable resources (computing systems, storage facilities, ...) to satisfy the user's request;
3. The Grid monitors request processing;
4. The Grid notifies the user when the results are available and eventually presents them.

On JetGrid, the user must specify what is to be mined, how it is to be mined and where it is to be mined. Under our approach, the user specifies what is to be mined by storing the names and grid locations of a set of data in a database associated with the EDG miner. These are communicated to a miner agent over the Grid. The user specifies how the data is to be mined by specifying a mining plan that lists the sequence of mining operations that are to be applied to the data and any parameters required. The user specifies where the mining is to take place by specifying the EDG computing elements (CEs) on which

the mining agent is to be staged. With these requirements specified, the user will then invoke the miner, which will send mining agents to the designated EDG computing elements. On these computing elements, each agent will acquire the data to be mined, mine it and send the results back to the user.

4. JetGrid Data Mining Architecture

We are currently implementing the ScanOnce algorithm in SQL to deploy it on top of the EDG by using useful Globus middleware services. The starting point for this development is the JetGrid data mining infrastructure that was preliminarily developed. The architecture of JetGrid is designed to be compatible with lower-level grid mechanisms. The JetGrid architecture is described in Fig.3. The JetGrid data mining system is extensible, using an object-oriented design that was coded in C++. The EDG provides a monitoring and information management service for distributed resources that expose a relational model with SQL support to provide static as well as dynamic information about Grid resources and for use within application monitoring. R-GMA is a relational implementation of the Grid Monitoring Architecture (GMA). The relational model R-GMA is very flexible and allows complex queries that make use of information in multiple objects. R-GMA makes information from Producers available to Consumers as relations (tables). The R-GMA implementation uses HTTP Servlet technology. Communication with the servlets is achieved via an API. This API has been implemented in Java and C++ but C and other languages will be provided soon. The response from a servlet is in the form of an XML document that corresponds to an XML schema definition. It is highly dynamic. For our infrastructure on top of R-GMA, we should consider a virtual organization (VO)'s information to be organized logically as one huge relational database - but the implementation is based on a number of loosely coupled components. The huge database is partitioned, with the description of the partitioning held in the Registry server [http://eu-datagrid.web.cern.ch/eu-datagrid]. To begin the mining operation, initially a mining agent will be staged to an EDG computing element. It is envisioned that these agent may acquire mining operations from multiple sites on the EDG. Some will be acquired from public repository sites that contain a standard set of mining operations. It is hoped that once the mining system is fully operational, mining users will contribute new operations to this mining repository [Hinke et al, 2000]. The agent also performs acquisition of data. Such data can be acquired from EDG-based repositories as well as various data repositories that provide FTP access to their data holdings. By using data delivery, the storage requirements for the target mining site are minimized.

It was mentioned in Section 2 that applying the ScanOnce algorithm implemented in SQL over the EDG to mine global association rules does not need to ship all of local data to one site thereby not causing excessive network communication costs. In this algorithm the contribution from each transaction is comprehensively taken into account by growing a prefix tree for each local transaction and enumerating all subsets of the transaction itemset. There is no need at all to store and re-scan the previously scanned transactions, which will be discarded after a single pass.

5. Preliminary Experiments

We performed a simple experiment on a synthetic database. These synthetic data sets will be generated using the procedure described in [Agrawal, 1993]. In this data set, the

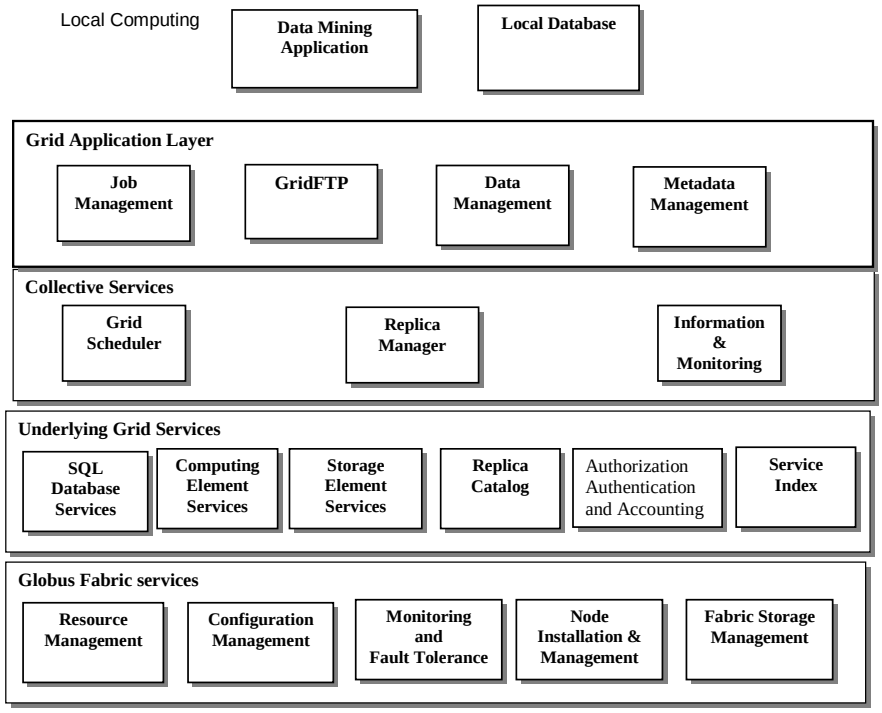


Fig.3 The prototype of JetGrid infrastructure.

average transaction size and average maximal potentially frequent itemset size will be set. Items are drawn from a universe of I = 10K unique items. Item_IDs are 4-byte integers. The number of transactions in the dataset ranges from 200,000 to 5 millions, which occupies up to 200 MB space. For this experiment, the data mining system used delivery to acquired data, similar to [Hinke, et al, 2000]. This data was acquired from a remote host using both FTP and Globus transport mechanisms. As a reference point, the EDG miner was also used to mine the same data that was stored locally. The Linux csh time command was used to time the commands, with its wall-clock time reported in the following table. These are preliminary results, since they represent only one run for each experiment. However, these results provide some indication of the performance of the JetGrid data mining infrastructure.

Source of Data	Time to Acquire and Mine Data
Using Globus Transport Mechanism from Remote Host	25 minutes 13 seconds
Using GridFTP from Remote Host	24 minutes 49 seconds
Local Host	15 minutes 21 seconds

We have reported our experimental results on a synthetic database. We are currently running our query over two other different datasets. The first dataset is a collection of

Web pages crawled by WebBase, a web crawler developed at Stanford University[Hirai, 2000]. Words in each document are identified. Common stop-words[Salton, 1988] are removed. Common stop-words[Salton, 1988] are removed. The resulting input file is 54 MB. The second dataset is the well-known Reuters newswire dataset, containing 806,000 news articles[http://www.reuters.co.uk/]. The input file resulting from this dataset after removing stop-words is roughly 210 MB.

6. Conclusion and Work-on-Progress

This paper represents a snap-shot into a project that is ongoing, presenting a scenario of grid-based mining, an architecture for a grid-based miner, and some preliminary experimental results. The implementation of a ScanOnce algorithm for quick association rule mining and the development of a data mining infrastructure (JetGrid) that is to operate on top of Relational Grid Monitoring Architecture (R-GMA) provided by European Data Grid (EDG) are described. JetGrid for quick knowledge discovery was preliminarily prototyped. The architecture of JetGrid is designed to be compatible with lower-level grid mechanisms. The JetGrid data mining system is extensible, using an object-oriented design that was coded in C++. Some preliminary experimental results are presented, which provide some indication of the performance of the JetGrid data mining infrastructure on top of EDG. Intensive tests are being carried out using GridFTP and other Globus transport mechanism for shipping raw data D across the grid to one node for processing (MD: Move Data Model)[Chan, 1999] and processing the data locally until a result R is obtained and ship the result to one node for further processing (MR: Move Results Model) [Sunderraman, 1998]. More test results will be reported at the Workshop.

References

Cannataro, M. and Talia, D., 2003. The knowledge grid, Communications of the ACM, Volume 46, Issue 1 (January 2003)

Chan P. and Kargupta, H., 1999. Proceedings of the Workshop on Distributed Data Mining, The Fourth International Conference on Knowledge Discovery and Data Mining, New York City, 1999.

Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S., 2001. The Data Grid: towards an architecture for the distributed management and analysis of large scientific datasets. J. of Network and Comp. Appl., (23):187–200, 2001.

Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., 1996. Advances in Knowledge Discovery and Data Mining. Cambridge: AAAI Press / MIT Press

Foster, I. and Kesselman. C., 1997. Globus: A metacomputing infrastructure toolkit. Intl J. of Supercomputer Applications, 11(2):115–128, 1997.

Foster, I. and Kesselman, C., 1999. editors. The Grid: Blueprint for a Future Computing Infrastructure. 1999.

Hinke, T., Novotny, J., 2000. "Data Mining on NASA's Information Power Grid", Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing, Pittsburgh, Pennsylvania, August 1-4, 2000

http://eu-datagrid.web.cern.ch/eu-datagrid

http://www.reuters.co.uk/

Manku, G., Motwani, R., 2002. Approximate Frequency Counts over Data Streams, Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002

Orlando, S., Palmerini, P., Perego, R. and Silvestri, F., 2002. Scheduling High Performance Data Mining Tasks on a Data Grid Environment, proceedings of Europar 2002.

Salton G. and Buckley. C., 1988. Term-weighting approaches in automatic text retrieval. Information Processing and Management, 24(1), 1988.

Sunderraman, R., 1998. The Terabyte Challenge: An Open, Distributed Testbed for Managing and Mining Massive Data Sets, Proceedings of the IEEE Conference on Supercomputing , 1998

Part 2.

Status and Future Directions in Data Mining Standards

XML for Analysis
Extended Abstract

Robert Chu
SAS

XML for Analysis is a Simple Object Access Protocol (SOAP)-based XML API, designed specifically for standardizing the data access interaction between a client application and a data provider working over the Web.

Under traditional data access techniques, such as OLE DB and ODBC, a client component that is tightly coupled to the data provider server must be installed on the client machine in order for an application to be able to access data from a data provider. Tightly coupled client components can create dependencies on a specific hardware platform, a specific operating system, a specific interface model, a specific programming language, and a specific match between versions of client and server components. The requirement to install client components and the dependencies associated with tightly coupled architectures are unsuitable for the loosely coupled, stateless, cross-platform, and language independent environment of the Internet. To provide reliable data access to Web applications the Internet, mobile devices, and cross-platform desktops need a standard methodology that does not require component downloads to the client. Extensible Markup Language (XML) is generic and can be universally accessed. What if, instead of invoking the proprietary interface of a client component, you could call methods and transfer data through XML HTTP messages without any client component? What if the application developer could build client components without concern for tight coupling to a server component or application? What if an application, developed with any programming language and running on any platform, could access data from any place on the Web without having to plan for specific platform support or even a specific provider version? The XML for Analysis specification answers these questions with XML for Analysis. XML for Analysis advances the concepts of OLE DB by providing standardized universal data access to any standard data source residing over the Web without the need to deploy a client component that exposes COM interfaces. XML for Analysis is optimized for the Web by minimizing roundtrips to the server and targeting stateless client requests to maximize the scalability and robustness of a data source. The specification defines two methods, Discover and Execute, which consume and send XML for stateless data discovery and manipulation.

The specification is built upon the open Internet standards of HTTP, XML, and SOAP, and is not bound to any specific language or technology. The specification references OLE DB so that application developers already familiar with OLE DB can see how XML for Analysis can be mapped and implemented. These references also provide background information on the OLE DB definitions that the specification extends. The specification cover data mining and OLAP (Online Analytical Processing). This talk will briefly walk you through a few simple steps from creating a data mining model to scoring a new data source with an existing mining model. How PMML (Predictive Modeling Markup Language) plays a role in the specification will also be discussed.

Java™ Data Mining (JSR-73): Overview and Status Extended Abstract

Mark Hornick

Data mining continues to gain mainstream acceptance for providing strategic advantage in the areas of customer relationship management, fraud detection, and national security, and life sciences, among others. Java technology, especially as leveraged within the scalable J2EE architecture, facilitates integration with applications such as B2C / B2B web sites, customer care centers, campaign management, and genomic / molecular pattern recognition and discovery.

Historically, application developers coded homegrown data mining algorithms into applications, or used sophisticated end-user GUIs. These GUIs packaged a suite of algorithms complete with support for data transformation, model building, testing, and scoring. However, it was difficult, if not impossible, to embed data mining end-to-end in applications using commercial data mining products due to inadequate APIs. If a vendor had an API, it was proprietary, making the development of a product using that API risky. If a different vendor's solution was required, rewriting that product was also potentially costly.

The ability to leverage data mining functionality via a standard API greatly reduces risk and potential cost. With a standard API customers can use multiple products for solving business problems by applying the most appropriate algorithm implementation without investing resources to learn each vendor's proprietary API. Moreover, a standard API makes data mining more accessible to developers while making developer skills more transferable. Vendors can now differentiate themselves on price, performance, accuracy, and features. JavaData Mining (JDM) addresses this need for the Java.

As with any standard, defining compliance for vendor implementations raises a myriad of issues. Should all implementations be required to support all algorithms and features? Should the results of data mining operations, e.g., rules in a decision tree model or scoring results, be the same for the same datasets across vendor implementations? For JDM, compliance is based on a core feature set with optional packages for each mining function and algorithm. This enables vendors that specialize in certain algorithms, e.g., neural networks, to conform to the JDM standard while only implementing relevant packages. JDM also provides *supportsCapability* methods that allow applications to determine at runtime if a vendor implementation supports a finer grained feature, e.g., whether classification model build accepts a cost matrix specification, or the clustering algorithm produces hierarchically arranged clusters. Regarding absolute correctness of results, JDM does not specify a particular result for mining algorithms, instead it focuses on the structure of that results. For example, if a decision tree is produced, each tree node can have at most one parent and their can be at most one root node.

In JDM, we defined one specialized conformance option: the *scoring engine*. Vendors who specialize in scoring data, or who provide a scoring engine option to their product offering, may support a subset of JDM features, namely tasks for *model import* and *apply*. A scoring engine minimally must support the import of models, either in a proprietary format or using a standard format such as PMML, and the scoring itself, either in batch or real-time. The scoring engine option fits well into enterprise architectures where model building is performed in one location and scoring in other, possibly geographically distant, locations. To this end, JDM also specifies an export task to move mining objects between data mining systems.

Through the course of designing the API, the expert group has made numerous tough choices of which features to include in the first release. We have selected classification, regression, associations, clustering, and attribute importance as first release mining functions, and build, apply, test, import, and export as first release mining tasks. Features being considered for the next release of JDM include: a web services interface, transformations, mining unstructured data such as text and images, multi-target models, ensembles, and additional mining functions such as feature extraction and forecasting.

As a Java Specification Request under SUN's Java Community Process (JCP), JDM must go through several reviews and final vote by the JCP Executive Committee before being accepted as a Java standard. In addition, the JCP requires that the API have a Reference Implementation (RI) and Technology Compatibility Kit (TCK). The RI ensures that the specification is implementable while providing potential users and implementors a working system to understand intended behavior. Vendors implementing the standard must certify their implementation by executing and passing the TCK, a test suite ensuring compliance. Both the RI and TCK pose certain challenges for a data mining API in terms of completeness and depth of conformance specification. Feedback from the JDM Community Review and Public Review has been positive and supportive. Initial experience with the RI has further helped to refine the model.

SQL/MM Data Mining and its Relation to other Data Mining Standards Extended Abstract

Christoph Lingenfelder

At the end of 2002, ISO published a data mining part of its multi-media extension SQL/MM (ISO/IEC 13249-6). Other data mining standards have been published earlier or are in public review now. We will start with a short introduction into the approach of SQL/MM data mining. Then, interfaces to other data mining standards are discussed, and an outlook to the next version of SQL/MM data mining is given.

SQL/MM is an extension to the SQL language (ISO/IEC 9075) and consists of standards for “Full Text”, “Spatial”, “Still Image”, and “Data Mining”, as well as “Foundation” to define common terms and features. All of them make use of the object-relational extensions to SQL. In particular, user-defined functions and user-defined types with methods are defined. These SQL extensions are called SQL/MM for historical reasons, because the early work focused on multi-media extensions. Data mining belongs to the same series as it uses the same object-relational foundation.

The data mining part focuses on an SQL API that allows the definition of data mining tasks for model creation and testing as well as the application of mining models. Four kinds of mining functions are covered, to which an implementation can independently claim conformance:

- Association Rules
- Clustering
- Classification
- Regression

For each of them, there is a Settings type that holds the parameters for model creation. Parameters pertaining to an input field can be defined using symbolic names so that a settings value can be used in different mining tasks. Task types combine all the information necessary to start a mining run. Creating a mining model is similar to aggregating data. That is, it operates on multiple rows and the result is a single value representing the mining model (there is one model type for each mining function).

On the application side, model values, together with row values, are used as input parameters into generic, i.e. model-independent scoring methods. These return result values containing the prediction or confidences.

Being SQL data types, all these values can be stored in database tables. There is no need to standardize functions for managing models such as copying, updating or deleting a model. The same is true for defining access rights, definition of model repositories, etc. All this is comes for free with existing SQL capabilities. Furthermore, this approach allows for convenient access to mining operations from within a variety of programming

languages and scripts as most of them can invoke SQL. Similarly, it is easy to reuse and schedule data mining actions in data warehouses or analytical applications or to make backups of important mining models.

For the purpose of model exchange, the SQL/MM data mining standard contains import and export methods. Compliant implementations must support import and export in PMML format. Currently the support of PMML 1.1 is mandatory, but supporting the latest version is recommended by ISO.

Methods for import and export of data mining specifications settings are also defined, but no particular format is required, so that the mechanism can now only be used for exchange between different servers of the same SQL/MM implementation.

Work on the next version has already started. There exists a roadmap laying out a tentative set of features that were suggested for the next version. This list can still be augmented, at least until the workgroup meeting in October 2003.

The most important extensions that were proposed to existing functionality are

- Standard format for import and export of data mining settings types
- Add get methods to retrieve metadata from mining objects
- More introspection into models

Among the new functions that were suggested is

- Single record scoring
- Model Application as a Table Function
- Additional mining functions
- Parameter settings for specific algorithms
- Taxonomies for Association Rule Settings
- Application mode for associations

As an exchange format for settings, it is seen as advantageous to come up with a common format between data mining standards, especially the emerging standard for Java (JSR-73) and SQL/MM. The structure might be based on features present in the CWM standard.

PMML Version 3 - Overview and Status
Extended Abstract

Gregor Meyer

The Predictive Model Markup Language (PMML) is developed by the Data Mining Group, a vendor led consortium, and consists of the following components:

1. **Data Dictionary:** defines the input attributes to models and specifies the type and value range for each attribute.
2. **Mining Schema**: Each model contains one mining schema that lists the attributes used in the model. These attributes are a subset of the attributes in the Data Dictionary. The mining schema contains information specific to a certain model, while the data dictionary contains data definitions that do not vary with the model. For example, the Mining Schema specifies the usage type of an attribute, which may be active (an input of the model), predicted (an output of the model), or supplementary (holding descriptive information and ignored by the model).
3. **Transformation Dictionary:** defined derived attributes. The derived attributes may be defined by *normalization*, which maps continuous or discrete values to numbers; by *discretization*, which maps continuous values to discrete values; by v*alue* mapping, which maps discrete values to discrete values; or by *aggregation*, which summarizes or collects groups of values, for example by computing averages.
4. **Model Statistics:** contain univariate statistics about the attributes used in the model.
5. **Models:** The parameters defined statistical and data mining models per se. Models in PMML Version 2.0 include regression models, clusters models, trees, neural networks, Bayesian models, association rules, and sequence models.

PMML Version 1.0 basically concerned itself with defining standards for various common data mining models assuming that the inputs to the models had already been defined. The inputs are called DataFields. PMML Version 2.0 introduced the TransformationDictionary, which contains DerivedFields. The inputs to models in PMML Version 2.0 may be DataFields or DerivedFields. In principle, this approach is powerful enough to capture the process of preparing data for statistical and data mining models.

PMML Version 2.1 was released as a Source Forge package in March 2003 and provided improved support for cleaning, transforming and aggregating data. These operations can be used to both prepare data and to shape it in real time enabling PMML Version 2.1 to be used for scoring data in one application using models developed by another application.

In this talk, we provide a quick overview of PMML Version 2.1 and describe some of the changes being considered for PMML Version 3.0.

This abstract is based in part on the article: Robert Grossman, Mark Hornick, and Gregor Meyer, Data Mining Standards Initiatives, Communications of the ACM, Volume 45-8, 2002, pages 59-61.