5-2005

Toward autonomic distributed data mining using intelligent web services.

Padmanabhan Ramaswamy 1976-
University of Louisville

TOWARD AUTONOMIC DISTRIBUTED DATA MINING USING INTELLIGENT WEB SERVICES

By

Padmanabhan Ramaswamy
BS., Kerala University, 1998

A Thesis
Submitted to the Faculty of the
Speed scientific School
University of Louisville
As Partial Fulfillment of the Requirements
For the Professional Degree

Master of Science

Computer Engineering and Computer Science Department
University of Louisville
Louisville KY

May 2005

TOWARD AUTONOMIC DISTRIBUTED DATA MINING USING INTELLIGENT WEB SERVICES

By

Padmanabhan Ramaswamy
B.S., Kerala University, 1998

A Thesis approved on

April 29, 2005

By the following Reading and Examination Committee

____
Dr. Anup Kumar, Co-Advisor (CECS)

____
Dr. Mehmed Kantardzic, Co-Advisor (CECS)

____
Dr. Adel Elmaghraby, Member (CECS)

____
Dr. Julius Wong, Member (ME)

ACKNOWLEDGEMENTS

I would like to gratefully acknowledge the enthusiastic supervision of Dr. Anup Kumar who provided me with insights that guided me and challenged my thinking, substantially improving the finished product. I thank Dr. Mehmed Kantardzic for providing me with on-going support throughout the project and for the technical discussions we had on the Distributed Data Mining model. I feel a lasting gratitude toward the faculty of Speed School who were excellent teachers and advisors.

I would also like to thank all my friends and family, wherever they are, particularly my Mum and Dad.

The study was in part supported by Kentucky Science & Engineering Foundation grant KSEF-323-RDE-003 and NSF grant INF 7-001-006.

ABSTRACT

TOWARD AUTONOMIC DISTRIBUTED DATA MINING USING INTELLIGENT WEB SERVICES

Padmanabhan Ramaswamy

May 1, 2005

This study defines a new approach for building a Web Services based infrastructure for distributed data mining applications. The proposed architecture provides a roadmap for *"autonomic"* functionality of the infrastructure hiding the complexity of implementation details and enabling the user with a new level of usability in data mining process. Web Services based infrastructure delivers all required data mining activities in a utility-like fashion enabling heterogeneous components to be incorporated in a unified manner. Moreover, this structure allows the implementation of data mining algorithms for processing data on more than one source in a distributed manner. The purpose of this study is to present a simple, but efficient methodology for determining when data distributed at several sites can be centralized and analyzed as data from the same theoretical distribution. This analysis also answers when and how the semantics of the sites is influenced by distribution in data. This hierarchical framework with advanced and core Web Services improves the current data mining capability significantly in terms of performance, scalability, efficiency, transparency of resources, and incremental extensibility.

TABLE OF CONTENTS

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 7

LIST OF FIGURES

CHAPTER 2

CHAPTER 3

CHAPTER 4

CHAPTER 6

LIST OF TABLES

CHAPTER 3

CHAPTER 4

CHAPTER 6

1. INTRODUCTION

Information technology and data mining technology have lingered too long in an era of over-specialization in which integration is just another specialty. We have made a 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. How do we deal with the complexity of the entire environment 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 the data mining domain. Data mining technology is a typical example where research has made a lot of progress in specific algorithms and tools, but there are not enough results on their integration and simplified use in complex, distributed Internet based environments [8, 11].

Moreover, the assumptions and approaches used for static and centralized data mining processes are not any more valid for distributed environment [9]. Distributed data mining **(DDM)** refers to the mining of distributed data sets using distributed computational resources. The physical distribution of data through different sites in general corresponds to a semantic distribution, i.e., the location of data may have a meaning that needs to be explicitly addressed in a distributed mining process [13]. Data mining algorithms should take place at a local level to discover characteristics of the local site, and at a global level where local data mining results may be combined to find global findings or compare the local semantics. In distributed data mining strategy, it is necessary to provide the means of learning how to analyze, combine, and integrate a number of separately learned dynamic local models. Real-time data mining analysis are highly desirable in many distributed applications on the Internet to update models when new events are detected [3, 4, 6]. Easy distribution of models in a networked environment is essential for maintaining up to date detection capabilities.

Furthermore, the simplified implementation of algorithms and methods developed for centralized data mining applications could make significant influence on a reduced quality of data mining results [14] in the distributed environment. Therefore, it is necessary to reevaluate all data mining phases and all algorithms in the context of distributed resources applied for real-time data mining. This research shows the need for new phases or updated phases in the data mining process [11, 15].

1.1 MOTIVATION

There are many reasons to investigate the use of web services in distributed data mining systems. Today most of the data mining is done in three-tier or four-tier system. In a three-tier system client sends the request to server, the server then runs mining algorithms on its local data. In a four-tier system, the server talks to another remote server, the remote server analyzes its local data. In both the systems, mining algorithms and the data are tied to the server making it a closed system. An extendable and open distributed data mining system can be built by separating the algorithm and data.

Using web services the interaction between algorithm and data can be made dynamic. Also, the algorithm can dynamically locate and analyze the data making it a much more robust system. The main motivation for this thesis involves the realization that using web services a dynamic distributed data mining system can be built.

1.2 CONTRIBUTIONS

The Web Services oriented approach proposed in this project is an attempt to build an infrastructure for DDM that will allow the integration of distributed, heterogeneous environment and complex interconnectivity. This approach will overcome the complexity of DDM applications and the limitations of existing infrastructures using Web Services technology [7, 8]. The proposed architecture will allow users to concentrate on what they want to accomplish rather than figuring how to solve all the technical details in tuning computing system for executing distributed data mining models. The Web Services based infrastructure delivers required data mining activities in a utility-like fashion enabling heterogeneous components to cooperate in a unified manner. The exchange and integration of data and tasks (tools, libraries, device drivers, middleware, etc.) will be implemented through open standards defining the identification of components, their communication protocol and negotiation protocol among them. The key characteristics of the proposed data mining framework are:

Dynamic: Dynamic discovery and use of data sets and data mining algorithms over the Internet based on user defined high-level specifications, and integration into distributed data mining process is one of the essential characteristics of the framework. It will tap the available resources, even negotiate their use, adapting to the conditions in the environment and requirements of the mining process.

Scalable: The Web Services based technologies used in this framework will allow integration of new data sources, algorithms, and tools available on the internet. In addition, clients from any platform will be able to use these new services. The framework will find a way to best interact with other neighboring systems.

Transparent: The complexity of data mining should be transparent to the users of the. In the proposed framework user focuses on what he/she wants to achieve with data mining and not on how to perform data mining. Therefore, the proposed framework hides the complexity of data mining operation built over heterogeneous and distributed Internet.

Extensible: The framework allows incremental and dynamic extensions of data sets and applied algorithms used in a distributed data mining process. Interactive configuration and dynamic reconfiguration of specified services under varying conditions, and adjustments of configurations to best handle changing web environment will be an important feature of our system.

1.3 THESIS ORGANIZATION

The thesis is organized in seven chapters. Chapter 2 summarizes the related research and a model for Distributed Data Mining. Chapter 3 discusses the proposed service oriented architecture. Registration and implementation details for various distributed datasets and data mining algorithms are discussed in Chapter 4 and the data mining algorithms and their functionalities are explained in Chapter 5. The results are provided in Chapter 6 and the summary and conclusions are included in Chapter 7.

2. RELATED RESEARCH

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. The first generation of data mining system utilized local data, with either client-server or 3-tier architectures. 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 source using ODBC, JDBC, or other related protocol. The next generation of data mining systems was component-based. The components could be local, relying on Microsoft's COM or DCOM platforms, for example, or global, relying on systems such as Suns J2EE platform. Angoss is an example of the former, and Kensington 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 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. This is probably because no agent-based infrastructure, over which an agent-based data mining system must be built, was ever widely adopted. Examples of agent-based distributed data mining systems include JAM [19], Papyrus [7], and BODHI [14].

Somewhat later, the next generation of service-based data mining systems began to emerge. These are generally built using W3C's standardization of web services. Examples include DataSpace [9] and data mining systems developed by IBM, Microsoft and SAS that employ the XML for Analysis standard [3]. More general service-based infrastructures, such as grids or data grids [4], 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. The Grid community has begun an effort called the Open Grid Service Architecture, or OGSA, that provides a web service based access to some grid services [17]. OGSA Database Access and Integration Services (OGSA DAIS) [16] combine grid services with web services for remotely accessing databases.

2.1 Existing Systems

There are a few integrated systems developed for distributed data mining. The JAM system developed by Professor Stolfo et al. 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 [10]. The Kensington [5] 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 [9, 10] 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 [4] developed by Professor Grossman is based on a layered infrastructure for high performance and wide area data mining. The distributed agent based mining environment (DAME) is developed by Krishnaswamy et al. The focus of DAME was to delivery data mining services via the Internet. It can support cluster of workstations connected by high performance network. The client server models for distributed data mining were developed by Chattratichat [2] called DecisionCenter, and the IntelliMiner [13] by Parthasarthy et al. These existing approaches in modeling data mining infrastructure suffer from one or more of the following limitations:

- Selected 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 [7].

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

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

- Most of the existing implementations of data mining infrastructures are tightly coupled and require both ends of the communication system to use the same distributed object model. This may not work across heterogeneous environments and firewall or proxy servers [6].

- 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 [4]. 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 web of distributed algorithms and other resources in addition to web of data [5].

In the existing data mining approaches, the success or failure of a distributed data-mining project is highly dependent on a particular person and a tool. Successful practice may not necessarily be repeated across the applications, especially when they are becoming more complex and more demanding in a distributed environment [8]. Therefore, data mining needs standard protocols in terms of a methodology and technology used. The objective of these standards should be to help user translate an application problem into data mining tasks on a higher abstract level with less efforts [11]. That is especially important for the users in the Internet environment where distributed data mining applications become extremely complex. This complexity increases when requirements for incremental or on-line mining [6] is added to the huge amount of data that is available for mining in distributed heterogeneous systems. Integrating time- and space-sensitive functionality of data in the mining process, and only partial availability of data, confronting privacy and security policies, all that enormously increase the complexity of the data-mining task [2, 8, 11]. This growing complexity of the distributed infrastructure threatens to undermine the very benefits data mining technology aims to provide.

2.2 DISTRIBUTED DATA MINING (DDM)

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 [10, 11, 13, 16]. The popularity of Internet and Web makes it imperative that data-mining framework is extended to include the distributed and time dependent information and tools. Moreover, the assumptions and the approaches used for static and centralized data mining process are not valid any more for distributed environment [14, 17]. Traditional development and deployment of different data mining techniques in a data mining process usually assume that:

- Data set for analysis is centralized, on a single computer, in a form of centralized database, data warehouse, data mart, or for simple problems as a flat file.
- The amount of available data is enough for successful application of a data mining techniques.
- Software tools supporting different data mining techniques are also centralized; they are located on the same computers where the data are stored.

For many classical data mining applications these assumptions are good enough [10, 11, 12] but in an Internet-based distributed information environment these assumptions about data, algorithms, and data mining process are not true and may represent constraints on the quality of data mining results. Standardization on one side, and 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 important, with new higher quality in data mining results. Our approach in standardization of distributed data mining methodology is based on a model of data mining process having four levels of abstraction [15]. They are organized in a hierarchical structure: a) phases of data mining, b) generic tasks, c) specialized tasks, and d) process instances. The hierarchy of the model is graphically presented in **Figure 2.1**.

The structure will be illustrated with examples of task hierarchies shown in **Figure 2.2**. For example, in the *exploration phase* one of generic tasks is "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, possible specialized task is "elimination of missing values", and corresponding process instances are: "mean value algorithm" or "clustering".

While higher level of abstract concepts in the hierarchy explain *what to do*, lower level components go in details of data processing and answer the question *how to do*. We have used the bottom-up approach for building service-oriented infrastructure for data mining. The implementation starts with process instances as components of Web services, and then integrate them into more abstract, and at the same time more complex and intelligent data mining specialization tasks.

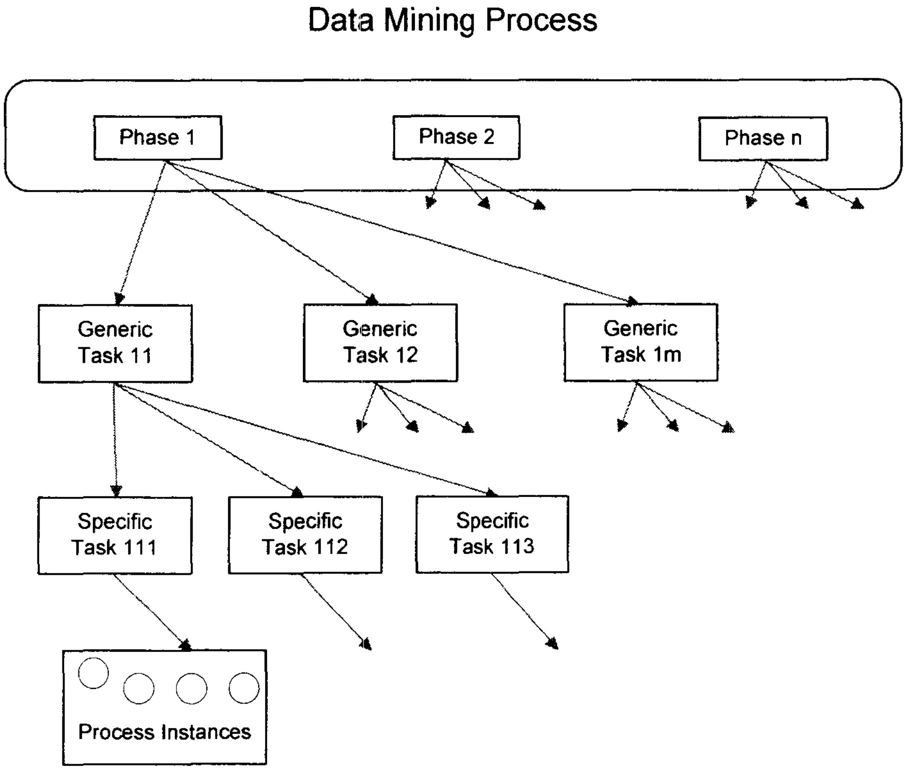


Figure 2.1: Hierarchical structure of a (Distributed) Data Mining Model

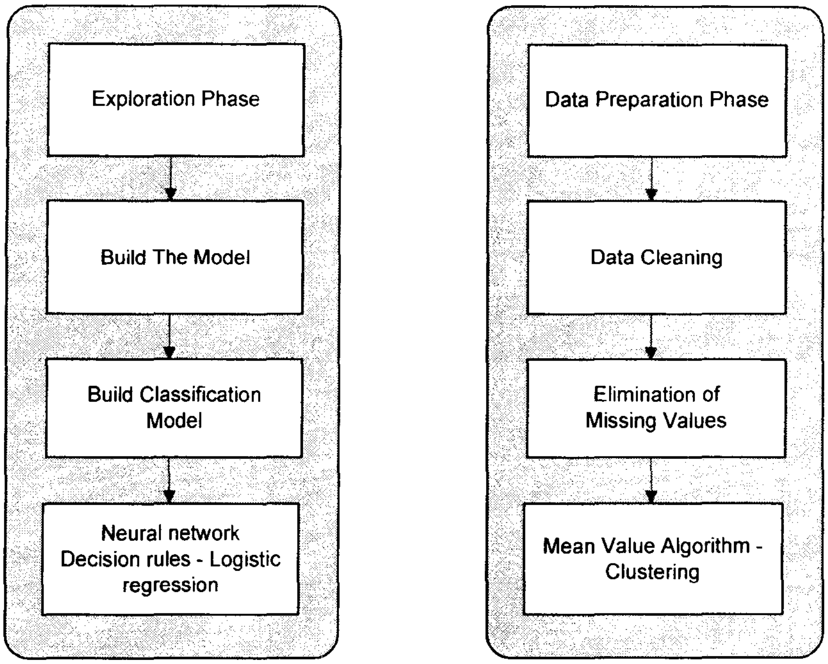

Exploration Phase
Build The Model
Build Classification Model
Neural network Decision rules - Logistic regression
Data Preparation Phase
Data Cleaning
Elimination of Missing Values
Mean Value Algorithm - Clustering

Figure 2.2: Examples of paths in hierarchical structure of a data mining process

2.3 Web Services

A term that refers to distributed or virtual applications or processes that use the Internet to link activities or software components. Web services use the following standards for communication and data processing:

- **XML** (eXtensible Markup Language) - An open standard for describing data.
- **UDDI** (Universal Description, Discovery, Integration specification) - Descriptive standard for documentation and how/where to publish it in an automated fashion.
- **SOAP** (Simple Object Access Protocol) - A technique to allow communications between applications over the web.
- **WSDL** (Web Services Description Language) - An XML specification for describing web services, what they do, and how to access them.

A web service makes itself available by describing itself in a Web Services Description Language (WSDL) document. WSDL document is a XML document that has all the information about web service, including its name, the operations it supports, parameters for those operations and the location where it is running. For service consumers or clients to locate the web service, web service provider has to publish the web service at registry server. Universal Description, Discovery, and Integration (UDDI) are a standard protocol to publish or to find web services. The description part in UDDI is for service provider to publish details about their organization and web services they provide. The discovery part in UDDI is for service consumers or clients to find these services. Service consumer or client invokes web service using Simple Object Access Protocol (SOAP). SOAP is a lightweight XML protocol used for information exchange between heterogeneous systems.

- Service provider registers the service in UDDI Registry.
- Client finds the service, gets the WSDL for the service.
- Using WSDL, client constructs a web service call and invokes the service.

Advantages of Web Services:

1. **Interoperability**: Client written in Java can invoke a service written in .NET or vice versa.
2. **Ubiquity**: Once the web service is registered in a registry server, any organization can use it.
3. **Loosely coupled**: Web service does not depend on the underlying system architecture.

Creating a Distributed Data Mining architecture using Web services involves a lot of new technologies. Java Web Services Developer Pack (JWSDP) from Sun Microsystems solves many of the issues and was suitable for this framework and hence chosen as the toolkit for its implementation.

2.4 SOAP Based Web Services

A web service may be implemented as a standalone TCP server, or it may be accessed via a URL through a web server. When running under a web server, the SOAP service can take advantage of firewall tunneling, although performance will be reduced. The SOAP service accepts XML that describes an action for the server to perform, and returns XML to the client describing the result of the operation. It is possible to maintain state between operations, and operations are essentially non-streaming, due to the marshaling rules of XML. There are two fundamental problems when using web services for data mining of moderate to large size remote and distributed data sets. First, due to the overhead of XML encoding and parsing, there is a limit to the speed of the data transmission and the total size of the return set. This is caused by the need to retain the entire dataset in local storage due to XML encoding and decoding rules. The specific issue is that redundant parts of XML documents can and must refer to the other similar parts of documents. This requires that the entire document be maintained for lookup purposes. Therefore, all data packaging mechanisms that are truly XML compliant are in essence non-streaming. While a server could, in theory, safely ignore this encoding rule when there are no circular data structures, a compliant client cannot safely do so.

3. WEB BASED DISTRIBUTED DATA MINING FRAMEWORK

Distributed environment used to be viewed as a solution for intensive computations with large datasets. However, operating with real world applications in distributed environment motivated new reasons to appear different than the traditional one. One application is in distributed data mining. Data mining algorithms are known with their combinatorial results which make distributed environment a good candidate for them. But sometimes, distribution of data is intended because the distribution itself has a meaning. A very popular example is in banking where banks like to exchange their models for fraud detection or loans approval. This is infeasible because of privacy of customers. Even if exchanging models were feasible it will lead to unexpressive results in case of loans approval because banks apply different policies for loans. In these cases, the following approach for mining distributed data is suggested:

- Build local models for each site
- Discuss relations between local models of sites i.e., whish sites are similar and which are different, to what extent they differ and the implication and significance of their differences.
- For similar sites build global model(s) and exclude those very different

3.1 Service Oriented Approach

Service oriented computing has become ever more popular due to the proliferation of web based computing [6]. In this paradigm, core Web Services [11] (as Web Components) form the fundamental elements for developing large and complex data mining environment. The core services, their description, operations including publication, discovery, selection and binding constitute the foundation of service oriented computing. Using these core services higher composition layers encompassing functionality of multiple layers at lower level can be built. The aggregate service components can then become part of the core services and can be used in building higher-level components. These aggregate components can be published and discovered by applications for providing effective solutions to users.

In general, service oriented Web Components are self-describing, based on open standards, allow rapid composition of distributed applications. Service or component providers develop these service-based components; they supply the detailed description of the functionality through the Web Service Definition Language (**WSDL**). The data or algorithm providers register the services with descriptions WSDL in the registries and the registries are searched by the clients for the required Web Services components [11, 15].

These components provide distributed computing framework across different platforms and languages. The design of core Web Services and aggregate services is a distributed programming task where service providers want to reuse existing services by extending or restricting their functionality without building from scratch. Currently, there is no data mining framework that allows the definition and implementation of data mining Web Services compositions. In order to alleviate this limitation, a framework for developing aggregate services for distributed data mining is proposed in this project.

3.2 EXECUTION FRAMEWORK

The design of core Web services and aggregate services is a distributed programming task where service providers want to reuse existing services by extending or restricting their functionality without building from scratch. Currently, there is no data mining framework that allows the definition and implementation of data mining Web service compositions. In order to alleviate this limitation, a framework for developing aggregate services for Distributed Data Mining is proposed in this paper.

In order to build service oriented framework for distributed data mining, we propose the architecture shown in **Figure 3.1**. In subsequent discussion, first the major components and their interaction are discussed followed by the details of major modules required for this architecture. In this framework data and algorithm providers will register their resources as follows:

Algorithm and data providers will describe their services using Web Services Description Language (WSDL) and interact with the registration service. Registration service will use WSDL description for entry in the registry.

Clients will interact with the Web interfaces and provide their data mining requirements in terms of data to be analyzed and types of mining activities to be performed. Detailed structure of data and the structure of algorithm are obtained from the registry by retrieving the WSDL for the data set and algorithm. Based on the client selection of data and algorithm, the execution framework will generate the execution sequence and will interact with the data and algorithm provider directly based on the information in WSDL.

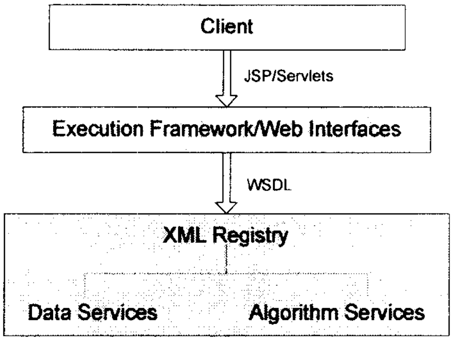


Figure 3.1: Execution framework

3.3 Detailed Architecture

In order to build service oriented framework for distributed data mining, we propose the architecture shown in **Figure 3.2**. In subsequent discussion, the major components and their interaction are discussed followed by the details of major modules required for this architecture. The core components of the proposed framework are described in **Table 3.1**.

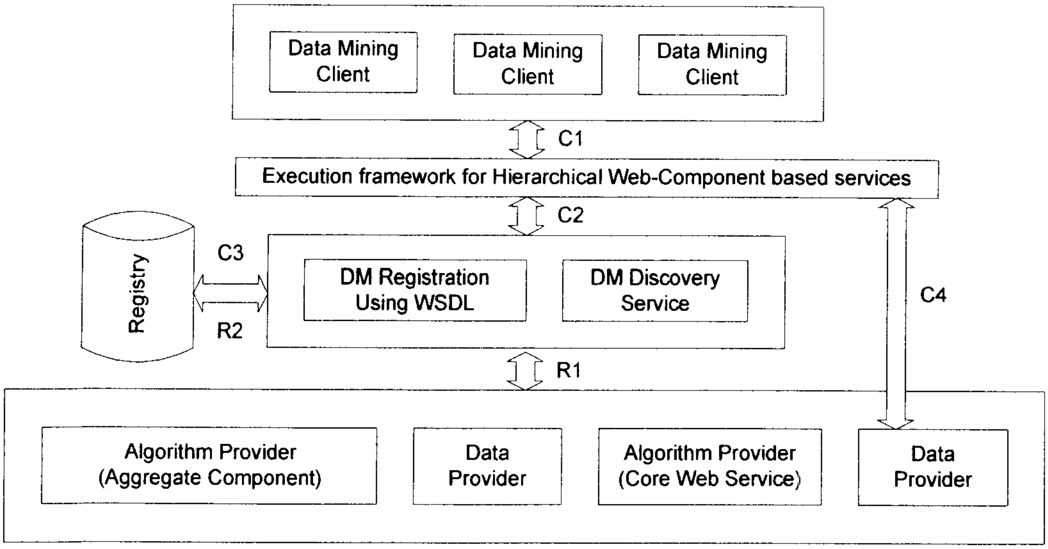

Data Mining Client
Data Mining Client
Data Mining Client
C1
Execution framework for Hierarchical Web-Component based services
C2
Registry
C3
R2
DM Registration Using WSDL
DM Discovery Service
C4
R1
Algorithm Provider (Aggregate Component)
Data Provider
Algorithm Provider (Core Web Service)
Data Provider

Figure 3.2: Architecture for hierarchical Web Component service execution

Component	Functionality
Clients	The user of the distributed data-mining framework.
Registry	This location has database for all the algorithm service providers and the data provides a to keeps their details in the form of WSDL This registry could be a local registry in an organization, or global UDDI registries [8].
Data Mining Discovery Service	Allows clients to discover and select distributed data sets and algorithms based on data mining problem requirement and returns WSDL description to Data Mining Web component execution framework.
Algorithm Provider	Provides the data mining algorithms. These algorithms could be core Web Services or aggregate components. It is the responsibility of the provider to register the WSDL of the dataset in registry using registration service.
Data Provider	Provides the datasets for analyses. It is the responsibility of the provider to register the WSDL of the dataset in registry using registration service.
Execution Framework	Provides support for invoking data and algorithm discovery service and the proper execution of selected algorithms.

Table 3.1: Functionality of the core components in the proposed framework given in Figure 3.2

In this framework data and algorithm providers will register their resources as follows:

R1: Algorithm and data providers will describe their services using Web Services Description Language (**WSDL**) and interact with Registration service.

R2: Registration service will use WSDL description for entry in the registry.

Clients will interact with data mining execution framework as follows:

C1: Clients will interact with Web Component execution framework and provide their data mining requirements in terms of data to be analyzed and types of mining activities to be performed.

C2: This framework will interact with data mining discovery service application program interface (API) by specifying data and algorithm requirements.

C3: The discovery service will provide a set of data and algorithms matching the client's requirement. Client will select the resources for data mining. Detailed structure of data and the structure of algorithm are obtained from registry by retrieving the WSDL for the data set and algorithm.

C4: Based on the client selection of data and algorithm, the Web component execution framework will generate the execution sequence and will interact with the data and algorithm provider directly based on the information in WSDL.

4. IMPLEMENTATION FRAMEWORK

4.1 DETAILED SYSTEM ARCHITECTURE

Registration is the fist step toward system initiation, during which services are registered with the registry. The services can be either data services or algorithm services. The registration process requires the following details from the service provider.

▪ Organization:	Name and Description
▪ Contact:	Name, Phone, and Email
▪ Service:	Name and Description
▪ Service Binding:	URI and Description

An organization can have one service with several bindings or multiple services. **Figure 4.1** shows the registration process.

This data registration process requires the data providers to specify the fields that are available for performing mining tasks, the data repository location, and how to get access the data or execute an algorithm on the data site. In the algorithm registration process, the provider gives the detailed information about the parameters associated with the service and other configuration details.

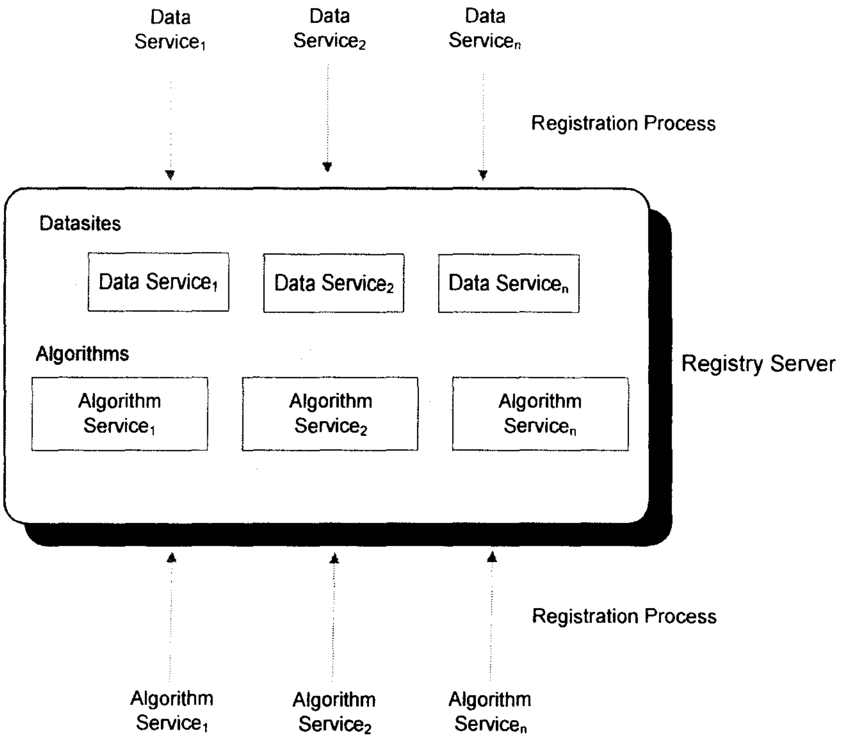


Figure 4.1: The registration process where the service providers register their services.

Once the registration process is complete, the services are ready to be used by the system for applying the various algorithms on the registered data sites. The first step of processing a client request is to select the dataset the client is interested in. In the proposed framework, a client is allowed to select a dataset from a list of available registered data services of their interest. Once the dataset is chosen the client has a choice of algorithms that can be applied to the selected dataset. All the dataset and algorithm information is retrieved from the registry.

In data registration, the data providers specify the data fields that are available for mining, the location of the data, and how to get access the data or execute an algorithm on the data site. In algorithm registration, the provider gives the detailed information about the service execution. For example information included for Fractal Dimension service is shown in **Table 4.1**.

Attribute	Value
Organization Name	UofL ASP
Organization Description	University of Louisville – Algorithm Service Providers
Service Name	Fractal Dimension
Service Description	Algorithm Service to calculate Fractal Dimension, number and types of inputs needed, outputs generated
Service Binding URI	http://136.165.147.86:8080/fdalgo-jaxrpc/fd
Service Binding Description	Service Binding for Fractal Dimension Algorithm

Table 4.1: Registration information for a service

Once the registration process is complete, the services are ready to be used by the system for applying the various algorithms on the registered data sites. The first step of processing a client request is to select the dataset the client is interested in. In the proposed framework a client is allowed to choose a dataset from a list of available registered data set services of their interest. Once the dataset is selected the client has a choice of algorithms that can be applied to the selected dataset. All the dataset and algorithm information is retrieved from the registry.

Figure 4.2 describes the general structure of most distributed data mining algorithms. In this figure distributed data mining algorithms are broken in three different parts that could be executed iteratively. The first part of the logic is performed on all the data sets involved for computing local models. These intermediate results are then integrated in part 2 of the processing model. Then, the integrated results are sent to part 3 of the computation that will distribute results for generating global models of data mining algorithm. One of the main goals in building distributed data mining algorithms is to reduce the amount of data exchanged between computing sites.

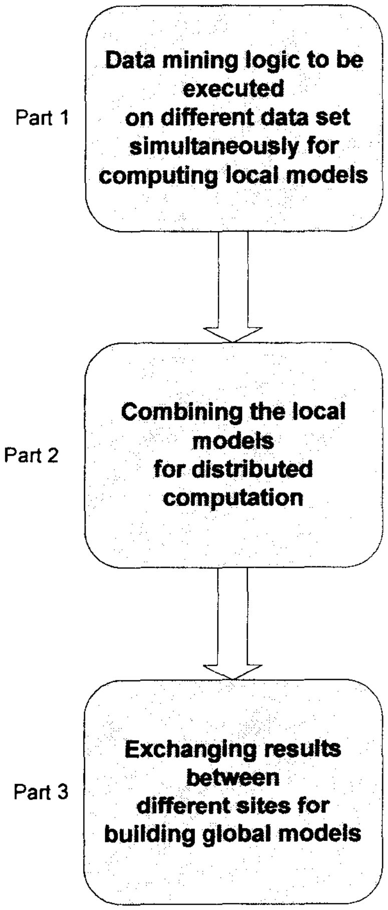

Part 1
Data mining logic to be executed on different data set simultaneously for computing local models
Part 2
Combining the local models for distributed computation
Part 3
Exchanging results between different sites for building global models

Figure 4.2: Structure of distributed data mining approach for distributed data

In our implementation framework parts 2 and 3 of the DDM service are generic, and will be offered as a core sub-service. This sub-service can perform different types of activities on various results obtained at different sites.

The implementation is broken in two parts:

i) the logic to be executed independently on various data sets.

ii) the logic that needs to be executed with the combined results.

We have chosen two approaches to implement this framework:

a) Centralized approach

In the centralized approach, the datasets residing on remote sites are collected and combined to form one data resource bundle on which the algorithm is executed as shown in **Figure 4.3**.

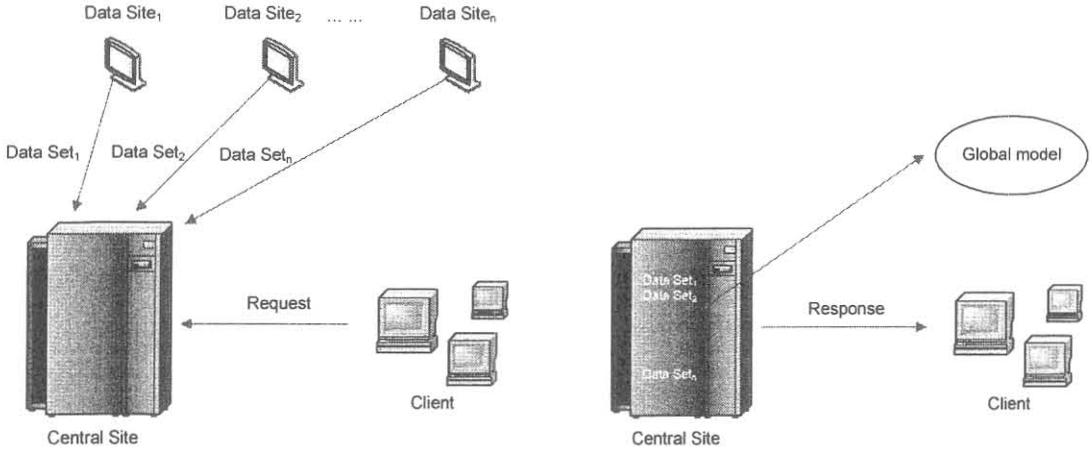


Figure 4.3: The datasets being collected into the central site and the results being sent to the client after processing the global model

b) Distributed approach

In the distributed approach, the datasets are not moved and they always reside on the remote sites. The algorithm is propagated to the remote data sites where it processes the individual dataset and returns local models to a coordination site. The results are then processed to generate the global model and thereafter sent to the client. The number of intermediate results directly depends on the data mining algorithm itself. The distributed approach may require multiple transfers of data between the co-ordination site and the data sites but it eventually proves to be the better of the two choices, especially for large real world data sets since the amount of data transferred is considerably lesser compared to the fetching of the actual datasets. The distributed approach is shown in **Figure 4.4**.

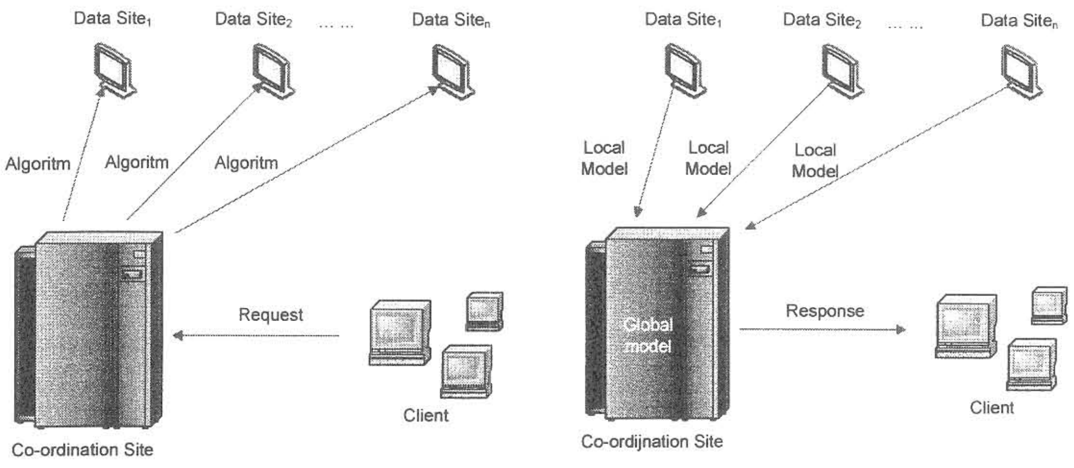


Figure 4.4: The algorithm being sent to the remote site and local models being collected to generate the global model and results being sent to the client after processing the global model

4.2 IMPLEMENTATION OF THE SERVICE FRAMEWORK

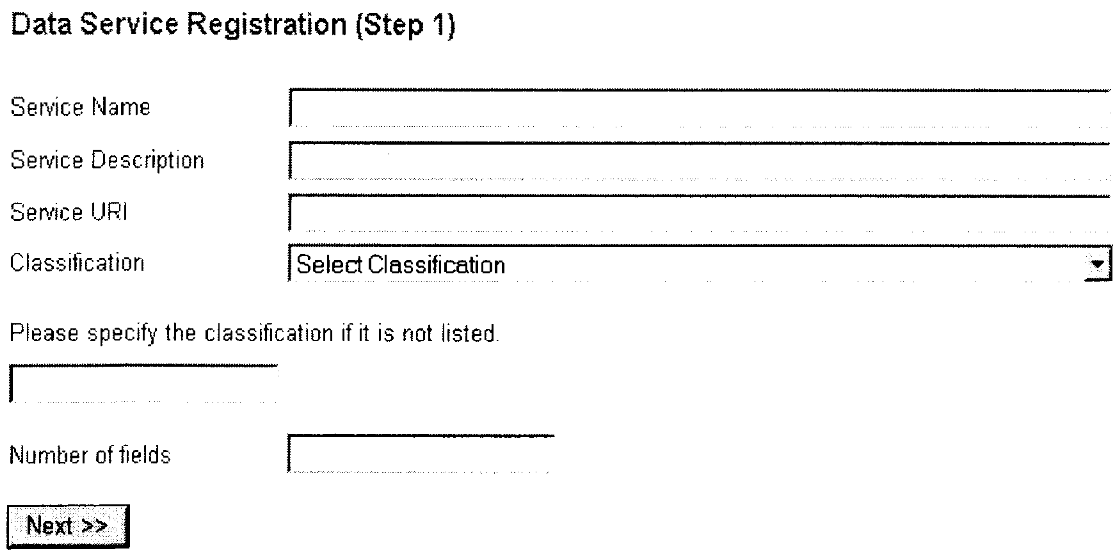
Registering Data and Algorithm Services require the following common parameters:

a. Service Name
b. Service Description
c. Service URI
d. Classification

The classification is an optional attribute but it plays a major role in reducing drastic amounts of search time.

4.2.1 Data Services

The data service registration is a two step process. The first step requests the user to provide the service information and the number of data fields that are available for processing. **Figure 4.5a** shows the interface the client uses as the first step towards registering the data service.



Data Service Registration (Step 1)

Service Name

Service Description

Service URI

Classification: Select Classification

Please specify the classification if it is not listed.

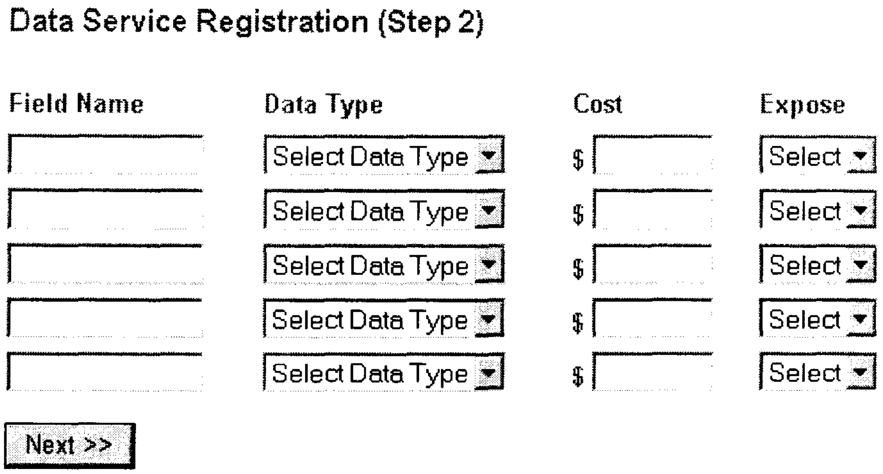
Number of fields

Figure 4.5a: Step 1 of the Data Service registration process

For each attribute of the data service, the following parameters need to be specified.

a) Attribute Name
b) Attribute Type
c) Attribute Cost
d) Attribute Exposure

The Attribute Type can be any of the data types supported by Web Services. The Attribute Cost specifies the amount to use the attribute. The Attribute Exposure specifies if the user can view this attribute. A value of **Yes** allows the user to view/copy the attribute. A value of **No** means that the attribute can be used to test algorithms on but the actual values cannot be viewed or copied. **Figure 4.5b** shows the interface the client uses as the second step to provide this information about the data service. Suppose the **Number of fields** was set to 5 in **Step 1**.

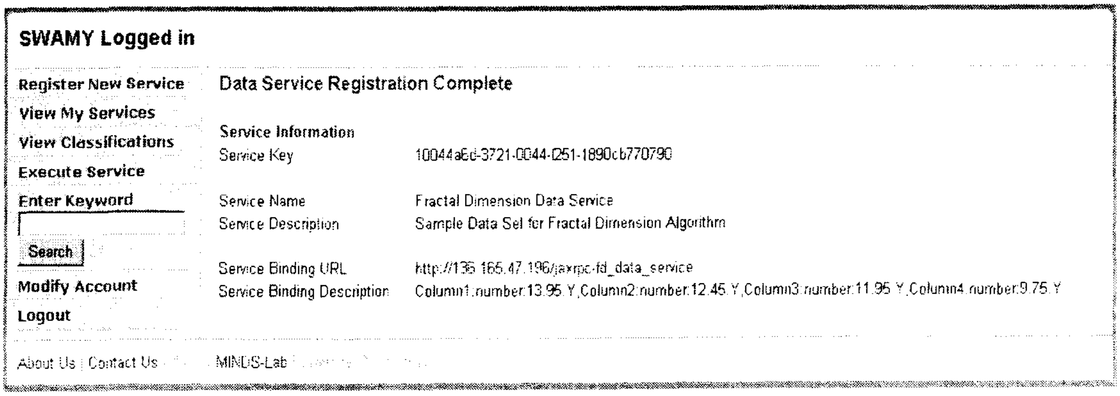


Data Service Registration (Step 2)

Field Name	Data Type	Cost	Expose
	Select Data Type	$	Select
	Select Data Type	$	Select
	Select Data Type	$	Select
	Select Data Type	$	Select
	Select Data Type	$	Select

Figure 4.5b: Step 2 of the Data Service registration process

The information provided is used to register the Service with the local registry and additional information is stored in a database. **Figure 4.5c** shows the registry response received once the data service registration is complete.



Data Service Registration Complete

Service Information

Service Key 10044a6d-3721-0044-c251-1890cb770790

Service Name Fractal Dimension Data Service

Service Description Sample Data Set for Fractal Dimension Algorithm

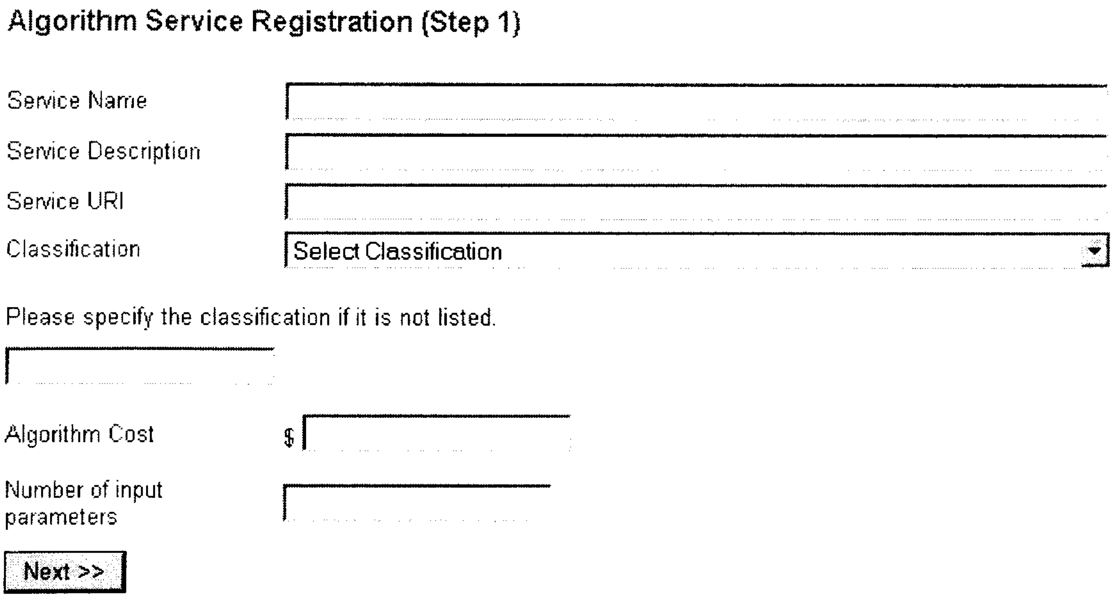
Service Binding URL http://136.165.47.196/jaxrpc-fd_data_service

Service Binding Description Column1:number:13.95:Y,Column2:number:12.45:Y,Column3:number:11.95:Y,Column4:number:9.75:Y

Figure 4.5c: Registry response for the Data Service registration process

4.2.2 Algorithm Services

The algorithm service registration is a two step process. The first step requests the user to provide the service information and the cost of using the algorithm and the number of input parameters accepted. **Figure 4.6a** shows the interface the client uses as the first step towards registering the algorithm service.



Algorithm Service Registration (Step 1)

Service Name

Service Description

Service URI

Classification: Select Classification

Please specify the classification if it is not listed.

Algorithm Cost $

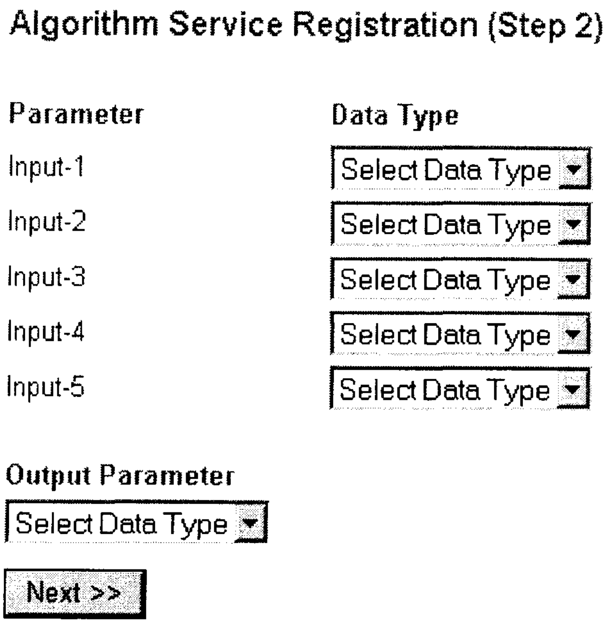
Number of input parameters

Figure 4.6a: Step 1 of the Algorithm Service registration process

The second step of the algorithm service registration process requires the following additional fields:

a) Number of input parameters
b) Output parameter type
c) Algorithm Cost

Based on the parameter Number of input parameters, Step 2 will dynamically allocate the required number of parameters to accept the data type. **Figure 4.6b** shows the interface the client uses as the second step to provide this information. Supposc the **Number of input parameters** was set to 5 in **Step 1**.



Algorithm Service Registration (Step 2)

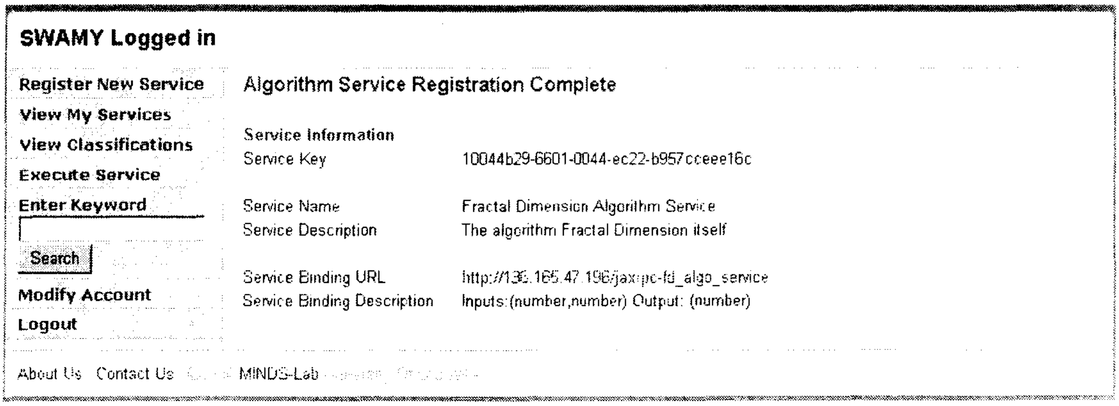
Parameter	Data Type
Input-1	Select Data Type ▾
Input-2	Select Data Type ▾
Input-3	Select Data Type ▾
Input-4	Select Data Type ▾
Input-5	Select Data Type ▾

Output Parameter

Select Data Type ▾

Figure 4.6b: Step 2 of the Algorithm Service registration process

The information provided is used to register the Service with the local registry and additional information is stored in a database. **Figure 4.6c** shows the registry response received once the algorithm service registration is complete.



SWAMY Logged in

Algorithm Service Registration Complete

Service Information

Service Key	10044b29-6601-0044-ec22-b957cceee16c
Service Name	Fractal Dimension Algorithm Service
Service Description	The algorithm Fractal Dimension itself
Service Binding URL	http://136.165.47.196/jaxrpc-fd_algo_service
Service Binding Description	Inputs:(number,number) Output: (number)

Figure 4.6c: Registry response for the Algorithm Service registration process

4.2.3 CENTRALIZED SERVICES

In the centralized approach, a collection of remote datasets is performed concurrently to form a single data resource bundle on which the data mining algorithm is executed. The algorithm also needs to be fetched from its remote location to the co-ordination site where the actual execution takes place. The following web services are used to achieve the execution of the centralized framework.

- **CopyRemoteData** (in **remote_location**, in **data_handler**, in **local_repository**): reads the data from the *"remote_location"* based on the specifications of the *data_handler"* and stores the data in the *"local_repository"*
- **CombineData** (in **local_repository**, out **combined_data**): combines the data read from the *"local_repository"* to create one large data source, *"combined_data"*.
- **FetchAlgorithm** (in **algorithm_location**): fetches the data mining algorithm from the *"algorithm_location"* and stores it locally for processing the data in the local repository
- **ExecuteAlgorithm** (in **local_repository**, out **results**): processes the data mining algorithm on the *"local_repository"* with the logic specified in the algorithm. Returns the intermediate *"results"* for performing other tasks or the final *"results"*.

4.2.4 Distributed Services

In the distributed approach, the datasets always reside on the remote sites and are never moved. The algorithm is propagated to the remote data sites where it processes the individual dataset and returns local models to a coordination site. The results are then processed to generate the global model and thereafter sent to the client. The number of intermediate results directly depends on the data mining algorithm itself.

- **SendAlgorithm** (in **algorithm_location**, in **remote_locations**): fetches the data mining algorithm from the *"algorithm_location"* and sends it to the *"remote-locations"* for processing the data.
- **ExecuteAlgorithm** (in **remote_locations**, out **results**): executes the algorithms concurrently on the *"remote_locations"* and stores the *"results*" on the remote location as the data model.
- **CopyRemoteModel** (in **remote_models**): reads the *"remote_models"* from the remote locations and copies it to the local coordination site for processing to get the final results.

The distributed approach may require multiple transfers of data between the coordination site and the data sites but it eventually proves to be the better of the two choices, especially for large real world data sets since the amount of data transferred is considerably less compared to the fetching of the actual datasets. The performance improvement can be measured in terms of execution time as well as the quality of service.

4.3 Selecting Dataset and Algorithm Service Information

The first step in the exceution phase is to choose the desired datasets and thc algorithm that can be applied on them. **Figure 4.8** shows the steps involved in the selection of dataset and algorithm information from the registry.

The client uses the web interface to query the registry for information about available datasets based on classifications or keyword search. The registry returns the appropriate information on all the datasets that match the search criteria. Once the client selects the datasets for processing, a search for algorithms that match the selected datasets is performed and the algorithm service information that can be applied on them is displayed. This step of the implementation is common for the centralized and distributed approach.

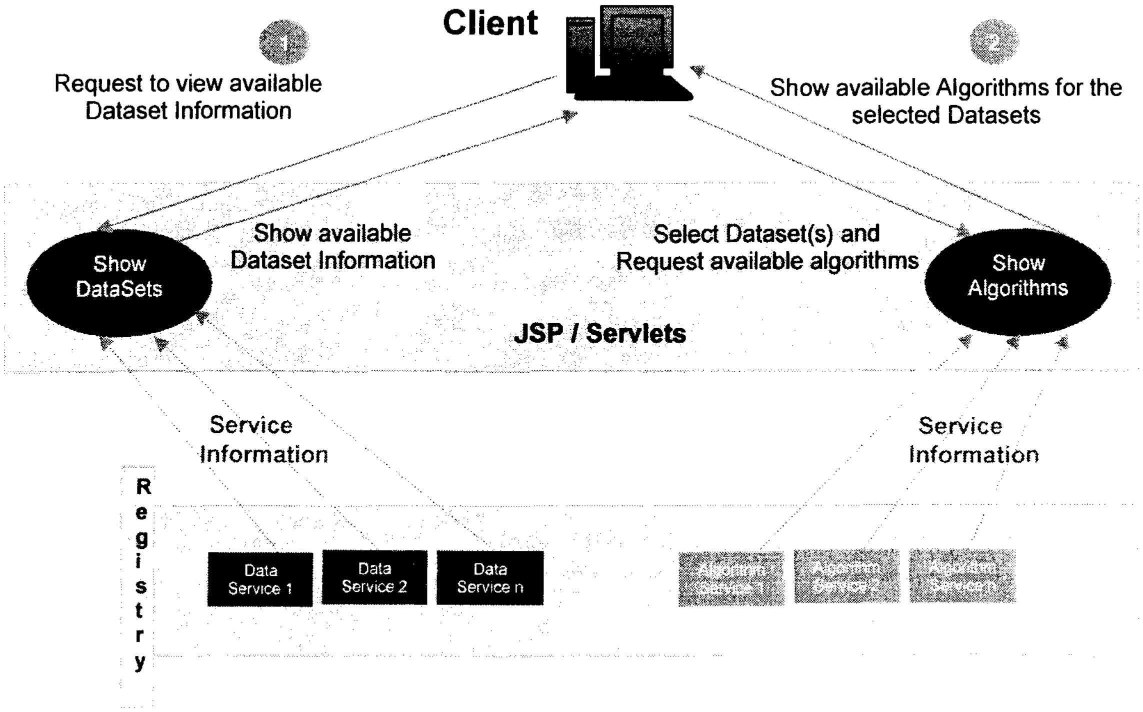


Figure 4.8: The client requests to view service information from the registry

4.3.1 EXECUTION - CENTRALIZED APPROACH

Step 1: The client invokes the data mining services from a URL. The client selects the required datasets located on remote sites.

Step 2: Servlet calls the service on the selected data sites to collect the data. The collected data is stored in a centralized location for processing as shown in **Figure 4.9**.

Step 3: The Apriori Service downloads the algorithm to the central site and is applied to build the global model for the datasets at the central site.

The datasets are now available as a single resource in the central site. The algorithm is applied on this combined data set and the results are returned to the client.

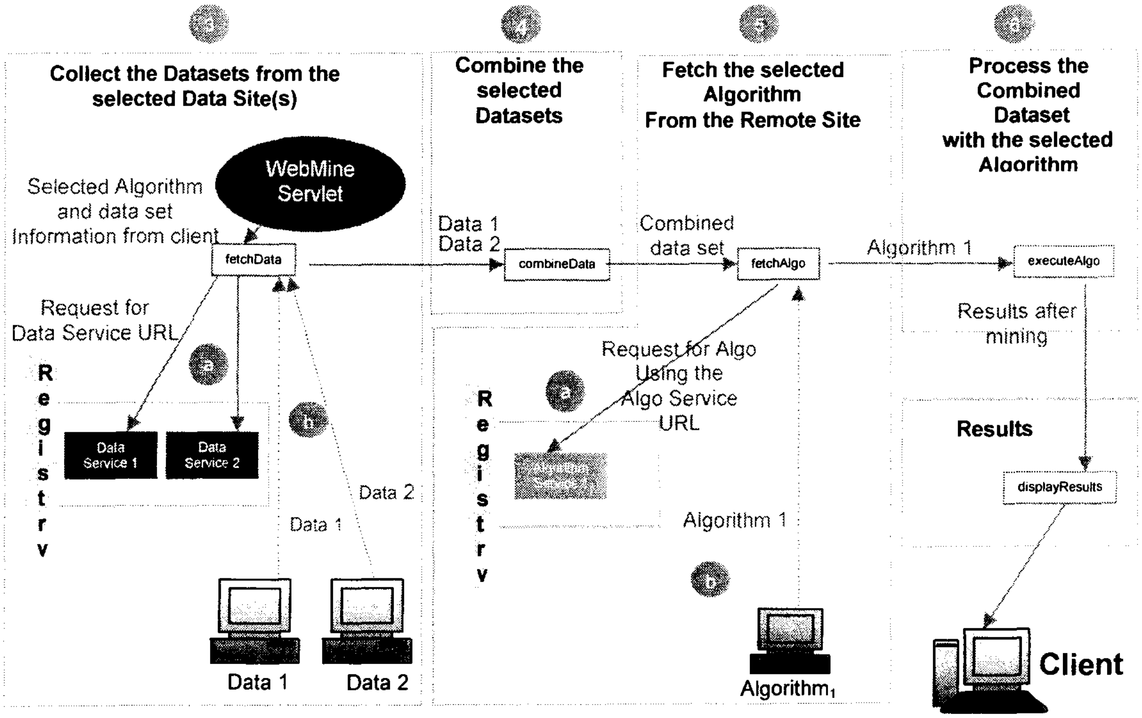
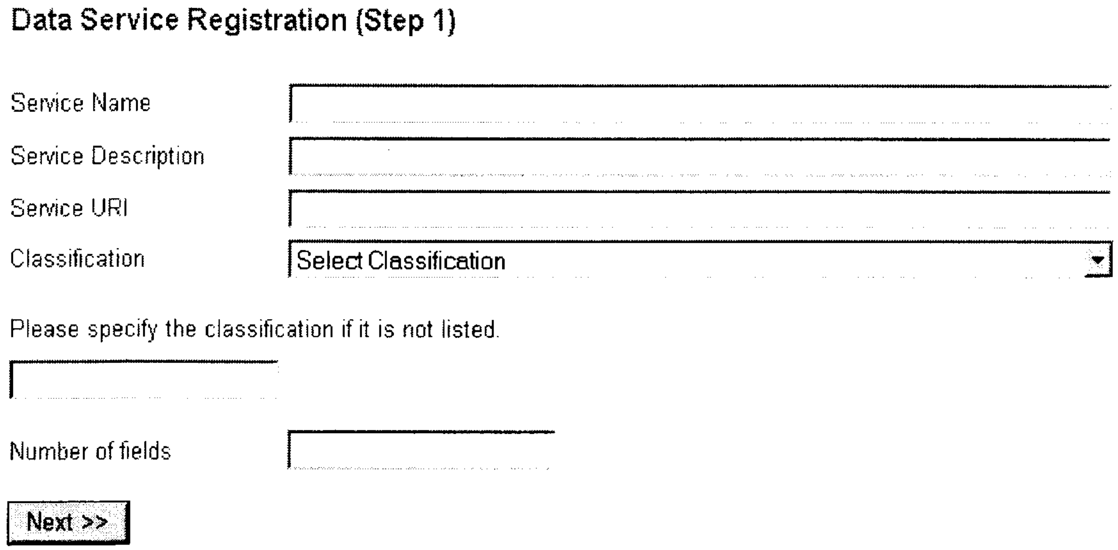

4
Collect the Datasets from the selected Data Site(s)
Combine the selected Datasets
Fetch the selected Algorithm From the Remote Site
Process the Combined Dataset with the selected Alaorithm
WebMine Servlet
Selected Algorithm and data set Information from client
fetchData
Request for Data Service URL
a
R
e
g
i
s
t
r
v
Data Service 1
Data Service 2
b
Data 1
Data 2
Data 1
Data 2
Data 1
Data 2
combineData
Combined data set
fetchAlgo
Request for Algo Using the Algo Service URL
a
R
e
g
i
s
t
r
v
Algorithm 1
b
Algorithm₁
Algorithm 1
executeAlgo
Results after mining
Results
displayResults
Client

Figure 4.9: Client interaction in Centralized Approach

4.2.1 Data Services

The data service registration is a two step process. The first step requests the user to provide the service information and the number of data fields that are available for processing. **Figure 4.5a** shows the interface the client uses as the first step towards registering the data service.



Data Service Registration (Step 1)

Service Name

Service Description

Service URI

Classification: Select Classification

Please specify the classification if it is not listed.

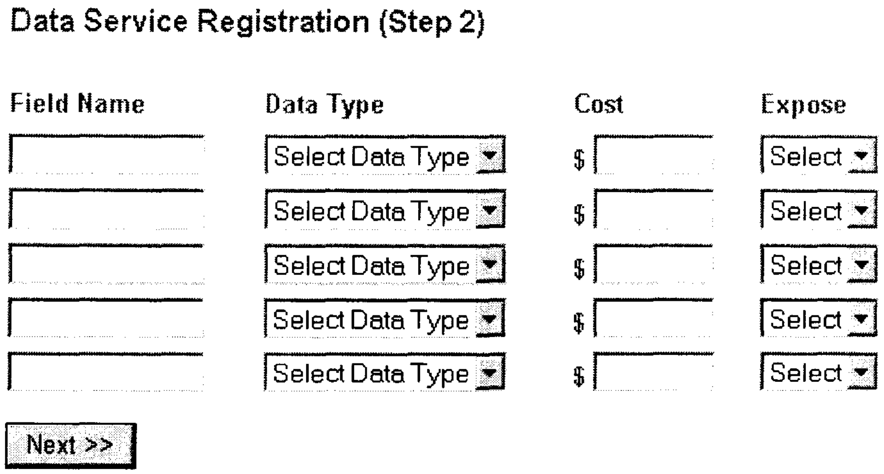
Number of fields

Figure 4.5a: Step 1 of the Data Service registration process

For each attribute of the data service, the following parameters need to be specified.

a) Attribute Name
b) Attribute Type
c) Attribute Cost
d) Attribute Exposure

The Attribute Type can be any of the data types supported by Web Services. The Attribute Cost specifies the amount to use the attribute. The Attribute Exposure specifies if the user can view this attribute. A value of **Yes** allows the user to view/copy the attribute. A value of **No** means that the attribute can be used to test algorithms on but the actual values cannot be viewed or copied. **Figure 4.5b** shows the interface the client uses as the second step to provide this information about the data service. Suppose the **Number of fields** was set to 5 in **Step 1**.

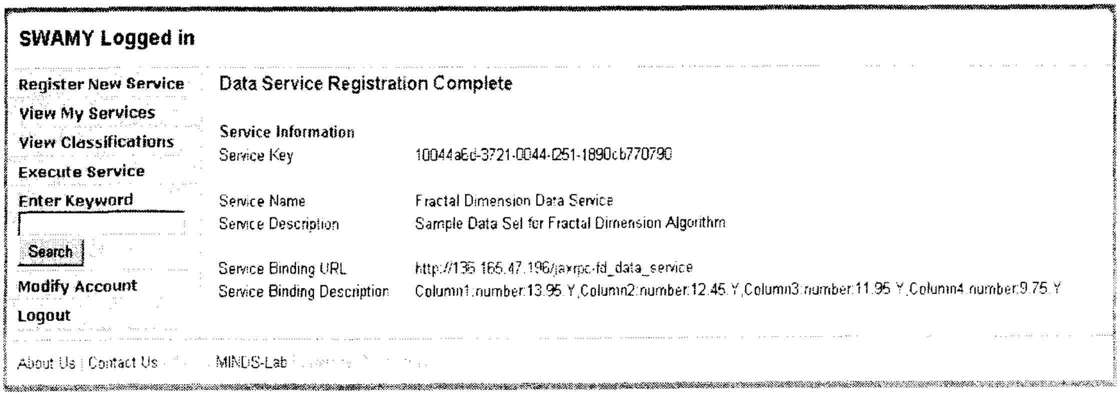


Data Service Registration (Step 2)

Field Name	Data Type	Cost	Expose
	Select Data Type	$	Select
	Select Data Type	$	Select
	Select Data Type	$	Select
	Select Data Type	$	Select
	Select Data Type	$	Select

Figure 4.5b: Step 2 of the Data Service registration process

The information provided is used to register the Service with the local registry and additional information is stored in a database. **Figure 4.5c** shows the registry response received once the data service registration is complete.



SWAMY Logged in

Data Service Registration Complete

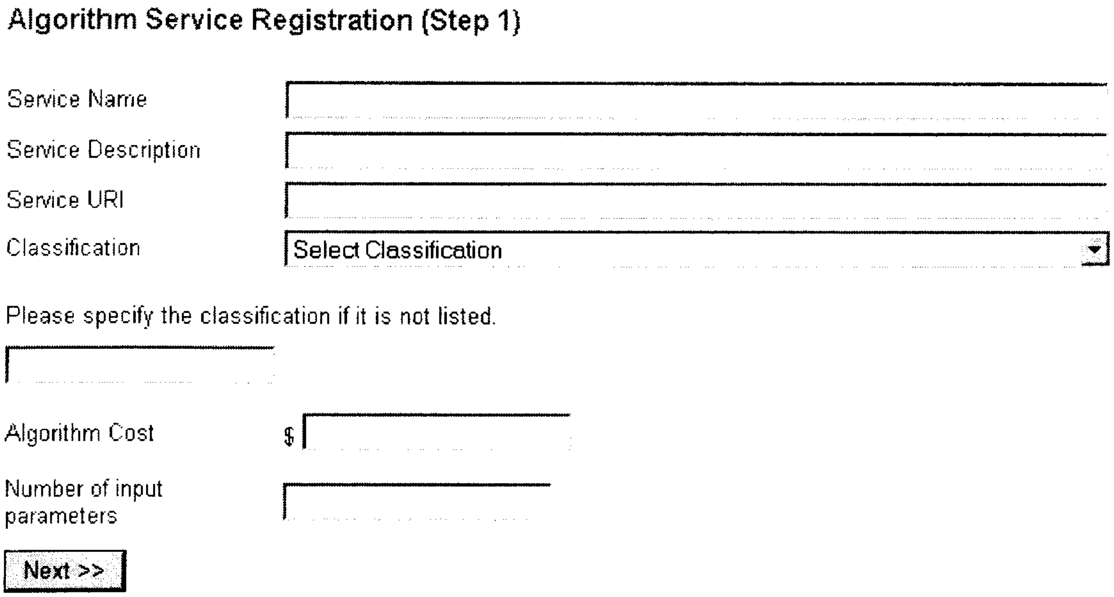
Service Information

Service Key	10044a6d-3721-0044-f251-1890cb770790
Service Name	Fractal Dimension Data Service
Service Description	Sample Data Sel for Fractal Dimension Algorithm
Service Binding URL	http://136.165.47.196/jaxrpc-fd_data_service
Service Binding Description	Column1:number:13.95:Y,Column2:number:12.45:Y,Column3:number:11.95:Y,Column4:number:9.75:Y

Figure 4.5c: Registry response for the Data Service registration process

4.2.2 Algorithm Services

The algorithm service registration is a two step process. The first step requests the user to provide the service information and the cost of using the algorithm and the number of input parameters accepted. **Figure 4.6a** shows the interface the client uses as the first step towards registering the algorithm service.



Algorithm Service Registration (Step 1)

Service Name

Service Description

Service URI

Classification: Select Classification

Please specify the classification if it is not listed.

Algorithm Cost $

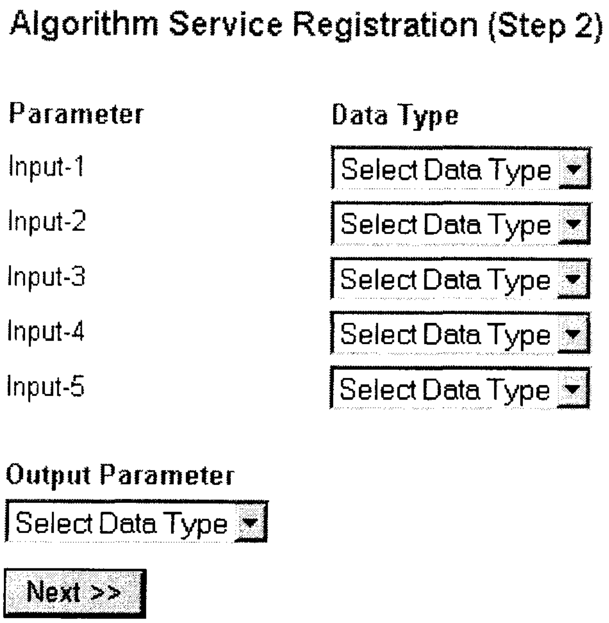
Number of input parameters

Figure 4.6a: Step 1 of the Algorithm Service registration process

The second step of the algorithm service registration process requires the following additional fields:

a) Number of input parameters
b) Output parameter type
c) Algorithm Cost

Based on the parameter Number of input parameters, Step 2 will dynamically allocate the required number of parameters to accept the data type. **Figure 4.6b** shows the interface the client uses as the second step to provide this information. Supposc the **Number of input parameters** was set to 5 in **Step 1.**



Algorithm Service Registration (Step 2)

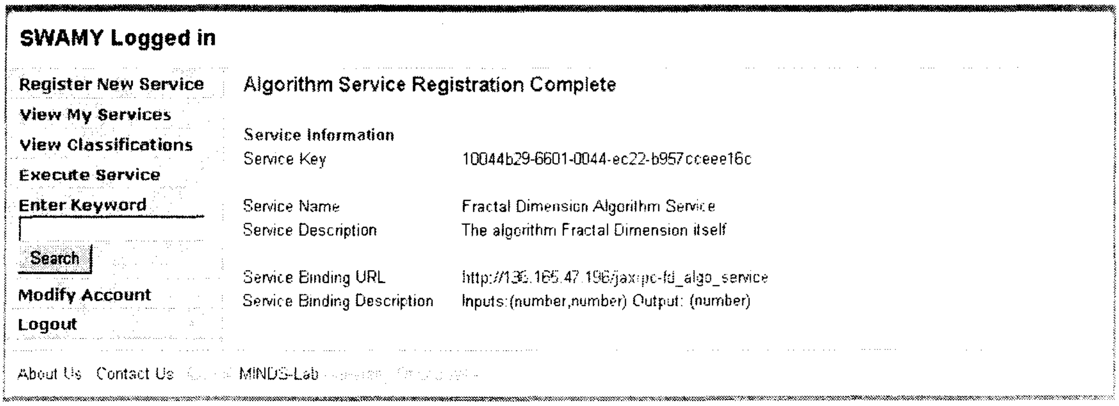
Parameter	Data Type
Input-1	Select Data Type ▾
Input-2	Select Data Type ▾
Input-3	Select Data Type ▾
Input-4	Select Data Type ▾
Input-5	Select Data Type ▾

Output Parameter

Select Data Type ▾

Figure 4.6b: Step 2 of the Algorithm Service registration process

The information provided is used to register the Service with the local registry and additional information is stored in a database. **Figure 4.6c** shows the registry response received once the algorithm service registration is complete.



SWAMY Logged in

Algorithm Service Registration Complete

Service Information	
Service Key	10044b29-6601-0044-ec22-b957cceee16c
Service Name	Fractal Dimension Algorithm Service
Service Description	The algorithm Fractal Dimension itself
Service Binding URL	http://136.165.47.196/jaxrpc-fd_algo_service
Service Binding Description	Inputs:(number,number) Output: (number)

Figure 4.6c: Registry response for the Algorithm Service registration process

4.2.3 Centralized Services

In the centralized approach, a collection of remote datasets is performed concurrently to form a single data resource bundle on which the data mining algorithm is executed. The algorithm also needs to be fetched from its remote location to the co-ordination site where the actual execution takes place. The following web services are used to achieve the execution of the centralized framework.

- **CopyRemoteData** (in **remote_location**, in **data_handler**, in **local_repository**): reads the data from the *"remote_location"* based on the specifications of the *data_handler"* and stores the data in the *"local_repository"*
- **CombineData** (in **local_repository**, out **combined_data**): combines the data read from the *"local_repository"* to create one large data source, *"combined_data"*.
- **FetchAlgorithm** (in **algorithm_location**): fetches the data mining algorithm from the *"algorithm_location"* and stores it locally for processing the data in the local repository
- **ExecuteAlgorithm** (in **local_repository**, out **results**): processes the data mining algorithm on the *"local_repository"* with the logic specified in the algorithm. Returns the intermediate *"results"* for performing other tasks or the final *"results"*.

4.2.4 DISTRIBUTED SERVICES

In the distributed approach, the datasets always reside on the remote sites and are never moved. The algorithm is propagated to the remote data sites where it processes the individual dataset and returns local models to a coordination site. The results are then processed to generate the global model and thereafter sent to the client. The number of intermediate results directly depends on the data mining algorithm itself.

- **SendAlgorithm** (in **algorithm_location**, in **remote_locations**): fetches the data mining algorithm from the *"algorithm_location"* and sends it to the *"remote-locations"* for processing the data.
- **ExecuteAlgorithm** (in **remote_locations**, out **results**): executes the algorithms concurrently on the *"remote_locations"* and stores the *"results*" on the remote location as the data model.
- **CopyRemoteModel** (in **remote_models**): reads the *"remote_models"* from the remote locations and copies it to the local coordination site for processing to get the final results.

The distributed approach may require multiple transfers of data between the coordination site and the data sites but it eventually proves to be the better of the two choices, especially for large real world data sets since the amount of data transferred is considerably less compared to the fetching of the actual datasets. The performance improvement can be measured in terms of execution time as well as the quality of service.

4.3 SELECTING DATASET AND ALGORITHM SERVICE INFORMATION

The first step in the exceution phase is to choose the desired datasets and thc algorithm that can be applied on them. **Figure 4.8** shows the steps involved in the selection of dataset and algorithm information from the registry.

The client uses the web interface to query the registry for information about available datasets based on classifications or keyword search. The registry returns the appropriate information on all the datasets that match the search criteria. Once the client selects the datasets for processing, a search for algorithms that match the selected datasets is performed and the algorithm service information that can be applied on them is displayed. This step of the implementation is common for the centralized and distributed approach.

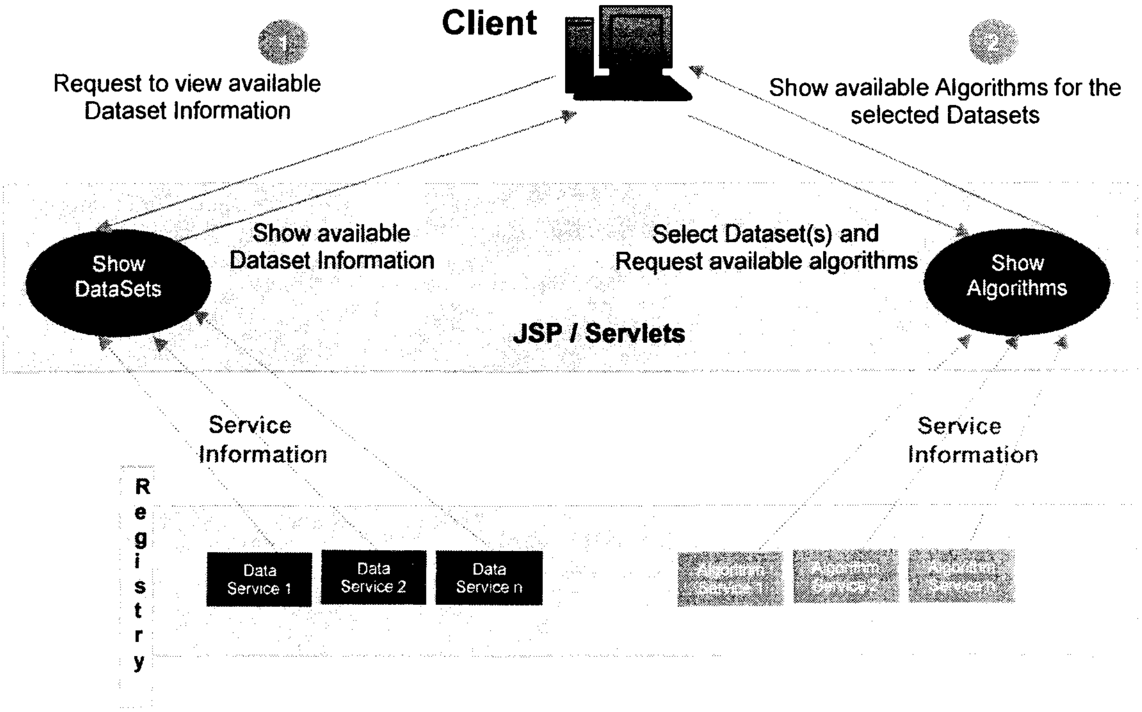


Figure 4.8: The client requests to view service information from the registry

4.3.1 EXECUTION - CENTRALIZED APPROACH

Step 1: The client invokes the data mining services from a URL. The client selects the required datasets located on remote sites.

Step 2: Servlet calls the service on the selected data sites to collect the data. The collected data is stored in a centralized location for processing as shown in **Figure 4.9**.

Step 3: The Apriori Service downloads the algorithm to the central site and is applied to build the global model for the datasets at the central site.

The datasets are now available as a single resource in the central site. The algorithm is applied on this combined data set and the results are returned to the client.

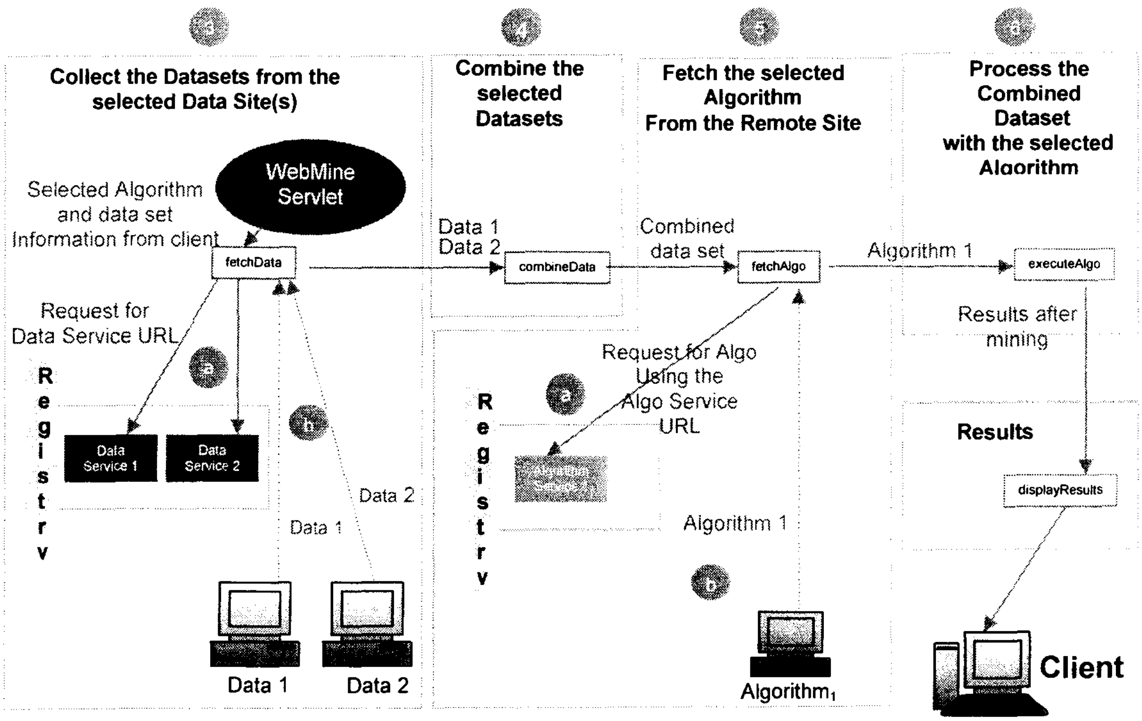

Collect the Datasets from the selected Data Site(s)
Combine the selected Datasets
Fetch the selected Algorithm From the Remote Site
Process the Combined Dataset with the selected Alaorithm
WebMine Servlet
Selected Algorithm and data set Information from client
fetchData
Request for Data Service URL
a
Registrv
Data Service 1
Data Service 2
b
Data 1
Data 2
Data 1
Data 2
Data 1
Data 2
combineData
Combined data set
fetchAlgo
Request for Algo Using the Algo Service URL
a
Registrv
Algorithm 1
b
Algorithm₁
Algorithm 1
executeAlgo
Results after mining
Results
displayResults
Client

Figure 4.9: Client interaction in Centralized Approach

3D, or in a higher dimensional space because only one dimension (one attribute) is assumed as independent variable and all others are dependent. Also, the dimensionality of rectangle and plane surface is always 2 regardless the dimensionality of the space for representation. These objects, representing an abstract relation in a data set, lead to the definition of the embedding and intrinsic dimensions of a data set [7]:

Definition 1 - *The embedding dimension E* of a data set is the dimension of its address space. In other words, it is the number of attributes of the data set.

Definition 2 - *The intrinsic dimension D* of a data set is the dimension of the spatial object represented by the data set, regardless of the space where it is embedded.

While the embedding dimension is given explicitly with the data set as the number of attributes, the intrinsic dimension is not computable directly. We can approximate it with the fractal dimension parameter [7, 16]. The fractal dimension characterizes multidimensional fractal sets. By embedding the data set in an n-dimensional grid with cell sides of size r, we can compute the frequency with which data points fall into the i-th cell, p_i. The generalized fractal dimension D_q represents a derivative or linear slope of discrete frequency function, as shown in the equation:

$$\mathbf{D_q = 1 / (q - 1) \;\; [\, \mathit{d} (\log \Sigma_i \, p_i^q) \, / \, \mathit{d} (\log r) \,]}$$

Where:

p_i - frequency of points falling in the i-th cell, and

r - size of the cell.

There is a family of fractal dimensions (i.e. Hausdorff fractal dimension for q=0, information fractal dimension for q=1, and correlation fractal dimension for q=2). Specifically, the correlation fractal dimension (q = 2) has gained attention in the literature [7]. Changes in the correlation fractal dimension means changes in the distribution of points in the multidimensional data set, and that is the parameter we are using in the simple and scalable analysis of large, distributed data sets. As many real data sets are self-similar, we can use their correlation fractal dimension as a measure of their intrinsic dimension *D*. The correlation fractal dimension is usually calculated by means of the box-counting algorithm. Let N(r) be $\Sigma_i p_i^2$, then the plot of N(r) for different values of r in a log-log scale is called the box-counting plot. The linear slope of the plot is the estimation of the correlation fractal dimension for the given data set.

5.3 APRIORI

Association analysis identifies relationships or affinities between items and/or between features. These relationships are then expressed as a collection of association rules. The approach has been particularly successful in mining very large transaction databases and is one of the core classes of techniques in data mining.

Each transaction is thought of as a basket of items, which we might represent as {A, B, C, D, E, F}. The algorithm searches for collections of items that often appear together e.g.{A, C, F}, and then from these *itemsets* it identifies rules like A, F → C which we read as indicating an association between A and F being in the transaction and C consequently appearing in the transaction.

The basis of an association analysis algorithm is the generation of frequent itemsets. However, naïve approaches will be quite expensive in computational time with even moderately sized databases. The *apriori* algorithm takes advantage of the simple *apriori* observation that all subsets of a frequent itemset must also be frequent. The observation allows the algorithm to consider a significantly reduced search space by starting with frequent individual items (eliminating rare items). We can then combine these into itemsets containing just two items and retain only those that are frequent enough. Similarly for itemsets containing three items, and so on.

Suppose we have a rule of the form A → C. We call A the *antecedent* and C the *consequent*, and both are non-empty sets of items. The concept of *frequent enough* is a parameter of the algorithm, used to control the number of association rules discovered. This *support* specifies how frequently the items must appear in the whole data set before the items can be considered as a candidate association rule. For example, the user may choose to consider only sets of items that occur in at least 5% of all transactions. Formally we define *support* for a collection of items I as the proportion of all baskets in which all items in I appear. Then we can define the support for an association rule as:

$$\mathbf{support\ (A \rightarrow C) = support\ (A\ U\ C)}$$

A second parameter, the *confidence*, calculates the proportion of transactions containing A that also contain C the use specifies a minimal probability for the association rule. For example, the user may choose to only generate rules which are true at least 90% of the time (that is, when A appears in the basket, C also appears in the same basket at least 90% of the time).

Formally:

confidence (A → C) = support (A → C) / support (A)

The Apriori algorithm is a breadth-first or generate-and-test type of search algorithm. Only after exploring all possibilities of associations containing k items does it then consider those containing $k+1$ items. For each k, all candidates are tested to determine whether they have enough support. The *apriori* algorithm uses a simple two step generate and merge process: generate frequent itemsets of size k then combine them to generate candidate frequent itemsets of size $k+1$. The algorithm is reasonably efficient even though the number of possible items is generally large and the baskets are generally small. The input data to the algorithm consists of records or transactions, each transaction representing a basket of items.

The two primary tuning parameters are minsup (**minimum support** expressed as a percentage of the total number of transactions in data) and mincon (**minimum confidence** also expressed as a percentage of the total number of transactions in data). Typically they have quite small values because of the size of the databases we are dealing with. Thus a support of 0.1% or smaller is not unusual.

6. Results and Discussion

In order to evaluate the effectiveness of the proposed service oriented framework three core data mining services including Normalization service (used by the Fractal Dimension algorithm), Fractal Dimension service and Apriori service were implemented. The services were implemented both in the centralized and distributed service oriented framework and their execution time was measured in a local and a wide area networks. The execution time for both implementations in a local area network is shown in the graphs included in this chapter. It is clear that as the number of records increase, the execution time of the centralized algorithm increases dramatically due to the overhead of combining all the data at the central site before performing the algorithm. The results illustrate that the distributed approach is more consistent and requires a shorter execution time. The performance results for the centralized and the distributed services in a wide area network are shown in various charts. It is interesting to observe that the increase in execution time for smaller number of records is much more significant here compared to the local area network. This shows the impact of additional communication delay when data sets are located on distant sites and are not on the same local area network. The distributed approach is consistent over time since it is dependent on the algorithms complexity rather than the dataset size. The apriori algorithm, whose complexity is higher than the fractal dimension algorithm showed similar results.

6.1 NORMALIZATION

Sample Input (Abridged)

```
1000   0       0       1000  1000000
750    433.012702  324759.5264 1183.012702 750000
625    649.519053  405949.408  1274.519053 812500
812.5 324.759526  263867.1152 1137.259526 765625
656.25      595.392465  390726.3052 1251.642465 785156.25
328.125     297.696233  97681.57631 625.821233  196289.0625
164.0625    148.848116  24420.39408 312.910616  49072.26563
332.03125   507.43676   168484.8617 839.46801   367736.8164
666.015625  253.71838   168980.4054 919.734005  507949.8291
833.007813  126.85919   105674.6964 959.867003  709995.2698
916.503906  63.429595   58133.47159 979.933501  844002.7237
708.251953  464.727499  329144.1591 1172.979453 717592.4778
354.125977  232.36375   82286.03978 586.489726  179398.1195
427.062988  549.194577  234540.6771 976.257565  483997.4791
713.531494  274.597288  195933.8135 988.128783  584530.8639
606.765747  570.311346  346045.39   1177.077093 693419.7033
.
.
.
587.075298  37.483753   22005.78552 624.559051  346062.4371
543.537649  451.754578  245545.6215 995.292227  499515.375
271.768824  225.877289  61386.40537 497.646114  124878.8437
635.884412  112.938645  71815.92365 748.823057  417104.1232
567.942206  489.482024  277997.5007 1057.42423  562151.0015
283.971103  244.741012  69499.37517 528.712115  140537.7504
641.985552  122.370506  78560.09682 764.356058  427119.9891
820.992776  61.185253   50232.65072 882.178029  677772.773
660.496388  463.605328  306209.6448 1124.101716 651185.3789
830.248194  231.802664  192453.7433 1062.050858 743044.5387
915.124097  115.901332  106064.1019 1031.025429 850885.2316
707.562048  490.963368  347387.0464 1198.525416 741689.0811
853.781024  245.481684  209587.6036 1099.262708 789203.2945
426.890512  122.740842  52396.90089 549.631354  197300.8236
213.445256  61.370421   13099.22522 274.815677  49325.20591
106.722628  30.68521    3274.806306 137.407839  12331.30148
553.361314  15.342605   8490.004201 568.703919  306444.1394
776.680657  7.671303    5958.152362 784.35196   603291.6919
388.340329  3.835651    1489.538091 392.17598   150822.923
444.170164  434.930528  193183.1639 879.100692  386451.6986
222.085082  217.465264  48295.79097 439.550346  96612.92465
361.042541  541.745334  195593.1119 902.787875  423839.7231
680.521271  270.872667  184334.6114 951.393937  536481.2013
590.260635  568.449035  335533.0887 1158.709671 671541.9233
```

The sample input shown here is an abridged version. The actual datasets were distributed over the network on multiple data sites. The results obtained for a test run with data of the nature above, the following results were obtained.

Results (Abridged)

```
Global Minimum and Maximum Values for each column
-------------------------------------------------
1.072523    0.0         0.0         1.40947     1.710467
1000.0      865.262736  432982.5973 1365.443297 1000000.0

Normalized Values
-----------------
0.4107373542594024      0.5495562795160058      0.4517744510282653
      0.6491557866621734      0.39533401203878155
0.20483183985938153     0.2747781397580029      0.11294361278016196
      0.32406123825549377     0.09883222015225111
0.6024159199296908      0.13738907045686063     0.16551352503977415
      0.5280737887448299      0.37754971381932634
0.3006711226945257      0.06869453465057115     0.04137838125994354
      0.2635202389302622      0.0943861455973873
0.6503355618477997      0.03434726790314474     0.044664000771838866
      0.4978032887156543      0.4243065234953063
0.32463094315304336     0.017173633373713207    0.011166000195269279
      0.2483849896487941      0.10607534797138221
0.161778634306202 0.00858681726471765     0.002791500047662539
      0.12367583974880411     0.026517554150401298
0.08035247988278132     0.004293408054498721    6.978750113382443E-4
      0.061321264432249306    0.006628105685156048
0.03963940317160783     0.0021467040272493605   1.744687534119515E-4
      0.030143977507091545    0.0016557435690947352
.
.
.
0.8536240324081105      0.2837076806691465      0.48405549069858655
      0.8048577801142757      0.7892029339385748
0.42627517893373557     0.14185384033457324     0.12101387265155102
      0.4019122349815449      0.1972994506071994
0.21260075219654812     0.07092692016728662     0.030253468157113852
      0.20043946241517954     0.04932357980935559
0.10576353882795438     0.035463459505784146    0.007563367041588024
      0.09970307649855666     0.012329612102394624
0.5528817694139772      0.017731729752892073    0.01960818807486053
      0.4158947071332418      0.3064429530935586
0.7764408847069886      0.0088658654543052      0.013760720174792114
      0.5739905231837039      0.6032910133423697
0.3876836055837115      0.004432932149293437    0.003440180044852809
      0.28647860651625906     0.1508214705081482
0.44357338365615906     0.5026571813442802      0.4461684259474971
      0.6434526802977885      0.3864506491440825
0.22124985455775986     0.2513285906721401      0.1115421064753264 6
      0.32120968507330133     0.09661137943357635
0.36035650864372015     0.6261050100278442      0.4517343494165417
      0.6608182195763096      0.423838737595174
0.6801782548223968      0.3130525050139221      0.4257321484731183
      0.6964522786721183      0.5364804084650348
0.5898207082755017      0.6569669666208762      0.7749343525405472
      0.848439516742278 0.671541361482338
```

Table 6.1 presents a summary of the total execution time for various sizes of datasets in a Local Area Network and **Figure 6.1** shows the difference in the execution times for both the approaches.

Number of Records	Centralized Approach (in ms)	Distributed Approach (in ms)
10	2	10093
100	953	10572
1000	1584	11174
10000	11280	12045
25000	19985	12986
50000	41063	14193

Table 6.1: Execution Times for the Normalization algorithm in a Local Area Network

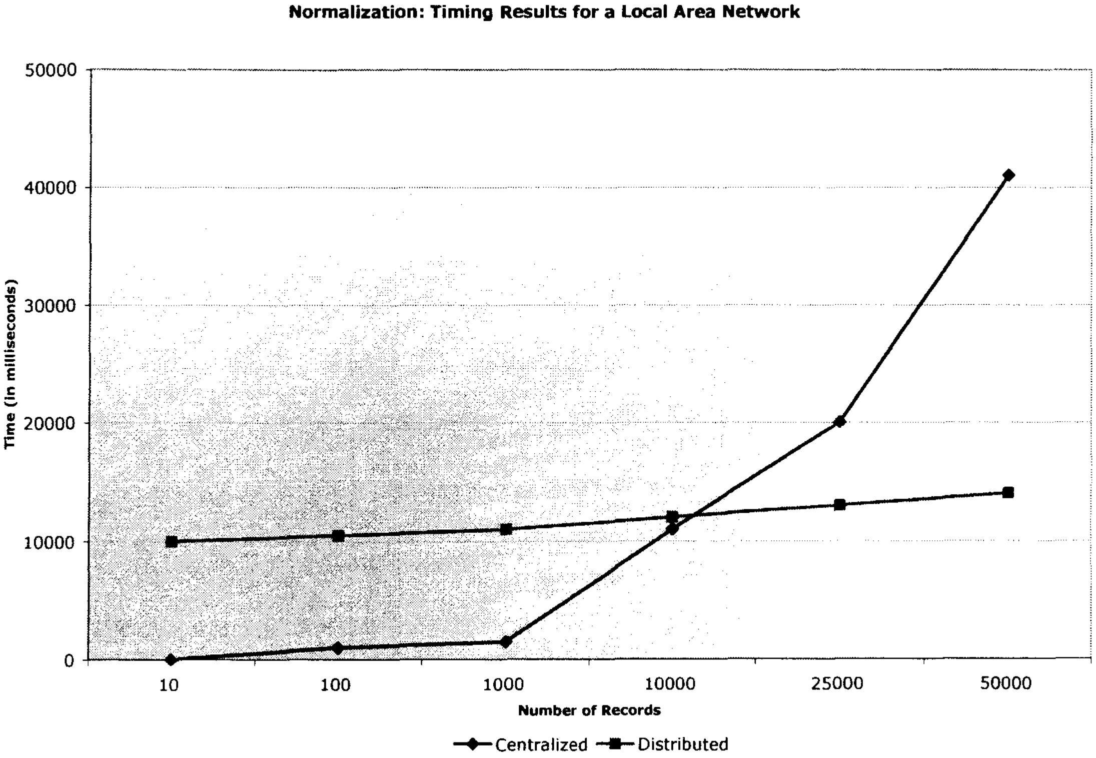


Figure 6.1: Timing Results the Normalization algorithm in a Local Area Network

Table 6.2 presents a summary of the total execution time for various sizes of datasets in a Wide Area Network and **Figure 6.2** shows the difference in the execution times for both the approaches.

Number of Records	Centralized Approach (in ms)	Distributed Approach (in ms)
10	3	20675
100	1121	22128
1000	14752	23284
10000	58163	24117
25000	70034	25283
50000	95178	26131

Table 6.2: Execution Times for the Normalization algorithm in a Wide Area Network

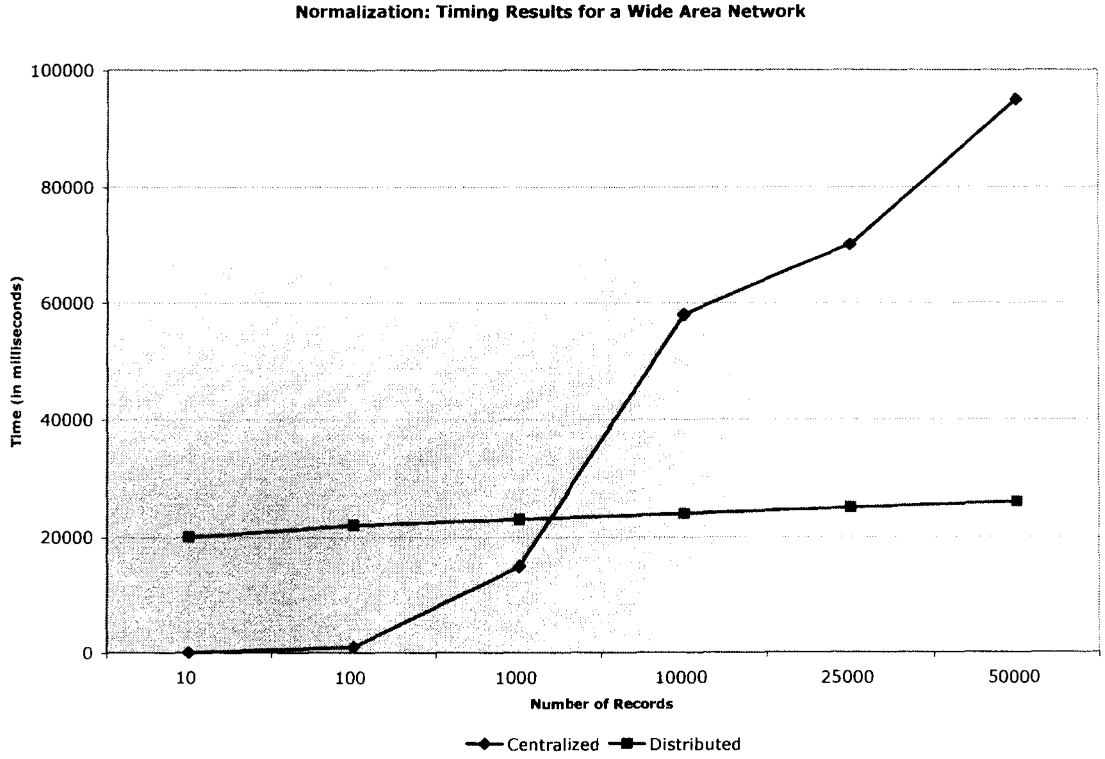


Figure 6.2: Timing Results the Normalization algorithm in a Wide Area Network

6.2 Fractal Dimension

Sample Input (Abridged)

```
0.904760414 0.750513162 0.001091628 0.932206113 0.407564752
0.739706064 0.864580806 0.259423693 0.011418937 0.356706385
0.424688613 0.078193912 0.153734069 0.033618253 0.376185509
0.729208003 0.763120257 0.81536836  0.943994791 0.184170159
0.94646033  0.398229453 0.222784329 0.176225524 0.922198721
0.124967392 0.516087951 0.711096562 0.890662873 0.411399253
0.563390336 0.267818823 0.459989708 0.77024769  0.813885662
0.382414158 0.900767957 0.653436678 0.612937901 0.874448675
0.171765921 0.124435936 0.296652074 0.631348493 0.684876036
0.378876905 0.161410775 0.42654536  0.313310403 0.232790797
0.449745042 0.157385243 0.891801018 0.589122929 0.987613927
0.567700996 0.602806213 0.371143355 0.21021309  0.714086604
0.255867668 0.087695473 0.468571623 0.053235229 0.034510664
0.14056215  0.672202741 0.914448435 0.009408939 0.616517722
0.031873775 0.243668039 0.275138193 0.343516135 0.883287067
0.97361543  0.088462958 0.006037269 0.503342929 0.542236925
0.520164863 0.750445934 0.340843093 0.631344558 0.646232528
0.860985541 0.950320833 0.312228982 0.612159987 0.234560139
.
.
.
0.397552305 0.572512484 0.856433221 0.241987076 0.766401098
0.036515194 0.827046468 0.634690871 0.5103693   0.548223592
0.315227562 0.061213551 0.389608188 0.553344671 0.609341084
0.55011439  0.884756061 0.40060435  0.546643909 0.801152882
0.333780101 0.265699934 0.650382835 0.621150656 0.53191915
0.189299685 0.323530236 0.601776414 0.257489409 0.014817414
0.109534611 0.950736971 0.399120502 0.973776989 0.675134591
0.708144024 0.893788511 0.10829746  0.800678545 0.675315414
0.48400863  0.660084574 0.901932356 0.088998619 0.266764217
0.102632789 0.167947575 0.624461557 0.048280511 0.374226558
0.4903537   0.975016942 0.852711187 0.687892242 0.001038837
0.98228721  0.254021122 0.952769735 0.362897777 0.23039182
0.889392779 0.937807216 0.415990575 0.654767107 0.679089268
0.547029756 0.590564596 0.146221548 0.253154055 1
0.437302612 0.960276301 0.08488191  1     0.836566837
0.134236786 0.550800623 1     0.624244059 0.912227382
0.196445825 1     0.505775257 0.430126011 0.478878893
1     0.280940063 0.323681051 0.082927406 0.641381264
0.216719789 0.616373528 0.615742464 0.418062896 0.007025302
```

The sample input shown here is an abridged version. The actual datasets were distributed over the network on multiple data sites. The results obtained for a test run with data of the nature above, the following results were obtained.

Results (Abridged)

r = 1/2		r = 1/4		r = 1/8		r = 1/16	
1.1.1.1.1	285	1.1.1.1.1	5	1.1.1.1.3	2	1.1.1.4.12	1
1.1.1.1.2	313	1.1.1.1.2	10	1.1.1.1.5	1	1.1.1.8.10	1
1.1.1.2.1	339	1.1.1.1.3	12	1.1.1.2.1	1	1.1.1.9.10	1
1.1.1.2.2	314	1.1.1.1.4	9	1.1.1.2.6	1	1.1.10.13.4	1
1.1.2.1.1	306	1.1.1.2.1	13	1.1.1.4.2	1	1.1.10.3.4	1
.		.		.		.	
.		.		.		.	
.		.		.		.	
2.2.1.2.2	323	1.1.2.1.3	10	8.8.8.7.8	1	9.9.8.5.11	1
2.2.2.1.1	341	1.1.2.1.4	6	8.8.8.8.2	2	9.9.8.7.10	1
2.2.2.1.2	303	1.1.2.2.1	7	8.8.8.8.6	1	9.9.8.7.15	1
2.2.2.2.1	317	1.1.2.2.2	6	8.8.8.8.7	1	9.9.9.15.10	1
2.2.2.2.2	324	1.1.2.2.3	11	8.8.8.8.8	1	9.9.9.4.2	1

r = 1/32		r = 1/64		r = 1/128	
1.1.26.31.24	1	1.10.13.13.31	1	1.103.51.112.13	1
1.1.28.3.24	1	1.10.39.45.3	1	1.104.115.60.44	1
1.1.6.28.10	1	1.10.52.52.24	1	1.104.121.123.18	1
1.10.13.30.10	1	1.10.9.41.21	1	1.105.26.21.119	1
1.10.19.29.5	1	1.11.25.19.5	1	1.105.31.32.87	1
.		.		.	
.		.		.	
.		.		.	
9.9.25.2.3	1	9.64.6.20.48	1	99.86.24.87.66	1
9.9.29.8.10	1	9.7.11.49.45	1	99.88.62.13.61	1
9.9.4.25.16	1	9.7.11.63.45	1	99.89.54.7.38	1
9.9.7.29.18	1	9.8.30.48.19	1	99.9.102.77.9	1
9.9.9.30.28	1	9.9.5.23.55	1	99.97.94.103.11	1

```
FD = 2.788957407590457
R = 0.9404928239071236

FD = 1.0678309573125557
R = 0.8293589297488905

FD = 0.11581529365873741
R = 0.7949366774769094

FD = 0.004621321783703003
R = 0.793912851301406
```

The FD: 2.788957407590457
The R: 0.9404928239071236

Table 6.3 presents a summary of the total execution time for various sizes of datasets in a Local Area Network and **Figure 6.3** shows the difference in the execution times for both the approaches.

Number of Records	Centralized Approach (in ms)	Distributed Approach (in ms)
10	28	20048
100	752	21613
1000	1759	21934
10000	58125	25912
20000	80474	30945
30000	92087	33117
40000	112790	42283
50000	146183	51743
60000	165251	56276

Table 6.3: Execution Times for the Fractal Dimension algorithm in a Local Area Network

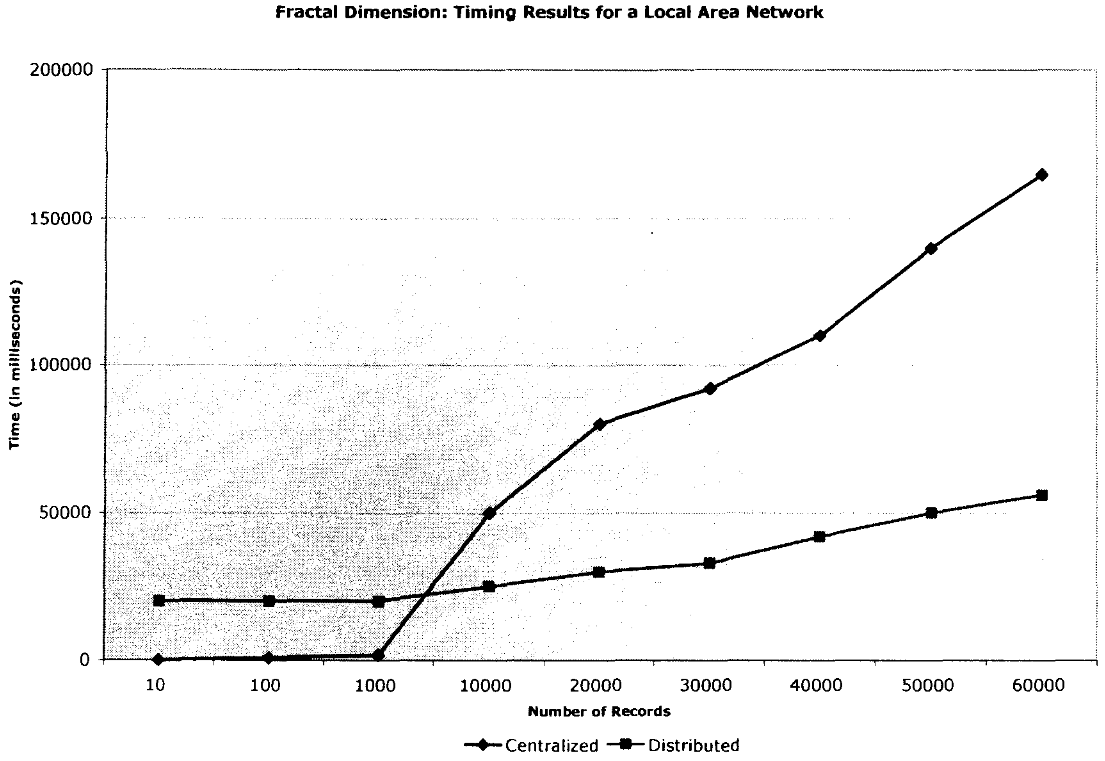


Figure 6.3: Timing Results the Fractal Dimension algorithm in a Local Area Network

Table 6.4 presents a summary of the total execution time for various sizes of datasets in a Wide Area Network and **Figure 6.4** shows the difference in the execution times for both the approaches.

Number of Records	Centralized Approach (in ms)	Distributed Approach (in ms)
10	27	103425
100	7512	121533
1000	17509	177342
10000	165433	185630
20000	189342	192153
30000	240342	213436
40000	453262	235121
50000	648211	247345
60000	812424	245841

Table 6.4: Execution Times for the Fractal Dimension algorithm in a Wide Area Network

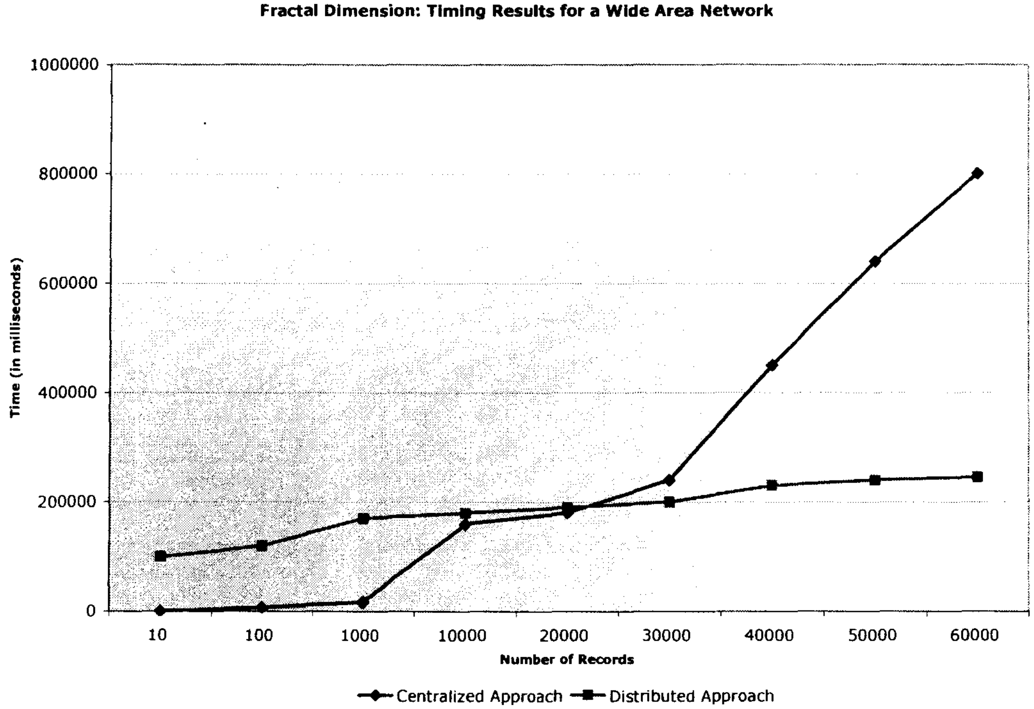


Figure 6.4: Timing Results the Fractal Dimension algorithm in a Wide Area Network

6.3 APRIORI

Sample Input (For generating Maximal Frequent Itemsets)

Site 1	Site 2
1 3 4	1 2 3 4
2 3 5	2 3 5
1 2 3 5	1 2 3 5
2 5	2 5
	3 4 5
	1 2 4
	1 3

Results (Maximal Frequent Itemsets)

Site 1	Site 2
Frequent Itemsets	Frequent Itemsets
1 (2)	1 (4)
1 3 (2)	1 2 (3)
2 (3)	1 3 (3)
2 3 (2)	2 (5)
2 3 5 (2)	2 3 (3)
2 5 (3)	2 5 (3)
3 (3)	3 (5)
3 5 (2)	3 5 (3)
5 (3)	5 (4)
Maximal Frequent Itemsets	Maximal Frequent Itemsets
1 3 (3)	1 2 (3)
2 3 5 (2)	1 3 (3)
	2 3 (3)
	2 5 (3)
	3 5 (3)

Similarity Matrix Generation

Iteration 1 (FI 1 to FI 2)

The first element {1 3} from FI 1 is searched in FI 2. The second element is a perfect match. Hence we consider the existence of {1 3} as a value of 1.

{1 3} ←→ {1 3} → 1

The second element {2 3 5} from FI 1 is searched in FI 2.

```
{2 3 5} ←→ {2 3 5} → 0
```

Since there exists no perfect match, a subset search is initiated. The subsets are created in a tree fashion to enable search based on the size of the subset. The results of the subset search yield the following values:

```
{2 3 5} ←→ {2 3 5} → 0
{2 3} ←→ {2 3} → 1
{2 5} ←→ {2 5} → 1
{3 5} ←→ {3 5} → 1
```

Note that {2 3 5} is a subset of itself. Therefore the existence of {2 3 5} is considered as ¾ since one of the four subsets failed to match. The similarity matrix entry for S_1S_2 is calculated as (1 + ¾) / 2 = 0.875.

To			
From		S_1	S_2
	S_1	1	0.875
	S_2		1

Iteration 1 (FI 2 to FI 1)

The first element {1 2} from FI 2 is searched in FI 1.

```
{1 2} ←→ {1 2} → 0
```

The second element {1 3} from FI 2 is searched in FI 1.

```
{1 3} ←→ {1 3} → 1
```

The third element {2 3} from FI 2 is searched in FI 1.

{2 3} ←→ {2 3} → 1 (A match based on the subset of {2 3 5} in FI 1)

The fourth element {2 5} from FI 2 is searched in FI 1.

{2 5} ←→ {2 5} → 1 (A match based on the subset of {2 3 5} in FI 1)

The fifth element {3 5} from FI 2 is searched in FI 1.

{3 5} ←→ {3 5} → 1 (A match based on the subset of {2 3 5} in FI 1)

The similarity matrix entry for S_2S_1 is calculated as (1 + 1 + 1 + 1) / 5 = 0.8

To			
From		S_1	S_2
	S_1	1	0.875
	S_2	0.8	1

Table 6.5 presents a summary of the total execution time for various sizes of datasets distributed over several sites and the algorithm execution performed using the centralized approach.

Number of Sites	10,000 Records	25,000 Records	50,000 Records	75,000 Records	100,000 Records
2	18,334	27,453	49,565	81,532	105,648
3	26,321	40,453	67,453	106,732	126,321
4	31,743	58,454	91,343	117,321	156,896
5	44,982	71,454	100,565	149,697	178,564
6	53,683	89,454	121,538	163,998	184,232
7	70,564	110,768	149,576	189,432	197,546

Table 6.5: Execution Times in Milliseconds for the Centralized approach

The results were plotted to measure performance changes as well as to determine the quality of the results. **Figure 6.5** shows the centralized approach implementation of the Apriori algorithm.

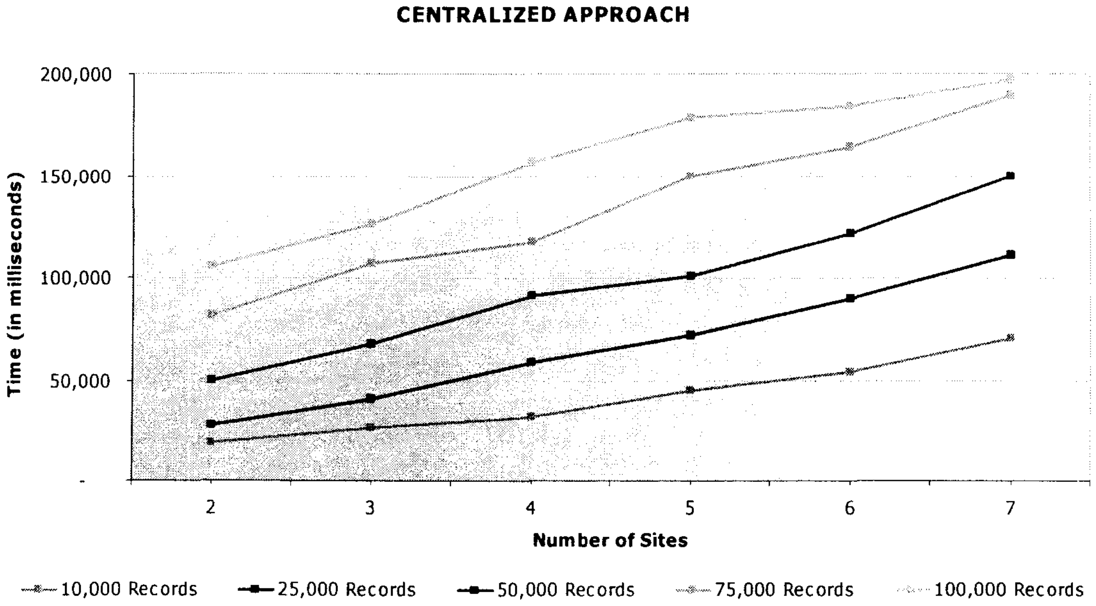


Figure 6.5: Timing Results for the Centralized approach

Table 6.6 presents a summary of the total execution time for various sizes of datasets distributed over several sites and the algorithm execution performed using the distributed approach.

Number of Sites	10,000 Records	25,000 Records	50,000 Records	75,000 Records	100,000 Records
2	29,528	33,632	40,985	48,345	57,345
3	32,743	37,458	47,684	53,896	60,684
4	34,743	40,975	51,564	60,657	69,489
5	36,965	44,953	55,907	67,343	76,984
6	39,564	47,932	59,567	71,569	83,673
7	43,742	50,785	61,843	75,311	89,565

Table 6.6: Execution Times in Milliseconds for the Distributed approach

The results were plotted to measure performance changes as well as to determine the quality of the results. **Figure 6.6** shows the distributed approach implementation of the Apriori algorithm.

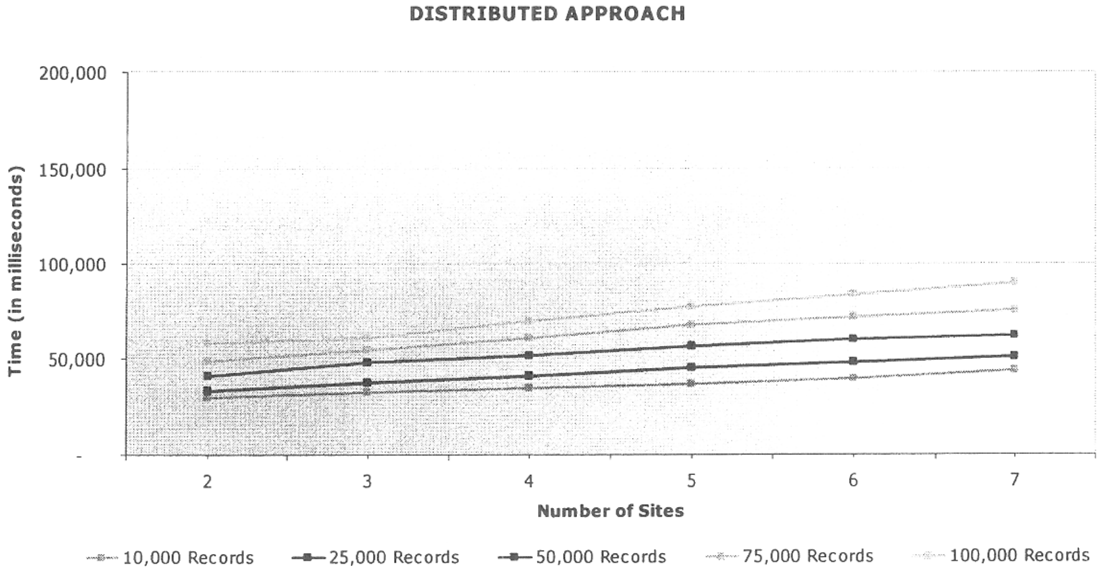


Figure 6.6: Timing Results for the Distributed approach

It is clear that as the number of records increase, the execution time of the centralized algorithm increases dramatically due to the overhead of combining all the data at the central site before executing the algorithm. **Table 6.7** shows the results obtained when a comparison of both the approaches was performed. The results illustrate that the distributed approach is more consistent and requires a shorter execution time in the long run.

Number of Sites	10,000 Records	25,000 Records	50,000 Records	75,000 Records	100,000 Records
2	-61.1%	-22.5%	17.3%	40.7%	45.7%
3	-24.4%	7.4%	29.3%	49.5%	52.0%
4	-9.5%	29.9%	43.5%	48.3%	55.7%
5	17.8%	37.1%	44.4%	55.0%	56.9%
6	26.3%	46.4%	51.0%	56.4%	54.6%
7	38.0%	54.2%	58.7%	60.2%	54.7%

Table 6.7: Percentage Improvement of Distributed approach Vs Centralized approach

The percentage of improvement of the distributed framework over the centralized framework is shown in **Figure 6.7**.

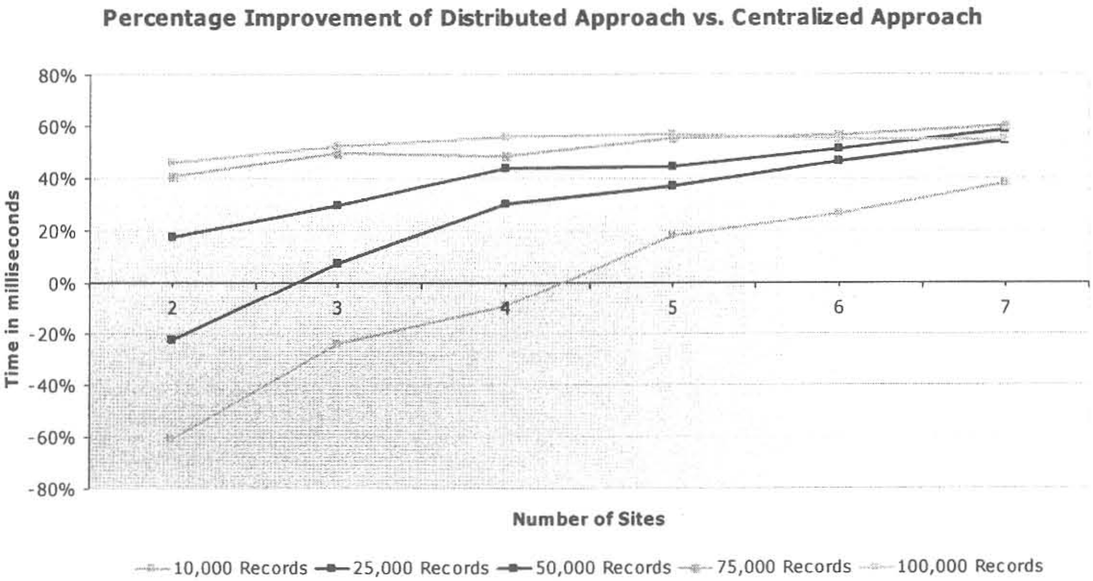


Figure 6.7: Percentage improvement

7. CONCLUSION

This project describes a service oriented distributed data mining framework that provides improved performance and hierarchical structure for developing composite service using core data mining services. This framework neither has any overhead of downloading data to a particular location for mining nor fixed location for data mining tools. In this framework a user will be able to mine data available at multiple sites, using distributed data mining algorithms and executing them on more than one site. The framework allows users ability to discover and use new algorithms and data sets on the Internet and for their data-mining tasks dynamically. The proposed framework alleviates the major limitations of existing approaches:

- The proposed framework allows coordination between different local models. This results in better accuracy in global learning models.
- Aggregate component development allows creation of new data mining components using the existing components. Moreover, the proposed middleware supports integration of heterogeneous technologies, data and algorithms.
- The proposed service oriented approach is dynamic where datasets and algorithms are searched based on the client's requirements. Thus, the location and data format for

datasets may not be known beforehand. The proposed framework improves the flexibility and performance of a data mining system.

- The proposed service oriented framework is built on open standards and is not tightly coupled but can execute components on heterogeneous platforms and languages. Moreover, Web provides a flexible platform for performing distributed data mining.

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 Web, http://www.globus.org/research/papers/osga.pdf, *2002.*

[4] Bailey S., Grossman R., Sivakumar H., Papyrus: A system for Data Mining over Local and Wide Area Clusters and Super Clusters, http://citeseer.nj.nec.com/408839.html, May 1999.

[5] Ning Chen, Nuno C. Marques, Narasimha Bolloju, AWeb Service-based approach for data mining in distributed environments, CENTRIA, Department of Information FCT, New University of Lisbon, Portugal, http://centria.di.fct.unl.pt/~nchen/papers/iceis.pdf

[6] Moore, Benjamin Seth " Distributed Mathematics as a Web Service," *M. Eng. Thesis*, University of Louisville, April 2002. (Kumar).

[7] Business Process Execution Language for Web Service, Version 1.1, www.ibm.com/developerworks/libraray/ws-bpel/.

[8] Box, Don, et al. *Simple Object Access Protocol (SOAP) 1.1.* NC: World Wide Web Consortium, May 2000. http://www.w3.org/ TR/SOAP/, March 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] Christensen, Erik, et al. *Web Services Description Language (WSDL).* NC: World Wide Web Consortium, http://www. w3.org/TR/wsdl.html, March 2001.

[11] Paul Donachy, et al., Web Enabled Distributed Data Mining and Conversion of Unstructured Data, Belfast e-Science Centre, ttp://www.nesc.ac.uk/events/ ahm2003/AHMCD/pdf/070.pdf

[12] Kassie, Fitsum"Smart Card Application Development Using Web Services", M.Engg. Thesis, University of Louisville, Dec. 2003. (Kumar)

[13] Elmaghraby, A. S. , Kantardzic, M and Wachowiak, M. P.," Data Mining From Multimedia Patient Records", book chapter, accepted for publication in the book "*Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques*", editors Triantaphyllou E. and Felici G., Kluwer Academic Publishers, Spring 2004.

[14] Goldszmidt M. and Jensen D., ed., Recommendations Report – DARPA *Workshop on Knowledge Discovery, Data Mining and Machine Learning*, Carnegie Mellon University, June 1998.

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

[16] 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.

[17] Grossman, Robert," Integrating Distributed Bioinformatics Data Using Data Webs", *Practical Innovation - BioCon 2003,* San Diego, CA, February 2003.

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

[19] Guo Y., Sutiwaraphun J., Probing Knowledge in Distributed Data Mining, *Proceedings of the PAKDD Conference,* Beijing, China, 1999.

[20] IEEE IPDSP, *Workshop on High Performance Data Mining,* http://www.cs.rpi.edu/zaki/HPMD/, 2000.

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

[22] Kantardzic M., Kumar A., Toward Autonomic Distributed Data Mining With Intelligent Web Services, *Proceedings of The 2003 International Conference on Information and Knowledge Engineering*, IKE'03, June 23-26, 2003, Las Vegas, pp. 544-550.

[23] Kantardzic M., Djulbegovic B., Hamdan H., A Data Mining Approach to Improving Polycythemia Vera Diagnostics, Special Issue of *Computers and Industrial Engineering Journal* on Data Mining and Knowledge Discovery, Vol 43., December 2002, pp. 765-773 .

[24] Kantardzic M., H. Hamdan, B. Djulbegovic, Artificial Neural Networks (ANN) Approach in Diagnostics of Polycythemia Vera, *International Journal of Computers and Their Applications*, Vol. 8, No. 2, June 2001, pp. 74-79.

[25] Kantardzic M., Sadeghian P., Knowledge Discovery in Semantically Distributed Data: The Fractal Dimension Approach, submitted for *The Nineteenth National Conference on Artificial Intelligence AAAI 2004*, San Jose, CA, July 2004.

[26] Kantardzic M., Soliman M., Global Model of Distributed Data Mining is not Summing of Local Models, in *"Data Mining IV"*, editors: N. Ebecken, C. Breibbia, and A. Zanasi, WIT Press, Southampton, UK, 2004., pp. 89-98.

[27] Kantardzic M.,. Emam A. Z., Elmaghraby A. S., Min H., A Novel Approach for Profile Association Rule Mining from Mixed Databases, *The 6th World Multiconference on Systems, Cybernetics, and Informatics SCI 2002*, Orlando, Florida, July 2002.

[28] Kantardzic, M., Zurada, J., "New Generation of Data Mining Applications", accepted for publications by IEEE Press and John Wiley, Spring 2004.

[29] Kantardzic, M., Sagedhian, P., Chun, S., "The Time Diversification Monitoring of a Stock Portfolio: An Approach Based on a Fractal Dimension", accepted for the *ACM Symposium on Applied Computing SAC 2004*, Nicosia, Cyprus, March 2004.

[30] Kantardzic, M., and Badia, A. "Efficient Implementation of Strong Negative Associations Rules", *Proceedings of the 2003 International Conference on Machine Learning and Applications*, Los Angeles, CA, June 2003, pp. 152-158 (The Best Paper Award).

[31] Kantardzic, M., Badia, A., "A Data Mining Framework for Semantically Distributed Databases", *Proceedings of The Seventh IASTED International Conference in Artificial Intelligence and Soft Computing (ASC 2003),* Banff, Canada, July 2003.

[32] Kubera, F. etal, "Unraveling the Web Service Web: An introduction to SOAP, WSDL, UDDI", IEEE Internet Computing, Vol. 6, No. 2, March/April 2002.

[33] 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.

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

[35] Kargupta H., Chan P., *Advances in Distributed and Parallel Knowledge Discovery*, AAAI Press/ MIT Press, 1999.

[36] 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.

[37] Kargupta H., Park B., Hershberger D., Johnson E., Collective Data Mining: A New Perspective Toward Distributed Data Mining, in "*Advances in Distributed and Parallel Knowledge Discovery*", eds.: Hillol Kargupta and Philip Chan, MIT/AAAI Press, 1999.

[38] Krishnaswamy, S., Zaslasvky, A., and Loke, S, W., Internet Delivery of Distributed Data Mining Services: Architectures, Issues and Prospects, Chapter 7 in the book *Architectural Issues of Web-enabled Electronic Business,* Murthy, V.K. and Shi, N. (eds.), 2003, pp. 113 - 127, Idea Group Publishing.

[39] Krishnaswamy, S., Loke, S, W., and Zaslavsky, A., Towards Anytime Anywhere Data Mining E-Services, *Proceedings of the Australian Data Mining Workshop*

(ADM'02) at the 15th Australian Joint Conference on Artificial Intelligence, (eds) S.J. Simoff, G.J. Williams, and M. Hegland. Canberra, Australia, December 2002, pp. 47 - 56.

[40] Kumar, A., Kantardzic, M., Ramaswamy, P., Sadeghian, P., "An Extensible Service Oriented Distributed Data Mining Framework", *Proceedings of International Conference on Machine Learning and Applications ICMLA04*, Louisville, December 2004, pp. 161-169.

[41] Little, M., "Transactions and Web Services", Communications of the ACM, Vol. 46, No. 10, pp.29-34, 2003.

[42] Kumar, A., Kantardzic, M., "Web Application Protocols and Services for Distributed Data Mining", *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - Workshop on Data Mining Standards, Services and Platforms (DM-SSP 03),* Washington, DC, August 2003.

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

[44] Papazoglou, M.P. and Georgakopoulos, D. ," Service Oriented Computing," *Communications of the ACM*, Vol. 46, No. 10, pp. 25-28, 2003.

[45] Park , B. H., Kargupta H, "Distributed Data Mining: Algoriths, Systems and Applications", http://citeseer.nj.nec.com/540909.html, 2000.

[46] Parsarthy, S. and Subramanian, R. "An Interactive Resource aware Framework for Distributed Daya mining", *Newsletter of the IEEE Technical Committee on Distributed Processing*, Spring 2001, pp. 24-32.

[47] 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.

[48] Software Architecture for DecisionCentre: A Web-based Distributed Data Mining System, Data Mining Group, Imperial College, *PCW*, 1997.

[49] Sayal, Mehmet and Scheuermann, Peter, " A Distributed Clustering Algorithm for Web-Based Access Patterns", in *Proceedings of the 2nd ACM-SIGMOD Workshop on Distributed and Parallel Knowledge Discovery*, Boston, Augusl 2000.

[50] Kantardzic, M., et al., "Data Mining Approach in a Selection of Laparoscopic Techniques", *Proceedings of the 10th International Conference on Intelligent Systems*, Arlington, VA, June 2001, pp. 1-4.

[51] Soliman, M., Kantardzic, M., "Towards Building a Distributed Mining Approach", *Proceedings of the 2nd IEEE International Symposium on Signal Processing and Information Technology,* Marrakesh, Marocco, December 2002, pp. 305-309.

[52] Steve J. , "A Wireless Printing Application Using Web Service Technology", *M.S. Thesis*, University of Louisville, May 2002, (Kumar).

[53] Subramonian R., Parthasarathy, "An Architecture for Distributed Data Mining", *Fourth International Conference on Knowledge Discovery and Data Mining*, New York, 1998, pp. 44-59.

APPENDIX

Publish.java – This program lets the algorithm/data service providers to publish their service with the registry.

```
package registry;

import java.net.PasswordAuthentication;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Properties;
import java.util.Set;

import javax.xml.registry.BulkResponse;
import javax.xml.registry.BusinessLifeCycleManager;
import javax.xml.registry.BusinessQueryManager;
import javax.xml.registry.Connection;
import javax.xml.registry.ConnectionFactory;
import javax.xml.registry.JAXRException;
import javax.xml.registry.RegistryService;
import javax.xml.registry.infomodel.EmailAddress;
import javax.xml.registry.infomodel.InternationalString;
import javax.xml.registry.infomodel.Organization;
import javax.xml.registry.infomodel.PersonName;
import javax.xml.registry.infomodel.Service;
import javax.xml.registry.infomodel.ServiceBinding;
import javax.xml.registry.infomodel.TelephoneNumber;
import javax.xml.registry.infomodel.User;

/**
 * @author swamy
 *
 * To change the template for this generated type comment go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
public class Publish {

        Connection connection = null;

        public void makeConnection(String queryUrl, String publishUrl) {

                //String httpProxyHost="";
                //String httpProxyPort="8080";
                //String httpsProxyHost="";
                //String httpsProxyPort="8080";


                Properties props = new Properties();
                props.setProperty("javax.xml.registry.queryManagerURL", queryUrl);
                props.setProperty("javax.xml.registry.lifeCycleManagerURL", publishUrl);

                //props.setProperty("com.sun.xml.registry.http.proxyHost",
                // httpProxyHost);
                //props.setProperty("com.sun.xml.registry.http.proxyPort",
                // httpProxyPort);
                //props.setProperty("com.sun.xml.registry.https.proxyHost",
    // httpsProxyHost);
    //props.setProperty("com.sun.xml.registry.https.proxyPort",
    // httpsProxyPort);

    try {

        // Create the connection, passing it the configuration properties

        ConnectionFactory factory = ConnectionFactory.newInstance();
        factory.setProperties(props);
        connection = factory.createConnection();
        System.out.println("Created connection to registry");
    } catch (Exception e) {
        e.printStackTrace();
        if (connection != null) {
            try {
                connection.close();
            } catch (JAXRException je) {
            }
        }
    }
}

public String executePublish(String username, String password,
        String orgName, String orgDescription, String personName,
        String phoneNumber, String email, String classificationScheme,
        String classificationName, String classificationValue,
        String serviceName, String serviceDescription,
        String svcbindingDescription, String svcbindingAccessURI) {

    RegistryService rs = null;
    BusinessLifeCycleManager blcm = null;
    BusinessQueryManager bqm = null;

    String id = null;

    try {
        rs = connection.getRegistryService();
        blcm = rs.getBusinessLifeCycleManager();
        bqm = rs.getBusinessQueryManager();
        System.out.println("Got registry service, query manager, and life
        cycle manager");

        // Get authorization from the registry
        PasswordAuthentication passwdAuth = new PasswordAuthentication(
                username, password.toCharArray());

        Set creds = new HashSet();
        creds.add(passwdAuth);
        connection.setCredentials(creds);
        System.out.println("Established security credentials");

        // Create organization name and description
        Organization org = blcm.createOrganization(orgName);
        InternationalString s = blcm
                .createInternationalString(orgDescription);
        org.setDescription(s);

        // Create primary contact, set name
        User primaryContact = blcm.createUser();
        PersonName pName = blcm.createPersonName(personName);
        primaryContact.setPersonName(pName);

        // Set primary contact phone number
        TelephoneNumber tNum = blcm.createTelephoneNumber();
        tNum.setNumber(phoneNumber);
        Collection phoneNums = new ArrayList();
        phoneNums.add(tNum);
        primaryContact.setTelephoneNumbers(phoneNums);

        // Set primary contact email address
```

```
EmailAddress emailAddress = blcm.createEmailAddress(email);
Collection emailAddresses = new ArrayList();
emailAddresses.add(emailAddress);
primaryContact.setEmailAddresses(emailAddresses);

// Set primary contact for organization
org.setPrimaryContact(primaryContact);

//                    // Set classification scheme to NAICS
//                    ClassificationScheme cScheme =
// bqm.findClassificationSchemeByName(null, classificationScheme);
//
//                    // Create and add classification
//                    Classification classification =
// blcm.createClassification(cScheme,
// classificationName,classificationValue);
//                    Collection classifications = new
                      ArrayList();
//                    classifications.add(classification);
//                    org.addClassifications(classifications);

// Create services and service
Collection services = new ArrayList();
Service service = blcm.createService(serviceName);
InternationalString is = blcm
               .createInternationalString(serviceDescription);
service.setDescription(is);

// Create service bindings
Collection serviceBindings = new ArrayList();
ServiceBinding binding = blcm.createServiceBinding();
is = blcm.createInternationalString(svcbindingDescription);
binding.setDescription(is);

// allow us to publish a fictitious URL without an error
binding.setValidateURI(false);
binding.setAccessURI(svcbindingAccessURI);
serviceBindings.add(binding);

// Add service bindings to service
service.addServiceBindings(serviceBindings);

// Add service to services, then add services to organization
services.add(service);
org.addServices(services);

// Add organization and submit to registry
// Retrieve key if successful
Collection orgs = new ArrayList();
orgs.add(org);

BulkResponse response = blcm.saveOrganizations(orgs);

Collection exceptions = response.getExceptions();

if (exceptions == null) {
       System.out.println("Organization saved");

       Collection keys = response.getCollection();
       Iterator keyIter = keys.iterator();
       if (keyIter.hasNext()) {
              javax.xml.registry.infomodel.Key orgKey =
       (javax.xml.registry.infomodel.Key) keyIter
                             .next();
              id = orgKey.getId();
              System.out.println("Organization key is " + id);
       }
} else {
       Iterator excIter = exceptions.iterator();
       Exception exception = null;
       while (excIter.hasNext()) {
```

```
                                        exception = (Exception) excIter.next();
                                        System.err.println("Exception on save: "
                                                       + exception.toString());
                                }
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        // At end, close connection to registry
                        if (connection != null) {
                                try {
                                        connection.close();
                                } catch (JAXRException je) {
                                }
                        }
                }

                return id;
        }

        public static void main(String[] args) {

                String username = "testuser";
                String password = "testuser";

                String orgName = "The Coffee Break by Swamy";
                String orgDescription = "Purveyor of the finest coffees. Established
                        1969";

                String personName = "Swamy";
                String phoneNumber = "(502) 314-8312";
                String email = "swamy@priest.com";

                String classificationScheme = "ntis-gov:naics";
                String classificationName = "Snack and Nonalcoholic Beverage";
                String classificationValue = "722213";

                String serviceName = "Published Using Program";
                String serviceDescription = "My Service Description";

                String svcbindingDescription = "My Binding Description";
                String svcbindingAccessURI = "http://Coffee.com:8080/sb/";

                //Additional parameters end here

                String queryURL = "http://localhost:8080/RegistryServer/";
                String publishURL = "http://localhost:8080/RegistryServer/";

                Publish p = new Publish();

                p.makeConnection(queryURL, publishURL);

                p.executePublish(username, password, orgName, orgDescription,
                               personName, phoneNumber, email, classificationScheme,
                               classificationName, classificationValue, serviceName,
                               serviceDescription, svcbindingDescription,
                               svcbindingAccessURI);

        }
}
```

QueryRegistry.java – This program lets the algorithm/data service providers to publish their service with the registry.

```
package registry;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.Properties;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.registry.BulkResponse;
import javax.xml.registry.BusinessQueryManager;
import javax.xml.registry.Connection;
import javax.xml.registry.ConnectionFactory;
import javax.xml.registry.FindQualifier;
import javax.xml.registry.JAXRException;
import javax.xml.registry.JAXRResponse;
import javax.xml.registry.RegistryService;
import javax.xml.registry.infomodel.Organization;
import javax.xml.registry.infomodel.RegistryObject;
import javax.xml.registry.infomodel.Service;

public class QueryRegistry extends HttpServlet {

        // edit these if behind firewall, otherwise leave blank
        String httpProxyHost = "";

        String httpProxyPort = "";

        PrintWriter out;

        public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
                doPost(req, res);
        }

        public void doPost(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {

                out = res.getWriter();
                res.setContentType("text/html");

                String regUrli = "http://localhost:8080/RegistryServer";
                String regUrlp = "https://localhost:8080/RegistryServer";
                String company = "%";

                out.println("regUrli = " + regUrli + "<br>");
                out.println("regUrlp = " + regUrlp + "<br>");
                out.println("company = " + company + "<br>");

                try {
                        executeQueryTest(regUrli, regUrlp, company);
                } catch (JAXRException e) {
                        out.println("Error during the test: " + e);
                } catch (NullPointerException e) {
                        out.println("Null Pointer Error during the test: " + e);
                }
                out.close();
        }

        public void executeQueryTest(String file, String filep, String cname)
                        throws JAXRException {
```

```
    try {
        Properties props = new Properties();
        props.setProperty("javax.xml.registry.queryManagerURL", file);
        props.setProperty("javax.xml.registry.lifeCycleManagerURL", filep);
        props.setProperty("javax.xml.registry.factoryClass",
                "com.sun.xml.registry.uddi.ConnectionFactoryImpl");

        props.setProperty("com.sun.xml.registry.http.proxyHost",
                httpProxyHost);
        props.setProperty("com.sun.xml.registry.http.proxyPort",
                httpProxyPort);

        ConnectionFactory factory = ConnectionFactory.newInstance();
        factory.setProperties(props);
        Connection conn = factory.createConnection();
        RegistryService rs = conn.getRegistryService();
        BusinessQueryManager bqm = rs.getBusinessQueryManager();

        ArrayList names = new ArrayList();
        names.add(cname);

        Collection fQualifiers = new ArrayList();
        fQualifiers.add(FindQualifier.SORT_BY_NAME_DESC);

        BulkResponse br = bqm.findOrganizations(fQualifiers, names, null,
                null, null, null);

        if (br.getStatus() == JAXRResponse.STATUS_SUCCESS) {
            out.println("Successfully queried the "
                    + "registry for organization matching the "
                    + "name pattern: \"" + cname + "\"" +
            "<br>");
            Collection orgs = br.getCollection();
            out.println("Results found: " + orgs.size() + "<br>");
            Iterator iter = orgs.iterator();
            while (iter.hasNext()) {
                Organization org = (Organization) iter.next();
                out.println("Organization Name: " + getName(org) +
            "<br>");
                out.println("Organization Key: " +
            org.getKey().getId()
                        + "<br>");
                out.println("Organization Description: "
                        + getDescription(org) + "<br>");

                Collection services = org.getServices();
                Iterator siter = services.iterator();
                while (siter.hasNext()) {
                    Service service = (Service) siter.next();
                    out.println("\tService Name: " +
                    getName(service)      + "<br>");
                    out.println("\tService Key: "
                    + service.getKey().getId() + "<br>");
                    out.println("\tService Description: "
                    + getDescription(service) + "<br>");
                }
            }
        } else {
            out.println("One or more JAXRExceptions "
            + "occurred during the query operation:");
            Collection exceptions = br.getExceptions();
            Iterator iter = exceptions.iterator();
            while (iter.hasNext()) {
                Exception e = (Exception) iter.next();
                out.println(e.toString());
            }
        }
    } catch (JAXRException e) {
        e.printStackTrace();
    }
}
```

```
        private String getName(RegistryObject ro) throws JAXRException {
                try {
                        return ro.getName().getValue();
                } catch (NullPointerException npe) {
                        return "";
                }
        }

        private String getDescription(RegistryObject ro) throws JAXRException {
                try {
                        return ro.getDescription().getValue();
                } catch (NullPointerException npe) {
                        return "";
                }
        }
}
```

ShowDataSets.java – This program displays the data sets published in the registry based on the search performed by the user.

```
package registry;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.Properties;
import java.util.StringTokenizer;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.registry.BulkResponse;
import javax.xml.registry.BusinessQueryManager;
import javax.xml.registry.Connection;
import javax.xml.registry.ConnectionFactory;
import javax.xml.registry.FindQualifier;
import javax.xml.registry.JAXRException;
import javax.xml.registry.JAXRResponse;
import javax.xml.registry.RegistryService;
import javax.xml.registry.infomodel.Organization;
import javax.xml.registry.infomodel.RegistryObject;
import javax.xml.registry.infomodel.Service;
import javax.xml.registry.infomodel.ServiceBinding;

public class ShowDataSets extends HttpServlet {

        // edit these if behind firewall, otherwise leave blank
        String httpProxyHost = "";
        String httpProxyPort = "";

        PrintWriter out;

        boolean showAll;

        String serviceField = "";

        int checkBoxCount;

        String dataAccessURI;

        public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
                doPost(req, res);
        }
```

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException {

        out = res.getWriter();
        res.setContentType("text/html");

        String regUrli = "http://localhost:8080/RegistryServer";
        String regUrlp = "https://localhost:8080/RegistryServer";
        String company = "%";

        checkBoxCount = 1;

        try {
                serviceField = req.getParameter("keyword");
                //out.println("Keyword: " + serviceField + "<br>");
                if (serviceField.equals(""))
                        showAll = true;
                else
                        showAll = false;
        } catch (NullPointerException npe) {
                showAll = true;
        }

        out.println("<html>");
        out.println("<head>");
        out.println("<title>DDM usign JWS: Data Sets</title>");
        out
                        .println("<meta http-equiv=\"Content-Type\"
                content=\"text/html; charset=iso-8859-1\">");
        out
                        .println("<link href=\"webmine.css\" rel=\"stylesheet\"
                type=\"text/css\">");
        out.println("</head>");
        out.println("<body>");

        //out.println("<center>");
        out.println("<table width=\"100%\">");
        out.println("<tr>");
        out
                        .println("<th align=\"center\" valign=\"middle\"
        bgcolor=\"#F6F9FE\">");
        out
                        .println("<font color=\"#006699\" size=\"5\"
        face=\"Verdana, Arial, Helvetica, sans-serif\">");
        out.println("<strong>Results</strong></font>");
        out.println("</th>");
        out.println("</tr>");
        out.println("</table>");
        out.println("<br>");

        out.println("<p><b>Registry URI:</b> " + regUrli + "<br><br>");

        try {
                executeQueryTest(regUrli, regUrlp, company);
        } catch (JAXRException e) {
                out.println("Error during the test: " + e);
        } catch (NullPointerException e) {
                out.println("Null Pointer Error during the test: " + e);
        }
        out.close();
}

public void executeQueryTest(String file, String filep, String cname)
                throws JAXRException {
        try {
                Properties props = new Properties();
                props.setProperty("javax.xml.registry.queryManagerURL", file);
                props.setProperty("javax.xml.registry.lifeCycleManagerURL", filep);
                props.setProperty("javax.xml.registry.factoryClass",
                              "com.sun.xml.registry.uddi.ConnectionFactoryImpl");
```

```
props.setProperty("com.sun.xml.registry.http.proxyHost",
                httpProxyHost);
props.setProperty("com.sun.xml.registry.http.proxyPort",
                httpProxyPort);

ConnectionFactory factory = ConnectionFactory.newInstance();
factory.setProperties(props);
Connection conn = factory.createConnection();
RegistryService rs = conn.getRegistryService();
BusinessQueryManager bqm = rs.getBusinessQueryManager();

ArrayList names = new ArrayList();
names.add(cname);

Collection fQualifiers = new ArrayList();
fQualifiers.add(FindQualifier.SORT_BY_NAME_DESC);

BulkResponse br = bqm.findOrganizations(fQualifiers, names, null,
                null, null, null);

if (br.getStatus() == JAXRResponse.STATUS_SUCCESS) {
        /*
         * out.println("Successfully queried the " + "registry for
         * organization matching the " + "name pattern: \"" +
         * serviceField + "\"" + " <br><br> ");
         */

        out
                        .println("<font face=verdana
        size=4px><strong>Data Sets</strong></font>");
        Collection orgs = br.getCollection();

        Iterator iter = orgs.iterator();
        int dsCount = 0;
        out
                        .println("<form name=datasetform method=post
        action=ShowAlgorithms>");
        out.println("<table border=1>");

        while (iter.hasNext()) {
                boolean printOnce = true;
                Organization org = (Organization) iter.next();

                Collection services = org.getServices();
                Iterator siter = services.iterator();

                while (siter.hasNext()) {
                        Service service = (Service) siter.next();
                        String serviceName = getName(service);
                        StringTokenizer st = new
                        StringTokenizer(serviceName);

                        while (st.hasMoreTokens()) {
                                String token = st.nextToken();
                                if (token.equalsIgnoreCase("Dataset")
                                                ||
                        token.equalsIgnoreCase("Data")) {
                                        if (printOnce) {
                                                out.println("<tr>");
                                                out

                                .println("<td bgcolor=#006699
                        bordercolor=#006699><strong><font
                        color=#FFFFFF>Organization Name
                        </font></strong></td>");
                                                out

                                .println("<td bgcolor=#006699
                        bordercolor=#006699 colspan=2><strong><font
                        color=#FFFFFF>"
```

```
+ getName(org) + "("  getDescription(org)+ ")</font></strong></td>");
out.println("</tr>");
out.println("<tr>");
out
.println("<td bordercolor=#FFFFFF><strong>Service Name</strong></td>");
out
.println("<td bordercolor=#FFFFFF><strong>Parameters</strong></td>");
out
.println("<td bordercolor=#FFFFFF><strong>Service Bindings</strong></td>");
}
printOnce = false;
if (showAll) {
        out.println("<tr>");
        out
        .println("<td bordercolor=#FFFFFF><input type=checkbox name=datasetKey"
        + checkBoxCount + " value=\""
        + service.getKey().getId()
        + "\">");
        //checkBoxCount++;
        out.println("<strong>" + serviceName
        + "</strong></td>");
        out
        .println("<td bordercolor=#FFFFFF>"
        + getDescription(service) + "</td>");
        Collection serviceBindings = service
        .getServiceBindings();
        Iterator sbiter = serviceBindings.iterator();
        out.println("<td bordercolor=#FFFFFF>");
        while (sbiter.hasNext()) {
                ServiceBinding serviceBinding = (ServiceBinding) sbiter.next();
                dataAccessURI = serviceBinding.getAccessURI();
                out.println("| <a href=\""
                + dataAccessURI + "?WSDL\">");
                out.println(dataAccessURI + "?WSDL");
                out.println("</a> |");
                out
                .println("<input type=hidden name=dataAccessURI"
                + checkBoxCount
                + " value=\""
                + dataAccessURI + "\"");
                checkBoxCount++;
        }
        out.println("</td>");
        out.println("</tr>");
        //out.println(serviceName +
        getDescription(service) + "<br>");
        dsCount++;
        continue;
} else {
        String serviceDescr = getDescription(service);
        StringTokenizer stDesc = new StringTokenizer(
        serviceDescr, ",");
        while (stDesc.hasMoreTokens()) {
                String tokenDesc = stDesc.nextToken()
                .trim();
                for (int i = 1; i < tokenDesc.length(); i++) {
                        String region = tokenDesc
                        .substring(0, i);
                        if (serviceField
                        .equalsIgnoreCase(region)) {
                                out.println("<tr>");
                                out
                                .println("<td bordercolor=#FFFFFF><input
                                type=checkbox name=datasetKey"
                                + checkBoxCount
                                + " value=\""
                                + service
                                .getKey()
                                .getId()
                                + "\">");
        out.println("<strong>"
 + serviceName
 + "</strong></td>");
 out
 .println("<td bordercolor=#FFFFFF>"
        + getDescription(service)
        + "</td>");

        Collection serviceBindings = service
        .getServiceBindings();

        Iterator sbiter = serviceBindings
        .iterator();

        out.println("<td bordercolor=#FFFFFF>");

        while (sbiter.hasNext()) {
                ServiceBinding serviceBinding = (ServiceBinding)
                sbiter.next();
                dataAccessURI = serviceBinding
                                .getAccessURI();
                out.println("| <a href=\""
                                + dataAccessURI
                                + "?WSDL\">");
                out.println(dataAccessURI
                                + "?WSDL");
                out.println("</a> |");
                out
                        .println("<input type=hidden
                name=dataAccessURI"
                                        + checkBoxCount
                                        + " value=\""
                                        + dataAccessURI
                                        + "\"");
                checkBoxCount++;

                        }


                                out.println("</td>");
                                out.println("</tr>");
                                //out.println(serviceName +
                                // getDescription(service) +
                                // "<br>");
                                dsCount++;
                                //break;
                                continue;
                                }
                        }
                }
        }
}
out.println("</table>");
        out.println("<br>");
        out
        .println("<input type=submit name=submit
        value=\"Show Algorithms\">");
        //out.println("<input type=hidden name=dataAccessURI
        value=\""
        // + dataAccessURI + "\"");
        out.println("</form>");
        out.println("<p><b>Datasets found:</b> " + dsCount +
        "</p>");
} else {
        out.println("One or more JAXRExceptions "
                        + "occurred during the query operation:");
        Collection exceptions = br.getExceptions();
        Iterator iter = exceptions.iterator();
        while (iter.hasNext()) {
                Exception e = (Exception) iter.next();
                out.println(e.toString());
```

```
                        }
                }
                out.close();
            } catch (JAXRException e) {
                e.printStackTrace();
            }
    }

    private String getName(RegistryObject ro) throws JAXRException {
            try {
                return ro.getName().getValue();
            } catch (NullPointerException npe) {
                return "";
            }
    }

    private String getDescription(RegistryObject ro) throws JAXRException {
            try {
                return ro.getDescription().getValue();
            } catch (NullPointerException npe) {
                return "";
            }
    }
}
```

ShowAlgorithms.java – This program queries the registry and displays the appropriate algorithms based on the data sets selected by the user.

```
package registry;

import java.io.*;
import java.util.*;

import javax.servlet.*;
import javax.servlet.http.*;

import javax.xml.registry.*;
import javax.xml.registry.infomodel.*;

public class ShowAlgorithms extends HttpServlet {

        // edit these if behind firewall, otherwise leave blank
        String httpProxyHost;
        String httpProxyPort;

        PrintWriter out;

        boolean showAll;

        String datasetField;

        String datasetDescription;

        Vector datafieldList;

        Vector parameterVals;

        String dataAccessURI;

        String algoAccessURI;

        //Vector algofieldList = new Vector();

        Service service;

        Vector dataAccessURIs;
Vector algoAccessURIs;

public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException {
        doPost(req, res);
}

public void doPost(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException {

        out = res.getWriter();
        res.setContentType("text/html");

        httpProxyHost = "";
        httpProxyPort = "";

        showAll = true;
        datasetField = "";
        datasetDescription = "";
        datafieldList = new Vector();
        parameterVals = new Vector();

        dataAccessURI = "";
        algoAccessURI = "";
        //Vector algofieldList = new Vector();

        service = null;
        dataAccessURIs = new Vector();
        algoAccessURIs = new Vector();

        String regUrli = "http://136.165.67.140:8080/RegistryServer";
        String regUrlp = "https://136.165.67.140:8080/RegistryServer";
        String company = "%";

        algoAccessURIs = new Vector();

        /*
         * try {
         *
         * datasetField = req.getParameter("datasetKey1");
         * //out.println("Keyword: " + serviceField + " <br> ");
         * if(datasetField.equals("")) showAll = true; else showAll = false; }
         * catch(NullPointerException npe) { showAll = true; }
         */

        out.println("<html>");
        out.println("<head>");
        out.println("<title>DDM usign JWS: Algorithms</title>");
        out
                        .println("<meta http-equiv=\"Content-Type\"
                content=\"text/html; charset=iso-8859-1\">");
        out
                        .println("<link href=\"webmine.css\" rel=\"stylesheet\"
                type=\"text/css\">");
        out.println("</head>");
        out.println("<body>");

        //out.println("<center>");
        out.println("<table width=\"100%\">");
        out.println("<tr>");
        out
                        .println("<th align=\"center\" valign=\"middle\"
        bgcolor=\"#F6F9FE\">");
        out
                        .println("<font color=\"#006699\" size=\"5\"
        face=\"Verdana, Arial, Helvetica, sans-serif\">");
        out.println("<strong>Results</strong></font>");
        out.println("</th>");
        out.println("</tr>");
        out.println("</table>");
        out.println("<br>");
        out.println("<p><b>Registry URI:</b> " + regUrli + "<br><br>");

        Enumeration en = req.getParameterNames();
        dataAccessURIs = new Vector();
        parameterVals = new Vector();

        Vector cnt = new Vector();
        while (en.hasMoreElements()) {
                String paramName = en.nextElement().toString();
                if (paramName.startsWith("datasetKey")) {
                        String parameterVal = req.getParameter(paramName);
                        parameterVals.add(parameterVal);
                        cnt.add(paramName.substring(10));
                        showAll = false;
                        //out.println("Name = " + parameterName + " Value = " +
                        // parameterVal);
                }
        }

        Enumeration en1 = req.getParameterNames();
        while (en1.hasMoreElements()) {
                String parameterName = en1.nextElement().toString();
                if (parameterName.startsWith("dataAccessURI")
                                && (cnt.contains(parameterName.substring(13)))) {
                        String parameterVal = req.getParameter(parameterName);
                        dataAccessURIs.add(parameterVal);
                        out.println("Name = " + parameterName + " Value = "
                                        + parameterVal + "<br>");
                        showAll = false;
                }
        }

        //if(parameterName.equals("dataAccessURI")) dataAccessURI =
        // req.getParameter("dataAccessURI");
        if (parameterVals.size() == 0)
                showAll = true;

        try {
                executeQueryTest(regUrli, regUrlp, company);
        } catch (JAXRException e) {
                out.println("Error during the test: " + e);
        } catch (NullPointerException e) {
                out.println("Null Pointer Error during the test: " + e);
        }
        out.close();
}

public void executeQueryTest(String file, String filep, String cname)
                throws JAXRException {
        try {
                Properties props = new Properties();
                props.setProperty("javax.xml.registry.queryManagerURL", file);
                props.setProperty("javax.xml.registry.lifeCycleManagerURL", filep);
                props.setProperty("javax.xml.registry.factoryClass",
                                "com.sun.xml.registry.uddi.ConnectionFactoryImpl");

                props.setProperty("com.sun.xml.registry.http.proxyHost",
                                httpProxyHost);
                props.setProperty("com.sun.xml.registry.http.proxyPort",
                                httpProxyPort);

                ConnectionFactory factory = ConnectionFactory.newInstance();
                factory.setProperties(props);
                Connection conn = factory.createConnection();
                RegistryService rs = conn.getRegistryService();
                BusinessQueryManager bqm = rs.getBusinessQueryManager();

                ArrayList names = new ArrayList();
                names.add(cname);
```

```
datafieldList = new Vector();

Collection fQualifiers = new ArrayList();
fQualifiers.add(FindQualifier.SORT_BY_NAME_DESC);

BulkResponse br = bqm.findOrganizations(fQualifiers, names, null,
               null, null, null);

if (br.getStatus() == JAXRResponse.STATUS_SUCCESS) {

       out
                      .println("<form name=algorithmform
       method=post action=WebMineServlet>");

       /*
        * out.println("Successfully queried the " + "registry for
        * organization matching the " + "name pattern: \"" +
        * serviceField + "\"" + " <br><br> ");
        */
       if (!showAll)
              out
                             .println("<font face=verdana
       size=4px><strong>Data Set(s)</strong></font><br>");
       Collection tempOrgs = br.getCollection();
       Iterator tempIter = tempOrgs.iterator();
       while (tempIter.hasNext()) {
              Organization tempOrg = (Organization)
       tempIter.next();
              Collection tempServices = tempOrg.getServices();
              Iterator tempSiter = tempServices.iterator();
              while (tempSiter.hasNext()) {
                     Service tempService = (Service)
              tempSiter.next();
                     String key = tempService.getKey().getId();
                     for (int i = 0; i < parameterVals.size();
                            i++) {
                            if (key.equals(parameterVals.get(i)))
                                   {
                                   out.println("<table
                                   border=1>");
                                   out.println("<tr>");
                                   out
                                                 .println("<td
                                   bgcolor=#006699
                            bordercolor=#006699><strong><font
                            color=#FFFFFF>Dataset
                            information</font></strong></td>");
                                   out.println("</tr>");
                                   out.println("<tr>");
                                   out
                                                 .println("<td
                                          bordercolor=#FFFFFF><s
                                   trong>Key: </strong>"

                                   + key + "</td>");
                                   out.println("</tr>");
                                   out.println("<tr>");
                                   out
                                                 .println("<td
                                          bordercolor=#FFFFFF><s
                                   trong>Service Name:
                                   </strong>"

                                   + getName(tempService)

                                   + "</td>");
                                   out.println("</tr>");
                                   out.println("<tr>");
                                   datasetDescription =
                                   getDescription(tempService);
```

```
out.println("<td bordercolor=#FFFFFF><strong>Parameters: </strong><br>");
StringTokenizer serviceParameters = new StringTokenizer(
getDescription(tempService), ",");
        while (serviceParameters.hasMoreTokens()) {
                String attribute = serviceParameters
                .nextToken();
                out.println("<input type=radio name=\"attribute--"
        + dataAccessURIs.get(i)
        + "\" value="
        + attribute
        + ">");
        out.println(attribute + "<br>");
        }
        out.println("</td>");
        out.println("</tr>");
        out.println("</table>");
        out.println("<br>");
        StringTokenizer datasetTokens = new StringTokenizer(
        datasetDescription);
        while (datasetTokens.hasMoreTokens()) {
                String datasetToken = datasetTokens
                .nextToken();
                int indexOfColon = datasetToken
        .indexOf(":");
        //out.println(indexOfColon + "<br>");
        String fieldType = datasetToken.substring(
        indexOfColon + 1, datasetToken
        .length());
        if (fieldType.endsWith(","))
        fieldType = fieldType.substring(0,
        fieldType.length() - 1);
        //out.println(fieldType + "<br>");
        datafieldList.add(fieldType.trim()
        .toLowerCase());
        }
        //out.println("<strong>Data Field
        // List:</strong> " + datafieldList +
        // "<br><br>");
                }
        }
        out
        .println("<font face=verdana size=4px><strong>Algorithms</strong></font>");
        Collection orgs = br.getCollection();
        Iterator iter = orgs.iterator();
                int dsCount = 0;

                for (int i = 0; i < parameterVals.size(); i++)
                        out.println("<input type=hidden name=datasetKey" + i
                                + " value=" + parameterVals.get(i) + ">");
                                out.println("<table border=1>");

                                while (iter.hasNext()) {
                                        boolean printOnce = true;
                                        Organization org = (Organization) iter.next();

                                        Collection services = org.getServices();
                                        Iterator siter = services.iterator();

                                        while (siter.hasNext()) {
                                                service = (Service) siter.next();
                                                String serviceName = getName(service);
                                                StringTokenizer st = new
                                                StringTokenizer(serviceName);

                                                while (st.hasMoreTokens()) {
                                                        String token = st.nextToken();
                                                        if
                                                (token.equalsIgnoreCase("Algorithm")
                                                                        ||
                                                token.equalsIgnoreCase("Algorithms")) {
                                                                if (printOnce) {
```

```
                out.println("<tr>");
                out.println("<td bgcolor=#006699 bordercolor=#006699><strong><font
                color=#FFFFFF>Organization Name </font></strong></td>");
                out
                .println("<td bgcolor=#006699 bordercolor=#006699 colspan=2><strong><font
                color=#FFFFFF>"
        + getName(org)
        + "("
        + getDescription(org)
        + ")</font></strong></td>");
        out.println("</tr>");
        out.println("<tr>");
        out
        .println("<td bordercolor=#FFFFFF><strong>Service Name</strong></td>");
        out
        .println("<td bordercolor=#FFFFFF><strong>Parameters</strong></td>");
        out
        .println("<td bordercolor=#FFFFFF><strong>Service Bindings</strong></td>");
        }
        printOnce = false;
        if (showAll) {
                out.println("<tr>");
                out
        .println("<td bordercolor=#FFFFFF><input type=radio name=algorithmKey value=\""
        + service.getKey().getId()
        + "\">");
        out.println("<strong>" + serviceName
        + "</strong></td>");
        out
        .println("<td bordercolor=#FFFFFF>"
        + getDescription(service)
        + "</td>");
        Collection serviceBindings = service
        .getServiceBindings();
        Iterator sbiter = serviceBindings.iterator();
        out.println("<td bordercolor=#FFFFFF>");
        while (sbiter.hasNext()) {
                ServiceBinding serviceBinding = (ServiceBinding) sbiter
                                              .next();
                algoAccessURI = serviceBinding.getAccessURI();
                out.println("| <a href=\""
                + algoAccessURI + "\"?WSDL>");

        out.println(algoAccessURI + "?WSDL");
        out.println("</a> |");
        algoAccessURIs.add(service.getKey()
        .getId());
        algoAccessURIs.add(algoAccessURI);
        }
        out.println("</td>");
        out.println("</tr>");
        //out.println(serviceName +
        // getDescription(service) + "<br>");
        dsCount++;
        continue;
} else {
        String serviceDescr = getDescription(service);
        StringTokenizer stDesc = new StringTokenizer(
        serviceDescr, ":");
        while (stDesc.hasMoreTokens()) {
                String tokenDesc = stDesc.nextToken();
                int indexOfOpenBracket = tokenDesc
                .indexOf("(");
                int indexOfCloseBracket = tokenDesc
                .indexOf(")");
                String fieldSet = tokenDesc.substring(
                        indexOfOpenBracket + 1,
                        indexOfCloseBracket);
                StringTokenizer fieldSetTokens = new StringTokenizer(
                fieldSet, ",");
                Vector algofieldList = new Vector();
while (fieldSetTokens.hasMoreElements()) {
        algofieldList.add(fieldSetTokens
        .nextToken().trim()
        .toLowerCase());
}


Collections.sort(algofieldList);
Collections.sort(datafieldList);
//out.println("Algo Field List: " +
// algofieldList + "<br>");
//out.println("Data Field List: " +
// datafieldList + "<br>");
if (new Subset().compare(datafieldList,

algofieldList)) {
out.println("<tr>");
out
.println("<td bordercolor=#FFFFFF><input type=radio name=algorithmKey value=\""
        + service.getKey()
        .getId()
        + "\">");

out.println("<strong>"
+ serviceName
+ "</strong></td>");
out
.println("<td bordercolor=#FFFFFF>"
                + getDescription(service)
                + "</td>");
Collection serviceBindings = service
.getServiceBindings();

Iterator sbiter = serviceBindings
.iterator();
        out
.println("<td bordercolor=#FFFFFF>");
while (sbiter.hasNext()) {

ServiceBinding serviceBinding = (ServiceBinding) sbiter
        .next();
algoAccessURI = serviceBinding
        .getAccessURI();
out.println("| <a href=\""
        + algoAccessURI
        + "?WSDL\">");
out.println(algoAccessURI
        + "?WSDL");
out.println("</a> |");
algoAccessURIs.add(service
        .getKey().getId());
algoAccessURIs
        .add(algoAccessURI);

out.println("</td>");
out.println("</tr>");

dsCount++;
                }
        break;
        }
}
}
out.println("</table>");
out.println("<br>");
out
.println("<input type=submit name=submit value=\"Run Algorithm\">");
        for (int i = 0; i < dataAccessURIs.size(); i++)
        out.println("<input type=hidden name=dataAccessURI" + i
                + " value=\"" + dataAccessURIs.get(i) + "\">");
                        out.println("<input type=hidden name=algoAccessURI
        value=\""
        + algoAccessURIs.toString() + "\">");
        out.println("</form>");
        out.println("<p><b>Algorithms found:</b> " + dsCount + "</p>");
                        } else {
                                out.println("One or more JAXRExceptions "
                                                + "occurred during the query operation:");

                                Collection exceptions = br.getExceptions();
                                Iterator iter = exceptions.iterator();
                                while (iter.hasNext()) {
                                        Exception e = (Exception) iter.next();
                                        out.println(e.toString());
                                }
                        }
                        out.close();
                } catch (JAXRException e) {
                        e.printStackTrace();
                }
        }

        private String getName(RegistryObject ro) throws JAXRException {
                try {
                        return ro.getName().getValue();
                } catch (NullPointerException npe) {
                        return "";
                }
        }

        private String getDescription(RegistryObject ro) throws JAXRException {
                try {
                        return ro.getDescription().getValue();
                } catch (NullPointerException npe) {
                        return "";
                }
        }
}
```

AprioriImpl.java – Implementation of the Apriori algorithm as a web service.

```
package aprioriservice;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Enumeration;
import java.util.StringTokenizer;
import java.util.Vector;

public class AprioriImpl implements AprioriIF {

        int pass; // number of passes

        int total; // total number of frequent itemsets

        int minsup; // minimal support of itemset

        String filename; // the filename of the database

        Item root; // the root item of the Trie
```

```
BufferedWriter writer;// the buffer to write the output to

DataHandler dh; // the handler for the database

StringBuffer returnStr; // the buffer that holds all the messages

String outputfilename;

/**
 * Constructur for creating a Apriori object.
 *
 */
public AprioriImpl() { }

/**
 * The workhorse method for the basic implementation of the Apriori
 * algorithm.
 */
public void findFrequentSets(String filename, String minsup, String outfile) {

        this.pass = 0;
        this.total = 0;
        this.outputfilename = outfile;
        this.minsup = Integer.parseInt(minsup);
        this.dh = new DataHandler(filename);
        this.root = new Item(0);
        returnStr = new StringBuffer();
        try {
                if (!outfile.equals("")) {
                        writer = new BufferedWriter(new FileWriter(outfile));
                }
        } catch (Exception e) {
        }

        boolean running = true;
        int candidates = 0, transactions = 0, pruned = 0, itemsets;

        while (running) {
                this.pass++;

                candidates = this.generateCandidates(this.root, new Vector(), 1);
                transactions = this.countSupport();
                pruned = this.pruneCandidates(this.root);

                itemsets = candidates - pruned;

                // correct the candidate count on first pass for printing
                if (this.pass == 1)
                        candidates = total;

                total += itemsets;
                if (itemsets <= this.pass && this.pass > 1) {
                        running = false;
                }

                //System.out.println("pass: " + this.pass + ", total: " + total
                //              + ", candidates: " + candidates + ", pruned: " +
                pruned);

                returnStr.append("pass: " + this.pass + ", total: " + total
                                + ", candidates: " + candidates + ", pruned: " +
                pruned + "\n");

        }

}

/**
 * Method for generating new candidates. Copies the siblings of an item to
 * its children.
 *
 * @param item
 *            the item to which generated items are added
 * @param depth
 *            the depth of recursion
 * @return the number of new candidates generated
 */
public int generateCandidates(Item item, Vector current, int depth) {
        Vector v = item.getChildren();
        Item child = item;
        int generated = 0;

        for (Enumeration e = v.elements(); e.hasMoreElements();) {
                child = (Item) e.nextElement();
                current.add(child);

                if (depth == this.pass - 1) {
                        generated += this.copySiblings(child, v, current);
                } else {
                        generated += this.generateCandidates(child, current, depth
                + 1);
                }

                current.remove(child);
        }
        return generated;
}

/**
 * Method for copying the siblings of an Item to its children.
 *
 * @param item
 *            the item to which the siblings are copied
 * @param siblings
 *            the siblings to be copied
 * @param current
 *            the current itemset to be generated
 * @return the number of siblings copied
 */
public int copySiblings(Item item, Vector siblings, Vector current) {
        Enumeration e = siblings.elements();
        Item parent = item;
        Item sibling = new Item();
        int copied = 0;

        while (sibling.getLabel() < parent.getLabel() && e.hasMoreElements()) {
                sibling = (Item) e.nextElement();
        }

        while (e.hasMoreElements()) {
                sibling = (Item) e.nextElement();
                current.add(sibling);
                if (this.pass <= 2
                                || this
                                                .checkSubsets(current,
                this.root.getChildren(), 0,
                                                                1)) {
                        parent.addChild(new Item(sibling.getLabel()));
                        copied++;
                }
                current.remove(sibling);
        }

        return copied;
}

/**
 * Checks if the subsets of the itemset to be generated are all frequent.
 *
 * @param current
 *            the current itemset to be generated
 * @param children
 *            the children in the trie on this depth
 * @param mark
 *            the mark in the current itemset
 * @param depth
 *            depth of recursion
 * @return true if the subsets are frequent, else false
 */
public boolean checkSubsets(Vector current, Vector children, int mark,
               int depth) {
        boolean ok = true;
        Item child;
        int index;
        int i = depth;

        if (children == null)
                return false;

        while (ok && (mark <= i)) {
                index = children.indexOf(current.elementAt(i));
                if (index >= 0) {
                        if (depth < this.pass - 1) {
                                child = (Item) children.elementAt(index);
                                ok = checkSubsets(current, child.getChildren(), i +
                                        1,depth + 1);
                        }
                } else {
                        ok = false;
                }
                i--;
        }

        return ok;
}

/**
 * Method for counting the supports of the candidates generated on this
 * pass.
 *
 * @return the number of transactions from which the support was counted
 */
public int countSupport() {
        int rowcount = 0;
        int[] items;
        this.dh.open();
        for (items = this.dh.read(); items.length > 0; items = this.dh.read()) {
                rowcount++;
                if (this.pass == 1) {
                        this.root.incSupport();
                        this.total += generateFirstCandidates(items);
                } else {
                        countSupport(root, items, 0, 1);
                }
        }

        if (this.pass == 1)
                this.minsup = this.minsup * rowcount / 100;

        return rowcount;

}

/**
 * Method generates the first candidates by adding each item found in the
 * database to the children of the root item. Also counts the supports of
 * the items found in the database.
 *
 * @param items
 *            the array of integer items from the database
 * @return the number of candidates generated
 */
public int generateFirstCandidates(int[] items) {
        Vector v = root.getChildren();
        Enumeration e = v.elements();
        Item item = new Item();
        int generated = 0;

        for (int i = 0; i < items.length; i++) {

                while (e.hasMoreElements() && item.getLabel() < items[i]) {
                        item = (Item) e.nextElement();
                }

                if (item.getLabel() == items[i]) {
                        item.incSupport();
                        if (e.hasMoreElements())
                                item = (Item) e.nextElement();
                } else if (item.getLabel() > items[i]) {
                        int index = v.indexOf(item);
                        Item child = new Item(items[i]);
                        child.incSupport();
                        this.root.addChild(child, index);
                        generated++;
                } else {
                        Item child = new Item(items[i]);
                        child.incSupport();
                        this.root.addChild(child);
                        generated++;
                }
        }
        return generated;
}

public void countSupport(Item item, int[] items, int i, int depth) {
        Vector v = item.getChildren();
        Item child;
        int tmp;
        Enumeration e = v.elements();

        // loop through the children to check
        while (e.hasMoreElements()) {
                child = (Item) e.nextElement();

                // break, if the whole transaction is checked
                if (i == items.length) {
                        break;
                }

                // do a linear search for the child in the transaction starting
                // from i
                tmp = i;
                while (tmp < items.length && items[tmp] < child.getLabel())
                        tmp++;

                // if the same item exists, increase support or go deaper
                if (tmp < items.length && child.getLabel() == items[tmp]) {
                        if (depth == this.pass) {
                                child.incSupport();
                        } else {
                                countSupport(child, items, tmp + 1, depth + 1);
                        }
                        i = tmp + 1;
                }

        }
}

/**
 * Method for pruning the candidates. Removes items that are not frequent
 * from the Trie.
 *
 * @param item
 *            the item the children of which will be pruned
 * @return the number of items pruned from the candidates
 */
public int pruneCandidates(Item item) {
        Vector v = item.getChildren();
        Item child = item;
        int pruned = 0;

        for (Enumeration e = new Vector(v).elements(); e.hasMoreElements();) {
                child = (Item) e.nextElement();

                // check infrequency, existence and that it is fully counted
                if (child.getSupport() < this.minsup) {
                        v.remove(child);
                        pruned++;
                } else {
                        pruned += pruneCandidates(child);
                }
        }
        return pruned;
}

/**
 * Method prints the itemsets to the system output and to a file if the name
 * of an output file exists.
 */
public String printFrequentSets() {
        if (this.writer != null) {
                print(root, "");
        }
        //System.out.println("\nnumber of frequent itemsets found: " +
        this.total);
        returnStr.append("\nnumber of frequent itemsets found: " + this.total +
                "\n");

        try {
                this.writer.close();
        } catch (IOException e) {
                e.printStackTrace();
        }

        //generateSortedFile();

        return returnStr.toString();
}

/**
 * Loops through the Trie recursively adding paths and subpaths to the
 * output string along the way.
 *
 * @param item
 *            the item where the recursion is
 * @param str
 *            the string of the gatherd itemset
 */
public void print(Item item, String str) {
        Vector v = item.getChildren();

        for (Enumeration e = v.elements(); e.hasMoreElements();) {
                item = (Item) e.nextElement();
                try {
                        this.writer.write(str + item.getLabel() + "\n");
                        //              " (" + item.getSupport() + ")\n");
                        this.writer.flush();
                } catch (Exception x) {
                        //System.out.println("no output file");
                        returnStr.append("no output file\n");
                }
                if (item.hasChildren()) {
                        print(item, str + item.getLabel() + " ");
                }
        }
        }

        public void generateSortedFile(String outputfilename) {

                try {

                        //Read the file contents into an arraylist
                        ArrayList al = new ArrayList(0);
                        BufferedReader in = new BufferedReader(new
                        FileReader(outputfilename));
                        String responseLine = null;
                        while((responseLine=in.readLine())!=null) {
                                StringTokenizer st = new StringTokenizer(responseLine);
                                //System.out.println(responseLine + ":" +
                        st.countTokens());
                                al.add(String.valueOf(st.countTokens()) + ":" +
                                responseLine);
                        }
                        in.close();

                        //Sort the elements
                        Object[] arrayForSorting = al.toArray();
                        Arrays.sort(arrayForSorting);

                        //System.out.println("Elements : " + arrayForSorting.length);
                        Object[] invertArray = new Object[arrayForSorting.length];
                        for(int forward=0,reverse = arrayForSorting.length-1;forward <
                        arrayForSorting.length; forward++, reverse--) {
                                //System.out.println("i = " + forward + " : " + "j = " +
                                reverse);
                                invertArray[reverse] = arrayForSorting[forward];
                        }
                        //Write to a new file
                        PrintWriter out = new PrintWriter(new BufferedWriter(new
                                FileWriter(outputfilename + "_sorted", false)));
                        for(int j=0;j<arrayForSorting.length;j++) {
                                StringTokenizer st = new
                                StringTokenizer(invertArray[j].toString(), ":");
                                st.nextToken();
                                out.println(st.nextToken());
                        }
                        out.close();

                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

        /**
         * Main method for testing the algorithm.
         *
         * @param args
         *            the arguments can contain the filename of the testfile and the
         *            minimal support threshold and a filename for output
         */
        public static void main(String args[]) {
                String testfile = "test.dat";
                String outfile = "";
                String support = "50";

                //StopWatch sw = new StopWatch();
                //sw.start();
                AprioriImpl apriori = new AprioriImpl();
                apriori.findFrequentSets(testfile, support, outfile);
                System.out.println(apriori.printFrequentSets());
                //sw.stop();
                //sw.print();
        }
}
```

FractalDimensionImpl.java – Implementation of the Fractal Dimension algorithm as a web service.

```
package fractaldimensionservice;

import java.util.*;
import java.io.*;

public class FractalDimensionImpl {

        public String findFD(String dataFileName, String normalizedFileName) {

                String result = new String();

                double elements[][] = null;
                double min[] = null;
                double max[] = null;

                BufferedReader in = null;
                try {
                        in = new BufferedReader(new FileReader(dataFileName));

                } catch (FileNotFoundException fne) {
                        result = "File not found.";
                }

                int dimension = 0;
                String responseLine = new String();
                int i = 0;
                int j = 0;
                try {
                        while ((responseLine = in.readLine()) != null) {
                                //System.out.println("line = " + responseLine);
                                StringTokenizer stMain = new StringTokenizer(responseLine);
                                j = 0;
                                if (dimension == 0) {
                                        dimension = stMain.countTokens();
                                        elements = new double[dimension][100000];
                                        min = new double[dimension];
                                        max = new double[dimension];
                                }
                                //System.out.println(responseLine);
                                while (stMain.hasMoreTokens()) {
                                        elements[j][i] =
Double.parseDouble(stMain.nextToken());
                //System.out.print("elements[" + j + "][" + i + "]" +
                                        // elements[j][i]);
                                        if (i == 0) {
                                                min[j] = elements[j][i];
                                                max[j] = elements[j][i];
                                        } else {
                                                if (elements[j][i] < min[j])
                                                        min[j] = elements[j][i];
                                                if (elements[j][i] > max[j])
                                                        max[j] = elements[j][i];
                                        }
                                //System.out.print("\tmin[" + j + "]" + min[j]);
                                //System.out.print("\tmax[" + j + "]" + max[j] + "\n");
                                        j++;
                                }
                                //System.out.println();
                                i++;
                                j = 0;
                        }

                        in.close();

                } catch (IOException ioe) {
```

```
                result = "Error reading file contents.";
        }

        double normalized = 0;

        try {
                PrintWriter out = new PrintWriter(new BufferedWriter(
                                new FileWriter(normalizedFileName)));
                for (int s = 0; s < i; s++) {
                        for (int r = 0; r < dimension; r++) {
                        normalized = (elements[r][s] - min[r]) / (max[r] - min[r]);
                                out.print(normalized);
                                //System.out.print(normalized);
                                //System.out.print(elements[r][s]);
                                if (r != dimension - 1)
                                        out.print("\t");
                        }
                        out.println();
                }
                out.close();
        } catch (IOException ioe) {
                result = "Error writing normalized file.";
        }

        result = processData(normalizedFileName);
        return result;

}

private String processData(String normalFileName) {

        double xArray[] = new double[7];
        double yArray[] = new double[7];
        StringBuffer returnVal = new StringBuffer();

        int arr = 0;
        for (int r = 2; r <= 128; r *= 2, arr++) {

                BufferedReader in = null;
                try {
                        in = new BufferedReader(new FileReader(normalFileName));
                } catch (FileNotFoundException fne) {
                        return "Normalized file not found.";
                }

                Hashtable cellTab = new Hashtable();
                String responseLine = new String();

                //System.out.println(recieveStr);
                try {

                        while ((responseLine = in.readLine()) != null) {

                        StringTokenizer st = new StringTokenizer(responseLine);
                                String cellId = "";
                                while (st.hasMoreTokens()) {
                                        //returnVal.append(responseLine + "\n");

                                int x = (int) Math.ceil(Double.parseDouble(st
                                                       .nextToken())
                                                       * r);
                                        if (x == 0)
                                                x = 1;
                                        cellId = cellId + String.valueOf(x) + ".";
                                }

                        if (cellId.trim().length() != 0) {

                                cellId = cellId.substring(0, cellId.length() - 1);
                        //If the table already contains the Cell ID, increment
                        // it.
                        if (cellTab.containsKey(cellId)) {
                                int cellValue = Integer.parseInt(cellTab
                                .get(cellId).toString());
                                                cellValue++;
                                                cellTab.remove(cellId);
                                                cellTab.put(cellId,
                                        String.valueOf(cellValue));
                                        } else {
                                                //Otherwise add a new entry
                                                cellTab.put(cellId, "1");
                                        }
                                }
                        }
                } catch (IOException ioe) {
                        return "Could not read file contents.";
                }
                int sum = 0;

                Enumeration enum = cellTab.keys();
                while (enum.hasMoreElements()) {
                        Object key = enum.nextElement();
                        sum += Integer.parseInt(cellTab.get(key).toString())
                        Integer.parseInt(cellTab.get(key).toString());
                }

                yArray[arr] = Math.log(sum) / Math.log(2);
                xArray[arr] = Math.log(1.0 / r) / Math.log(2);

                //returnVal.append("arr =" + arr + " r = 1/" + r + " x = " +
                // xArray[arr] + " y = " + yArray[arr] + "\n");

                writeResults(cellTab, r, returnVal);
        }

        calculateFD(xArray, yArray, returnVal);

        return returnVal.toString();
}

private void calculateFD(double[] xVal, double[] yVal,
                StringBuffer returnVal) {

        double meanX = 0;
        double meanY = 0;

        double[] coeff_corr = new double[4];
        double fd[] = new double[4];

        for (int i = 0; i < 4; i++) {
                double sumX = 0;
                double sumY = 0;

                //System.out.println("Iteration " + (i+1));
                for (int j = i; j < i + 4; j++) {
                        sumX += xVal[j];
                        sumY += yVal[j];
                }
                meanX = sumX / 4;
                meanY = sumY / 4;

                //System.out.println("Mean of X: " + meanX);
                //System.out.println("Mean of Y: " + meanY);

                double SXX = 0;
                double SYY = 0;
                double SXY = 0;

                for (int k = i; k < i + 4; k++) {
                        SXX += (xVal[k] - meanX) * (xVal[k] - meanX);
                        SYY += (yVal[k] - meanY) * (yVal[k] - meanY);
                        SXY += (xVal[k] - meanX) * (yVal[k] - meanY);
                        }

                        //System.out.println("SXX: " + SXX);
                        //System.out.println("SYY: " + SYY);
                        //System.out.println("SXY: " + SXY);

                        fd[i] = SXY / SXX;
                        //System.out.println("FD = " + fd[i]);
                        returnVal.append("FD = " + fd[i] + "\n");

                        coeff_corr[i] = SXY / Math.pow(SXX * SYY, 0.5);
                        //System.out.println("R = " + coeff_corr[i]);
                        returnVal.append("R = " + coeff_corr[i] + "\n");

                }

                double max = 0;
                double theFD = 0;
                for (int l = 0; l < 4; l++) {
                        if (coeff_corr[l] > max) {
                                max = coeff_corr[l];
                                theFD = fd[l];
                        }
                }

                //System.out.println("The FD: " + theFD);
                //System.out.println("The R: " + max);

                returnVal.append("The FD: " + theFD + "\n");
                returnVal.append("The R: " + max + "\n");

        }

        private void writeResults(Hashtable ht, int r, StringBuffer returnVal) {

                returnVal.append("r = 1/" + r + "\n");

                Object[] keys = new Object[ht.size()];
                Enumeration enum = ht.keys();
                int i = 0;
                while (enum.hasMoreElements()) {
                        Object hashKey = enum.nextElement();
                        keys[i++] = hashKey + "." + ht.get(hashKey);
                }

                Arrays.sort(keys);

                for (int j = 0; j < keys.length; j++) {
                        String hashRow = keys[j].toString();
                        String hashKey = hashRow.substring(0, hashRow.lastIndexOf("."));
                        String hashValue = hashRow.substring(hashRow.lastIndexOf(".") + 1,
                                        hashRow.length());
                        returnVal.append(hashKey + "\t" + hashValue + "\n");
                }
        }

        public static void main(String args[]) {
                String result = new FractalDimensionImpl().findFD("C:\\comb_data.out",
                                "C:\\normalized.out");
                System.out.println(result);
        }

}
```

CURRICULUM VITAE

Name	Padmanabhan Ramaswamy
Address	Department of Computer Engineering and Computer Science University of Louisville Louisville KY 40208
Date of Birth	August 5 1976
Education	M.S., Computer Engineering and Computer Science, 2002 – 2005 University of Louisville, Louisville, KY Kerala University, Trivandrum, India B.S., Computer Science, 1994 – 1998
Publications	"An Extensible Service Oriented Distributed Data Mining Framework", Proceedings of International Conference on Machine Learning and Applications ICMLA – 04.