Agent-Enriched Data Mining Using an Extendable Framework

Kamal Ali Albashiri
Department of Computer Science
The University of Liverpool, Ashton Building,
Ashton Street, Liverpool L69 3BX, United Kingdom
ali@csc.liv.ac.uk

Frans Coenen
Department of Computer Science
The University of Liverpool, Ashton Building,
Ashton Street, Liverpool L69 3BX, United Kingdom
frans@csc.liv.ac.uk

ABSTRACT

An extendable and generic Agent Enriched Data Mining (AEDM) framework, EMADS (the Extendable Multi-Agent Data mining System) is described. The central feature of the framework is that it avoids the use of ontologies or agreed meta-language formats by supporting a system of wrappers. The advantage offered is that the system is easily extendable, further data agents and mining agents can simply be added to the system. A demonstration EMADS framework is currently available. The paper includes details of the EMADS architecture and the wrapper principle incorporated into it. A full description of the framework's operation is provided by considering two AEDM scenarios; the scenarios are also the focus for an evaluation of the framework.

1. MOTIVATION AND GOALS

Agent-Enriched Data Mining (AEDM), also known as multi-agent data mining, seeks to harness the general advantageous of MAS in the application domain of Data Mining (DM). MAS technology has much to offer DM, particularly in the context of various forms of distributed and cooperative DM. Distributed (and parallel) DM is directed at reducing the time complexity of computation associated with the increasing sophistication, size and availability of the data sets we wish to mine. Cooperative DM encompasses ensemble mechanisms and techniques such as bagging and boosting. MAS have a clear role in both these areas. MAS technology also offers some further advantageous for AEDM, namely:

- Extendibility of DM frameworks,
- Resource and experience sharing,
- Greater end-user accessibility,
- Information hiding, and
- The addressing of privacy and security issues.

The last of the above advantageous merits some further comment. By its nature DM is often applied to sensitive data. The MAS approach would allow data to be mined remotely. Similarly, with respect to DM algorithms, MAS can make use of algorithms without necessitating their transfer to users, thus contributing to the preservation of intellectual property rights. MAS make it possible for software services to be provided through the cooperative efforts of distributed collections of autonomous agents. Communication and cooperation between agents are brokered by one or more facilitators, which are responsible for matching requests, from users and agents, with descriptions of the capabilities of other agents. Thus, it is not generally required that a user or agent know the identities, locations, or number of other agents involved in satisfying a request.

The challenge of generic AEDM is the disparate nature and variety of modern DM, and the necessary communication mechanism required to cope with this disparate nature. One approach is to make use of the established Agent Communication Languages (ACLs) and mechanisms; well known examples include the Knowledge Query and Manipulation Language (KQML), the Knowledge Interchange Format (KIF), and the Foundation for Intelligent Physical Agents (FIPA) ACL [14]. All these ACLs have their advantageous and disadvantageous and tend to address particular forms of intra-agent communication; for example FIPA ACL is directed at agent negotiation. Each can be employed in the context of AEDM communication but on its own will not facilitate the shared agent understanding required to achieve generic AEDM. This would require recourse to the use of ontologies and/or some agreed meta-language. It is suggested in this work that a method of addressing the communication requirements of AEDM is to use a system of mediators and wrappers coupled with an ACL such as FIPA ACL, and that this can more readily address the issues concerned with the variety and range of contexts to which AEDM can be applicable.

To investigate and evaluate the expected advantageous of wrappers and mediators, in the context of generic AEDM, the authors have developed and implemented (in JADE) a multi-agent platform, EMADS (the Extendable Multi-Agent Data mining System). Extendibility is seen as an essential feature of the framework primarily because it allows its functionality to grow in an incremental manner. The vision is of an "anarchic" collection of agents, contributed to by a community of EMADS users, that exist across an "internet space"; that can negotiate with each other to attempt to perform a variety of DM tasks (or not if no suitable collection of agents come together) as proposed by other (or the same) EMADS users. An EMADS demonstrator is currently in operation.

The primary goal of the EMADS framework is to provide a means for integrating new DM algorithms and data sources in a distributed infrastructure and collaborative environment. However, EMADS also seeks to address some of the issues of DM that would benefit from the rich and complex interactions of communicating agents. The broad advantages offered by the framework are:

- Flexibility in assembling communities of autonomous service providers, including the incorporation of existing applications.

Cite as: Agent-enriched data mining using an extendable framework, Kamal Ali Albashiri, and Frans Coenen, *Proc. of 4th Int. ws on Agents and Data Mining Interaction (ADMI-2009)*, May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

- Minimization of the effort required to create new agents, and to wrap existing applications.
- Support for end users to express DM requests without having detailed knowledge of the individual agents.

The rest of this paper is organised as follows. A brief review of some related work on Agent-enriched Data Mining (AEDM) is presented in Section 2. The conceptual framework together with an overview of the wrapper principle is presented in Section 3. The framework operation is illustrated in Section 4 using two DM scenarios: Meta Association Rule Mining (MARM) and single label classification. Finally some conclusions are presented in Section 5.

2. RELATED WORK

There are a number of reports in the literature of the application of Agent techniques to DM. The contribution of this section is a broad review of prominent AEDM approaches in the literature and discussion of the benefits that agent-driven DM architectures provide in coping with such problems. This section is not concerned with particular DM techniques; it is however concerned with work on the design of distributed and multi-agent system directed at DM.

The most fundamental approach to distributed DM is to move all of the data to a central data warehouse and then to analyze this with a single DM system, even though this approach intuitively guarantees accurate DM results, it might be infeasible in many cases.

An alternative approach is high level learning with meta-learning strategies in which all the data can be locally analyzed (local data model), and the local results at their local sites combined at the central site to obtain the final result (global data model). Meta-learning methods have been widely used within DM [31, 10], particularly in the area of classification and regression. These approaches are less expensive but may produce ambiguous and incorrect global results. In addition, Distributed DM approaches require centralised control that causes a communication bottleneck that sometimes leads, in turn, to inefficient performance and system failure.

To make up for such a weakness, many researchers have investigated more advanced approaches of combining local models built at different sites. Most of these approaches are agent-based high level learning strategies.

One of the earliest references to AEDM can be found in Kargupta et al. [20] who describe a parallel DM system (PADMA) that uses software agents for local data accessing and analysis, and a web based interface for interactive data visualization. PADMA has been used in medical applications. They describe a distributed DM architecture and a set of protocols for a multi-agent software tool. Peng et al. [26] presented an interesting comparison between single-agent and multi-agent text classification in terms of a number of criteria including response time, quality of classification, and economic/privacy considerations. Their results indicate, not unexpectedly, in favour of a multi-agent approach.

A popular AEDM approach is described in the METAL project [25] whose emphasis is on helping the user to obtain a ranking of suitable DM algorithms through an online advisory system. Gorodetsky et al. [16] correctly consider that the core problem in AEDM is not the DM algorithms themselves (in many case these are well understood), but the most appropriate mechanisms to allow agents to collaborate. Gorodetsky et al. present an AEDM system to achieve distributed DM and, specifically, classification. A more recent system, proposed in [24], uses the MAGE middleware [29] to build an execution engine that uses a directed acyclic graph to formalize the representation of KDD process. In [11] a multi-agent system F-Trade has been proposed. It is a web-based DM infrastructure for trading and surveillance support in capital markets.

The meta-learning strategy offers a way to mine classifiers from homogeneously distributed data. It follows three main steps. The first is to generate base classifiers at each site using a classifier learning algorithms. The second step is to collect the base classifiers at a central site, and produce meta-level data from a separate validation set and predictions generated by the base classifier on it. The third step is to generate the final classifier (meta-classifier) from meta-level data via a combiner or an arbiter. Copies of the classifier agent will exist, or be deployed, on nodes in the network being used (see for example [27]). Perhaps the most mature agent-based meta-learning systems are: JAM [30], BODHI [19], and Papyrus [6]. Papyrus is designed to support both learning strategies; meat-learning and central learning. A hybrid learning strategy is a technique that combines local and remote learning for model building [17]. In contrast to JAM and BODHI, Papyrus can not only move models from site to site, but can also move data when such a strategy is desirable. Papyrus is a specialized system which is designed for clusters while JAM and BODHI are designed for data classification. These are reviewed in details in [21].

Most of the previously proposed AEDM systems are used to improve the performance of one specific DM task. To the best knowledge of the authors, there have been only few AEDM systems that define a generic framework for the AEDM approach. An early attempt was IDM [9], a multiple agent architecture that attempts to do direct DM that helps businesses gather intelligence about their internal commerce agent heuristics and architectures for KDD. In [5] a generic task framework was introduced but was designed to work only with spatial data. The most recent system is introduced in [15] where the authors proposed a multi-agent system to provide a general framework for distributed DM applications. The effort to embed the logic of a specific domain has been minimized and is limited to the customization of the user. However, although its customizable feature is of a considerable benefit, it still requires users to have very good DM knowledge.

3. EMADS OVERVIEW

A high level view of the framework conceptualization showing the various categories of agents and their contributors is given in Figure 1. The housekeeping (DF and AMS) agents are specialized server agents that are responsible for helping agents to locate one another. They do not participate in problem-solving; they only play a role of a facilitator in the system. Note that any system configuration is not limited to single MAS. Larger systems can be assembled from multiple MASs, each having the sort of structure shown in Figure 2.

3.1 System Structure

The EMADS framework has several different modes of operation according to the nature of the participant. Each mode of operation has a corresponding category of *User Agent*. Broadly, the supported categories are:

- Developers: Developers are participants, who have full access and may contribute DM algorithms in the form of *Data Mining Agents* (DM Agents).
- Data Miners: These are participants, with restricted access to the system, who may pose DM requests through User Agents and *Task Agents* (see below for further details).
- Data Contributors: These are participants, again with restricted access, who are prepared to make data available, by launching *Data Agents*, to be used by DM agents.

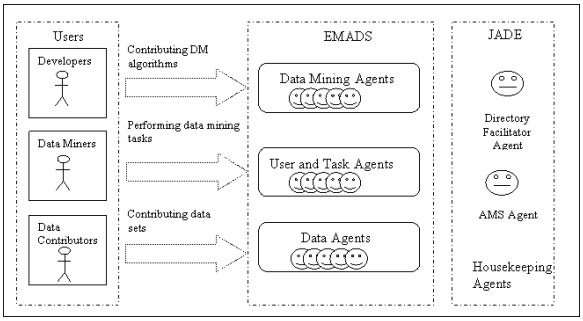


Figure 1: High level view of EMADS conceptual framework.

The various categories of agents are illustrated in Figure 1: DM agents, Task Agents, User Agents and Data Agents. DM agents are usually specialist agents that posses an algorithm for a particular DM task or sub-task. DM agents may be based on legacy applications, in which case the agent may be little more than a wrapper that calls a pre-existing API (see subsection 3.3 for further detail).

Note that, before interaction with EMADS can commence, appropriate software needs to be downloaded and launched by the participant. Note also that any individual participant may be as a user, contributor and developer at the same time.

Conceptually the nature of the requests that may be posted by users is extensive. In the current demonstration implementation, the following types of generic request are supported:

- Find the "best" classifier (to be used by the requester at some later date in off line mode) for a data set provided by the user.
- Find the "best" classifier for the indicated data set (i.e. provided by some other participant).
- Find a set of Association Rules (ARs) contained within the data set(s) provided by the user.
- Find a set of Association Rules (ARs) contained within the indicated type of data set(s) (i.e. provided by other participants).

In the above a "best" classifier is defined as a classifier that will produce the highest accuracy on a given test set (identified by the mining agent) according to the detail of the request. To obtain the "best" classifier EMADS will attempt to access and communicate with as many classifier generator DM agents as possible and select the best result. The classification style of user request will be discussed further in subsection 4.2 to illustrate the operation of EMADS in more detail.

The Association Rule Mining (ARM) style of request is discussed further in subsection 4.1. The scenario investigated here is one where an agent framework is used to implement a form of Meta-ARM where the results of the parallel application of ARM to a collection of data sets, with not necessarily the same schema but conforming to a global schema, are combined. Some further details of this process can be found in Albashiri et al. [3, 4].

3.2 Agent Interactions

Conceptually the EMADS system is a hybrid peer to peer agent based system comprising a collection of collaborating agents that exist in a set of containers. Agents may be created and contributed to the system by any user/contributor. One of these containers, the main container, holds a number of housekeeping agents that provide various facilities to maintain the operation of the framework.

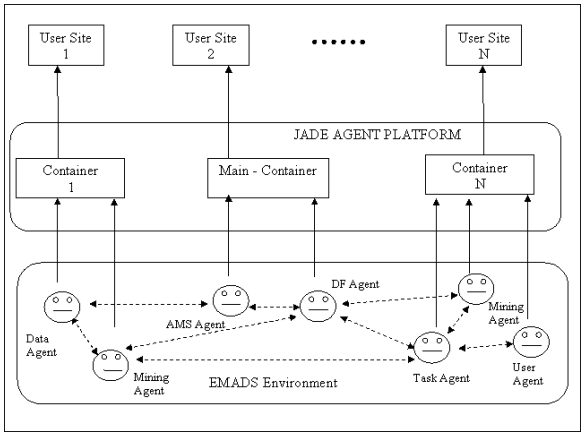


Figure 2: EMADS Architecture as Implemented in Jade

In particular the main container holds an Agent Management System (AMS) agent and a Directory Facilitator (DF) agent. The terminology used is taken from the JADE (Java Agent Development) [7] framework in which the framework is implemented. Briefly the AMS agent is used to control the life cycles of other agents in the platform, and the DF agent provides an agent lookup service. Both the main container and the remaining containers can hold various DM agents. Note that the EMADS main container is located on the EMADS host organisation site, while the other containers may be held at any other sites worldwide.

Figure 2 presents the EMADS architecture as implemented in JADE. It shows a sample collection of several application agents and housekeeping agents, organized as a community of peers by a common relationship to each other.

With reference to Figure 2, a user agent runs on the user's local host and is responsible for: (i) accepting user input (request), (ii) launching the appropriate Task Agent to processes the user request, and (iii) displaying the results of the (distributed) computation. The user expresses a task to be executed using standard interface dialogue mechanisms by clicking on active areas in the interface and, in some cases, by entering thresholds values; note that the user does not need to specify which agent or agents should be employed to perform the desired task. For instance, if the question "What is the best classifier for my data?" is posed in the user interface, this request will trigger a Task Agent. The Task Agent requests the facilitator to match the action part of the request to capabilities published by other agents. The request is then routed by the Task Agent to the appropriate agents to execute the request, this will typically involve communication among various relevant agents within the system. On completion the results are sent back to the user agent for display.

The key elements of the operation of EMADS that should be noted are:

1. The mechanism whereby a collection of agents can be harnessed to identify a "best solution".
2. The process whereby new agents connect to the facilitator and registering their capability specifications.
3. That the interpretation and execution of a task is a distributed process, with no one agent defining the set of possible inputs to the system.
4. That a single request can produce cooperation and flexible communication among many agents spread across multiple machines.

3.2.1 Agent Cooperation

Cooperation among the various EMADS agents is achieved via messages expressed in FIPA ACL and is normally structured around a three-stage process:

1. **Service Registration** where providers (agents who wish to provide *services*) register their capability specifications with a facilitator.
2. **Request Posting** where User Agents (*requesters* of services) construct requests and relay them to a Task Agent,
3. **Processing** where the Task Agent coordinates the efforts of the appropriate service providers (Data Agents and DM Agents) to satisfy the request.

Note that Stage 1 (service registration) is not necessarily immediately followed by stage 2 and 3, it is possible that a provider services may never be used. Note also that the facilitator (the DF and AMS agents) maintains a knowledge base that records the capabilities of the various EMADS agents, and uses this knowledge to assist requesters and providers of services in making contact. When a service provider (i.e. Data Agent or DM Agent) is created, it makes a connection to a facilitator, which is known as its *parent facilitator*. Upon connection, the new agent informs its parent facilitator of the services it can provide. When the agent is needed, the facilitator sends its address to the requester agent. An important element of the desired EMADS agent cooperation model is the function of the Task Agent; this is therefore described in more detail in the following subsection.

3.2.2 The Task Agent

A Task Agent is designed to handle a user request. This involves a three step process:

1. Determination of whom (which specific agents) will execute a request;
2. Optimization of the complete task, including parallelization where appropriate; and
3. Interpretation of the optimized task.

Thus determination (step 1) involves the selection of one or more agents to handle each sub-task given a particular request. In doing this, the Task agent uses the facilitator's knowledge of the capabilities of the available EMADS agents (and possibly of other facilitators, in a multi-facilitator system). The facilitator may also use information specified by the user (such as threshold values). In processing a request, an agent can also make use of a variety of capabilities provided by other agents. For example, an agent can request data from Data Agents that maintain shared data. The optimization step results in a request whose interpretation will require as few communication exchanges as possible, between the Task Agent and the satisfying agents (typically DM Agents and Data Agents), and can exploit the parallel processing capabilities of the satisfying agents. Thus, in summary, the interpretation of a task by a Task Agent involves: (i) the coordination of requests directed at the satisfying agents, and (ii) assembling the responses into a coherent whole, for return to the user agent.

3.3 System Extendibility

One of the principal objectives of the EMADS framework is to provide an easily extendable framework that can accept new data sources and new DM techniques. In general, extendibility can be defined as the ease with which software can be modified to adapt to new requirements or changes in existing requirements. Adding a new data source or DM techniques should be as easy as adding new agents to the system. The desired extendibility is achieved by a system of wrappers. EMADS wrappers are used to "wrap" up DM artifacts so that they become EMADS agents and can communicate with other EMADS agents. Such EMADS wrappers can be viewed as agents in their own right that are subsumed once they have been integrated with data or tools to become EMADS agents. The wrappers essentially provide an application interface to EMADS that has to be implemented by the end user, although this has been designed to be a fairly trivial operation.

In the current demonstration EMADS system two broad categories of wrapper have been defined: (i) data wrappers and (ii) tool wrappers; the first is used to create data agents and the second to create DM agents. Each is described in further detail in the following two subsections.

3.3.1 Data Wrappers

Data wrappers are used to "wrap" a data source and consequently create a data agent. Broadly a data wrapper holds the location (file path) of a data source, so that it can be accessed by other agents; and meta information about the data. To assist end users in the application of data wrappers a data wrapper GUI is available. Once created, the data agent announces itself to the DF agent as a consequence of which it becomes available to all EMADS users.

3.3.2 Tool Wrappers

Tool wrappers are used to "wrap" up DM software systems and thus create a mining agent. Generally the software systems will be DM tools of various kinds (classifiers, clusters, association rule miners, etc.) although they could also be (say) data normalization/discretization or visualization tools. It is intended that the framework will incorporate a substantial number of different tool wrappers each defined by the nature of the desired I/O which in turn will be informed by the nature of the generic DM tasks that it is desirable for EMADS to be able to perform. Currently the research team has implemented two tool wrappers:

1. The binary valued data, single label, classifier generator.
2. The data normalization/discretization wrapper.

Many more categories of tool wrapper can be envisaged. Mining tool wrappers are more complex than data wrappers because of the different kinds of information that needs to be exchanged.

In the case of a "binary valued, single label, classifier generator" wrapper the input is a binary valued data set together with meta information about the number of classes and a number slots to allow for the (optional) inclusion of threshold values. The output is then a classifier expressed as a set of Classification Rules (CRs). As with data agents, once created, the DM agent announce themselves to the DF agent after which they will becomes available for use to EMADS users.

For example, in the case of the data normalization/discretization wrapper, the LUCS-KDD (Liverpool University Computer Science - Knowledge Discovery in Data) ARM DN (Discretization/ Normalization) software [1] is used to convert data files, such as those available in the UCI data repository [8], into a binary format suitable for use with Association Rule Mining (ARM) applications. This tool has been "wrapped" using the data normalization/discretization wrapper.

[1] $http://www.csc.liv.ac.uk/\tilde{f}rans/KDD/Software/$

4. SYSTEM DEMONSTRATION

Perhaps the best way to obtain an intuitive sense of how the framework typically functions is to briefly look at an example of how it has been applied to real world scenarios. The following subsections describe two demonstration applications (Scenarios) implemented within the EMADS framework.

The first (discussed further in subsection 4.1) is a distributed meta mining scenario where EMADS agents are used to merge the results of a number of ARM operations, a process referred to as meta-ARM, to produce a global set of Association Rules (ARs). The challenge here is to minimise the communication overhead, a significant issue in distributed and parallel DM (regardless of whether it is implemented in an agent framework or not).

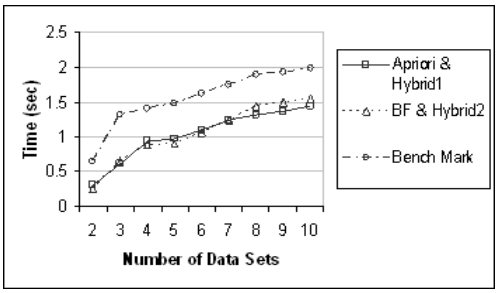
The second scenario (subsection 4.2) is a classification scenario where the objective is to generate a classifier (predictor) fitted to EMADS user's specified data set. It has been well established within the DM research community, for reasons that remain unclear but are concerned with the nature of the input data, that there is no single "best" classification algorithm. The aim of this second scenario is therefore to identify a "best" classifier given a particular data set. Best in this context is measured in terms of classification accuracy. This experiment not only serves to illustrate the advantageous of EMADS but also provides an interesting comparison of a variety of classification techniques and algorithms.

4.1 Meta ARM (Association Rule Mining) scenario

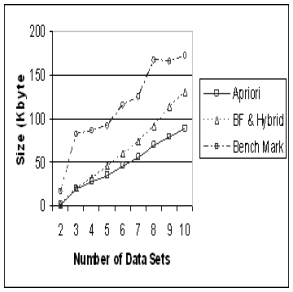
The term *meta mining* is defined, in the context of EMADS, as the process of combining the individually obtained results of N applications of a DM activity. The motivation behind the scenario is that data relevant to a particular DM application may be owned and maintained by different, geographically dispersed, organizations. There is therefore a "privacy and security" issue, privacy preserving issues [1] are of major concerns in inter enterprise DM when dealing with private databases located at different sites. One approach to addressing the meta mining problem is to adopt a distributed approach. The meta mining scenario considered here is a meta Association Rule Mining (meta ARM) scenario where the results of N ARM operations, by N agents, are brought together.

4.1.1 Dynamic Behaviour of System for Meta ARM operations

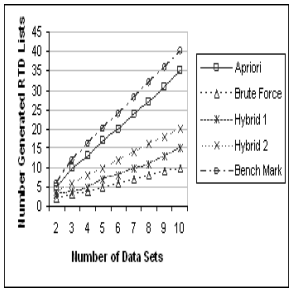
The meta ARM EMADS illustration described here was used to identify the most efficient Meta ARM agent process given a number of alternatives. The first algorithm was a bench mark algorithm, against which other Meta ARM algorithms were compared. Four comparison meta ARM algorithms were constructed (Apriori, Brute Force, Hybrid 1 and Hybrid 2). Full details of the algorithms can be found in [3]. In each case it was assumed that each data source would produce a set of frequent sets, using some ARM algorithm, with the results stored in a common data structure. These data structures would then be merged in some manner through a process of agent collaboration. Each of the Meta ARM algorithms made use of a Return To Data (RTD) lists, one per data set, to contain lists of itemsets whose support was not included in the original ARM operation and for which the count was to be obtained by a return to the raw data held at a data agent. The RTD lists comprised zero, one or more tuples of the form $< I, sup >$, where I is an item set for which a count is required and sup is the desired count. RTD lists are constructed as a meta ARM algorithm progresses. During RTD list construction the sup value will be 0, it is not until the RTD list is processed that actual values are assigned to sup. The processing of RTD lists may occur during, and/or at the end of, the meta ARM process depending on the nature of the meta ARM algorithm used.



(a) Processing Time



(b) Total size of RTD lists



(c) Number of RTD lists

Figure 3: Effect of number of data sources

The meta ARM scenario comprises a set of N data agents and $N+1$ DM agents: N ARM agents and one meta ARM agent. Note that each ARM agent could have a different ARM algorithm associated with it, however a common data structure was assumed to facilitate data interchange. The common data structure used was a T-tree [12], a set enumeration tree structure for storing item sets. Once generated the N local T-trees were passed to the Meta ARM agent which created a global T-tree. During the global T-tree generation process the Meta ARM agent interacted with the various ARM agents in the form of the exchange of RTD lists.

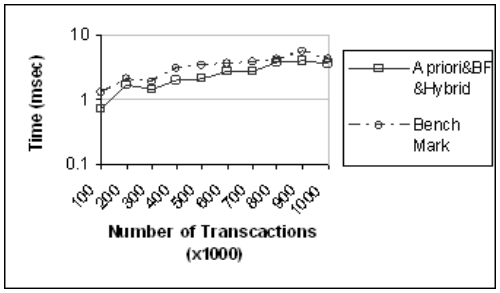
4.1.2 Experimentation and Analysis

To evaluate the five Meta ARM algorithms (including the bench mark algorithm), in the context of the EMADS vision, a number of experiments were conducted. These are described and analyzed in this subsection. The experiments were designed to analyze the effect of the following:

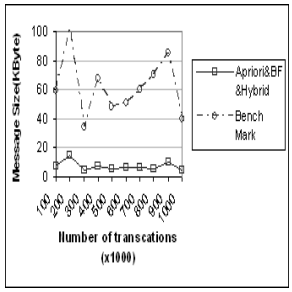
1. The number of data sources (data agents).
2. The size of the data sets (held at data agents) in terms of number of records.
3. The size of the data sets (held at data agents) in terms of number of attributes.

Experiments were run using two Intel Core 2 Duo E6400 CPU (2.13GHz) computers with 3GB of main memory (DDR2 800MHz), Fedora Core 6, Kernel version 2.6.18 running under Linux except for the first experiment where two further computers running under Windows XP were added. For each of the experiments we measured:

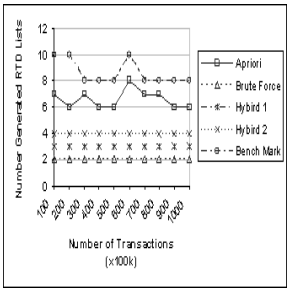
- Processing time (seconds/mseconds),



(a) Processing Time



(b) Total size of RTD lists



(c) Number of RTD lists

Figure 4: Effect of number of data sources

- The size of the RTD lists (Kbytes), and
- The number of RTD lists generated.

Note that the authors did not use the well known IBM QUEST generator [2] because many different data sets (with the same input parameters) were required and it was found that the QUEST generator always generated the same data given the same input parameters. Instead the authors used the LUCS KDD data generator[2]. Figure 3 shows the effect of adding additional data sources using the four Meta ARM algorithms and the bench mark algorithm. For this experiment thirteen different artificial data sets were generated and distributed among four machines using $T = 4$ (average number of items per transactions), $N = 20$ (Number of attributes), $D = 100k$ (Number of transactions. Note that the slight oscillations in the graphs result simply from a vagary of the random nature of the test data generation.

Figure 4 demonstrates the effect of increasing the number of records. The input data for this experiment was generated by producing a sequence of ten pairs of data sets (with T =4, N =20) representing two sources on two differentmachines. From graph 3(a) it can be seen that the Brute Force and Hybrid 1 algorithms work best because the size of the return to data lists are limited as no unnecessary candidate sets are generated. This is illustrated in graph 3(b). Graph 3(b) also shows that the increase in processing time in all cases is due to the increase in the number of records only; the size of the RTD lists remains constant throughout as does the number of RTD lists generated (graph 3(c)).

Figure 3 and 4 also indicate, at least with respect to meta ARM, that EMADS offers positive advantages in that all the Meta ARM algorithms were more computationally efficient than the bench mark algorithm. The results of the analysis also indicated that the Apriori Meta ARM approach coped best with a large number of data sources, while the Brute Force and Hybrid 1 approaches coped best with increased data sizes (in terms of column/rows).

[2] *http : //www.csc.liv.ac.uk/f̃rans/KDD/Software//LUCS − KDD − DataGen/*

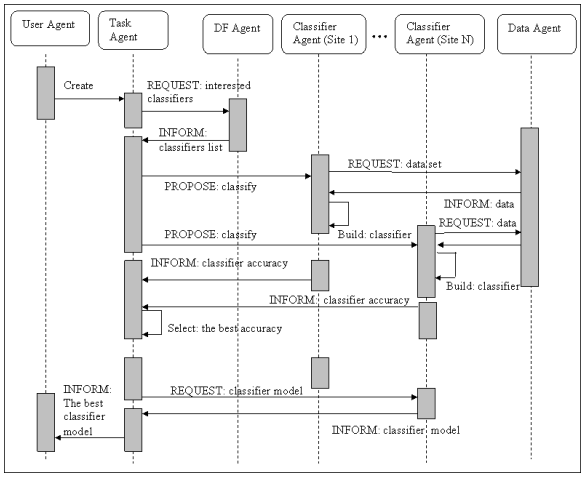


Figure 5: Classification Task Sequence Diagram.

4.2 Classifier Generation scenario

In this subsection the operation of EMADS is illustrated in the context of a classifier generation task, however much of the discussion is equally applicable to other generic DM tasks. The scenario is that of an end user who wishes to obtain a "best" classifier founded on a given, pre-labelled, data set; which can then be applied to further unlabelled data. The assumption is that the given data set is binary valued and that the user requires a single-label, as opposed to a multi-labelled, classifier. The request is made using the individual's user agent which in turn will spawn an appropriate task agent.

For this scenario the task agent interacts with mining agents that hold single labelled classifier generators that take binary valued data as input. Each of these mining agents is then accessed and a classifier, together with an accuracy estimate, requested. Once received the task agent selects the classifier with the best accuracy and returns this to the user agent.

The DM agent wrapper in this case provides the interface that allows input of: (i) the identifier for the data set to be classified, and (ii) the number of class attributes (a value that the mining agent cannot currently deduce for itself); while the user agent interface allows input for threshold values (such as support and confidence values).

The output is a classifier together with an accuracy measure. To obtain the accuracy measures the classifier generators (DM agents) build their individual classifier using the first half of the input data as the "training" set and the second half of the data as the "test" set. An alternative approach might have been to use Ten Cross Validation (TCV) to identify the best accuracy. It should be noted that the objective here is to return a classifier, using TCV ten classifiers will be built and thus one of them would have to be selected.

From the literature there are many reported techniques available for generating classifiers. For the scenario reported here the authors' used implementations of eight different algorithms[3]:

1. FOIL (First Order Inductive Learner) [28]: The well estab-

[3] Taken from the LUCS-KDD repository at *http : //www.csc.liv.ac.uk/f̃rans/KDD/Software/*

Table 1: Classification Results

Data Set	Classifier	Accuracy	Time(sec)
connect4.D129.N67557.C3	RDT	79.76	502.65
adult.D97.N48842.C2	IGDT	86.05	86.17
letRecog.D106.N20000.C26	RDT	91.79	31.52
anneal.D73.N898.C6	FOIL	98.44	5.82
breast.D20.N699.C2	IGDT	93.98	1.28
congres.D34.N435.C2	RDT	100	3.69
cylBands.D124.N540.C2	RDT	97.78	41.9
dematology.D49.N366.C6	RDT	96.17	11.28
heart.D52.N303.C5	RDT	96.02	3.04
auto.D137.N205.C7	IGDT	76.47	12.17
penDigits.D89.N10992.C10	RDT	99.18	13.77
soybean.D118.N683.C19	RDT	98.83	13.22
waveform.D101.N5000.C3	RDT	96.81	11.97

lished inductive learning algorithm for the generation of Classification Association Rules (CARs).

2. TFPC (Total From Partial Classification): A CAR generator [13] founded on the P and T-tree set enumeration tree data structures.
3. PRM (Predictive Rule Mining) [32]: An extension of FOIL.
4. CPAR (Classification based on Predictive Association Rules) [32]: A further development from FOIL and PRM.
5. IGDT (Information Gain Decision Tree) classifier: An implementation of the well established C4.5 style of decision tree based classifier using information gain as the "splitting criteria".
6. RDT (Random Decision Tree) classifier: A decision tree based classifier that uses most frequent current attribute as the "splitting criteria".
7. CMAR (Classification based on Multiple Association Rules): A well established Classification Association Rule Mining (CARM) algorithm [23].
8. CBA (Classification Based on Associations): Another well established CARM algorithm [22].

These were placed within an appropriately defined tool wrapper to produce eight (single label binary data classifier generator) DM agents. This was found to be a trivial operation indicating the versatility of the wrapper concept.

Thus each mining agent's basic function was to generate a classification model using its own classifier and provide this to the task agent. The task agent then evaluates all the classifier models and chooses the most accurate model to be returned to the user agent. The negotiation process amongst the agents is represented by the sequence diagram given in Figure 5 (the figure assumes that appropriate data agents exist). The figure includes *N* classification agents. The sequence of processing events commences with a user agent which spawns a (classification) task agent, which in turn announces itself to the DF agent. The DF agent returns a list of classifier DM agents that can potentially be used to generate the desired classifier.

The task agent then interacts with these DM agents who each generate a classifier and return statistical information regarding the accuracy of their classifier. The task agent selects the DM agent that has produced the best accuracy and requests the associated classifier; this is then passed back to the user agent.

4.2.1 Experimentation and Analysis

To evaluate the EMADS classification scenario, as described above, a sub-set of the data sets available at the UCI machine learning data repository [8] were used (preprocessed by data agents so that they were discretized/normalized into a binary form). The results are presented in Table 1. Each row in the table represents a particular request and gives the name of the data set, the selected best algorithm as identified from the interaction between the EMADS agents, the resulting best accuracy and the total EMADS execution time from creation of the initial Task Agent to the final "best" classifier being returned to the user agent. The naming convention used in the Table is that: D equals the number of attributes (after discretization/normalization), N the number of records and C the number of classes (although EMADS has no requirement for the adoption of this convention).

The results demonstrate firstly, that EMADS can usefully be adopted to produce a best classifier from a selection of classifiers. Secondly, that the operation of EMADS is not significantly hindered by agent communication overheads, although this has some effect. Generation time, in most cases does not seem to be an issue, so further classifier generator mining agents could easily be added. The results also reinforce the often observed phenomena that there is no single best classifier generator suited to all kinds of data.

5. CONCLUSIONS

This paper described EMADS, a multi-agent framework for DM. The architecture provides a framework for the construction and operation of distributed software agents. The principal advantages offered by the system are that of experience and resource sharing, flexibility and extendibility, and (to an extent) protection of privacy and intellectual property rights. The use of a single facilitator offers both advantages and weaknesses with respect to scalability and fault tolerance. On the plus side, the grouping of a facilitator with a collection of agents provides a faster look-up service. However, even though the intention is that the facilitator assists agents in finding one another and then to "step aside" while other agents communicate over a direct, dedicated channel so as to prevent a communication bottleneck; there is still the potential for a facilitator to become a critical point of system failure. Further work in this area is therefore required, one solution is to use more than one facilitator deployed on multiple machines for a better fault-tolerant platform.

The framework's operation is illustrated using both meta ARM and classification scenarios. Extendibility is presented by showing how wrappers are used to incorporate existing software into EMADS. Experience to date indicates that, given an appropriate wrapper, existing DM software can very easily be packaged to become a DM agent. Flexibility is illustrated using a classification scenario. Information hiding is illustrated in that users need have no knowledge of how any particular piece of DM software works or the location of the data to be used.

A good foundation has been established for both DM research and genuine application based DM. It is acknowledged that the current functionality of the framework is limited to classification and meta ARM. The research team is at present working towards increasing the diversity of mining tasks that can be addressed. There are many directions in which the work can (and is being) taken forward. One interesting direction is to build on the wealth of distributed DM research that is currently available and progress this in an MAS context. The research team is also enhancing the system's robustness so as to make it publicly available. It is hoped that once the system is live other interested DM practitioners will be prepared

to contribute algorithms and data.

6. REFERENCES

[1] Aggarwal, C. and Yu, P., “A Condensation Approach to Privacy Preserving DataMining”. Lecture Notes in Computer Science, Vol. 2992, pp. 183-199, (2004).

[2] Agrawal, R., Mehta, M., Shafer, J., Srikant, R., Arning, A. and Bollinger, T., “The Quest Data Mining System”. Proceedings 2nd Int. Conf. Knowledge Discovery and Data Mining, KDD, (1996).

[3] Albashiri, K., Coenen, F., Sanderson, R. and Leng. P., “Frequent Set Meta Mining: Towards Multi-Agent Data Mining”. In Bramer, M., Coenen, F.P. and Petridis, M. (Eds.), Research and Development in Intelligent Systems XXIII., Springer, London, pp139-151, (2007).

[4] Albashiri, K., Coenen, F., and Leng. P., “Agent Based Frequent Set Meta Mining: Introducing EMADS”. Artificial Intelligence in Theory and Practice II, Proceedings IFIP, Springer, pp23-32, (2007).

[5] Baazaoui H., Faiz S., Ben Hamed R., Ben Ghezala H., “A Framework for data mining based multi-agent: an application to spatial data”. 3rd World Enformatika Conference WEC’05, Avril, Istanbul, (2005).

[6] Bailey, S., Grossman, R., Sivakumar, H. and Turinsky, A., “Papyrus: a system for data mining over local and wide area clusters and super-clusters". In Proceedings Conference on Supercomputing, page 63. ACM Press, (1999).

[7] Bellifemine, F. Poggi, A. and Rimassi, G., “JADE: A FIPA-Compliant agent framework”. Proceedings Practical Applications of Intelligent Agents and Multi-Agents, pg 97-108, (1999). (See http://sharon.cselt.it/projects/jade for latest information).

[8] Blake, C. and Merz, C. , “UCI Repository of machine learning databases”. Irvine, CA: University of California, Department of Information and Computer Science. http://www.ics.uci.edu/mlearn/MLRepository.html, (1998).

[9] Bose, R. and Sugumaran, V., “IDM: An Intelligent Software Agent Based DataMining Environment”. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 2888-2893 San Diego, CA: IEEE Press, (1998).

[10] Bota, J., Gmez-Skarmeta, A., Valds, M., and Padilla, A., “Metala: A meta-learning architecture". Fuzzy Days, pages 688 Ű 698, (2001).

[11] Cao, L., and Zhang, C., “F-Trade: Agent-mining symbiont for financial services,” AAMAS, ACM Press, pp. 1363Ű1364, (2007).

[12] Coenen, F., Leng, P., and Goulbourne, G., “Tree Structures for Mining Association Rules”. Journal of Data Mining and Knowledge Discovery, Vol 8, No 1, pp25-51, (2004).

[13] Coenen, F., Leng, P. and Zhang, L. “Threshold Tuning for Improved Classification Association Rule Mining”. Proceeding PAKDD, LNAI3158, Springer, pp216-225, (2005).

[14] Foundation for Intelligent Physical Agents, FIPA 2002 Specification. Geneva, Switzerland. (See http://www.fipa.org/specifications/index.html), (2002).

[15] Giuseppe, D., Giancarlo, F., “A customizable multi-agent system for distributed data mining”. Proceedings of the 2007 ACM symposium on applied computing, Pages: 42 47, (2007).

[16] Gorodetsky, V., Karsaeyv, O., Samoilov, V., “Multi-agent technology for distributed data mining and classification". Proceedings Int. Conf. on Intelligent Agent Technology (IAT 2003), IEEE/WIC, pp438-441, (2003).

[17] Grossman, R. and Turinsky, A., “A framework for finding distributed data mining strategies that are intermediate between centralized strategies and in-place strategies”, In KDD Workshop on Distributed Data Mining, (2000).

[18] Kamber, M.,Winstone, L.,Wan, G., Shan, S. and Jiawei, H., “Generalization and Decision Tree Induction: Efficient Classification in Data Mining”. Proceedings of the Seventh International Workshop on Research Issues in Data Engineering, pp.111-120, (1997).

[19] Kargupta, H., Byung-Hoon, et al., “Collective Data Mining: A New Perspective Toward Distributed Data Mining". Advances in Distributed and Parallel Knowledge Discovery, MIT/AAAI Press, (1999).

[20] Kargupta, H., Hamzaoglu, I. and Stafford B., “Scalable, Distributed Data Mining Using an Agent Based Architecture". Proceedings of Knowledge Discovery and Data Mining, AAAI Press, 211-214, (1997).

[21] Klusch, M., Lodi, G., “Agent-based Distributed Data Mining: The KDEC Scheme. Intelligent Information Agents" Ű The AgentLink Perspective. Lecture Notes in Computer Science 2586, Springer, (2003).

[22] Liu, B. Hsu, W. and Ma, Y., “Integrating Classification and Assocoiation Rule Mining”. Proceedings KDD-98, New York, 27-31 August. AAAI. pp80-86, (1998).

[23] Li W., Han, J. and Pei, J., “CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules”. Proceedings ICDM, pp369-376, (2001).

[24] Luo, P., Huang, R., He, Q., Lin, F., and Shi, Z., “Execution engine of meta-learning system for kdd in multi-agent environment". Technical report, Institute of Computing Technology, Chinese Academy of Sciences, (2005).

[25] METAL Project. Esprit Project METAL, (2002). http://www.metal-kdd.org

[26] Peng, S., Mukhopadhyay, S., Raje, R., Palakal, M. and Mostafa, J., “A Comparison Between Single-agent and Multi-agent Classification of Documents". Proceedings 15th International Parallel and Distributed Processing Symposium, pp935-944, (2001).

[27] Prodromides, A., Chan, P. and Stolfo, S., “Meta-Learning in Distributed Data Mining Systems: Issues and Approaches". In Kargupta, H. and Chan, P. (Eds), Advances in Distributed and Parallel Knowledge Discovery. AAAI Press/The MIT Press, pp81-114, (2000).

[28] Quinlan, J. R. and Cameron-Jones, R. M., “FOIL: A Midterm Report”. Proceedings ECML, Vienna, Austria, pp3-20, (1993).

[29] Shi, Z., Zhang, H., Cheng, Y., Jiang, Y., Sheng, Q., and Zhao, Z., “Mage: An agent-oriented programming environment". In Proceedings of the IEEE International Conference on Cognitive Informatics, pages 250 Ű 257, (2004).

[30] Stolfo, S., Prodromidis, A. L., Tselepis, S. and Lee, W., “JAM: Java Agents for Meta-Learning over Distributed Databases". Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 74-81, (1997).

[31] Vilalta, R., Christophe, G., Giraud-Carrier, Brazdil, P., Soares, C., “Using Meta-Learning to Support Data Mining". IJCSA 1(1): 31-45, (2004).

[32] Yin, X. and Han, J., “CPAR: Classification based on Predictive Association Rules”. Proceedings SIAM Int. Conf. on Data Mining (SDM’03), San Fransisco, CA, pp. 331-335, (2003).