A Generic and Extendible Multi-Agent Data Mining Framework

Kamal Ali Albashiri, and Frans Coenen

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

Abstract. A generic and extendible Multi-Agent Data Mining (MADM) framework, EMADS (the Extendible Multi-Agent Data mining System) is described. The central feature of the framework is that it avoids the use of agreed meta-language formats by supporting a system of wrappers. The advantage offered is that the system is easily extendible, so that 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 and evaluation of the framework's operation is provided by considering two MADM scenarios.

1 Motivation and Goals

Multi-Agent Data Mining (MADM) seeks to harness the general advantages of Multi-Agent Systems (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. The main issues with MADM are the disparate nature of DM and the wide range of tasks encompassed. Any desired generic MADM framework therefore requires a sophisticated communication mechanism to support it. In the work described here we address the communication requirements of MADM by using a system of mediators and wrappers coupled with an Agent Communication Language (ACL) such as FIPA ACL [8]. We believe this can more readily address the issues concerned with the variety and range of contexts to which a generic MADM can be applicable. The use of wrappers also avoids the need for agreed meta-language formats.

To investigate and evaluate the expected advantages of wrappers and mediators in the context of generic MADM, we have developed and implemented (in JADE) a multi-agent framework, EMADS (the Extendible Multi-Agent Data mining System). The primary goal of the EMADS framework is extendibility; we wish to provide a means for integrating new DM algorithms and data sources in our MADM framework. However, EMADS also seeks to address some of the issues of DM that would benefit from the use of a generic framework. EMADS provides:

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

- Minimisation 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 paper's organisation is as follows. A brief review of some related work on MADM is presented in Section 2. The conceptual framework, together with an overview of the wrapper principle, is presented in Section 3 and Section 4. The framework's operation is illustrated in Section 5 using two DM scenarios, and finally some conclusions are presented in Section 6.

2 Related Work

MAS have shown much promise for flexible, fault-tolerant, distributed problem solving. Some MADM frameworks focus on developing complex features for specific DM tasks, without attempting to provide much support for usability or extendibility [10]. The success of peer-to-peer systems and negotiating agents has engendered a demand for more generic, flexible, robust frameworks.

There have been only few such generic MADM systems. An early example was IDM [6], a multi-agent architecture for direct DM to help businesses gather intelligence about their internal commerce agent heuristics and architectures for KDD. In [3] a generic task framework was introduced, but designed to work only with spatial data. The most recent system was introduced in [9] where the authors proposed a multi-agent system to provide a general framework for distributed DM applications. In this system the effort to embed the logic of a specific domain has been minimised and is limited to the customisation of the user. However, although its customisable feature is of a considerable benefit, it still requires users to have very good DM knowledge. The EMADS system which we describe below aims to allow DM algorithms to be embedded in a flexible framework with minimum effort by the user.

3 System Architecture

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.

Conceptually the nature of the requests that may be posted by EMADS users is extensive. In the current demonstration implementation a number of generic requests are supported directed at classification and Association Rule Mining (ARM) scenarios. Two exemplar scenarios are used to illustrate this paper (Section 5).

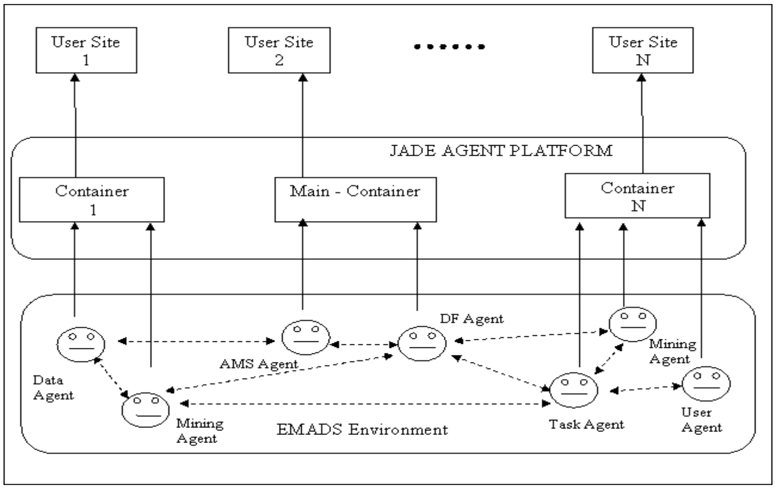


Fig. 1. EMADS Architecture as Implemented in Jade

Fig. 1 presents the EMADS architecture as implemented in JADE (The Java Agent Development Environment) [4]. It shows a sample collection of several application agents and housekeeping agents, organised as a community of peers by a common relationship to each other, that exist in a set of containers. In particular the main container holds the housekeeping agents (an Agent Management System (AMS) agent and a Directory Facilitator (DF) agent). These are specialized server agents responsible for facilitating agents to locate one another.

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 process user requests, 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 threshold 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 combination of agents to execute the request. On completion the results are sent back to the user agent for display.

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 services provider 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.

4 System Extendibility

One of the principal objectives of EMADS is to provide an easily extendible MADM framework that can easily accept new data sources and new data mining techniques. The desired extendibility is achieved by a system of wrappers. EMADS wrappers are used to "wrap" data mining artefacts so that they become EMADS agents and can communicate with other EMADS agents. As 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 data or data mining agents. The wrappers essentially provide an application interface to EMADS that has to be implemented by the end user; this has been designed to be a fairly trivial operation.

EMADS provides the definition of an abstract parent agent class and every instance agent object (i.e., a program that implements a learning DM algorithm) is then defined as a subclass of this parent class. Through the variables and methods inherited by all agent subclasses, the parent agent class describes a simple and minimal interface that all subclasses have to comply to. As long as an agent conforms to this interface, it can be introduced and used immediately as part of the EMADS system. Two broad categories of wrapper have been defined: (i) data wrappers and (ii) tool wrappers.

4.1 Data Wrappers

Data wrappers are used to "wrap" a data source and consequently create a data agent. 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.

4.2 Tool Wrappers

Tool wrappers are used to “wrap” up data mining software systems and thus create a mining agent. Generally the software systems will be data mining tools of various kinds (classifiers, clusters, AR miners, etc.) although they could also be (say) data normalisation/discretization or visualization tools. It is intended that EMADS 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 data mining tasks that it us desirable for EMADS to be able to perform.

5 System Demonstration

The operation of EMADS is described in the following two subsections by considering two demonstration applications (scenarios).

5.1 Meta ARM (Association Rule Mining) scenario

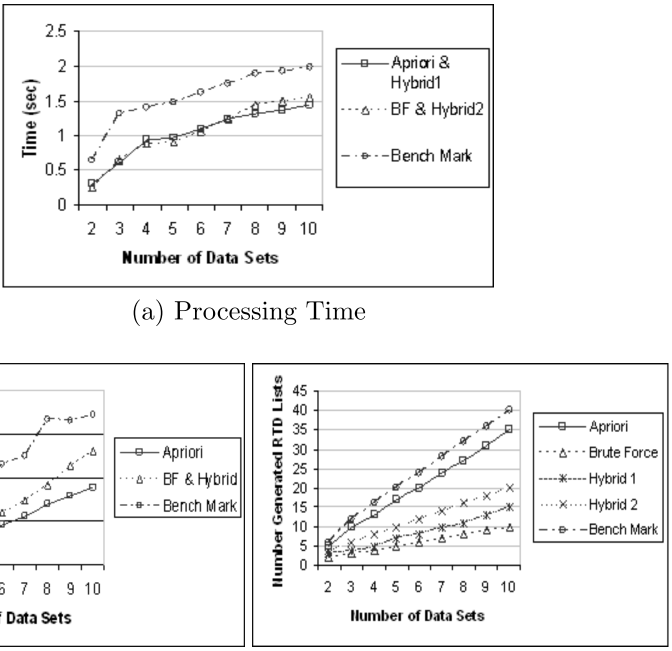
Meta Mining is defined here as the process of combining individually obtained results of N applications of a DM activity. The motivation behind the scenario is that data relevant to a particular DM application may be owned and maintained by different, geographically dispersed, organizations.

The meta ARM scenario comprises a set of N data agents, N ARM mining agents and a meta ARM agent. Note that each ARM mining agent could have a different ARM algorithm associated with it, although, it is assumed that a common data structure is used to facilitate data interchange. For the scenario described here a set enumeration tree structure called a T-tree [7] was used. Once generated the N *local* T-trees are passed to the Meta ARM agent which creates a *global* T-tree. During the global T-tree generation process the Meta ARM agent interacts with the various ARM agents. There are a number of strategies that can be adopted with respect to when in the process intra agent communication should be made. The authors identified five distinct strategies (Benchmark, Apriori, Brute Force, Hybrid 1 and Hybrid 2). A full description of the algorithms can be found in [1].

5.1.1 Experimentation and Analysis

To evaluate the five Meta ARM algorithms, in the context of the EMADS vision, a number of experiments were conducted designed to analyze the effect of: (i) the number of data agents, (ii) the size of the data agents’ datasets in terms of number of records, and (iii) the size of the data agents’ datasets in terms of number of attributes. For each of the experiments we measured: (i) processing time, (ii) the overall size of the communications (Kbytes), and (iii) the number of individual communications.

The results shown in Fig. 2 indicate, 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 (no intra agent cooperation). 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).



(a) Processing Time

(b) Total size of RTD lists

(c) Number of RTD lists

Fig. 2. Effect of number of data sources.

5.2 Classifier Generation scenario

The Classifier Generation 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 generate a classifier, together with an accuracy estimate. Once received the task agent selects the classifier with the best accuracy and returns this to the user agent.

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[1]. 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.

Data Set	Classifier Name	Accuracy	Generation 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-large.D118.N683.C19	RDT	98.83	13.22
waveform.D101.N5000.C3	RDT	96.81	11.97

Table 1. Classification Results

5.2.1 Experimentation and Analysis

To evaluate the classification scenario, a sequence of data sets taken from the UCI machine learning data repository [5] were used (pre-processed by data agents so that they were discretized/normalized into a binary valued format). 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 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.

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 phenomenon that there is no single best classifier generator suited to all kinds of data set. Further details of this process can be also found in Albashiri et al. [2].

6 Conclusions

This paper described EMADS, a generic multi-agent framework for DM. The principal advantages offered by the system are that of experience and resource

[1] Taken from the LUCS-KDD software repository at $http://www.csc.liv.ac.uk/\tilde{f}rans/KDD/Software/$

sharing, flexibility and extendibility, protection of privacy and intellectual property rights and information hiding. The framework's operation was illustrated using meta ARM and classification scenarios. Extendibility is demonstrated 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 the classification scenario. Information hiding is demonstrated in that users need have no knowledge of how any particular piece of DM software works or the location of the data used.

A good foundation has been established for both DM research and genuine application based DM. 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 a 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.

References

1. 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.), Springer, London, pp139-151, (2007).
2. Albashiri, K., Coenen, F., and Leng. P., "EMADS: An Extendible Multi-Agent Data Miner". In Bramer, M., Coenen, F.P. and Petridis, M. (Eds.), Springer, London, pp263-276, (2008).
3. Baazaoui H., Faiz S., Hamed R., and Ghezala H., "A Framework for data mining based multi-agent: an application to spatial data". 3rd World Enformatika Conference, Istanbul, (2005).
4. 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).
5. 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).
6. Bose, R. and Sugumaran, V., "IDM: An Intelligent Software Agent Based Data Mining Environment". In Proceedings of IEEE Press, San Diego, CA, (1998).
7. Coenen, F., Leng, P., and Goulbourne, G., "Tree Structures for Mining Association Rules". Journal of DM and Knowledge Discovery, Vol 8, No 1, pp25-51, (2004).
8. Foundation for Intelligent Physical Agents, FIPA 2002 Specification. Geneva, Switzerland. (See http://www.fipa.org/specifications/index.html), (2002).
9. Giuseppe, D., Giancarlo, F., "A customisable multi-agent system for distributed data mining". Proceedings ACM symposium on applied computing, (2007).
10. Klusch, M., Lodi, G., "Agent-based Distributed Data Mining: The KDEC Scheme. Intelligent Information Agents" The AgentLink Perspective. Lecture Notes in Computer Science 2586, Springer, (2003).